ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τεχνικές Ανωνυμοποίησης Δεδομένων σε Γενεαλογικά Δέντρα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ρεβέκκα Κ. Καλλή

Επιβλέπων: Μιλτιάδης Αναγνώστου
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2019

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τεχνικές Ανωνυμοποίησης Δεδομένων σε Γενεαλογικά Δέντρα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ρεβέκκα Κ. Καλλή

Επιβλέπων: Μιλτιάδης Αναγνώστου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11/11/2019

……..…..	……..…….	……..……..
Μιλτιάδης Αναγνώστου	Ιωάννα Ρουσσάκη	Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π	Επίκουρη Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2019

……..…..

Ρεβέκκα Καλλή
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

2019, η εποχή της τεχνολογίας και της πληροφορίας. Η εποχή όπου εκατομμύρια δεδομένα ανταλλάσσονται καθημερινά μπροστά από μια οθόνη, είτε αυτή είναι υπολογιστή είτε ενός «smartphone». Η εποχή όπου τα δεδομένα του κάθε ασθενή ενός νοσοκομείου ή ενός ιατρείου, των υπαλλήλων σε μια εταιρεία, των πελατών σε ένα κατάστημα ή μια τράπεζα, ακόμη και ενός απλού πελάτη του σουπερμάρκετ είναι πλέον ψηφιακά.

Ανάπτυξη, ευτύχημα ή κίνδυνος; Δυστυχώς, όπως κάθε είδους ανάπτυξη, έτσι και η τεχνολογική έχει τις συνέπειές της. Η ευκολία που έχουν προσφέρει τα δεδομένα σε ψηφιακή μορφή τόσο σε επιχειρήσεις όσο σε δημόσιες υπηρεσίες είναι αδιαμφισβήτητη. Μαζί της όμως έρχεται και μία μάστιγα κερδοσκόπων, που συχνά είναι αυτοί που νομίμως τους παραχωρούνται τα δεδομένα αυτά, και που δεν χάνουν ευκαιρία να καταπατήσουν τα δικαιώματα του κάθε «απλού πολίτη» για να χρησιμοποιήσουν τα στοιχεία του σε παράνομες έρευνες ή καλύτερα , να τα πουλήσουν σε τρίτους εώς ότου επιτευχθεί ο σκοπός τους και λάβουν ένα χρηματικό ποσό.

Κινήσεις για προστασία του κάθε χρήστη έχουν γίνει. Μία από αυτές το πλέον γνωστό σε όλους GDPR (General Data Protection Regulation) που στόχος του είναι να προστατεύει όλους τους πολίτες της Ευρωπαικής Ένωσης από παραβιάσεις και καταπατήσεις της ιδιωτικότητας τους και έχει τη δυνατότητα να εκδόσει πρόστιμο στον κάθε παραβάτη. Αυτό όμως δεν είναι αρκετό. Τις περισσότερες φορές, τα άτομα που τείνουν να «επιτεθούν» και να κλέψουν πληροφορίες είναι έμπειρα και τα ίχνη τους είναι άφαντα με αποτέλεσμα ο νόμος αυτός να μην τους επηρεάζει στην τελική. Γι’αυτόν ακριβώς το λόγο, είναι αναγκαία η προστασία των δεδομένων του κάθε ατόμου και με άλλους τρόπους.

Βασικός και κύριος τρόπος προστασίας των προσωπικών δεδομένων, η ανωνυμοποίηση. Διάφορες τεχνικές επεξεργασίας πληροφοριών έχουν εξελιχθεί ραγδαία τα τελευταία χρόνια. Τα ψηφιακά δεδομένα σε σημαντικές κυρίως επιχειρήσεις είναι πλέον «ανώνυμα». Σκοπός αυτού; Ακόμη και αν κλαπούν, ο επιτιθέμενος να μην μπορεί να τα αποκωδικοποιήσει όυτως ώστε να αποτραπεί η περαιτέρω εκμετάλλευση τους. Όμως, κάποιες τεχνικές είναι αποδοτικές, κάποιες όχι.

Σκοπός λοιπόν αυτής της διπλωματικής είναι να μελετήσει τα παραπάνω και συγκεκριμένα να προσπαθήσει να δει την πλευρά τόσο του επιτιθέμενου όσο και του θύματος σε μία επίθεση πάνω στα δεδομένα ενός γενεαλογικού δέντρου. Η αναπόφευκτη σύνδεση ατόμων σε ένα γενεαλογικό δέντρο καθιστά πιο δύσκολη την αποδοτική ανωνυμοποίηση.

Λέξεις Κλειδιά

Ανωνυμοποίηση, αποανωνυμοποίηση, GDPR, Gedcom Standard, δεδομένα, ιδιωτικότητα, προστασία προσωπικών δεδομένων, παραβίαση, γενεαλογικό δέντρο, γενεαλογικός γράφος, επίθεση, ταυτοποίηση.

Abstract

2019, the era of technology and information. The time when millions of data are exchanged daily in front of a screen, be it a computer or a smartphone. The time when the data of every patient in a hospital or medical office, company employees, customers in a store or bank, even a single supermarket customer is now digital.

Growth, happiness or risk? Unfortunately, just like any kind of development, technology has its consequences. The convenience of providing data in digital form to both businesses and public services is unquestionable. But with it comes a scourge of speculators, often the ones who are legally granted this data, who miss the opportunity to infringe on the rights of any "ordinary citizen" to use his information in illegal investigations or, better, sell to third parties until their purpose is met and receive a sum of money.

Actions to protect each user have been made. One of the most well-known of all is the GDPR (General Data Protection Regulation), which aims to protect all citizens of the European Union from violations and violations of their privacy and has the power to issue a fine to any offender. But that's not enough. Most of the time, people who tend to "attack" and steal information are experienced and their traces are invisible so this law does not affect them in the end. For this very reason, it is necessary to protect the data of each individual in other ways.

The main way of protecting personal data is anonymization. Various information processing techniques have evolved rapidly in recent years. Digital data in major businesses is now "anonymous". The purpose of this? Even if they are stolen, the attacker cannot decode them to prevent further exploitation. But some techniques are efficient, some are not.

The purpose of this thesis is therefore to study the above and in particular to try to see both the attacker and the victim's side in an attack on the data of a pedigree tree. The inevitable attachment of individuals to a pedigree tree makes it difficult to efficiently anonymize.

Keywords

Anonymization, de-anonymization, GDPR, Gedcom, Gedcom Standard, data, privacy, breach, family tree, attack, techniques

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε κατά τη διάρκεια του ακαδημαϊκού έτους 2018-2019 στον τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής του Εθνικού Μετσόβιου Πολυτεχνείου και σηματοδοτεί το τέλος των προπτυχιακών μου σπουδών.
Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή Μιλτιάδη Αναγνώστου για την ευκαιρία που μου έδωσε να αναλάβω τη διπλωματική μου εργασία και να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα, καθώς και για την αμέριστη βοήθεια του σε οτιδήποτε χρειαζόμουν. Επίσης, θα ήθελα να ευχαριστήσω τους γονείς μου, Κυριάκο και Μαρίνα, καθώς και την αδερφή μου Μαρία και το γιο μου Αντρέα, για την συμπαράσταση τους κατά τη διάρκεια των σπουδών μου, αλλά και γενικότερα καθ' όλη τη διάρκεια της πορείας μου μέχρι σήμερα. Τέλος, ένα μεγάλο ευχαριστώ στους φίλους και φίλες μου, οι οποίοι ήταν και είναι πάντα κοντά μου όταν τους χρειάζομαι.

Ρεβέκκα Καλλή,
Αθήνα, Νοέμβριος 2019

Πίνακας Περιεχομένων

Πίνακας Πινάκων

Πίνακας Εικόνων

1 Εισαγωγή

1.1 Το πρόβλημα

Στη σύγχρονη εποχή που ζούμε, η αύξηση της χρήσης των μέσων κοινωνικής δικτύωσης κυρίως από νεαρά σε ηλικία άτομα που έχουν ως σκοπό να προβάλουν την προσωπική τους ζωή, η είσοδος σε παραπλανητικούς ιστότοπους, ιστότοπους δηλαδή που έχουν ως σκοπό να ξεγελάσουν τον χρήστη παίρνοντας προσωπικά δεδομένα του, η εισαγωγή πληροφοριών σε υποτιθέμενα ερωτηματολόγια και ιατρικές φόρμες ή ακόμη και η απλή καθημερινή συναλλαγή καθιστούν απαραίτητη την προστασία των προσωπικών δεδομένων του κάθε χρήστη. Καλώς ή κακώς η τεχνολογία και η μετατροπή των πληροφοριών του κάθε ατόμου σε ψηφιακά δεδομένα, αποτελεί σημαντικό κομμάτι σήμερα, κάτι που δεν είναι απαραίτητα κακό, φτάνει να υπάρχει η γνώση και η «καλή καχυποψία» που θα βοηθήσουν στην προστασία της ιδιωτικότητας του καθενός.

1.2 Σκοπός της διπλωματικής εργασίας

Σκοπός αυτής τη διπλωματικής είναι η εφαρμογή της τεχνικής της ανωνυμοποίησης σε γενεαλογικά δέντρα καθως και η διερεύνηση της αποτελεσματικότητάς της σε επιθέσεις για αποανωνυμοποίηση και υποκλοπή των δεδομένων αυτών. Η επιλογή αυτού του είδους δεδομένων, δηλαδή του γενεαλογικού δέντρου, θεωρήθηκε σημαντική λόγω του ότι τα άτομα που το αποτελούν συνδέονται μεταξύ τους και έτσι η καλή ανωνυμοποίηση τους είναι απαραίτητη για να μην οδηγήσει σε αλυσιδωτή αποανωνυμοποίηση.
Θα χρησιμοποιηθεί η ανωνυμοποίηση σε διάφορες μορφές της καθώς και πρακτικά αποτελέσματα μέσα από εκτέλεση προγραμμάτων σε Python για να φανεί κατά πόσο είναι όντως αποδοτική όσον αφορά την προστασία προσωπικών δεδομένων, ή αν είναι ευάλωτη σε οποιαδήποτε είδους «επίθεση». Θα γίνει δηλαδή επίθεση σε κάθε μορφή ανωνυμοποίησης των δεδομένων του δέντρου αυτού με σκοπό την λήψη των ευαίσθητων πληροφοριών του κάθε ατόμου που το αποτελεί. Όλα τα παραπάνω θα εκτιμηθούν με βάση κάποια στατιστικά στοιχεία που παράγονται μέσα από τα αποτελέσματα.

1.3 Οργάνωση Τόμου

Η διπλωματική εργασία αποτελείται από 6 κεφάλαια.
Στο παρόν κεφάλαιο (1ο) έχει γίνει εισαγωγή στο αντικείμενο της εργασίας. Δηλαδή στην προστασία των προσωπικών δεδομένων και έχει τεθεί ο σκοπός της εργασίας που είναι η δημιουργία ενός γενεαλογικού δέντρου και η μετέπειτα δοκιμή τεχνικών ανωνυμοποίησης και αποανωνυμοποίησης σε αυτό.

Στο 2ο κεφάλαιο γίνεται μια επεξήγηση διαφόρων εννοιών που θα συναντηθούν παρακάτω όσον αφορά τα γενεαλογικά δέντρα ώστε να είναι εύκολο για τον αναγνώστη να μπει πλήρως στο νόημα της διπλωματικής αυτής.
Στο 3ο κεφάλαιο όπως και στο 2ο γίνεται επεξήγηση διαφόρων εννοιών, εδώ όσον αφορά την προστασία δεδομένων.
Στο 4ο κεφάλαιο επεξηγείται βήμα-βήμα η ανάκτηση των δεδομένων που θα χρειαστούν και η δημιουργία ενός γενεαλογικού δέντρου βάσει αυτών.
Στο 5ο κεφάλαιο γίνεται προσπάθεια ανωνυμοποίησης με διάφορες τεχνικές τόσο σε δεδομένα όσο σε γράφους καθώς και επεξήγηση τους.
Στο 6ο κεφάλαιο γίνεται προσπάθεια αποανωνυμοποίησης βασισμένης στις τεχνικές που εφαρμόστηκαν στο 5ο κεφάλαιο.
Στο 7ο κεφάλαιο γίνεται εξαγωγή γενικών συμπερασμάτων.
Στο Παράρτημα Ι βρίσκονται όλα τα προγράμματα που γράφτηκαν σε γλώσσα Python και βοήθησαν στην διπλωματική αυτή.
Στο Παράρτημα ΙΙ βρίσκονται όλα τα αποτελέσματα για τις εκτελέσεις που έγιναν στο κεφάλαιο 6.
Στο Παράρτημα ΙΙΙ βρίσκονται τα μέσα/τεχνολογίες που χρησιμοποιήθηκαν στην παρούσα διπλωματική.

2 Γενεαλογικά Δέντρα

2.1 Ορισμός

Το γενεαλογικό δέντρο ή αλλιώς οικογενειακό δέντρο, είναι ένας γράφος που παρουσιάζει οικογενειακές σχέσεις. Περιγράφει δηλαδή σχέσεις γονέων-παιδιών, συζύγων κ.λπ. Μπορεί να χρησιμοποιηθεί για να δείξει βαθμούς συγγένειας μεταξύ ανθρώπων. Σ' ένα τυπικό γενεαλογικό γράφο για κάθε άτομο περιλαμβάνονται πληροφορίες όπως το ονοματεπώνυμο, το φύλο, ημερομηνίες γέννησης, θανάτου, γάμου και άλλων σημαντικών γεγονότων, τους τόπους όπου έγιναν αυτά τα γεγονότα, επαγγέλματα κ.λπ. Συχνά όμως δημιουργούνται γενεαλογικοί γράφοι με εξειδικευμένες πληροφορίες, όπως ιατρικά δεδομένα, γενετικές πληροφορίες, σωματικά και ψυχολογικά χαρακτηριστικά και γενικότερα οποιοδήποτε χαρακτηριστικό για το οποίο υπάρχει υποψία ότι παρουσιάζει σχέση με την κληρονομικότητα. Πολλά από αυτά τα χαρακτηριστικά είναι προσωπικά ή ευαίσθητα (σύμφωνα με τους ορισμούς που δίνονται στην GDPR και σε άλλα κείμενα προστασίας της ιδιωτικότητας).

2.2 Γράφος

Η αναπαράσταση των στοιχείων ενός γενεαλογικού δέντρου μπορεί να γίνει με την χρήση γράφων. Σε ένα τέτοιο γράφο οι κόμβοι ή κορυφές είναι πρόσωπα και οι ακμές συγκεκριμένες σχέσεις μεταξύ τους, συνήθως γονέα-παιδιού ή συζύγων/συντρόφων. Γενικότερα, ένας γράφος/γράφημα είναι ένα διατεταγμένο ζεύγος $G = (V, E)$ αποτελούμενο από ένα σύνολο V «κορυφών» ή «κόμβων» μαζί με E σύνολο από «ακμές», οι οποίες συνδέουν κορυφές.

Ένα δέντρο με βάση την θεωρία γράφων, είναι ένας μη κατευθυνόμενος γράφος στον οποίο οποιεσδήποτε δύο κορυφές συνδέονται μα ένα και μόνο απλό μονοπάτι. Εάν σε ένα γενεαλογικό δέντρο, κάθε κόμβος ήταν συνδεδεμένος μόνο με ένα εκ των δύο γονιών και ο άλλος δεν υπήρχε στο δέντρο, τότε θα υπήρχε περίπτωση για δέντρο όπως ορίζεται στην θεωρία γράφων. Στην συγκεκριμένη περίπτωση όμως ένας κόμβος εκτός του ότι πιθανόν να συνδέεται και με τους δύο γονείς του, μπορεί επίσης να έχει και κάποιο πλήθος συζύγων. Έτσι, μεταξύ δύο κορυφών είναι αναμενόμενο να υπάρχουν περισσότερα του ενός μονοπάτια (π.χ. το ένα να περνάει από τον πατέρα και το άλλο από τη μητέρα).
Τι συμβαίνει όμως στην περίπτωση αυτή; Θα χρησιμοποιηθεί μια άλλη δομή δεδομένων η οποία παρουσιάζεται πιο κάτω και είναι τα multitrees.

2.3 Multitrees

Πρόσφατα, οι Furnas & Zacks προσπάθησαν να βρουν ένα νέο τύπο δομής για να αντιπροσωπεύει πληροφορίες, κάτι που θα ήταν μεταξύ δέντρων και ακυκλικών γραφημάτων αλλά θα διεύρυνε τις επιλογές του κάθε χρήστη τους όσον αφορά τον σχεδιασμο αλλά και την εισαγωγή πληροφοριών. Κατέληξαν στα multitrees, τα οποία είναι ακυκλικές γραφικές παραστάσεις στις οποίες ο κάθε κόμβος μπορεί να έχει πολλούς γονείς και επίσης μέσα σε ολόκληρη τη δομή μπορούν να περιέχουν πολλά υπόδεντρα. Ακόμη, τα diamonds (ένα διαμάντι εμφανίζεται όταν υπάρχουν δύο μονοπάτια μεταξύ δύο κόμβων) είναι απαγορευμένα σε ένα multitree. Έτσι λοιπόν, τα γενεαλογικά δέντρα δεν είναι εν γένει δέντρα με τη γραφοθεωρητική έννοια, αλλά multitrees.

2.4 Gedcom Standard

Η ονομασία *Gedcom* που θα χρησιμοποιηθεί αρκετά σε αυτή τη διπλωματική είναι ένα ακρωνύμιο του *Genealogical Data Communication* (Επικοινωνία γενεαλογικών δεδομένων). Αναπτύχθηκε από το τμήμα οικογενειακής ιστορίας της *LDS Church* (*The Church of Jesus Christ of Latter-day Saints*) με σκοπό την τυποποίηση της περιγραφής ενός γενεαλογικού δέντρου. Στη συνέχεια μπορεί να χρησιμοποιηθεί για την ανταλλαγή γενεαλογικών δεδομένων μεταξύ διαφόρων γενεαλογικών λογισμικών. Ένα αρχείο Gedcom (gedcom file), που έχει την κατάληξη .ged, περιέχει πληροφορίες για το κάθε άτομο που περιέχεται στο δέντρο και τις σχέσεις μεταξύ αυτών. Το GEDCOM Standard είναι ένα τεχνικό έγγραφο που γράφτηκε για προγραμματιστές ηλεκτρονικών υπολογιστών, προγραμματιστές συστημάτων και τεχνικούς χρήστες. Καλύπτει τα παρακάτω θέματα:

- Γραμματική αναπαράστασης δεδομένων GEDCOM
- Γραμματική γραμμικής σύνδεσης
- Ετικέτες γραμμικής σύνδεσης
- Κώδικες του The Church of Jesus Christ of Latter-day Saints' temple
- Κώδικες χαρακτήρων ANSEL

Στην παρούσα διπλωματική, από τα παραπάνω κεφάλαια ενδιαφέρουν οι παρακάτω ετικέτες (Tags):

Ετικέτα	**Επίσημο Όνομα**	**ΕΠΕΞΗΓΗΣΗ**
HEAD	HEADER	Προσδιορίζει πληροφορίες σχετικά με ολόκληρο το GEDCOM transmissionAG
CHAR	CHARACTER	Μια ένδειξη του συνόλου χαρακτήρων που χρησιμοποιήθηκε για τη σύνταξη αυτών των αυτοματοποιημένων πληροφοριών
SOUR	SOURCE	Το αρχικό ή το πρωτότυπο υλικό από το οποίο ελήφθησαν οι πληροφορίες
GEDC	GEDCOM	Πληροφορίες σχετικά με τη χρήση του GEDCOM σε μια μετάδοση

VERS	VERSION	Υποδεικνύει ποια έκδοση ενός προϊόντος, είδους ή δημοσίευσης χρησιμοποιείται ή αναφέρεται
FORM	FORMAT	Ένα δεδομένο όνομα που δίνεται σε μια συνεπή μορφή στην οποία μπορούν να μεταφερθούν πληροφορίες
INDI	INDIVIDUAL	Ενα άτομο
NAME	NAME	Μια λέξη ή συνδυασμός λέξεων που χρησιμοποιούνται για να βοηθήσουν στην ταυτοποίηση ενός ατόμου, ενός τίτλου ή άλλου στοιχείου. Περισσότερες από μία γραμμές NAME πρέπει να χρησιμοποιούνται για άτομα που ήταν γνωστά με πολλαπλά ονόματα
SEX	SEX	Υποδηλώνει το φύλο ενός ατόμου - αρσενικό ή θηλυκό.
BIRT	BIRTH	Το γεγονός της γέννησης
DATE	DATE	Η ώρα ενός γεγονότος σε μορφή ημερολογίου
PLAC	PLACE	Όνομα της τοποθεσίας ενός γεγονότος
DEAT	DEATH	Το γεγονός του θανάτου
FAMS	FAMILY_SPOUSE	Προσδιορίζει την οικογένεια στην οποία ένα άτομο εμφανίζεται ως σύζυγος
FAMC	FAMILY_CHILD	Προσδιορίζει την οικογένεια στην οποία ένα άτομο εμφανίζεται ως παιδί.
FAM	FAMILY	Προσδιορίζει ένα νόμιμο, κοινό νόμο ή άλλη συνήθη σχέση ανδρών και γυναικών και των παιδιών τους, εάν υπάρχουν, ή μιας οικογένειας που δημιουργείται λόγω της γέννησης ενός παιδιού στον βιολογικό πατέρα και τη μητέρα του
HUSB	HUSBAND	Ένα άτομο στο οικογενειακό ρόλο ενός παντρεμένου άνδρα ή και πατέρα
WIFE	WIFE	Ένα άτομο στο ρόλο μητέρας ή και παντρεμένης γυναίκας
CHIL	CHILD	Το φυσικό, υιοθετημένο, ή σφραγισμένο (LDS) παιδί ενός πατέρα και μιας μητέρας.
TRLR	TRAILER	Στο επίπεδο 0, καθορίζει το τέλος μιας μετάδοσης GEDCOM

Πίνακας 1: Ετικέτες GEDCOM

Για μεγαλύτερη ευκολία, δίνεται ένα παράδειγμα ενός απλού αρχείου GEDCOM που περιέχει όλες τις παραπάνω έννοιες και έπειτα ακολουθεί επεξήγηση του.

```
0 HEAD
1 CHAR UTF-8
1 SOUR Ancestry.com Family Trees
2 VERS (2010.3)
2 NAME Ancestry.com Family Trees
1 GEDC
2 VERS 5.5
2 FORM LINEAGE-LINKED
0 @P1@ INDI
1 BIRT
2 DATE 1993
2 PLAC Denmark
1 NAME Kate
1 SEX F
1 FAMC @F1@
0 @P2@ INDI
1 BIRT
2 DATE 1965
1 NAME Rachel
1 SEX F
1 FAMC @F3@
1 FAMS @F1@
0 @P3@ INDI
1 BIRT
2 DATE 1961
2 PLAC Denmark
1 NAME Peter
1 SEX M
1 FAMC @F2@
1 FAMS @F1@
0 @P4@ INDI
1 BIRT
2 DATE 1937
1 NAME Mary
1 SEX F
1 FAMS @F2@
0 @P5@ INDI
1 NAME Paul
1 BIRT
2 DATE 1940
1 SEX M
1 FAMS @F2@
0 @P6@ INDI
1 BIRT
```

```
2 DATE 1989
2 PLAC Denmark
1 NAME Mary
1 SEX F
1 FAMC @F1@
0 @P7@ INDI
1 BIRT
2 DATE 1926
1 DEAT
2 DATE 2008
1 NAME Kate
1 SEX F
1 FAMS @F3@
0 @P8@ INDI
1 BIRT
2 DATE 1918
2 PLAC Nederlands
1 DEAT
2 DATE 1996
2 PLAC Denmark
1 NAME Michael
1 SEX M
1 FAMS @F3@
0 @F1@ FAM
1 HUSB @P3@
1 WIFE @P2@
1 CHIL @P6@
1 CHIL @P1@
0 @F2@ FAM
1 HUSB @P5@
1 WIFE @P4@
1 CHIL @P3@
0 @F3@ FAM
1 HUSB @P8@
1 WIFE @P7@
1 CHIL @P2@
0 TRLR
```

Το παραπάνω αρχείο GEDCOM χωρίζεται «άτυπα» σε 3 μέρη. Στο 1ο μέρος δίνονται πληροφορίες για το αρχείο που δημιουργήθηκε, στο 2ο δίνονται πληροφορίες για κάθε άτομο του γενεαλογικού δέντρου ξεχωριστά, και στο 3ο μέρος δίνονται πληροφορίες για την κάθε οικογένεια που δημιουργείται. Επίσης, παρατηρείται πως αριστερά σε όλο το αρχείο υπάρχει μια στήλη με τους αριθμούς 0,1 ή 2. Οι αριθμοί αυτοί υποδηλώνουν επίπεδα με πρώτο το 0. Για παράδειγμα στο παρακάτω απόσπασμα ενός gedcom αρχείου:

0 INDI (το επίπεδο 0 δηλώνει το άτομο)
1 BIRT (το επίπεδο 1 δηλώνει τη γέννηση του ατόμου του επιπέδου 0)
2 DATE 25 MAY 1949 (το επίπεδο 2 δηλώνει την ακριβή ημερομηνία γέννησης)
1 DEAT (εδώ δηλώνεται ο θάνατος πάλι ως επίπεδο 1 αφού αφορά το άτομο του επιπέδου 0)
2 DATE 2019 (στο επίπεδο 2 δηλώνεται όπως και πριν η ημερομηνία που αφορά το επίπεδο 1)

Στο αρχείο gedcom τα παραπάνω γράφονται στοιχισμένα το ένα κάτω από το άλλο. Εδώ γράφηκαν έτσι για ευκολία στην επεξήγηση.

Επεξήγηση του πιο πάνω παραδείγματος:

1ο μέρος:

Level	Tag	Value	Επεξήγηση
0	HEAD		Επικεφαλίδα
1	CHAR	UTF-8	Κωδικοποίηση χαρακτήρων που χρησιμοποιείται: UTF-8
1	SOUR	Ancestry.com Family Trees	Πηγή: Ancestry.com
2	VERS	(2010.3)	Έκδοση του Ancestry.com
2	NAME	Ancestry.com Family Trees	Όνομα του Ancestry.com
1	GEDC		Gedcom αρχείο
2	VERS	5.5	Έκδοση του GEDCOM
2	FORM	LINEAGE-LINKED	Δεδομένο όνομα

Πίνακας 2: Εισαγωγή GEDCOM αρχείου

2ο μέρος: Πάρθηκε το κομμάτι που αφορά το άτομο P3

Επίπεδο	Ετικέτα	Τιμή	Επεξήγηση
0	INDI	@P3@	Μοναδικός αριθμός/Ταυτότητα ατόμου = P5
1	BIRT		Δήλωση Γέννησης
2	DATE	1961	Τιμή ημερομηνίας γέννησης: 1961
2	PLAC	Denmark	Τοποθεσία γέννησης: Denmark
1	SEX	M	Φύλο: M (Male=Αρσενικό)
1	NAME	Peter	Όνομα: Peter
1	FAMC	@F2@	Ο P5 είναι παιδί της οικογένειας F2
1	FAMS	@F1@	Ο P5 είναι ο σύζυγος (αφού είναι αρσενικό) της οικογένειας F1

Πίνακας 3: Στοιχεία κάθε ατόμου ξεχωριστά

Θα μπορούσε αντίστοιχα με το BIRT να υπάρχει και η ημερομηνία και τοποθεσία θανάτου όπως πιο κάτω, εάν το άτομο δεν βρίσκεται εν ζωή:

1	DEAT		Δήλωση θανατου
2	DATE	1996	Ημερομηνία θανάτου: 1996

2	PLAC	Denmark	Τοποθεσία θανάτου: Denmark

<u>3ο μερος</u>: Πάρθηκε το κομμάτι που αφορά την οικογένεια F1

Επίπεδο	Ετικέτα	Τιμή	Επεξήγηση
0	FAM	@F1@	Θα δοθούν στοιχεία της οικογένειας F1
1	HUSB	@P3@	Ο σύζύγος / Πατέρας στην οικογένεια: ο P3
1	WIFE	@P2@	Η σύζυγος / Μητέρα στην οικογένεια: η P2
1	CHIL	@P6@	Παιδί στην οικογένεια: P6
1	CHIL	@P1@	Παιδί στην οικογένεια: P1

Πίνακας 4: Στοιχεία οικογενειών ξεχωριστά

3 Προστασία Δεδομένων

3.1 GDPR

GDPR είναι τα αρχικά του *General Data Protection Regulation*, στα ελληνικά *Γενικός Κανονισμός Προστασίας Δεδομένων*. Είναι νόμος που ισχύει από την 25 Μαΐου 2018 για όλους τους πολίτες της Ευρωπαϊκής Ένωσης και για όλες τις ιδιωτικές επιχειρήσεις και ιδιωτικούς φορείς που εδρεύουν στην Ευρωπαϊκή Ένωση. Ισχύει επίσης για επιχειρήσεις και φορείς που δεν βρίσκονται στο έδαφος της Ευρωπαϊκής Ένωσης αλλά δραστηριοποιούνται σε αυτό, φυσικά ή ψηφιακά. Ο GDPR ορίζει ότι η ανταλλαγή ευαίσθητων προσωπικών πληροφοριών με επιπλέον παροχές, λειτουργίες ή υπηρεσίες "premium" απαγορεύεται επειδή σε τέτοιες περιπτώσεις υποκλέπτεται η συγκατάθεσή σας με πιθανό δόλο. Αν τα προσωπικά δεδομένα αφορούν σε ανήλικο κάτω των 16 ετών, πρέπει να υπάρχει ρητή συναίνεση του γονέα. Επίσης όλοι οι πολίτες της Ευρωπαϊκής Ένωσης έχουν δικαίωμα στην πρόσβαση, διόρθωση ή διαγραφή των προσωπικών δεδομένων τους που βρίσκονται στα χέρια τρίτων. Αν κάποιος θεωρείται ευρωπαίος πολίτης ή ζεις μόνιμα στην Ε.Ε., μπορεί ανά πάσα στιγμή να ζητήσε πλήρες αντίγραφο όλων των πληροφοριών που τον αφορούν και μπορεί να βρίσκονται στα αρχεία κάποιας ευρωπαϊκής ιδιωτικής επιχείρησης ή κάποιου άλλου που βρίσκεται εκτός ΕΕ. Σύμφωνα με τον GDPR, είναι υποχρεωμένοι παρέχουν τον ευρωπαίο πολίτη τις πληροφορίες του σε διάστημα ενός μηνός. Εάν κάποια επιχείρηση αρνηθεί τα παραπάνω, οφείλει να έχει άκρως σημαντικούς λόγους αφού θα έχει αντιμέτωπη του την δικαιοσύνη. Όλοι όσοι έχουν στην κατοχή τους προσωπικά δεδομένα άλλων, δηλαδή ευαίσθητες πληροφορίες που τους αφορούν, είναι υποχρεωμένοι κατά τον GDPR να φροντίζουν συνεχώς για την ασφάλεια των δεδομένων αυτών ώστε να είναι αδύνατη η κλοπή τους. Τέλος, παράβαση του GDPR από μια επιχείρηση για παράδειγμα, επιβάλλει πρόστιμο που μπορεί να φτάσει ακόμα και τα €20 εκατομμύρια.

3.2 Ανωνυμοποίηση

Η ανωνυμοποίηση είναι μια τεχνική επεξεργασίας δεδομένων, η οποία καταργεί ή τροποποιεί στοιχεία που μπορεί να οδηγήσουν σε προσωπική ταυτοποίηση. Τα δεδομένα αποκτούν ανώνυμη μορφή και ιδανικά δεν μπορούν να συσχετιστούν με οποιοδήποτε άτομο.
Η ανωνυμοποίηση δεδομένων είναι ένας τρόπος τροποποίησης δεδομένων, του οποίου η πρόθεση είναι η προστασία ευαίσθητων πληροφοριών. Είναι η διαδικασία κρυπτογράφησης ή αφαίρεσης προσωπικών στοιχείων από σύνολα δεδομένων ή χρήση διαφόρων άλλων τεχνικών, έτσι ώστε οι άνθρωποι που περιγράφονται με τα δεδομένα να παραμένουν ανώνυμοι.

3.3 Αποανωνυμοποίηση

Η αποανωνυμοποίηση είναι μια τεχνική ανάκτησης ανωνυμοποιημένων δεδομένων, δηλαδή συσχέτισης των δεδομένων με συγκεκριμένα πρόσωπα. Η αποανωνυμοποίηση χρησιμοποιεί τεχνικές και μέσα, με τα οποία κανείς «επιτίθεται» στις ανώνυμες πληροφορίες συνδυάζοντας πιθανώς άλλα διαθέσιμα δεδομένα.

Ένα παράδειγμα στο σήμερα, που δείχνει πόσο σημαντική είναι η αποανωνυμοποίηση για εταιρείες-μεγαθήρια, είναι το Βραβείο Netflix. Έργο της εταιρείας Netflix, γνωστής πλεόν ανά το παγκόσμιο, είναι διαδικτυακή ενοικίαση ταινιών. Πριν από μερικά χρόνια, το 2006 συγκεκριμένα , η εταιρεία αυτή αποφάσισε να δημιουργήσει ένα ανοιχτό διαγωνισμό σκοπός του οποίου είναι η ανάδειξη του καλύτερου συνεργατικού αλγορίθμου φιλτραρίσματος της πρόβλεψης των αξιολογήσεων χρηστών για ταινίες. Ως δεδομένα, δίνονται μόνο οι προηγούμενες αξιολογήσεις χρηστών και καμία άλλα προσωπική-ευαίσθητη πληροφορία για τους χρήστες. Στόχος της εταιρείας, ήταν να βελτιώσει τον αλγόριθμο της, σε σημείο που να είναι σχεδόν αδύνατο να «σπάσει». Με την προβολή του διαγωνισμού ανά το παγκόσμιο και με το χρηματικό ποσό να ανέρχεται στο ένα εκατομμύριο, πολλοί από τους συμμετέχοντες δήλωσαν ότι σκοπός τους ήταν μόνο το κέρδος και όχι να καταστρέψουν τη συγκεκριμένη εταιρεία.

Όλα τα παραπάνω λοιπόν, δείχνουν πόσο σημαντική είναι η ανωνυμοποίηση αλλά και η αποανωνυμοποίηση, όχι μόνο σε δημόσιες υπηρεσίες που συναντά ο καθένας καθημερινά, αλλά και σε μεγαλοεταιρείες που κρίνεται τόσο το όνομα τους αλλά και τα ακατανόμαστου βαθμού κέρδη τους.

4 Γενεαλογικό δέντρο και Gedcom

4.1 Δημιουργία Γενεαλογικού δέντρου

«Τα οικογενειακά δέντρα έχουν τεράστιες εφαρμογές σε ποικίλους τομείς όπως η γενετική, η ανθρωπολογία και τα οικονομικά. Ωστόσο, η συλλογή εκτεταμένων οικογενειακών δέντρων είναι κουραστική και συνήθως βασίζεται σε πόρους με περιορισμένη γεωγραφική εμβέλεια και σύνθετους περιορισμούς χρήσης δεδομένων. Συλλέξαμε 86 εκατομμύρια προφίλ από διαθέσιμα στο κοινό ηλεκτρονικά δεδομένα που μοιράζονται οι λάτρεις της γενεαλογίας.»

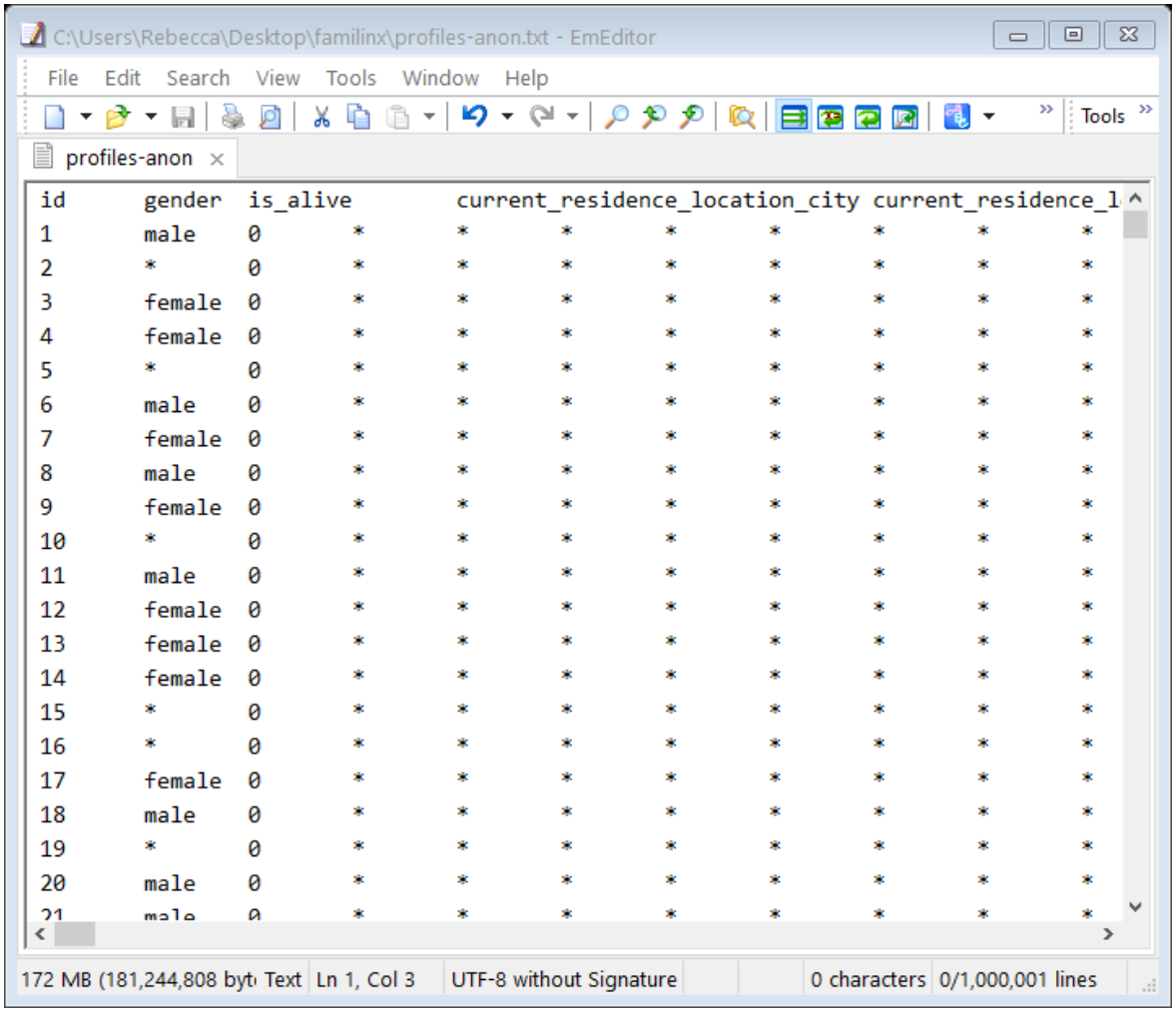
Την βάση για το γενεαλογικό δέντρο της συγκεκριμένης διπλωματικής έθεσε το αποτέλεσμα της παραπάνω έρευνας. Οι συγκεκριμένοι ερευνητές, επικέντρωσαν έναν γενεαλογικό ιστότοπο στον ιστότοπο Geni.com, στον οποίο ο κάθε χρήστης μπορεί να δημιουργεί προφίλ και να ανεβάζει γενεαλογικά δέντρα. Έπειτα ο ιστότοπος σαρώνει αυτόματα αυτά τα προφίλ, εντοπίζει ομοιότητες και συγχωνεύει όπου θεωρείται απαραίτητο. Έτσι, δημιουργούνται μεγαλύτερα γενεαλογικά δέντρα και συγκεντρώθηκαν 86 εκατομμύρια άτομα στο σύνολο, τα οποία οι παραπάνω ερευνητές αφού «έκοψαν, καθάρισαν και επικύρωσαν», στη συνέχεια τα μετέτρεψαν σε ψηφιακά δεδομένα τα οποία είναι πλέον διαθέσιμα σε όλο το διαδικτυακό κοινό μέσω της ιστοσελίδας http://familinx.org/download.html .

4.1.1 Αρχικά Δεδομένα

Τα αρχικά δεδομένα τα οποία θα επεξεργαστούν-κοπούν ώστε να επέλθει το επιθυμητο αποτέλεσμα δεδομένων, έχουν ληφθεί από τον ιστότοπο familinx. Ο ιστότοπος αυτός είναι ανοιχτός σε όλους και βάζοντας μια ηλεκτρονική διεύθυνση, δίνεται η δυνατότητα για δωρεάν λήψη ενός συμπιεσμένου αρχείου το οποίο περιέχει τα 3 παρακάτω text files που συμπεριλαμβάνουν όλα όσα αναφέρθηκαν στην αρχή της ενότητας.

- **«profiles-anon.txt»**

  Περιέχει τα 86.000.000 άτομα που αναφέρθηκαν στην αρχή της ενότητας καθώς και 75 ιδιότητες/πεδία για το καθένα από αυτά. Τα άτομα προσδιορίζονται μοναδικά από ένα id, καθώς όπως είναι λογικό, δεν υφίσταται το να γίνεται δημοσίως γνωστό το πλήρες όνομα τους.



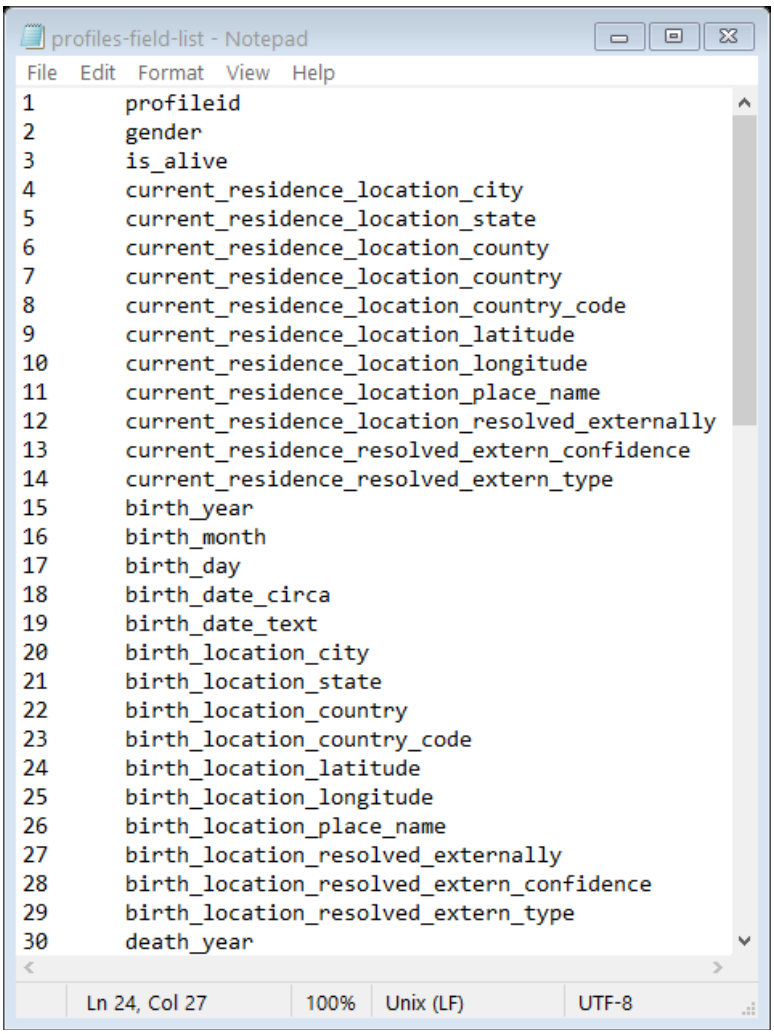
Εικόνα 1: Αρχείο «profiles-anon.txt»

- **«profiles-filed-list.txt»**

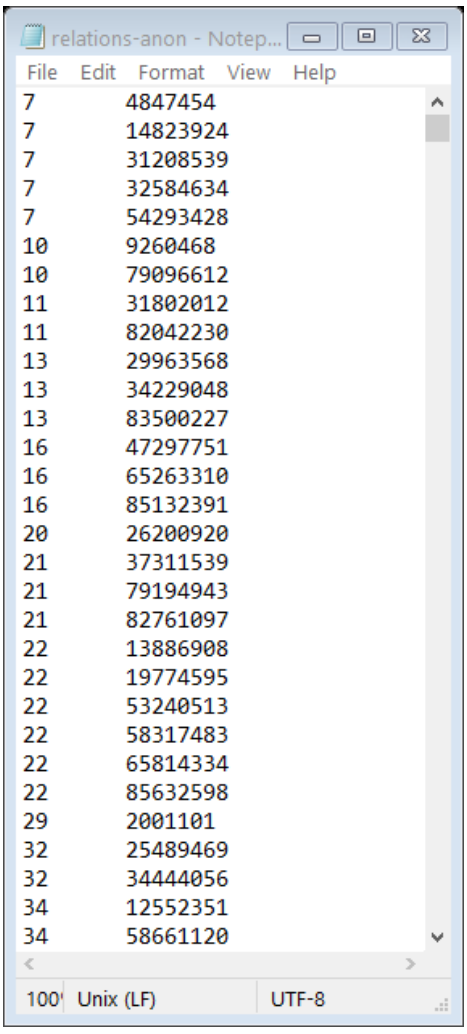
  Περιέχει τα 75 παραπάνω πεδία εξηγώντας τι εκπροσωπεί το καθένα από αυτά.

- **«relations-anon.txt»**

  Περιέχει τις σχέσεις γονέα-παιδιού. Αναλυτικά, το κάθε άτομο (όπως φαίνεται και στο «profiles-anon» έχει για 1ο πεδίο τον μοναδικό αριθμό id. Έτσι λοιπόν στο «relations-anon» έχουμε 2 στήλες, στην αριστερά βρίσκεται το id του γονέα και στην αντίστοιχη γραμμή, στην δεξιά στήλη υπάρχει το id του παιδιού του.



Εικόνα 2: Αρχείο «profiles-filed-list.txt»

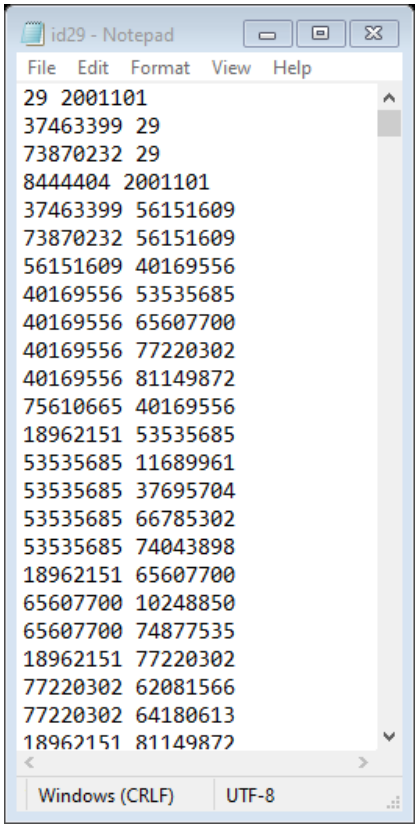


Εικόνα 3: Αρχείο «relations-anon.txt»

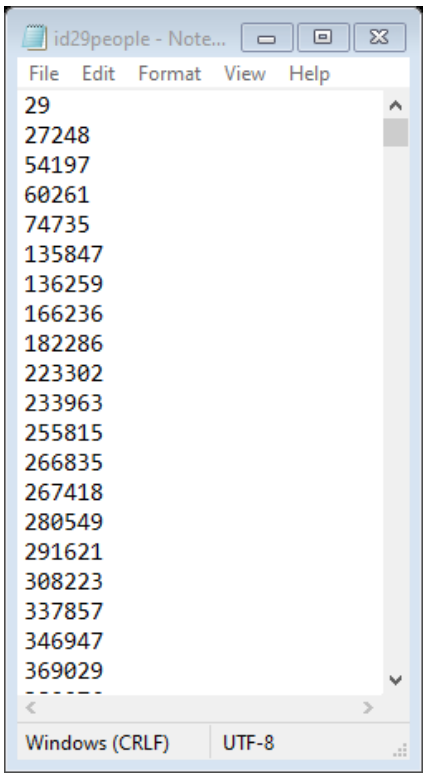
4.1.2 Δεδομένα Γενεαλογικού Δέντρου

Τα 86 εκατομμύρια προφίλ είναι αριθμός εξαιρετικά δύσκολος για μετέπειτα επεξεργασία και εξαγωγή αποτελεσμάτων. Γι' αυτό το λόγο, αρχικά, σκοπός της διπλωματικής είναι να πάρει ένα κομμάτι από τα παραπάνω άτομα το οποίο όμως ταυτόχρονα να αποτελεί γενεαλογικό δέντρο, δηλαδή τα άτομα αυτά να μην είναι «ασύνδετα» μεταξύ τους.
Ένα εύρος 3000-4000 ατόμων θεωρήθηκε αρκετό για να μπορέσει να προχωρήσει η διπλωματική με το βασικό της στόχο την ανωνυμοποίηση και αποανωνυμοποίηση.
Για την λήψη του παραπάνω κομματιού από τα αρχικά δεδομένα, εκτελέστηκε το **Πρόγραμμα 1, Findtree.py (Παράρτημα Ι)** , το οποίο διαβάζοντας το αρχείο relations-anon.txt, με είσοδο ενός id, βρίσκει-κτίζει τους συνδέσμους για το γενεαλογικό δέντρο γύρω από αυτό. Δηλαδή, ξεκινώντας από αυτό βρίσκει όλους τους απογόνους και προγόνους του. Οι τελικοί συνδέσμοι τυπώνονται στο αρχείο idX.txt (όπου X=id) , το οποίο αρχείο έχει την ίδια δομή με το relations-anon.txt.
Έπειτα, με εκτέλεση του **προγράμματος 2, Numberofpeople.py (Παράρτημα Ι)** , μετριέται μοναδικά το πλήθος των ατόμων που βρίσκονται στο αρχείο idX.txt και έπειτα τυπώνονται στο αρχείο idXpeople.txt.
Μετά από αρκετές δοκιμές για διαφορετικά id, καταληκτικό ήταν το id=29, για το οποίο το πρόγραμμα 2 είχε ως αποτέλεσμα 3619 διαφορετικά άτομα, κάτι που ήταν μέσα στο επιθυμητό εύρος της διπλωματικής.

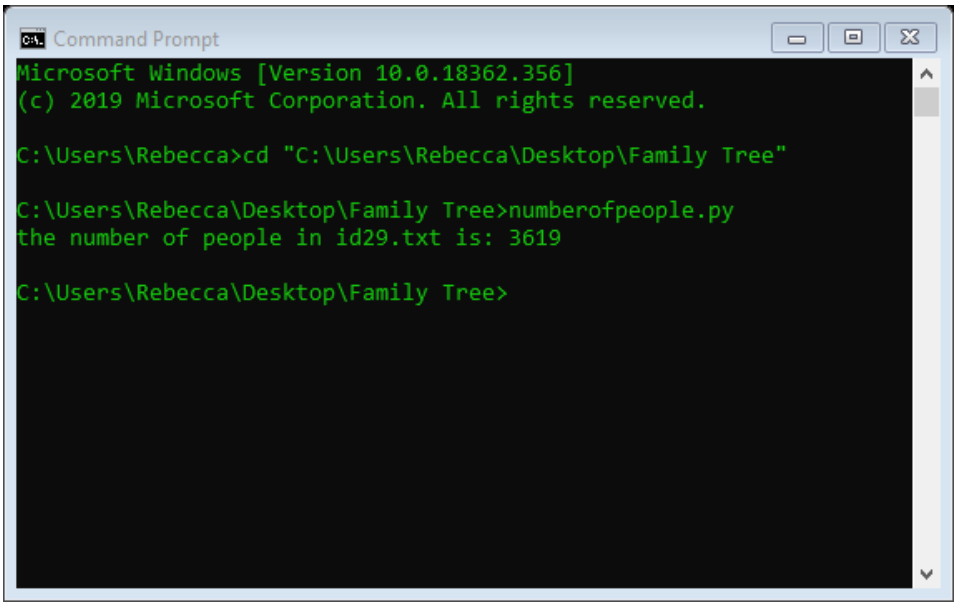
Αποτελέσματα εκτέλεσης Findtree.py και Numberofpeople.py:



Εικόνα 4: Αρχείο «id29.txt»



Εικόνα 5: Αρχείο «id29people.txt»



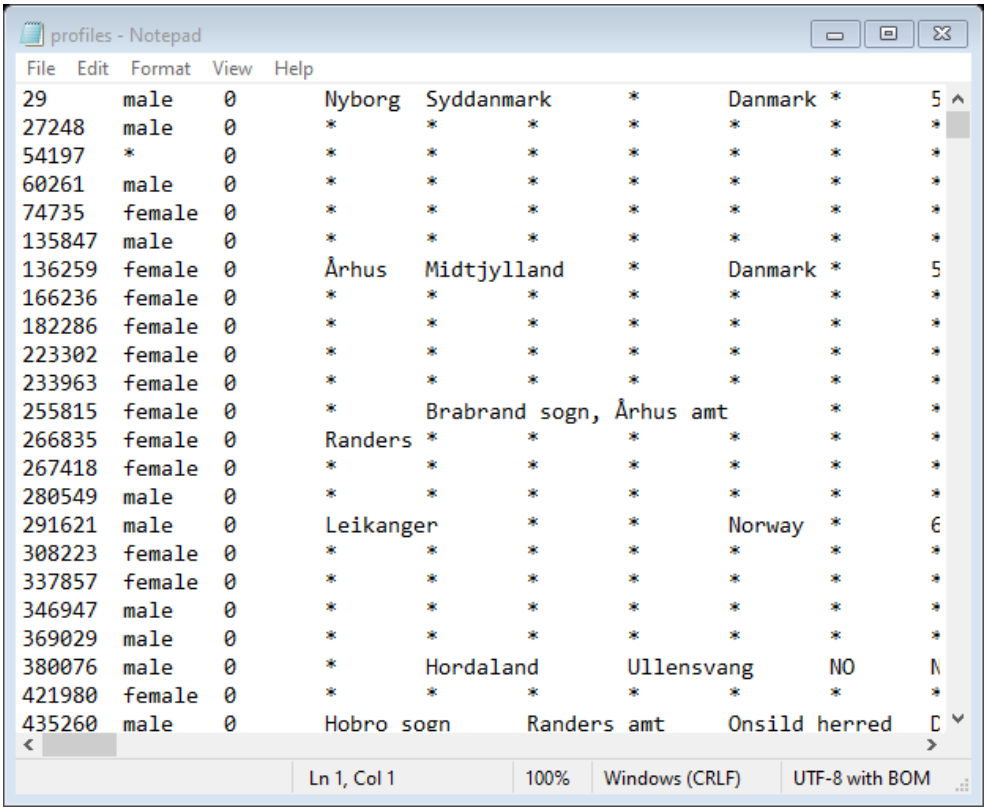
Εικόνα 6: Αποτέλεσμα εκτέλεσης Numberofpeople.py

4.1.3 Στοιχεία ατόμων του Γενεαλογικού Δέντρου

Όπως αναφέρθηκε στην αρχή της διπλωματικής, για τη δημιουργία ενός γενεαλογικού δέντρου, και την μετατροπή του σε σε αρχείο gedcom, χρειάζονται πληροφορίες για το κάθε άτομο ξεχωριστά οι οποίες θα αντιστοιχηστούν στις ανάλογες ετικέτες (tags).

Τα 3619 άτομα του δέντρου είναι γραμμένα στο αρχείο id29people.txt. Για την εύρεση των στοιχείων του κάθε ατόμου ξεχωριστά, εκτελέστηκε το **Πρόγραμμα 3, Findprofiles.py (Παράρτημα Ι) ,** το οποίο θα πρέπει να διαβάσει τα αρχεία id29people.txt και profiles-anon.txt και να κάνει τις ανάλογες αντιστοιχήσεις για να τυπωθούν σε ένα τελικό αρχείο profiles.txt τα 3619 άτομα μαζί με τα 75 πεδία-στοιχεία τους. Το αρχείο profiles-anon.txt όμως, καθώς είναι αυτό με τα 86 εκατομμύρια άτομα, χρειάστηκε να σπάσει σε 86 μικρότερα αρχεία για να είναι εφικτή η ανάγνωση του.

Αποτέλεσμα:



Εικόνα 7: Αποτέλεσμα εκτέλεσης Findprofiles.py – Αρχείο «profiles.txt»

4.2 Δημιουργία αρχείου Gedcom

4.2.1 Τιμές Ετικετών

Όπως αναφέρθηκε στην παράγραφο 2.1.3 , για τη δημιουργία ενός gedcom αρχείου χρειάζονται κάποιες ετικέτες. Οι τιμές των ετικετών που χρειάζονται για το συγκεκριμένο γενεαλογικό δέντρο εξηγούνται παρακάτω:

- **HEAD**
  Ένα αρχείο gedcom ξεκινά με επικεφαλίδα τη συγκεκριμένη ετικέτα. Είναι δεδομένο δηλαδή από το Gedcom Standard.

- **GEDC**
  Όπως πιο πάνω, έτσι και εδώ η ετικέτα είναι δεδομένο από το Gedcom Standard.

- **VERS**
  Η έκδοση Gedcom που χρησιμοποιήθηκε είναι η 5.5.1.

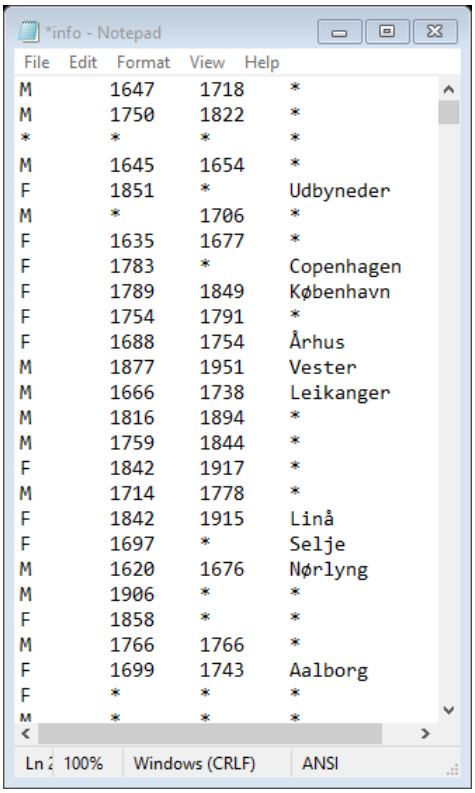
- **FORM**
  Δεδομένο από το Gedcom Standard.

- **CHAR**
  Η κωδικοποίηση χαρακτήρων που χρησιμοποιήθηκε είναι η: UTF-8.

- **INDI**
  Από την εκτέλεση του προγράμματος Numberofpeople.py βρέθηκε το πλήθος των ατόμων του γενεαλογικού δέντρου, που είναι ίσο με 3619. Τα id τους βρίσκονται ταξινομημένα στο αρχείο id29people.txt. Ξεκινώντας από τον πρώτο, η αντιστοίχιση (με τη χρήση της excel για ευκολία) έγινε ως εξής:
  I0 - 29
  I1 - 27248
  I2 - 54197
  I3 - 60261
  …

- **NAME**
  Ως όνομα του κάθε ατόμου, θεωρήθηκε ο μοναδικός αριθμός του id αφού δεν υπάρχουν ονοματεπώνυμα στα δεδομένα.

Για την εύρεση τιμών για τις ετικέτες SEX, BIRT, DEAT, DATE, PLAC, εκτελέστηκε το **πρόγραμμα 4, Tags.py, (Παράρτημα Ι)**. Τα αποτελέσματα του προγράμματος φαίνονται στην παρακάτω εικόνα:



Εικόνα 8: Αποτέλεσμα εκτέλεσης Tags.py – Αρχείο «info.txt»

- **SEX**
  1η στήλη του αρχείου info.txt

- **BIRT**
  Δήλωση γέννησης, η χρονολογία και τοποθεσία της οποίας θα φανούν στις επόμενες 2 ετικέτες.

- **DATE**
  2η στήλη του αρχείου info.txt

- **PLAC**
  4η στήλη του αρχείου info.txt

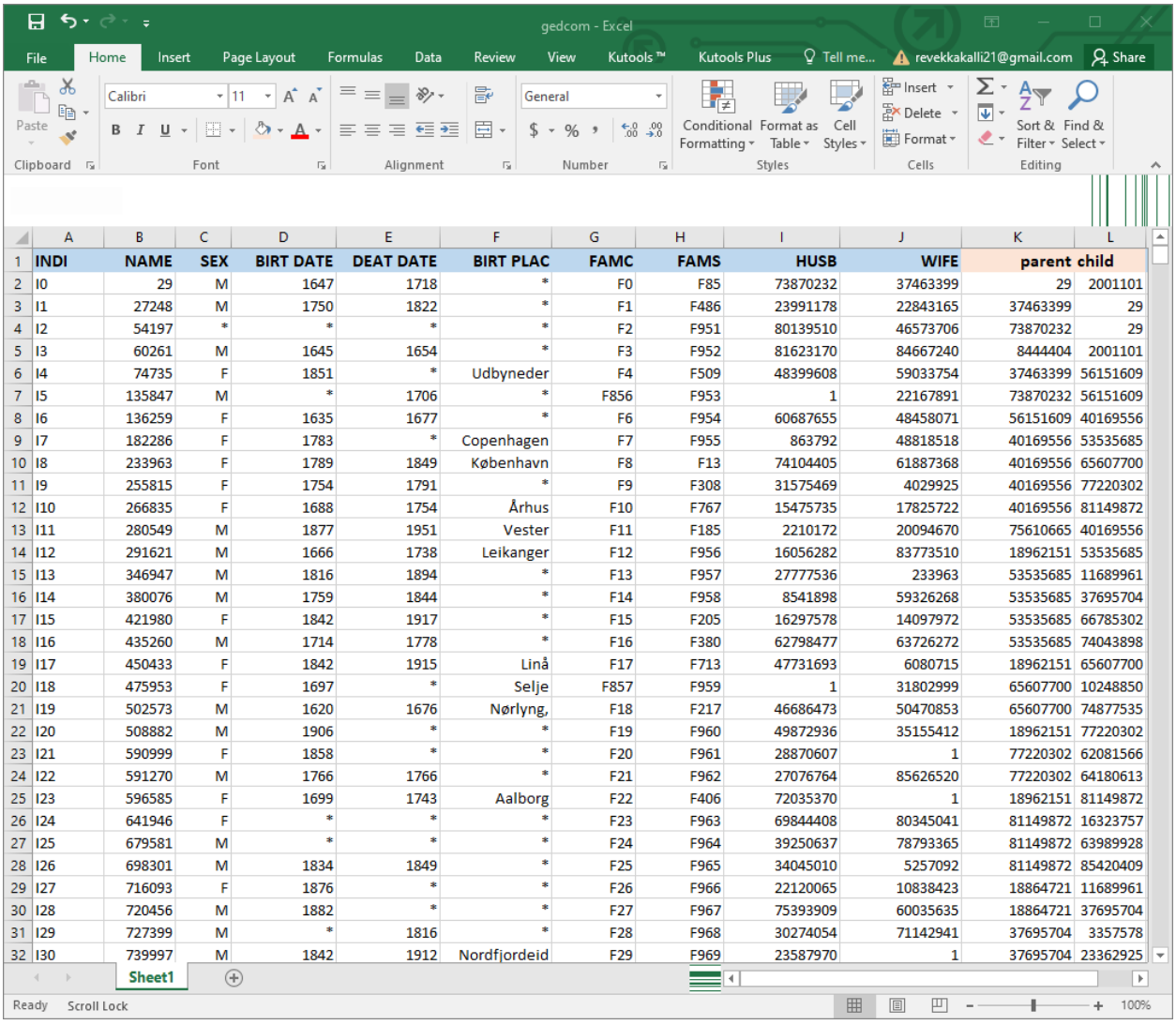
- **DEAT**
  Δήλωση θανάτου, η χρονολογία του οποίου θα φανεί στην επόμενη ετικέτα.

- **DATE**
  3η στήλη του αρχείου info.txt

  Σε όλες τις παραπάνω τιμές, επισημαίνεται ότι όπου υπάρχει η τιμή « * », σημαίνει ότι δεν υπάρχει τιμή για την συγκεκριμένη ετικέτα.

- **FAMC**
  Από τις στήλες NAME, SEX με απλά φίλτρα της excel δημιουργήθηκαν οι στήλες HUSB (sex = M), WIFE(sex = F). Έπειτα, με τη βοήθεια του αρχείου id29.txt και συγκεκριμένα της αριστερής στήλης (parent), διαγράφηκαν από τις στήλες HUSB, WIFE όσοι δεν ανήκουν στην στήλη parent. Έτσι, πλέον για κάθε NAME[i] οι γονείς του είναι οι HUSB[i] (πατέρας) και WIFE[i] (μητέρα) για iєR. Επίσης όπου υπάρχει ο αριθμός 1 είναι εκεί που υπάρχει κάποια τιμή για το αντίστοιχο δεδομένο.
  Έπειτα, δημιουργείται στο παραπάνω αρχείο μια στήλη FAMC (στήλη G).
  Για την αντιστοίχιση του FAMC ακολουθήθηκε η εξής διαδικασία:
  Ξεκινώντας από την αρχή θεωρείται ότι ο 29 έχει τιμή FAMC=F0 δηλαδή ανήκει στην οικογένεια F0 ως παιδί. Προχωρώντας κάθε φορά στο επόμενο άτομο, ελέγχεται εάν έχει ίδιους γονείς (στήλες HUSB-WIFE), με κάποιον από τους πάνω στους οποίους ήδη δόθηκε τιμή FAMC. Εάν ισχύει, τότε παίρνουν την αντίστοιχη τιμή, αλλιώς αυξάνεται η τιμή του FAMC κατά 1. Δηλαδή ο 27248, με διαφορετικούς από τον 29 γονείς, θα πάρει την τιμή FAMC=F1
  Πλέον είναι εμφανές ότι ξεκινώντας από την αρχή, ο 29 είναι παιδί του 73870232 και της 37463399, ο 27248 είναι παιδί του 23991178 και της 22843165 κ.ο.κ.
  Έστω λοιπόν ότι η οικογένεια F0 είναι αυτή με πατέρα τον 73870232 μητέρα την 37463399 και παιδί τον 29 , η οικογένεια F1 είναι αυτή με πατέρα τον 23991178 μητέρα την 22843165 και παιδί τον 27248. Με απλή python, και ξεκινώντας από την αρχή, τοποθετήθηκαν σε 3 λίστες father,mother,famc οι τιμές των πατέρας (εδώ 73870232) ,μητέρα (εδώ 37463399), αριθμός οικογένειας (εδώ F0). Για κάθε επόμενη γραμμή που ελέγχεται εάν ο πατέρας και η αντίστοιχη μητέρα υπάρχουν ήδη στις λίστες father,mother. Εάν υπάρχουν, τότε θα δίνεται ο υπάρχων αριθμός αλλιώς η τιμή του FAMC θα αυξάνεται κατά ένα (F1,F2,F3…).

- **FAMS**
  Έχοντας τις τιμές για NAME, FAMC πάμε στις τιμές HUSB, WIFE και όπου NAME = HUSB ή NAME = WIFE αναγράφεται δίπλα από το NAME η τιμή του FAMC του HUSB/WIFE ανάλογα.
  Π.χ: Εάν η τιμή NAME[2] = HUSB[98] τότε FAMS[2]=FAMC[90].



	A	B	C	D	E	F	G	H	I	J	K	L
1	INDI	NAME	SEX	BIRT DATE	DEAT DATE	BIRT PLAC	FAMC	FAMS	HUSB	WIFE	parent	child
2	I0	29	M	1647	1718	*	F0	F85	73870232	37463399	29	2001101
3	I1	27248	M	1750	1822	*	F1	F486	23991178	22843165	37463399	29
4	I2	54197	*	*	*	*	F2	F951	80139510	46573706	73870232	29
5	I3	60261	M	1645	1654	*	F3	F952	81623170	84667240	8444404	2001101
6	I4	74735	F	1851	*	Udbyneder	F4	F509	48399608	59033754	37463399	56151609
7	I5	135847	M	*	1706	*	F856	F953	1	22167891	73870232	56151609
8	I6	136259	F	1635	1677	*	F6	F954	60687655	48458071	56151609	40169556
9	I7	182286	F	1783	*	Copenhagen	F7	F955	863792	48818518	40169556	53535685
10	I8	233963	F	1789	1849	København	F8	F13	74104405	61887368	40169556	65607700
11	I9	255815	F	1754	1791	*	F9	F308	31575469	4029925	40169556	77220302
12	I10	266835	F	1688	1754	Århus	F10	F767	15475735	17825722	40169556	81149872
13	I11	280549	M	1877	1951	Vester	F11	F185	2210172	20094670	75610665	40169556
14	I12	291621	M	1666	1738	Leikanger	F12	F956	16056282	83773510	18962151	53535685
15	I13	346947	M	1816	1894	*	F13	F957	27777536	233963	53535685	11689961
16	I14	380076	M	1759	1844	*	F14	F958	8541898	59326268	53535685	37695704
17	I15	421980	F	1842	1917	*	F15	F205	16297578	14097972	53535685	66785302
18	I16	435260	M	1714	1778	*	F16	F380	62798477	63726272	53535685	74043898
19	I17	450433	F	1842	1915	Linå	F17	F713	47731693	6080715	18962151	65607700
20	I18	475953	F	1697	*	Selje	F857	F959	1	31802999	65607700	10248850
21	I19	502573	M	1620	1676	Nørlyng,	F18	F217	46686473	50470853	65607700	74877535
22	I20	508882	M	1906	*	*	F19	F960	49872936	35155412	18962151	77220302
23	I21	590999	F	1858	*	*	F20	F961	28870607	1	77220302	62081566
24	I22	591270	M	1766	1766	*	F21	F962	27076764	85626520	77220302	64180613
25	I23	596585	F	1699	1743	Aalborg	F22	F406	72035370	1	18962151	81149872
26	I24	641946	F	*	*	*	F23	F963	69844408	80345041	81149872	16323757
27	I25	679581	M	*	*	*	F24	F964	39250637	78793365	81149872	63989928
28	I26	698301	M	1834	1849	*	F25	F965	34045010	5257092	81149872	85420409
29	I27	716093	F	1876	*	*	F26	F966	22120065	10838423	18864721	11689961
30	I28	720456	M	1882	*	*	F27	F967	75393909	60035635	18864721	37695704
31	I29	727399	M	*	1816	*	F28	F968	30274054	71142941	37695704	3357578
32	I30	739997	M	1842	1912	Nordfjordeid	F29	F969	23587970	1	37695704	23362925

Εικόνα 9: Στιγμιότυπο από το βοηθητικό αρχείο της excel για δημιουργία του Gedcom file

Από τα παραπάνω δεδομένα υπάρχουν πλέον όλες οι οικογένειες συμπληρωμένες και μπορεί να δημιουργηθεί και το «2ο κομμάτι» του gedcom αρχείου με τα στοιχεία της κάθε οικογένειας ξεχωριστά όπως φαίνεται παρακάτω.

- **FAM**
  Τιμές της στήλης G.

- **HUSB**
  Στήλη I αντίστοιχη σε αριθμό κελιού με την παραπάνω στήλη G.

- **WIFE**
  Στήλη J αντίστοιχη σε αριθμό κελιού με τις δύο παραπάνω στήλες I, G.

- **CHIL**
  Στήλη A (όσα id από την στήλη A έχουν για famc την τρέχουσα τιμή της ετικέτας FAM μπαίνουν στην ίδια οικογένεια)

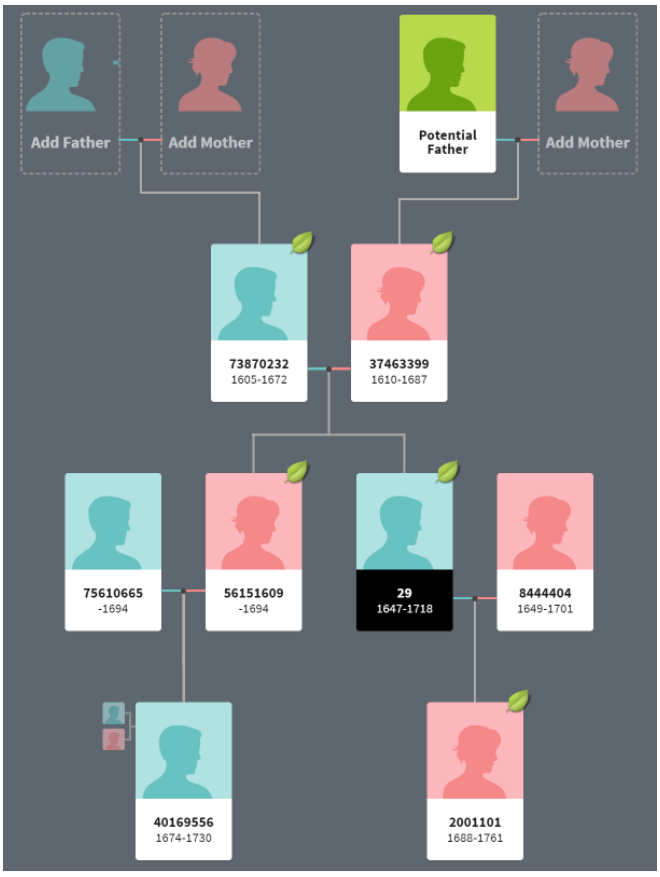
- **TRLR**
  Δεδομένο από το Gedcom Standard.

4.2.2 Τελική μορφή Gedcom αρχείου

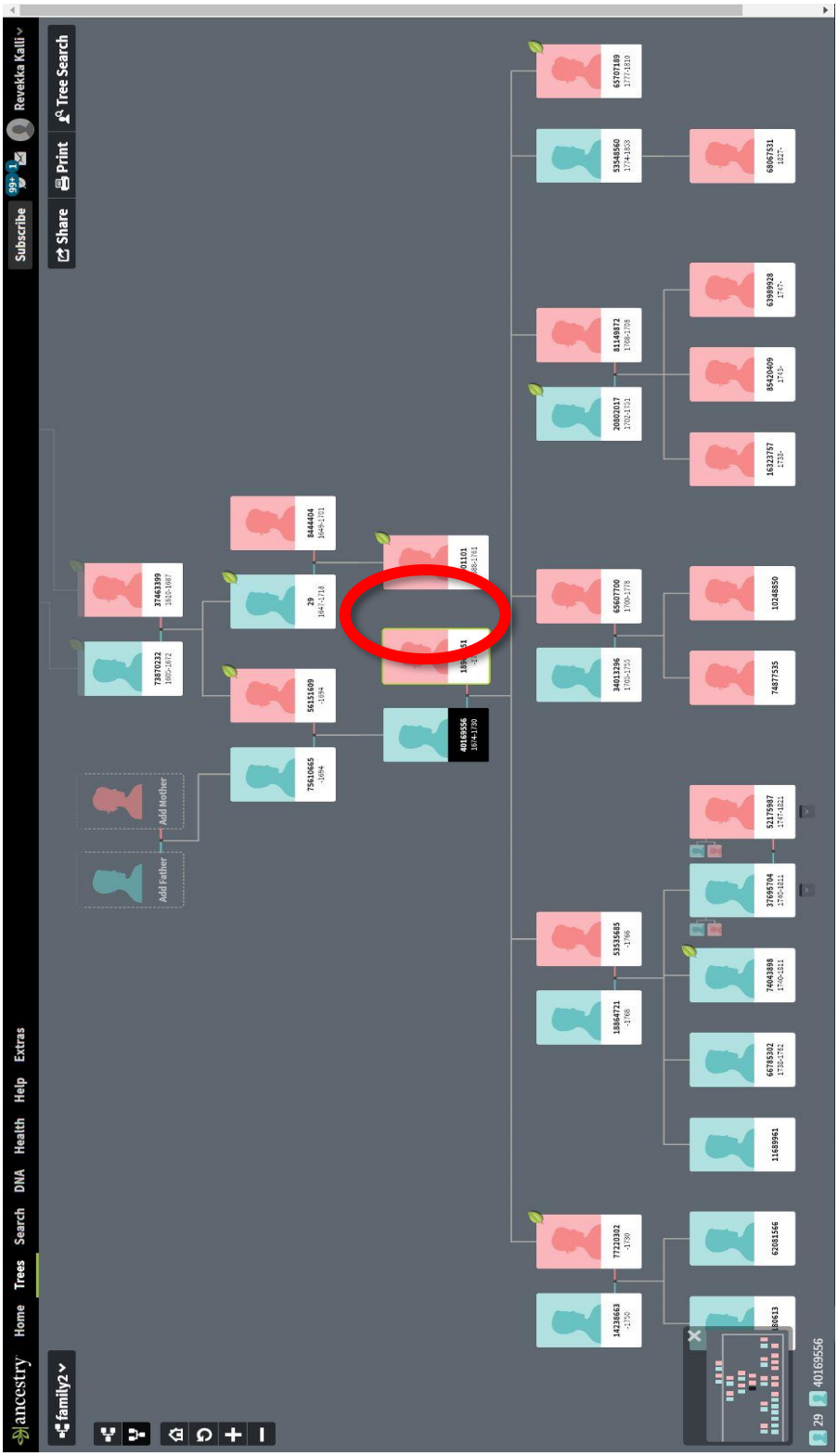
Τα δεδομένα που χρειάζεται ένα γενεαλογικό δέντρο, έχουν πλέον συμπληρωθεί. Όμως για την δημιουργία ενός αρχείου που είναι gedcom compliant (.ged file) δηλαδή ενός αρχείου που να μπορεί να παρασταθεί ως γενεαλογικό δέντρο με οποιοδήποτε λογισμικό που δέχεται .ged files, χρειάζεται να γραφούν τα παραπάνω δεδομένα με ένα συγκεκριμένο τρόπο γραφής.

Για να δημιουργηθεί λοιπόν το τελικό αρχείο με όνομα Family.ged εκτελέστηκε με είσοδο όλα τα παραπάνω δεδομένα, το **Πρόγραμμα 5, CreateGed.py (Παράρτημα Ι)** το οποίο τα τύπωσε στο τελικό αρχείο με τον επίσημο τρόπο γραφής ενός .ged αρχείου, που περιγράφεται στο Gedcom Standard.

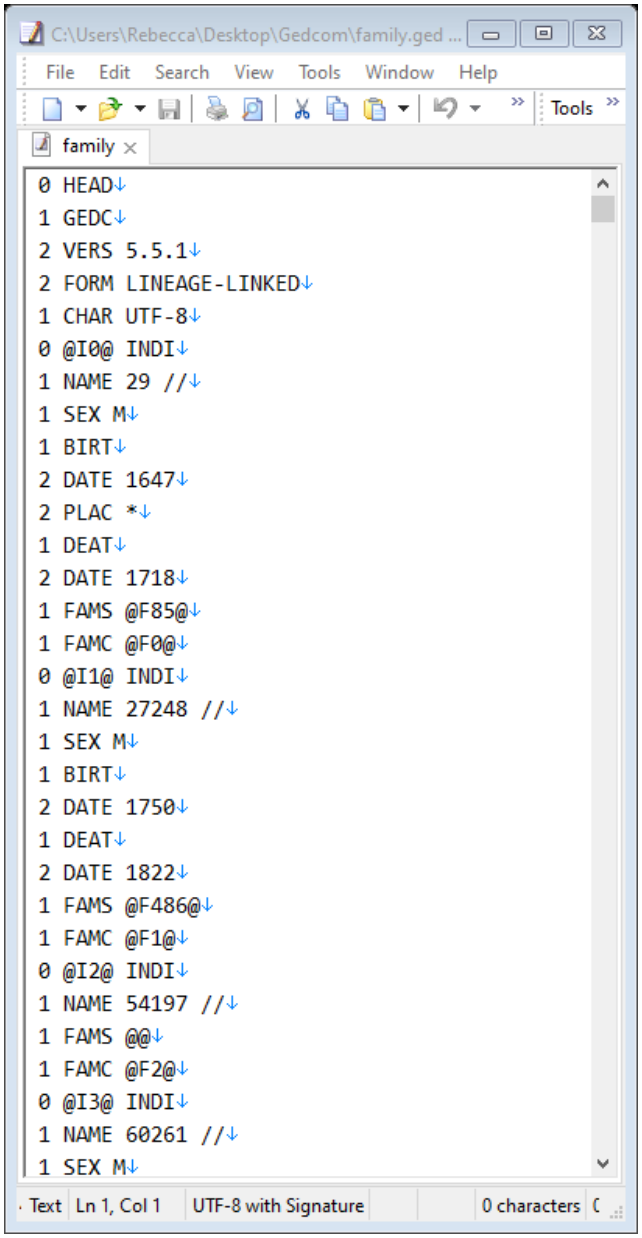
Αποτελέσματα:



Εικόνα 10: Στιγμιότυπο του γενεαλογικού δέντρου που δημιουργήθηκε με επιλεγμένο το αρχικό άτομο «29»



Εικόνα 11: Στιγμιότυπο μεγαλύτερου εύρους από το τελικό γενεαλογικό δέντρο με χρήση της ιστοσελίδας ancestry.com με το αρχικό άτομο «29» να βρίσκεται στον κόκκινο κύκλο



Εικόνα 12: Αρχείο Family.ged

5 Ανωνυμοποίηση Δεδομένων

5.1 Τεχνικές ανωνυμοποίησης σε δεδομένα

Υπάρχουν διάφορες τεχνικές ανωνυμοποίησης που εφαρμόζονται σε δεδομένα, κάποιες από τις οποίες θα αναφερθούν αλλά και μελετηθούν πιο κάτω.

5.1.1 Αφαίρεση

Αυτή η διαδικασία περιλαμβάνει την αφαίρεση ολόκληρων πεδίων δεδομένων για να μειωθεί ο κίνδυνος σύνδεσης με οποιαδήποτε πηγή. Δεν χρησιμοποιείται ευρέως σε επιχειρήσεις, αλλά μερικές έχουν τη δυνατότητα και την ανάγκη να απομακρύνουν ολόκληρα κομμάτια των δεδομένων από τα σύνολα δεδομένων τους και να τα διαγράψουν εξ ολοκλήρου είτε για να δημιουργήσουν πιο ασφαλείς συλλογές πληροφοριών είτε για να δώσουν ανωνυμοποιημένες συλλογές σε εξωτερικούς χρήστες, π.χ. στατιστικούς, ερευνητές κ.α.

Μια ισοδύναμη τεχνική θα ήταν η αντικατάσταση των ονομάτων με τυχαίους αριθμούς ή τυχαία ψευδώνυμα ή ακόμη με χρήση μιας hash function. Το χαρακτηριστικό αυτής της τεχνικής είναι ότι δεν επιτρέπει (στο βαθμό που τα νέα ονόματα είναι τυχαία) την αντιστροφή, δηλαδή τον προσδιορισμό του αρχικού ονόματος από το ψευδώνυμο.

Στην περίπτωση της απουσίας ενός πεδίου με ψευδώνυμο ή κωδικό, ο επιτιθέμενος πιθανότατα θα προσθέσει μια δική του κωδικοποίηση, προκειμένου να δημιουργήσει στο τέλος ένα πίνακα που θα αντιστοιχίζει κωδικούς με πιθανά πραγματικά ονόματα.

Χρήση της σε αυτή τη διπλωματική:

Οι πληροφορίες που υπάρχουν για το κάθε άτομο του γενεαλογικού δέντρου στο οποίο έχουμε καταλήξει από το αρχικό σύνολο είναι ο μοναδικός αριθμός (id), το φύλο, η ημερομηνία και τοποθεσία γέννησης και η ημερομηνία θανάτου (εάν υπάρχει). Ο μοναδικός αριθμός προέρχεται ούτως ή άλλως από ανωνυμοποίηση της αρχικής συλλογής γενεαλογικών δεδομένων. Κατά πόσο μπορεί να συνδεθεί με πραγματικά πρόσωπα είναι ένα ζήτημα που δεν μας απασχολεί σ' αυτήν την εργασία.

Μας ενδιαφέρει όμως να βρούμε κατά πόσο είναι δυνατή η αποανωνυμοποίηση σε ένα γράφο με τη δεδομένη δομή. Υποθέτουμε επίσης ότι στον ανωνυμοποιημένο γράφο περιλαμβάνεται κάποιο χαρακτηριστικό που ενδιαφέρει τον επιτιθέμενο, αλλά δεν μπορεί να τον βοηθήσει στην ταυτοποίηση (δεν είναι *quasi-identifier*). Για παράδειγμα, τέτοιο χαρακτηριστικό μπορεί να είναι αν ένα άτομο έχει παρουσιάσει μια συγκεκριμένη ασθένεια. Ο σκοπός επομένως του επιτιθέμενου είναι να ταυτοποιήσει τα άτομα, ώστε να μάθει ποιο τελικά πραγματικό πρόσωπο παρουσιάζει το συγκεκριμένο χαρακτηριστικό.

Με ποια γνώση θα προσπαθήσει την αποανωνυμοποίηση ο επιτιθέμενος; Θεωρούμε ότι οι αριθμοί ανήκουν σε πραγματικά άτομα που με τα ονόματά τους είναι γνωστά στον επιτιθέμενο και ότι αυτό συμβαίνει είτε για όλα τα άτομα είτε για μερικά. Π.χ. μπορεί να είναι άτομα που κατοικούν στην ίδια κωμόπολη και ο επιτιθέμενος κατάγεται από την ίδια κωμόπολη. Κατά συνέπεια μπορεί να χρησιμοποιήσει τη δομή του ανωνυμοποιημένου γράφου και την δική του ανεξάρτητη γνώση για να ταυτοποιήσει άτομα αυτού του γράφου.

Σε ένα πραγματικό γενεαλογικό δέντρο θα είχαν πιθανώς συμπεριληφθεί και άλλες πληροφορίες εκτός από το ονοματεπώνυμο κάθε ατόμου, όπως π.χ. η ημερομηνία γέννησης (ή και γάμου, θανάτου κ.λπ.) Τέτοιες πληροφορίες μπορούν να χρησιμοποιηθούν εν γένει ως quasi-identifiers, δηλαδή ως βοηθητικά στοιχεία για την ταυτοποίηση ενός ατόμου. Π.χ. αν ο επιτιθέμενος έχει πρόσβαση σε ληξιαρχικά δεδομένα, μπορεί να περιορίσει δραστικά το σύνολο ατόμων που έχει γεννηθεί σε συγκεκριμένη μέρα. Στα πειράματα που κάναμε σ' αυτήν την εργασία έχουμε υποθέσει ότι τέτοιες πληροφορίες έχουν ήδη αφαιρεθεί, πριν γίνει αντικατάσταση των ονομάτων από τυχαίους κωδικούς.

5.1.2 Κρυπτογράφηση

Η κρυπτογράφηση δίνει την ευκαιρία να γίνει ένας συμβιβασμός ανάμεσα στην ασφάλεια/ιδιωτικότητα και στην δυνατότητα για ανάκτηση των αρχικών δεδομένων από το νόμιμο ιδιοκτήτη ή χρήστη τους. Στην περίπτωση αυτή επομένως ο κωδικός ή το ψευδώνυμο μπορούν να αναστραφούν αν κάποιος γνωρίζει το σωστό κλειδί με το οποίο έχει γίνει η κρυπτογράφηση. Φυσικά δημιουργείται ο κίνδυνος να σπάσει η κρυπτογράφηση.

5.1.3 K-ανωνυμία

Στη συγκεκριμένη μέθοδο, χρησιμοποιείται ως βάση δεδομένων ένας πίνακας με n γραμμές και m στήλες. Κάθε γραμμή του πίνακα αντιπροσωπεύει μια καταχώρηση στην βάση δεδομένων για ένα συγκεκριμένο άτομο και κάθε στήλη ένα διαφορετικό χαρακτηριστικό για το άτομο αυτό, μοναδικό ή μη, κάτι αντίστοιχο με το αρχείο profiles-anon.txt που προαναφέρθηκε.
Έστω ο παρακάτω πίνακας:

Name	Gender	Birth Year	Favourite Music
Mary	F	1995	Rock
George	M	1947	Jazz
Paul	M	1940	Metal
Mary	F	1966	Rock

Πίνακας 5: Βάση Δεδομένων για εφαρμογή k-ανωνυμίας

Σκοπός της k-ανωνυμίας είναι με την χρήση κάποιων μεθόδων, που θα αναφερθούν πιο κάτω, ο πίνακας στο τέλος της εφαρμογής τους να έχει k σε αριθμό ίδιες γραμμές.

Υπάρχουν δύο γνωστές μεθόδοι για να επιτευχθεί η k-ανωνυμία οι οποίες είναι οι εξής:

1. **Γενίκευση:**
   Η γενίκευση είναι η διαδικασία κατά την οποία η τιμή ενός χαρακτηριστικού που μπορεί να χρησιμεύσει ως ψευδο-αναγνωριστικό (quasi-identifier) αντικαθίστανται με μία πιο γενική τιμή ή καλύτερα ένα σύνολο τιμών. Για παράδειγμα, το έτος γέννησης μπορεί να αντικατασταθεί από μια τιμή δεκαετίας (το 1996 από τη δεκαετία του 1990, δηλαδή τα έτη 1990 ως 1999). Όσο όμως είναι μεγαλύτερη η γενίκευση, τόσο μειώνεται εν γένει ο κίνδυνος. Στο συγκεκριμένο παράδειγμα η χρήση 50ετίας αντί 10ετίας εν γένει μειώνει την πιθανότητα ταυτοποίησης.

2. **Απόκρυψη:**
   Η απόκρυψη είναι η διαδικασία κατά την οποία η τιμή ενός χαρακτηριστικού που μπορεί να χρησιμεύσει ως ψευδο-αναγνωριστικό (quasi-identifier) αντικαθίσταται από έναν αστερίσκο (*). Μπορούν να αντικατασταθούν όλες ή κάποιες από τις τιμές κάποιας στήλης. Για παράδειγμα

Η χρήση τόσο της γενίκευσης όσο και της απόκρυψης, εν γένει οδηγεί σε μη αναστρέψιμη απώλεια πληροφορίας για τον ερευνητή που θα χρησιμοποιήσει το κατ΄αυτόν τον τρόπο επεξεργασμένο και ανωνυμοποιημένο σύνολο δεδομένων. Ταυτόχρονα δεν εκμηδενίζει τον κίνδυνο αποανωνυμοποίησης.

Για να μπορέσει να επιτευχθεί η 2-ανωνυμία στον παραπάνω πίνακα, θα γίνει χρήση της 50ετίας στην μέθοδο της γενίκευσης και απόκρυψη των στηλών Name και Favourite Music, και έτσι θα έχει ως εξής:

Name	Gender	Birth Year	Favourite Music
*	F	1950-1999	*
*	M	1900-1949	*
*	M	1900-1949	*
*	F	1950-1999	*

Πίνακας 6: Παράδειγμα 2-ανωνυμίας

Έτσι, οι γραμμές είναι ίδιες ανά δύο. Η 1^{η} με την 4^{η} και η 2^{η} με την 3η.

Χρήση της k-ανωνυμίας σε αυτή τη διπλωματική:

Στην παρούσα διπλωματική έχουμε υποθέσει ότι έχουν αφαιρεθεί τα ψευδο-αναγνωριστικά, οπότε δεν τίθεται θέμα γενίκευσης ή απόκρυψης, και γενικά k-ανωνυμίας.

5.2 Τεχνικές ανωνυμοποίησης σε γράφους

Στην περίπτωση που στο αρχικό σύνολο δεδομένων περιλαμβάνεται γενεαλογική πληροφορία αι θέλουμε να εμποδίσουμε την αξιοποίησή της από τον επιτιθέμενο, μπορούμε να επέμβουμε στη δομή του γενεαλογικού γράφου. Όσο πιο δραστική είναι αυτή η επέμβαση, τόσο πιο πιθανό είναι να καταστραφούν πληροφορίες που θα χρησίμευαν σε ενα ερευνητή που θα διερευνούσε τη σχέση ενός χαρακτηριστικού με την κληρονομικότητα. Σε ένα ακραίο παράδειγμα, αν διαγραφούν από το σύνολο δεδομένων όλοι οι δείκτες που περιγράφουν σχέσεις γονέων-παιδιών (δηλαδή όλες οι ακμές του γενεαλογικού γράφου), αφενός θα μηδενιστεί η πιθανότητα να βρει ο επιτιθέμενος πρόσωπα βάσει γονέων και παιδιών, αφετέρου όμως θα ακυρωθεί οποιαδήποτε δυνατότητα ενός επιστήμονα να διερευνήσει κατά πόσο ένα χαρακτηριστικό είναι κληρονομικό.
Μια λιγότερο δραστική επέμβαση θα ήταν να μειωθεί ο αριθμός των παιδιών ως το πολύ *n*, π.χ. *n*=5 και θα μπορούσε να χρησιμοποιηθεί αν η πλευρά της άμυνας θεωρεί ότι ο επιτιθέμενος βασίζει την επίθεσή του στον εντοπισμό ατόμων με μεγάλο αριθμό παιδιών.
Μια τέτοια επέμβαση ακυρώνει την δυνατότητα του επιτιθέμενου να εντοπίσει ένα μικρό αριθμό ατόμων (ιδανικά ένα άτομο) που έχουν τον μέγιστο αριθμό παιδιών, δεδομένου ότι με τον τρόπο αυτόν μεγαλώνει ο αριθμός ατόμων με τον μέγιστο αριθμό παιδιών, που πλέον είναι ίσος με *n*.
Κατά συνέπεια η επιλογή του *n* αποτελεί ένα συμβιβασμό ανάμεσα στο πλήθος του συνόλου ατόμων με μέγιστο αριθμό παιδιών με σκοπό να γίνεται δύσκολη η επίθεση και στην καταστροφή πληροφορίας που δημιουργεί η διαγραφή ατόμων ή σχέσεων γονέα-παιδιού.
Πιο κάτω περιγράφονται τρεις διαφορετικές τεχνικές, οι οποίες εφαρμόστηκαν στα δεδομένα της διπλωματικής αυτής, καθώς και τα αποτελέσματα τους.

5.2.1 Διαγραφή περισσότερων των *n* παιδιών σε κάθε κόμβο πατέρα

Για την πρώτη τεχνική εκτελέστηκε το **Πρόγραμμα 6, DeleteNodes.py (Παράρτημα Ι)** το οποίο κάνει τα εξής:
Ξεκινώντας από έναν αρχικό κόμβο Χ και δεδομένου ενός αριθμού *n*, εάν τα παιδιά-κόμβοι του Χ υπερβαίνουν σε αριθμό τον *n*, τότε επιλέγονται τυχαία κάποια από αυτά και διαγράφονται από το γενεαλογικό δέντρο-γράφο, μαζί με τις αντίστοιχες ακμές (που καταλήγουν στον διαγραφόμενο κόμβο και φεύγουν από τον διαγραφόμενο κόμβο). Η διαδικασία επαναλαμβάνεται για κάθε κόμβο με περισσότερα των *n* παιδιά έτσι ώστε στον γράφο τελικά να μην υπάρχει κόμβος με περισσότερα του *n* παιδιά.

Αποτελέσματα:

Έγιναν 50 εκτελέσεις του προγράμματος 6 για n=0,…,16. Για κάθε μια από αυτές, ως αρχικός κόμβος Χ θεωρείτουν ένα τυχαίο άτομο (R.P = Random person στον πίνακα) από τα 3619 που αποτελούν το δέντρο.
Η εκτέλεση έγινε με την βοήθεια του GitBush και χρήση της εντολής:

```
for i in {1..50}; do python DeleteNodes.py $i; done
```

Τα αποτελέσματα για τα αντίστοιχα n φαίνονται στον παρακάτω πίνακα όπου:
Avg= μέσος όρος των αποτελεσμάτων για κάθε n

	1η	2η	3η	4η	5η	6η	7η	8η	…	50η	Avg
n=1	17	18	103	41	7	22	40	20	…	7	27.62
n=2	37	123	56	19	140	100	241	73	…	324	100.78
n=3	25	49	6	22	39	548	196	63	…	449	302.94
n=4	568	1004	1264	105	1108	75	1097	1130	…	168	692.54
n=5	87	89	181	286	489	247	1440	226	…	93	1152.02
n=6	2361	1890	2224	1774	2388	2371	2470	2265	…	2260	2079.60
n=7	2916	386	2683	2744	2490	2830	2935	2695	…	2853	2504.22
n=8	3178	13	3207	3176	3113	9	3074	3166	…	3022	2918.20
n=9	3321	3231	3358	3377	3349	3340	3353	3344	…	3360	3188.52
n=10	3490	3438	3438	3488	3426	3378	3340	3448	…	3456	3447.18
n=11	3531	3546	3545	3510	3540	3510	3530	3519	…	3519	3532.24
n=12	3560	3574	3576	3575	3558	3573	3575	3562	…	3552	3489.88
n=13	3591	3580	3584	3591	3591	3575	3576	3570	…	3591	3583.56
n=14	3597	3597	3597	3597	3597	3597	3597	3597	…	3597	3597.00
n=15	3599	3599	3599	3599	3599	3599	3599	3599	…	3599	3598.84
n=16	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3600.00

Πίνακας 7: Αποτελέσματα διαγραφής περισσότερων των n παιδιών σε κάθε κόμβο πατέρα

5.2.2 Διαγραφή περισσότερων των *n* παιδιών και των απογόνων τους σε κάθε κόμβο πατέρα

Ομοίως με πριν, ξεκινώντας από έναν αρχικό κόμβο Χ και δεδομένου ενός αριθμού ν, εάν τα παιδιά-κόμβοι του Χ υπερβαίνουν σε αριθμό τον ν, τότε επιλέγονται τυχαία κάποια από αυτά και διαγράφονται από το γενεαλογικό δέντρο-γράφο. Η διαφορά εδώ είναι ότι εκτός από τα παιδιά που θα διαγραφούν, θα διαγραφούν και όλοι οι απογόνοι τους. Η διαδικασία επαναλαμβάνεται για κάθε κόμβο με περισσότερα των ν παιδιά.

Ο αλγόριθμος εδώ είναι παρόμοιος με αυτόν της περίπτωσης 5.2.1. Η διαφορά τους είναι ότι σε αυτή την περίπτωση, προστίθεται στο πρόγραμμα η συνάρτηση FindDescendands (η οποία με ορίσματα τις λίστες deleted,husb,wife,name,notq καλείται αναδρομικά μέχρι να αποθηκευτούν στη λίστα notq όλοι οι απογόνοι του κάθε διεγραμμένου παιδιού της λίστας deleted).
Για την δεύτερη τεχνική λοιπόν, εκτελέστηκε το **Πρόγραμμα 6, DeleteNodes.py (Παράρτημα Ι)** με προσθήκη της συνάρτησης αυτής και μετατροπή της γραμμής 65 του κώδικα 5.2.1 από σχόλιο σε εντολή. Τα υπόλοιπα είναι ακριβώς ίδια με πριν.

Συνάρτηση FindDescendands:

```
1. def FindDescendands(deleted,husb,wife,name,notq):
2.     cofc=[]
3.     for i in range(0,len(deleted)):
4.         for k in range(0,len(husb)):
5.             if(deleted[i]==husb[k] or deleted[i]==wife[k]):
6.                 notq.append(name[k])
7.                 cofc.append(name[k])
8.
9.     if(cofc!=[]):
10.         FindDescendands(cofc,husb,wife,name,notq)
11.
12.     return notq
```

Ομοίως με πριν, έγιναν 50 εκτελέσεις του προγράμματος για n=0,…,16. Για κάθε μια από αυτές, ως αρχικός κόμβος X θεωρείτουν ένα τυχαίο άτομο από τα 3619 που αποτελούν το δέντρο.

Αποτελέσματα 50 εκτελέσεων:

	1η	2η	3η	4η	5η	6η	7η	8η	…	50η	Avg
n=1	17	15	17	28	4	8	29	26	…	40	27.14
n=2	52	20	36	17	37	27	10	40	…	222	96.24
n=3	52	84	11	10	92	264	69	307	…	60	240.92
n=4	561	882	1202	507	25	347	202	606	…	547	435.62
n=5	197	981	959	1385	1092	979	43	871	…	1160	836.52
n=6	1826	1995	11	1881	1932	378	1653	1744	…	2235	1546.44
n=7	312	31	2553	2829	2446	2654	2612	2629	…	2604	2169.06
n=8	3057	2752	3025	3022	2879	2971	2629	3001	…	3161	2800.30
n=9	3175	3223	3278	3350	3254	3211	51	3295	…	3283	3204.54
n=10	3464	3461	3356	3350	3465	3475	3508	3349	…	3435	3362.76
n=11	3529	3516	3538	3532	3504	3533	3525	3516	…	3479	3522.92
n=12	3567	3553	3575	3539	3560	3529	3577	3544	…	3565	3487.00
n=13	3591	3591	3590	3556	3591	3591	3585	3564	…	3591	3583.14
n=14	3597	3597	3597	3597	3597	3597	3597	3597	…	3597	3596.82
n=15	3599	3599	3599	3599	3599	3599	3599	3599	…	3599	3598.98
n=16	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3528.24

Πίνακας 8: Αποτελέσματα διαγραφής περισσότερων των n παιδιών και των απογόνων τους σε κάθε κόμβο πατέρα

5.2.3 Διαγραφή περισσότερων των *n* ακμών που συνδέουν γονέα-παιδί

Ξεκινώντας από έναν αρχικό κόμβο X και δεδομένου ενός αριθμού *n*, εάν τα παιδιά-κόμβοι του X (που έχει έστω με την γυναίκα του Y) υπερβαίνουν σε αριθμό τον *n*, τότε επιλέγονται τυχαία κάποια από αυτά και διαγράφονται οι ακμές που συνδέουν τα παιδιά αυτά με τους γονείς X,Y. Η διαφορά εδώ σε σχέση με τα 5.2.1 και 5.2.2 είναι ότι ναι μεν απαλείφεται η σύνδεση-ακμή μεταξύ γονιού και παιδιού, το παιδί όμως παραμένει στον γράφο μαζί με τους απογόνους του απλά τώρα χωρίς γονείς. Η διαδικασία επαναλαμβάνεται για κάθε κόμβο με περισσότερα των *n* παιδιά έτσι ώστε στον γράφο τελικά να μην υπάρχει κόμβος με περισσότερες των *n* ακμές.

Για την τρίτη τεχνική, εκτελέστηκε το **Πρόγραμμα 7, DeleteEdges.py (Παράρτημα Ι)**

Αποτελέσματα 50 εκτελέσεων:

	1η	2η	3η	4η	5η	6η	7η	8η	…	50η	Avg
n=1	3371	3348	3406	3370	3361	3362	3389	3357	…	3370	3128.50
n=2	3512	3449	3530	3435	3465	3497	3503	3521	…	3466	3489.52
n=3	3558	3559	3513	3555	3532	3514	3569	3502	…	3521	3333.56
n=4	3571	3566	3573	3577	3519	3535	3528	3567	…	3574	3556.64
n=5	3569	3573	3575	3564	3566	3571	3579	3579	…	3536	3497.12
n=6	3585	3580	3585	3585	3579	3581	3586	3580	…	3579	3582.56
n=7	3592	3594	3587	3585	3587	3591	3586	3594	…	3590	3517.88
n=8	3598	3599	3598	3599	3593	3598	3593	3599	…	3599	3453.66
n=9	3600	3594	3600	3594	3599	3598	3599	3593	…	3594	3596.74
n=10	3600	3600	3600	3600	3600	3600	3599	3599	…	3595	3526.42
n=11	3600	3595	3600	3595	3595	3599	3600	3600	…	3595	3527.38
n=12	7	3595	3595	3600	3600	3600	3600	3600	…	3600	3526.94
n=13	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3599.90
n=14	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3600.00
n=15	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3600.00
n=16	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3528.14

Πίνακας 9: Αποτελέσματα διαγραφής περισσότερων των n ακμών που συνδέουν γονέα-παιδί

6 Αποανωνυμοποίηση Δεδομένων

6.1 Στατιστικά στοιχεία γενεαλογικού δέντρου

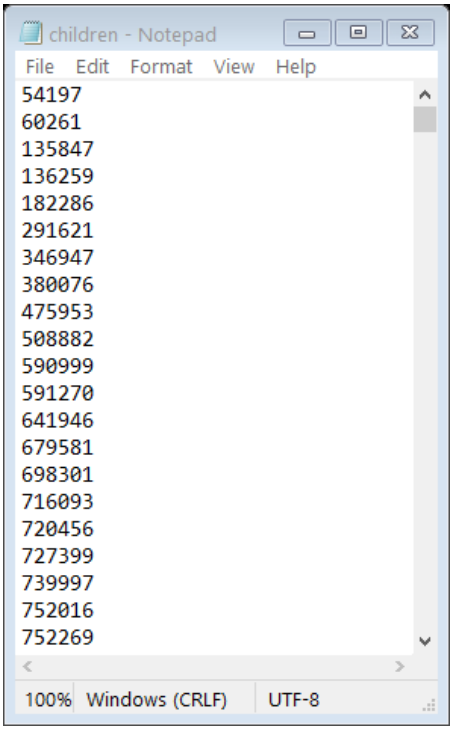
Η λογική που θεωρείται σ' αυτό το κεφάλαιο πως ακολουθείται από τον επιτιθέμενο για να γίνει η αποανωνυμοποίηση δεδομένων (ή *ταυτοποίηση* ατόμων) είναι να εντοπίσει αρχικά ένα άτομο του δέντρου με συγκεκριμένο μοναδικό (αν είναι δυνατόν) χαρακτηριστικό. Στον ανωνυμοποιημένο γράφο υποθέτουμε ότι διατηρείται μόνο η σχέση γονέα-παιδιού και, κατά περίπτωση, το φύλο κάθε ατόμου. Η ύπαρξη ενός ή περισσοτέρων άλλων χαρακτηριστικών που δεν είναι quasi-identifiers θεωρείται μη αξιοποιήσιμη. Επομένως ο επιτιθέμενος βασικά οφείλει να εστιάσει την προσοχή του στα παιδιά καθενός και συγκεκριμένα σε όσους έχουν ακραία μεγάλο αριθμό παιδιών.

Γι'αυτό θα χρειαστούν τα στατιστικά του Γ.Δ, δηλαδή ποιος έχει ποια παιδιά, πόσα κ.ο.κ. Στην περίπτωση αυτού του Γ.Δ. όπως θα φανεί, αυτά τα άτομα είναι δύο και το χαρακτηριστικό τους που τα κάνει να «ξεχωρίζουν» από τα υπόλοιπα είναι ότι έχουν τον μέγιστο αριθμό παιδιών που παρατηρείται σε όλο το δέντρο που είναι ίσος με 16.

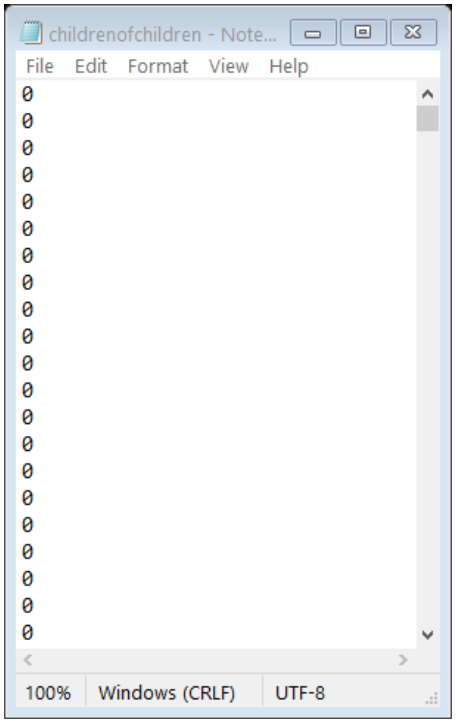
Μια άλλη χρήσιμη παρατήρηση είναι ότι στο συγκεκριμένο δέντρο που θα εξετάσουμε η ύπαρξη σχέσης «συντροφικότητας» (που μπορεί να υποδηλώνει γάμο ή και όχι) μεταξύ δύο ατόμων (διαφορετικού φύλου) προκύπτει από την ύπαρξη ενός τουλάχιστον κοινού παιδιού και όχι από μια ειδικού τύπου ακμή μεταξύ συντρόφων. Αν δύο άτομα έχουν βρεθεί σε τέτοια σχέση χωρίς να έχουν αποκτήσει παιδιά, η σχέση αυτή δεν είναι δυνατό να φανεί στο γράφο.

Εδώ, εκτελέστηκε το **Πρόγραμμα 8, GedStats.py (Παράρτημα Ι)** το οποίο βρίσκει τον αριθμό των παιδιών κάθε ατόμου προκειμένου να εντοπίσει ο επιτιθέμενος άτομα με μέγιστο αριθμό παιδιών. Τα στατιστικά τυπώνονται ομαδοποιημένα σε αρχεία όπως φαίνεται στις παρακάτω εικόνες.

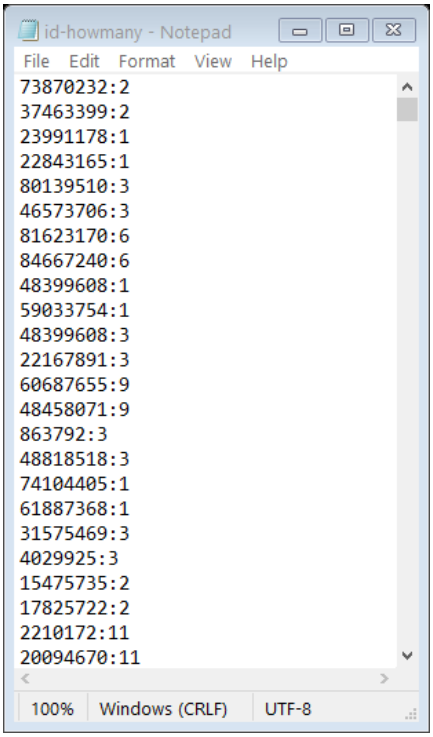
Αποτελέσματα:



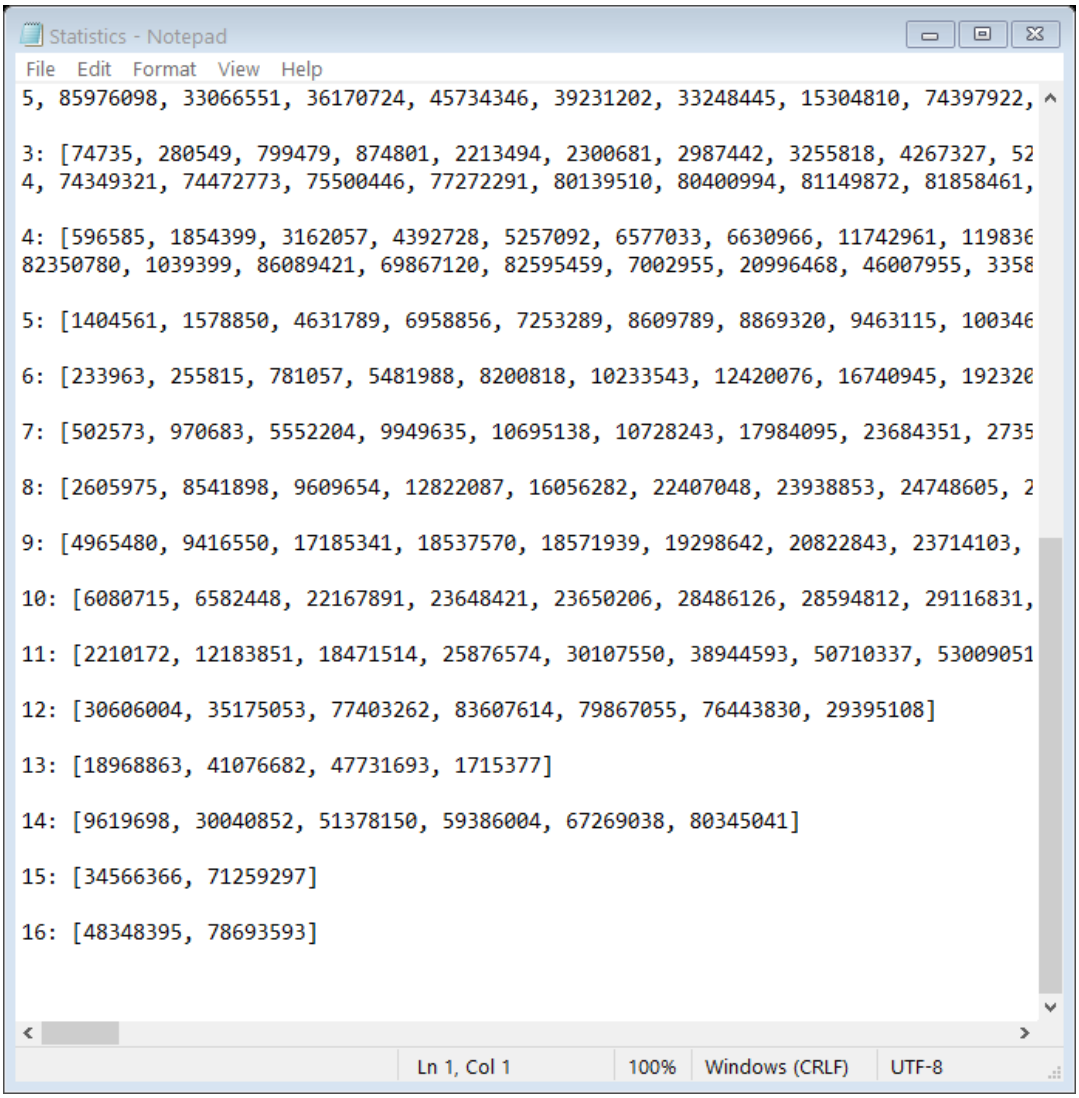
Εικόνα 14: Αρχείο «children.txt»



Εικόνα 13: Αρχείο «childrenofchildren.txt»



Εικόνα 15: Αρχείο «id-howmany.txt»



Εικόνα 16: Αρχείο «Statistics.txt»

6.2 Εκτέλεση Αποανωνυμοποίησης

Ιδανικά ο επιτιθέμενος περιμένει να βρει ένα άτομο με μέγιστο αριθμό παιδιών ή ένα ζευγάρι, δηλαδή δύο άτομα με κοινά όλα τους τα παιδιά. Εναλλακτικά μπορεί να βρεθεί ένα άτομο με μέγιστο αριθμό παιδιών ακόμη και αν όλα τα εν λόγω παιδιά διατηρούν και τους δύο γονείς τους στο γράφο, επειδή μπορεί από την άλλη πλευρά να προέρχονται από συντρόφους περισσότερες της μιας (ή περισσότερους του ενός). Εφόσον διατηρείται το φύλο στον ανωνυμοποιημένο γράφο, θα μπορούσε επίσης αυτή η πληροφορία να αξιοποιηθεί. Για παράδειγμα, αν 4 άτομα έχουν από 16 παιδιά, ήτοι δυο ζευγάρια με 16 παιδιά το κάθε ζευγάρι, ο επιτιθέμενος μπορεί να διακρίνει ανάμεσα στα δυο ζευγάρια χρησιμοποιώντας το πλήθος αγοριών-κοριτσιών μέσα στα παιδιά κάθε ζευγαριού. Σε μια κάπως διαφορετική προσέγγιση μπορεί να εντοπισθεί το ζευγάρι με τα περισσότερα παιδιά αντί του ατόμου με τα περισσότερα παιδιά κ.ο.κ.

Εναλλακτικά, αν δεν είναι η ιδιότητα υπαρκτή σε ένα μόνο άτομο, ο επιτιθέμενος μπορεί να χρησιμοποιήσει δύο ή περισσότερα «σενάρια» ή να διερευνήσει το δέντρο απογόνων καθενός από τα άτομα που είναι «ισόπαλα».

Η λογική των περαιτέρω ταυτοποιήσεων έχει ως εξής:

- Οι πρόγονοι κάθε ταυτοποιημένου ατόμου ταυτοποιούνται.
- Μοναδικά αγόρια ή μοναδικά κορίτσια ενός ταυτοποιημένου ατόμου ή ενός ταυτοποιημένου ζευγαριού ταυτοποιούνται επίσης.
- Μοναδικός/-ή σύντροφος ενός ταυτοποιημένου ατόμου, ταυτοποιείται επίσης.
- Αν το ταυτοποιημένο άτομο έχει >1 αριθμό συντρόφων, οι σύντροφοι μπορούν ενδεχομένως να διαφοροποιηθούν και ταυτοποιηθούν βάσει αριθμού παιδιών.
- Αν το ταυτοποιημένο άτομο έχει >1 αριθμό συντρόφων, οι σύντροφοι μπορούν ενδεχομένως να διαφοροποιηθούν και ταυτοποιηθούν βάσει διαφορετικού δέντρου απογόνων.
- Παιδιά του ταυτοποιημένου ατόμου μπορούν ενδεχομένως να διαφοροποιηθούν και ταυτοποιηθούν βάσει αριθμού δικού τους παιδιών (δηλαδή εγγονιών).
- Παιδιά του ταυτοποιημένου ατόμου μπορούν ενδεχομένως να διαφοροποιηθούν και ταυτοποιηθούν βάσει διαφορετικού δέντρου απογόνων.

Εν προκειμένω προσδιορίζεται ως κατάλληλο άτομο με 16 παιδιά αυτό με τον κωδικό αριθμό 48348395 (διατηρούμε τους κωδικούς του αρχικού δέντρου των Kaplanis κ.α.).

Η διαδικασία ακολουθεί τα παρακάτω βήματα:

1. Είναι γνωστό ότι ο 48348395 έχει 16 παιδιά, άρα και ταυτοποιείται.
   Τοποθετείται σε ένα σύνολο ταυτοποιημένων ατόμων Q και σε ένα σύνολο ατόμων που δεν έχουν εξεταστεί R.

Q	X1
R	X1

2. Οι X2=31778447 και X3=67522707 ταυτοποιούνται ως γονείς του X1

Q	X1	X2	X3
R	X1	X2	X3

3. Η X4=80345041 ταυτοποιείται ως σύζυγος του X1 με την οποία έχει μοναδικό αριθμό παιδιών ίσο με 13

Q	X1	X2	X3	X4
R	X1	X2	X3	X4

4. Η X5=11441472 ταυτοποιείται ως σύζυγος του X1 με την οποία έχει μοναδικό αριθμό παιδιών ίσο με 3

Q	X1	X2	X3	X4	X5
R	X1	X2	X3	X4	X5

5. Ο X11=18984480 ταυτοποιείται ως το παιδί του X1 που έχει τα περισσότερα παιδιά (7 παιδιά)

Q	X1	X2	X3	X4	X5	X11
R	X1	X2	X3	X4	X5	X11

6. Ο X12=31802999 ταυτοποιείται ως το δεύτερο παιδί του X1 με τα περισσότερα παιδιά (5 παιδιά)

Q	X1	X2	X3	X4	X5	X11	X12
R	X1	X2	X3	X4	X5	X11	X12

7. Ο X13=32047612 ταυτοποιείται ως το τρίτο παιδί του X1 με τα περισσότερα παιδιά (3 παιδιά)

Q	X1	X2	X3	X4	X5	X11	X12	X13
R	X1	X2	X3	X4	X5	X11	X12	X13

8. Ο X14=57585723 ταυτοποιείται ως το τέταρτο παιδί του X1 με τα περισσότερα παιδιά (2 παιδιά)

Q	X1	X2	X3	X4	X5	X11	X12	X13	X14
R	X1	X2	X3	X4	X5	X11	X12	X13	X14

9. Ο X15=1210147 ταυτοποιείται ως παιδί του X1 λόγω του διαφορετικού δέντρου απογόνων που έχει σε σύγκριση με τα υπόλοιπα παιδιά του X1

Q	X1	X2	X3	X4	X5	X11	X12	X13	X14	X15
R	X1	X2	X3	X4	X5	X11	X12	X13	X14	

10. Όλα τα υπόλοιπα παιδιά του X1 έχουν 1 ή κανένα παιδί και δεν μπορούν να ταυτοποιηθούν. Έτσι εξετάσαμε τον X1, τον διαγράφουμε από το σύνολο R και προχωράμε στο επόμενο άτομο προς εξέταση, τον X2.

Q	X1	X2	X3	X4	X5	X11	X12	X13	X14	X15
R	~~X1~~	X2	X3	X4	X5	X11	X12	X13	X14	

11. Ο X6=10053541 ταυτοποιείται ως γονιός του X2.
(Ο 31778447 ταυτοποιείται ως σύζυγος του X2 αλλά υπάρχει ήδη ως X3 άρα δεν προστίθεται στα σύνολα).
Δεν ταυτοποιείται κανείς άλλος.
Επόμενό άτομο προς εξέταση: X3

Q	X1	X2	X3	X4	X5	X11	X12	X13	X14	X15	X6
R	~~X2~~	X3	X4	X5	X11	X12	X13	X14	X6		

12. Δεν ταυτοποείται κανείς άλλος.
Επόμενο άτομο προς εξέταση: X4

Q	X1	X2	X3	X4	X5	X11	X12	X13	X14	X15	X6
R	~~X3~~	X4	X5	X11	X12	X13	X14	X6			

13. Οι X7= 55288219 και X8 = 39128549 ταυτοποιούνται ως γονείς του X4.

Q	X1	X2	X3	X4	X5	X11	X12	X13	X14	X15	X6	X7	X8
R	X4	X5	X11	X12	X13	X14	X6	X7	X8				

14. Ο X9 = 69844408 ταυτοποιείται ως ο σύζυγος της X4 με τον οποίο έχει μοναδικό αριθμό παιδιών ίσο με 1.

Q	X1	X2	X3	X4	X5	X11	X12	X13	X14	X15	X6	X7	X8	X9
R	X4	X5	X11	X12	X13	X14	X6	X7	X8	X9				

15. Αφού έχουν 1 μόνο παιδί ταυτοποιείται και αυτό ως ο X41=641946

Q	X1	X2	X3	X4	X5	X11	X12	X13	X14	X15	X6	X7	X8	X9
	X41													
R	X4	X5	X11	X12	X13	X14	X6	X7	X8	X9	X41			

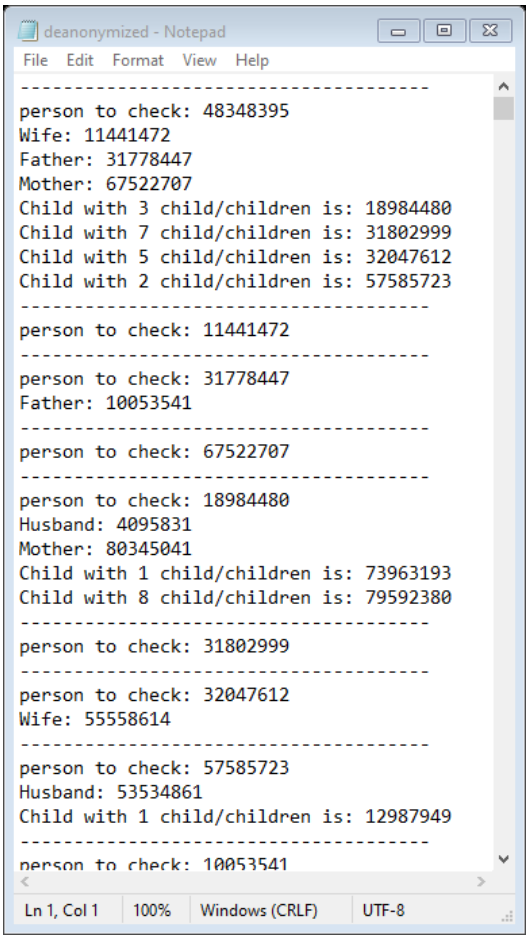
16. Ο X42=70910982 ταυτοποιείται ως το παιδί του X4 με μοναδικό αριθμό παιδιών ίσο με 1.
Ο X4 διαγράφεται.
Επόμενο άτομο προς εξέταση: X5

Q	X1	X2	X3	X4	X5	X11	X12	X13	X14	X15	X6	X7	X8	X9
	X41	X42												
R	~~X4~~	X5	X11	X12	X13	X14	X6	X7	X8	X9	X41	X42		

…

Κάπως έτσι ταυτοποιούνται 1739 από τα 3619 άτομα του δέντρου, όπως φαίνεται και από την εκτέλεση του προγράμματος 9.
Το **Πρόγραμμα 9, Deanonymize.py (Παράρτημα Ι)** είναι ο αλγόριθμος που εκτελεί τα παραπάνω βήματα.

Αποτελέσματα:



Εικόνα 17: Αποτελέσματα εκτέλεσης Deanonymize.py - Αρχείο «deanonymized.txt»

1739 άτομα ταυτοποιήθηκαν από τα 3619

1739/3619= 0.480519481= 48% ποσοστό επιτυχίας αποανωνυμοποίησης

6.3 Αποανωνυμοποίηση σε δέντρο με παιδιά που έχουν διαγραφεί

Στην περίπτωση που ο γράφος έχει υποστεί κλάδεμα οποιουδήποτε είδους (δηλ. διαγραφή κόμβων και ακμών ή μόνο διαγραφή ακμών) με αποτέλεσμα τον περιορισμό μέγιστου αριθμού παιδιών σε δεδομένο αριθμό *n*, ο αλγόριθμος από-ανωνυμοποίησης που έχουμε εξετάσει ως τώρα έχει δυσκολία να ξεκινήσει από ένα συγκεκριμένο άτομο (ή λίγα αρχικά άτομα).

Ο εντοπισμός ενός ατόμου με μοναδικές ιδιότητες είναι ένα ευρύτερο πρόβλημα, που θα μπορούσε να γίνει με αναζήτηση ατόμων με μοναδικό δέντρο απογόνων ή και προγόνων και γενικότερα σχετίζεται με το πρόβλημα του ισομορφισμού γράφων. Εδώ δεν πρόκειται να εξετάσουμε μεθόδους που βασίζονται σε αυτήν την λογική, αλλά θα περιοριστούμε σε μεθόδους που αποτελούν συνέχεια της λογικής που έχει αναπτυχθεί στα προηγούμενα κεφάλαια.

Στον αλγόριθμο που θα εξεταστεί θεωρείται ως προϋπόθεση ότι είναι γνωστό ένα οποιοδήποτε άτομο του καινούριου κλαδεμένου δέντρου. Αυτό το άτομο επιλέγεται τυχαία στην αρχή, δηλαδή θεωρούμε ότι ο επιτιθέμενος έχει «εξωτερική» πληροφόρηση για την ταυτότητα ενός ατόμου.

Στις προσομοιώσεις εφαρμόζεται το **Πρόγραμμα 9 Deanonymize.py** με δύο σημαντικές προσθήκες στον αλγόριθμο.

1. Για αρχεία εισόδου διαβάζονται τα επεξεργασμένα βάσει των νέων δεδομένων αρχεία αντί για τα πρωτότυπα.

2. Διαβάζεται το αρχείο cut_tree.txt το οποίο περιέχει τα άτομα του «κομμένου» δέντρου και κάθε φορά που βρίσκεται ένα νέο άτομο για αποανωνυμοποίηση, πριν προστεθεί στην λίστα ταυτοποιημένων, ελέγχεται εάν είναι άτομο του καινούριου δέντρου. Αν όχι δεν λαμβάνεται υπόψη.

6.3.1 Αποτελέσματα

Οι γραμμές πάνε ανά δύο:
Στην πρώτη (n = i, όπου i=0,…,16) είναι τα αποτελέσματα της ανωνυμοποίησης, δηλαδή το πλήθος ατόμων που έχει το τελικό δέντρο μετά την εφαρμογή της τεχνικής 5.2.1 (Διαγραφή περισσότερων των ν παιδιών σε κάθε κόμβο πατέρα) - εκτέλεσης του προγράμματος DeleteNodes.py.
Στην δεύτερη (n = i D, όπου i=0,…,16) είναι τα αποτελέσματα της αποανωνυμοποίησης των προηγουμένων, δηλαδή το πλήθος των ατόμων που κατάφεραν να αποανωνυμοποιηθούν από τους παραπάνω μετά την εκτέλεση του προγράμματος Deanonymize.py.

	1η	2η	3η	4η	5η	6η	7η	8η	…	50η	Avg
n=1	17	18	103	41	7	22	40	20	…	7	27.62
n=1 D	17	18	103	41	7	22	40	20	…	7	27.28
n=2	37	123	56	19	140	100	241	73	…	324	100.78
n=2 D	35	111	49	15	127	88	217	67	…	292	91.74
n=3	25	49	6	22	39	548	196	63	…	449	302.94
n=3 D	20	32	3	16	27	329	148	38	…	294	221.98
n=4	568	1004	1264	105	1108	75	1097	1130	…	168	692.54
n=4 D	417	667	862	68	662	20	713	603	…	85	437.34
n=5	87	89	181	286	489	247	1440	226	…	93	1152.02
n=5 D	40	39	89	160	257	121	900	104	…	48	612.30
n=6	2361	1890	2224	1774	2388	2371	2470	2265	…	2260	2079.60
n=6 D	1286	1031	1244	1046	1319	1308	1355	1229	…	1258	1102.04
n=7	2916	386	2683	2744	2490	2830	2935	2695	…	2853	2504.22
n=7 D	1544	157	1409	1439	1324	1531	1536	1418	…	1519	1302.22
n=8	3178	13	3207	3176	3113	9	3074	3166	…	3022	2918.20
n=8 D	1612	6	1634	1589	1624	2	1565	1614	…	1556	1450.28
n=9	3321	3231	3358	3377	3349	3340	3353	3344	…	3360	3188.52
n=9 D	1662	1623	1684	1681	1679	1654	1677	1663	…	1667	1591.94
n=10	3490	3438	3438	3488	3426	3378	3340	3448	…	3456	3447.18
n=10 D	1720	1693	1701	1712	1698	1690	1651	1705	…	1703	1704.12
n=11	3531	3546	3545	3510	3540	3510	3530	3519	…	3519	3532.24
n=11 D	1727	1730	1724	1710	1737	1728	1710	1718	…	1719	1725.98
n=12	3560	3574	3576	3575	3558	3573	3575	3562	…	3552	3489.88
n=12 D	1736	1734	1739	1739	1721	1740	1732	1733	…	1731	1697.34
n=13	3591	3580	3584	3591	3591	3575	3576	3570	…	3591	3583.56
n=13 D	1743	1739	1732	1740	1740	1723	1738	1733	…	1738	1736.90
n=14	3597	3597	3597	3597	3597	3597	3597	3597	…	3597	3597.00
n=14 D	1746	1740	1741	1737	1740	1740	1739	1738	…	1740	1739.28
n=15	3599	3599	3599	3599	3599	3599	3599	3599	…	3599	3598.84
n=15 D	1746	1741	1734	1741	1741	1737	1740	1740	…	1738	1739.30
n=16	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3600.00
n=16 D	1744	1740	1738	1740	1743	1740	1737	1734	…	1732	1739.86

Πίνακας 10: Αποτελέσματα αποανωνυμοποίησης σε δέντρο με παιδιά που έχουν διαγραφεί

6.3.2 Διαστήματα Εμπιστοσύνης

Για την εύρεση διαστημάτων εμπιστοσύνης 90% τόσο σε αυτήν όσο και στις μετέπειτα υποενότητες χρησιμοποιήθηκαν τα εξής:

Δεδομένα: Ως δεδομένα χρησιμοποιήθηκαν τα κλάσματα

$$\frac{\alpha\pi o\alpha\nu\omega\nu\nu\mu o\pi o\iota\mu\acute{\varepsilon}\nu\omega\nu}{\alpha\nu\omega\nu\nu\mu o\pi o\eta\mu\acute{\varepsilon}\nu\omega\nu} = \frac{\Gamma\rho\alpha\mu\mu\acute{\eta}\ n{=}i\ D\ (i{=}0,\ldots,16)}{\Gamma\rho\alpha\mu\mu\acute{\eta}\ n{=}i\ (i{=}0,\ldots,16)}$$

Για παράδειγμα, για n=15 θα χρησιμοποιηθούν ως δεδομένα τα κλάσματα:

Δεδομένα = $[\frac{1746}{3599}, \frac{1741}{3599}, \ldots, \frac{1738}{3599}]$

Μέσος όρος:

Ο μέσος όρος για 50 εκτελέσεις θα βρεθεί από την διαίρεση: $\frac{\acute{A}\theta\rho o\iota\sigma\mu\alpha\ \Delta\varepsilon\delta o\mu\acute{\varepsilon}\nu\omega\nu}{\pi\lambda\acute{\eta}\theta o\varsigma\ \Delta\varepsilon\delta o\mu\acute{\varepsilon}\nu\omega\nu}$

Στο πιο πάνω παράδειγμα για n=15 θα είναι:

Μέσος όρος = $\bar{X} = \frac{[\frac{1746}{3599} + \frac{1741}{3599} + \ldots + \frac{1738}{3599}]}{50}$ = 0.483257

Διάστημα Εμπιστοσύνης:

Τύπος**:** ($\bar{X}$ - z * $\frac{s}{\sqrt{m}}$, $\bar{X}$ + z * $\frac{s}{\sqrt{m}}$) = ($\bar{X}$ − H , $\bar{X}$ + H)

Εδώ:
z = 1.645 για 90% Δ.Ε.
m = 50 τιμές

s = $\sqrt{\frac{\Sigma\,(X-\bar{X})^2}{50}}$

	Μέσος Όρος	Η	Διάστημα Εμπιστοσύνης 90%	
	$\bar{X}$	$z * \frac{s}{\sqrt{50}}$	$\bar{X}-z * \frac{s}{\sqrt{50}}$	$\bar{X}+z * \frac{s}{\sqrt{50}}$
n=1	0.994785	0.007067	0.987718	1.001851
n=2	0.887591	0.024521	0.863070	0.912112
n=3	0.703984	0.021763	0.682221	0.725748
n=4	0.619840	0.026515	0.593325	0.646356
n=5	0.528156	0.033404	0.494752	0.561561
n=6	0.525882	0.024564	0.501318	0.550445
n=7	0.506977	0.018801	0.488177	0.525778
n=8	0.491033	0.021223	0.469810	0.512255
n=9	0.494369	0.008014	0.486355	0.502382
n=10	0.494409	0.000760	0.493649	0.495170
n=11	0.488631	0.000454	0.488177	0.489084
n=12	0.479834	0.010948	0.468886	0.490782
n=13	0.484672	0.000234	0.484438	0.484907
n=14	0.483498	0.000153	0.483346	0.483651
n=15	0.483257	0.000153	0.483104	0.483410
n=16	0.483271	0.000194	0.483077	0.483465

Πίνακας 11: Διαστήματα εμπιστοσύνης 90% για αποανωνυμοποίηση σε δέντρο με παιδιά που έχουν διαγραφεί

Άρα, με μέγεθος δείγματος = 50, για κάθε n η καλύτερη εκτίμηση για το πλήθος ανωνυμοποιημένων είναι ο μέσος όρος. Με βάση αυτό το δείγμα , είμαστε 90% σίγουροι ότι το πλήθος αποανωνυμοποιημένων για κάθε n, βρίσκεται στα παραπάνω διαστήματα.
Π.χ: για n=15 είμαστε 90% σίγουροι ότι το πλήθος αποανωνυμοποιημένων ατόμων είναι μεταξύ (0.483104, 0. 483410) δηλαδή (1739.17 , 1740.28).
Λόγω του μεγάλου μεγέθους του δείγματος. το περιθώριο σφάλματος είναι μικρότερο εδώ.

6.4 Αποανωνυμοποίηση σε δέντρο με διεγραμμένα παιδιά και απογόνους

Ο ίδιος αλγόριθμος της υποενότητας 6.3 εφαρμόζεται και εδώ.
Τα αποτελέσματα και εδώ βρίσκονται με την προϋπόθεση ότι ένα άτομο του καινούριου δέντρου είναι γνωστό.

6.4.1 Αποτελέσματα

Οι γραμμές πάνε ανά δύο:
Στην πρώτη (n = i, όπου i=0,…,16) είναι τα αποτελέσματα της ανωνυμοποίησης, δηλαδή το πλήθος ατόμων που έχει το τελικό δέντρο μετά την εφαρμογή της τεχνικής 5.2.2 (Διαγραφή περισσότερων των ν παιδιών και των απογόνων τους σε κάθε κόμβο πατέρα) και εκτέλεσης του προγράμματος DeleteNodes.py με την προσθήκη της συνάρτησης finddescendands.py.

Στην δεύτερη (n = i D, όπου i=0,…,16) όπως και πριν είναι τα αποτελέσματα της αποανωνυμοποίησης των προηγουμένων, δηλαδή το πλήθος των ατόμων που κατάφεραν να αποανωνυμοποιηθούν από τους παραπάνω μετά την εκτέλεση του προγράμματος Deanonymize.py.

	1η	2η	3η	4η	5η	6η	7η	8η	…	50η	Avg
n=1	17	15	17	28	4	8	29	26	…	40	27.14
n=1 D	17	15	17	28	4	8	29	26	…	40	25.70
n=2	52	20	36	17	37	27	10	40	…	222	96.24
n=2 D	36	12	26	13	33	27	5	36	…	176	82.28
n=3	52	84	11	10	92	264	69	307	…	60	240.92
n=3 D	31	24	7	6	57	202	44	105	…	46	158.66
n=4	561	882	1202	507	25	347	202	606	…	547	435.62
n=4 D	374	563	753	46	13	231	115	417	…	361	258.06
n=5	197	981	959	1385	1092	979	43	871	…	1160	836.52
n=5 D	100	607	586	832	675	531	15	1	…	711	458.08
n=6	1826	1995	11	1881	1932	378	1653	1744	…	2235	1546.44
n=6 D	992	959	6	1080	1071	199	865	980	…	1241	826.04
n=7	312	31	2553	2829	2446	2654	2612	2629	…	2604	2169.06
n=7 D	138	13	1395	1485	1	1396	1421	1393	…	1382	1107.26
n=8	3057	2752	3025	3022	2879	2971	2629	3001	…	3161	2800.30
n=8 D	1493	1466	1534	5	1516	1535	1382	1550	…	1577	1369.16
n=9	3175	3223	3278	3350	3254	3211	51	3295	…	3283	3204.54
n=9 D	1614	1617	1591	1687	1602	1567	25	1653	…	1635	1560.10
n=10	3464	3461	3356	3350	3465	3475	3508	3349	…	3435	3362.76
n=10 D	1715	1700	1655	1639	1707	1714	1713	1667	…	1653	1657.60
n=11	3529	3516	3538	3532	3504	3533	3525	3516	…	3479	3522.92
n=11 D	1718	1719	1729	1729	1716	1726	1722	1725	…	1714	1722.98
n=12	3567	3553	3575	3539	3560	3529	3577	3544	…	3565	3487.00
n=12 D	1739	1730	1738	1726	1721	1724	1736	1727	…	1734	1695.72
n=13	3591	3591	3590	3556	3591	3591	3585	3564	…	3591	3583.14
n=13 D	1738	1733	1738	1727	1742	1741	1738	1728	…	1740	1737.20
n=14	3597	3597	3597	3597	3597	3597	3597	3597	…	3597	3596.82
n=14 D	1741	1741	1740	1740	1740	1737	1743	1742	…	1740	1739.94
n=15	3599	3599	3599	3599	3599	3599	3599	3599	…	3599	3598.98
n=15 D	1738	1736	1740	1740	1738	1746	1740	1741	…	1740	1739.74
n=16	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3528.24
n=16 D	1734	1740	1740	1741	1740	1739	1741	1743	…	1740	1704.96

Πίνακας 12: Αποτελέσματα αποανωνυμοποίησης σε δέντρο με διεγραμμένα παιδιά και απογόνους τους

6.4.2 Διαστήματα Εμπιστοσύνης

Για διάστημα εμπιστοσύνης 90% έχουμε τα παρακάτω αποτελέσματα:

	Μέσος Όρος	Η	Διάστημα Εμπιστοσύνης 90%	
	$\bar{X}$	$z * \frac{s}{\sqrt{50}}$	$\bar{X}-z * \frac{s}{\sqrt{50}}$	$\bar{X}+z * \frac{s}{\sqrt{50}}$
n=1	0.976856	0.022821	0.954035	0.999677
n=2	0.838733	0.03268	0.806053	0.871413
n=3	0.640517	0.041233	0.599284	0.681751
n=4	0.599473	0.035181	0.564292	0.634654
n=5	0.542933	0.037962	0.504971	0.580895
n=6	0.519280	0.022862	0.496418	0.542142
n=7	0.502638	0.020165	0.482473	0.522804
n=8	0.482861	0.026726	0.456135	0.509587
n=9	0.486705	0.017032	0.469673	0.503737
n=10	0.486565	0.010685	0.475881	0.497250
n=11	0.489127	0.000470	0.488656	0.489597
n=12	0.484298	0.003349	0.480949	0.487647
n=13	0.484844	0.000260	0.484584	0.485104
n=14	0.483738	0.000148	0.483591	0.483886
n=15	0.483408	0.000132	0.483276	0.483540
n=16	0.476825	0.010838	0.465987	0.487664

Πίνακας 13: Διαστήματα Εμπιστοσύνης 90% για αποανωνυμοποίησ σε δέντρο με διεγραμμένα παιδιά και απογόνους τους

Άρα, με μέγεθος δείγματος = 50, για κάθε n η καλύτερη εκτίμηση για το πλήθος ανωνυμοποιημένων είναι ο μέσος όρος. Με βάση αυτό το δείγμα , είμαστε 90% σίγουροι ότι το πλήθος αποανωνυμοποιημένων για κάθε n, βρίσκεται στα παραπάνω διαστήματα.
Π.χ: για n=15 είμαστε 90% σίγουροι ότι το πλήθος αποανωνυμοποιημένων ατόμων είναι μεταξύ (0.483276, 0.483540) δηλαδή (1739.79 , 1740.44).
Λόγω του μεγάλου μεγέθους του δείγματος. το περιθώριο σφάλματος είναι μικρότερο εδώ.

6.5 Αποανωνυμοποίηση σε δέντρο με διεγραμμένες ακμές μεταξύ γονέα-παιδιού

Ομοίως με πριν εφαρμόζεται ο ίδιος αλγόριθμος για τυχαίο γνωστό άτομο X κάθε φορά.

6.5.1 Αποτελέσματα

Οι γραμμές πάνε ανά δύο:
Στην πρώτη (n = i, όπου i=0,…,16) είναι τα αποτελέσματα της ανωνυμοποίησης, δηλαδή το πλήθος ατόμων που έχει το τελικό δέντρο μετά την εφαρμογή της τεχνικής 5.2.3 (Διαγραφή περισσότερων των ν ακμών που συνδέουν γονέα-παιδί) και εκτέλεσης του προγράμματος DeleteEdges.py με την προσθήκη της συνάρτησης finddescendands.py.
Στην δεύτερη (n = i D, όπου i=0,…,16) όπως και πριν είναι τα αποτελέσματα της αποανωνυμοποίησης των προηγουμένων, δηλαδή το πλήθος των ατόμων που κατάφεραν να αποανωνυμοποιηθούν από τους παραπάνω μετά την εκτέλεση του προγράμματος Deanonymize.py.

	1η	2η	3η	4η	5η	6η	7η	8η	…	50η	Avg
n=1	3371	3348	3406	3370	3361	3362	3389	3357	…	3370	3128.50
n=1 D	1	1	10	7	3	5	3	1	…	1	4.90
n=2	3512	3449	3530	3435	3465	3497	3503	3521	…	3466	3489.52
n=2 D	11	33	9	1	1	1	1	14	…	32	30.76
n=3	3558	3559	3513	3555	3532	3514	3569	3502	…	3521	3333.56
n=3 D	1	345	1	11	1	115	1	1	…	1	51.62
n=4	3571	3566	3573	3577	3519	3535	3528	3567	…	3574	3556.64
n=4 D	1	34	530	33	1	632	486	1	…	2	208.18
n=5	3569	3573	3575	3564	3566	3571	3579	3579	…	3536	3497.12
n=5 D	5	19	58	909	1200	52	957	217	…	1	549.32
n=6	3585	3580	3585	3585	3579	3581	3586	3580	…	3579	3582.56
n=6 D	31	1	16	1403	1348	52	5	4	…	1300	722.38
n=7	3592	3594	3587	3585	3587	3591	3586	3594	…	3590	3517.88
n=7 D	1451	1460	1	1481	1507	1512	1492	19	…	1359	1207.44
n=8	3598	3599	3598	3599	3593	3598	3593	3599	…	3599	3453.66
n=8 D	1620	4	1642	1605	1568	1642	1622	1650	…	1636	1227.54
n=9	3600	3594	3600	3594	3599	3598	3599	3593	…	3594	3596.74
n=9 D	1667	1685	1668	1661	1676	1	1642	1670	…	1664	1565.58
n=10	3600	3600	3600	3600	3600	3600	3599	3599	…	3595	3526.42
n=10 D	1717	1715	1713	1719	1730	1710	1726	1702	…	1682	1607.54
n=11	3600	3595	3600	3595	3595	3599	3600	3600	…	3595	3527.38
n=11 D	1741	1717	1714	1728	1715	1723	1728	1	…	1727	1658.14
n=12	7	3595	3595	3600	3600	3600	3600	3600	…	3600	3526.94
n=12 D	1	1734	1723	1728	1721	1730	1740	1737	…	1742	1696.84

n=13	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3599.90
n=13 D	1741	1731	1731	1740	1730	1743	1746	1734	…	1741	1737.52
n=14	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3600.00
n=14 D	1741	1739	1740	1741	1742	1741	1740	1741	…	1743	1705.86
n=15	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3600.00
n=15 D	1743	1739	1741	1738	1741	1738	1738	1741	…	1740	1739.70
n=16	3600	3600	3600	3600	3600	3600	3600	3600	…	3600	3528.14
n=16 D	1741	1740	1741	1743	1740	1739	1741	1740	…	1739	1705.06

Πίνακας 14: Αποτελέσματα αποανωνυμοποίησης σε δέντρο με διεγραμμένες ακμές μεταξύ γονέα-παιδιού

6.5.2 Διαστήματα Εμπιστοσύνης

Για διάστημα εμπιστοσύνης 90% έχουμε τα παρακάτω αποτελέσματα:

	Μέσος Όρος	**H**	**Διάστημα Εμπιστοσύνης 90%**	
	$\bar{X}$	$\mathbf{z} * \frac{s}{\sqrt{50}}$	$\bar{X}\text{-}\mathbf{z} * \frac{s}{\sqrt{50}}$	$\bar{X}\text{+}\mathbf{z} * \frac{s}{\sqrt{50}}$
n=1	0.010684	0.013686	-0.003000	0.024369
n=2	0.008918	0.003557	0.005361	0.012475
n=3	0.026365	0.017365	0.008999	0.043730
n=4	0.059737	0.021807	0.037931	0.081544
n=5	0.160485	0.033740	0.126745	0.194225
n=6	0.205571	0.041147	0.164424	0.246718
n=7	0.338385	0.035902	0.302484	0.374287
n=8	0.344147	0.043557	0.300590	0.387705
n=9	0.434728	0.026815	0.407914	0.461543
n=10	0.451956	0.023469	0.428486	0.475425
n=11	0.463632	0.019423	0.444209	0.483054
n=12	0.481122	0.000377	0.480745	0.481498
n=13	0.482638	0.000321	0.482317	0.482960
n=14	0.473651	0.016493	0.457158	0.490144
n=15	0.483231	0.000161	0.483070	0.483392
n=16	0.479244	0.006764	0.472479	0.486008

Πίνακας 15: Διαστήματα Εμπιστοσύνης 90% για αποανωνυμοποίηση σε δέντρο με διεγραμμένες ακμές μεταξύ γονέα-παιδιού

Άρα, με μέγεθος δείγματος = 50, για κάθε n η καλύτερη εκτίμηση για το πλήθος ανωνυμοποιημένων είναι ο μέσος όρος. Με βάση αυτό το δείγμα , είμαστε 90% σίγουροι ότι το πλήθος αποανωνυμοποιημένων για κάθε n, βρίσκεται στα παραπάνω διαστήματα.
Π.χ: για n=15 είμαστε 90% σίγουροι ότι το πλήθος αποανωνυμοποιημένων ατόμων είναι μεταξύ (0.483070 , 0.483392) δηλαδή (1739.05 , 1740.21). Λόγω του μεγάλου μεγέθους του δείγματος. το περιθώριο σφάλματος είναι μικρότερο εδώ.

6.6 Εναλλακτικές μεθόδοι αποανωνυμοποίησης

6.6.1 Διαφορετικός τρόπος ταυτοποίησης αρχικού κόμβου

Ένα ιδανικότερο σενάριο θα ήταν να μην χρειάζεται η γνώση κάποιου αρχικού κόμβου Χ για να μπορέσει να ξεκινήσει η αποανωνυμοποίηση.

Σ' αυτήν την περίπτωση, μια λύση θα ήταν να εφαρμοστεί μια παραλλαγή της συνάρτησης FindDescendandsTree η οποία ως «διαφορετικό» δέντρο απογόνων θα θεωρούσε μόνο ένα δέντρο με μοναδικό αριθμό γενεών. Εάν η συνάρτηση αυτή εφαρμοστεί πάνω σε οποιοδήποτε κλαδεμένο/κομμένο δέντρο ξεκινώντας από τους κόμβους-ρίζες, δηλαδή αυτούς που δεν έχουν γονείς, και βρει ένα μοναδικό δέντρο απογόνων για κάποιον από αυτούς τότε ο συγκεκριμένος κόμβος-ρίζα ταυτοποιείται. Το σημαντικό όμως είναι ότι αυτός ο κόμβος, αυτόματα ταυτοποιείται και στο αρχικό δέντρο. Αυτό συμβαίνει διότι, για n>=1 όσα παιδιά και να «κοπούν» από το δέντρο, το ύψος του παραμένει το ίδιο.
Έτσι, εάν κλαδευτεί κάποιο δέντρο για n=2 και βρεθεί μοναδικό δέντρο απογόνων για τον Χ, τότε ο Χ ταυτοποιείται στο αρχικό δέντρο και μπορεί μετέπειτα να χρησιμοποιηθεί αυτός ως αρχικός κόμβος για τις μετέπειτα αποανωνυμοποιήσεις που αναφέρθηκαν στην αρχή του κεφαλαίου και να γίνει μια πιο «σωστή» αποανωνυμοποίηση.

Φυσικά αυτό είναι αρκετά δύσκολο στο δέντρο του συγκεκριμένου εύρους, αφού είναι σχεδόν απίθανο να βρεθεί ένα και μοναδικό υπόδεντρο με m μοναδικές γενιές.

6.6.2 Χρήση συνδυαστικής για περαιτέρω ταυτοποιήσεις

Ας κάνουμε την απλοποιητική υπόθεση ότι σε όλα τα ζευγάρια και οι δύο πλευρές τους δεν έχουν άλλους συντρόφους, δηλαδή όλα τα παιδιά τους είναι παιδιά του ζευγαριού. Έστω τώρα ένα ζευγάρι με N παιδιά, όπου $N>n$. Όταν γίνει κλάδεμα των παιδιών ώστε να έχουν το πολύ n, υπάρχουν $\binom{N}{n}$ τρόπου να επιλεγούν τα τελικά n παιδιά από τα αρχικά N.

Ας υποθέσουμε τώρα ότι υπάρχει μια μόνο τέτοια οικογένεια. Ο επιτιθέμενος, με την υπόθεση ότι γνωρίζει τον πραγματικό γράφο, γνωρίζει και ότι έχει γίνει κλάδεμα ως n, αλλά δεν μπορεί να γνωρίζει ποια ακριβώς παιδιά έχουν αφαιρεθεί στο κλάδεμα, θα πάρει τον πραγματικό γράφο και θα δοκιμάσει όλα τα δυνατά κλαδέματα και στη συνέχεια θα προσπαθήσει να δει ποιος από τους κλαδεμένους πραγματικούς γράφους φαίνεται να είναι ισόμορφος με τον ανωνυμοποιημένο και κλαδεμένο. Στην περίπτωση που τα εν λόγω παιδιά είναι φύλλα και στους δύο γράφους, η προσπάθεια του επιτιθέμενου δεν χρειάζεται να πάει παρακάτω, επειδή όποια κι αν είναι τα κλαδεμένα παιδιά δεν υπάρχει τρόπος να βρεθεί ποιο είναι ποιο σ' αυτά που απομένουν. Ωστόσο αν όλα έχουν απογόνους, πράγμα που αποτελεί το χειρότερο σενάριο για τον επιτιθέμενο, θα προκύψουν $\binom{N}{n}$ διαφορετικοί κλαδεμένοι γράφοι.

Στη συνέχεια ας υποθέσουμε ότι στον αρχικό (πραγματικό ή ανωνυμοποιημένο) γράφο υπάρχουν K ζευγάρια με παιδιά περισσότερα από n, συγκεκριμένα με αριθμό παιδιών N_1, N_2, …, N_k, τότε από το i-οστό ζευγάρι θα προκύψουν $\varphi_i=\binom{N_i}{n}$ διαφορετικές n-άδες παιδιών, οπότε όλοι οι γράφοι θα μπορούσαν να είναι $\varphi=\varphi_1\varphi_2\ldots\varphi_N$. Ωστόσο ο αριθμός αυτός πρέπει να νοηθεί ως

άνω φράγμα, επειδή κάποια παιδιά θα είναι «φύλλα» κι επειδή γενικότερα ο υπολογισμός του πλήθους των ισόμορφων υπογράφων ενός γράφου με δεδομένους περιορισμούς είναι ένα πιο πολύπλοκο πρόβλημα.

Στη συνέχεια δίνουμε ενδεικτικά νούμερα για τον γράφο που έχουμε χρησιμοποιήσει στις προσομοιώσεις.

Στον πίνακα παρουσιάζονται τα αποτελέσματα για το πόσες οικογένειες έχουν n=0,…,16 παιδιά.

n=1	450
n=2	145
n=3	103
n=4	82
n=5	53
n=6	51
n=7	35
n=8	23
n=9	19
n=10	10
n=11	11
n=12	3
n=13	3
n=14	3
n=15	1
n=16	0

Πίνακας 16: Αντιστοίχηση πλήθους παιδιών ανά αριθμό οικογενειών

Υπολογισμός φ:

Συνδυασμοί: $\binom{5}{3} = \frac{5!}{3!(5-3)!}$

Για n=14

φ = $\binom{15}{14}$ = 15

Για n=13

φ = $\binom{15}{13} * \binom{14}{13} * \binom{14}{13} * \binom{14}{13}$ = 288120

Για n=12

$\varphi = \binom{15}{12} * \binom{14}{12} * \binom{14}{12} * \binom{14}{12} * \binom{13}{12} * \binom{13}{12} * \binom{13}{12} = 753295946585.00$

…

<u>Αποτελέσματα για φ:</u>

Όπου υπάρχει η τιμή #NUM! τα νούμερα είναι πολύ μεγάλα και δεν μπορούν να υπολογιστούν.

	φ
n=1	#NUM!
n=2	#NUM!
n=3	#NUM!
n=4	#NUM!
n=5	6.4667E+250
n=6	4.9008E+184
n=7	4.4792E+131
n=8	5.11472E+90
n=9	2.17757E+58
n=10	5.77972E+36
n=11	5.39839E+19
n=12	7.53296E+11
n=13	288120
n=14	15
n=15	-
n=16	-

Πίνακας 17: Αποτελέσματα φ

7 Συμπεράσματα

7.1 Σύνοψη

Στα πλαίσια της διπλωματικής αυτής εργασίας, στόχος ήταν η εφαρμογή διαφόρων τεχνικών ανωνυμοποίησης σε ένα γενεαλογικό δέντρο 3619 ατόμων, το οποίο λήφθηκε με τη χρήση αλγορίθμων από ένα μεγαλύτερο έυρος δεδομένων των 86 εκατομμυρίων ατόμων, και η μετέπειτα προσπάθεια αποανωνυμοποίησης του με διάφορους ελέγχους. Θεωρήθηκε ότι τα άτομα του συγκεκριμένου δέντρου, είναι ο πληθυσμός μιας μικρής πόλης, ο οποίος είναι αποθηκευμένος σε μια γενεαλογική βάση. Θα μπορούσε για παράδειγμα να ήταν η βάση δεδομένων ενός τοπικού νοσοκομείου. Στην βάση αυτή, περιλαμβάνονται μοναδικά (id) και μη (sex, birth year, etc.) χαρακτηριστικά για το κάθε άτομο της πόλης αυτής.

Η ανωνυμοποίηση σε πρώτο στάδιο έγινε με την απλή υπόθεση ότι το μοναδικό αυτό χαρακτηριστικό id του κάθε ατόμου, έχει διαγραφεί. Η μετέπειτα αποανωνυμοποίηση ξεκίνησε με αρχικό πρόσωπο τον κάτοικο με τα περισσότερα παιδιά , 16 σε αριθμό. Λόγω του ότι τα 3619 άτομα είναι σχετικά μικρός αριθμός για μια πόλη, θεωρήθηκε λογικό πως οι κάτοικοι λίγο εώς πολύ, γνωρίζονται μεταξύ τους άρα γνωρίζουν και αυτόν με τα περισσότερα παιδιά. Όταν γίνεται γνωστό ένα άτομο, αυτόματα γίνονται και οι γονείς του σε ένα γενεαλογικό δέντρο. Παρ'όλα αυτά, για παιδιά και συζύγους δεν ισχύει το ίδιο γι'αυτό και εφαρμόστηκαν περαιτέρω ελέγχοι (κεφάλαιο 6, υποενότητα 6.2). Οι διάφοροι ελέγχοι για το συγκεκριμένο πρόσωπο, βοήθησαν στην αποανωνυμοποίηση αρκετών από τους προγόνους και απογόνους του, και η μετέπειτα εφαρμογή των ελέγχων πάνω σε αυτούς βοήθησε στην αποανωνυμοποίηση περισσότερων κ.ο.κ. Τελικά, αποανωνυμοποιήθηκαν 1739 άτομα από τους 3619 συνολικά. Δηλαδή, το ποσοστό επιτυχίας ήταν 48% και κάτω από τους μισούς κατάφεραν να ταυτοποιηθούν.

Μετά από αυτή την απλή τεχνική ανωνυμοποίησης η οποία συνεχίζει να εφαρμόζεται και στα επόμενα στάδια, εφαρμόστηκαν και τρεις άλλες:
1. Ξεκινώντας από ένα τυχαίο άτομο του συνολικού δέντρου, από κάθε άτομο που έχει αριθμό παιδιών μεγαλύτερο του n, όπου n οποιοσδήποτε αριθμός μεταξύ 1 και 16, κόβονται τυχαία κάποια από τα παιδιά του ώστε στο τέλος να παραμείνουν n. Κόβονται δηλαδή οι ακμές που ενώνουν το «προς διαγραφή παιδί» με τους γονείς του, οι ακμές που τον ενώνουν με τα δικά του παιδιά αλλά και ο ίδιος ο κόμβος.
2. Με την ίδια λογική όπως στην πρώτη, δεν διαγράφονται μόνο τα παιδιά που επιλέγονται τυχαία, αλλά και όλοι οι απογόνοι τους.
3. Τέλος, η τρίτη τεχνική μοιάζει με την πρώτη, με την διαφορά ότι εδώ τα παιδιά παραμένουν ως κόμβοι στο δέντρο και κόβονται μόνο οι απαραίτητες ακμές.

Στις τρεις αυτές τεχνικές, για να ξεκινήσει η αποανωνυμοποίηση, θεωρήθηκε ότι ήταν γνωστό ένα αρχικό άτομο Χ που μπορούσε να ήταν οποιοδήποτε άτομο του «επεξεργασμένου-κλαδεμένου» δέντρου. Όπως ήταν αναμενόμενο, όσο μεγαλύτερο το n τόσο περισσότερες

ταυτοποιήσεις γίνονταν και συγκεκριμένα για n ≥10 το πλήθος των αποανωνυμοποιημένων ήταν αρκετά κοντά στον αριθμό 1739.
Παρ'όλ'αυτά, με τις παραπάνω ανωνυμοποιήσεις στον πραγματικό κόσμο, δηλαδή σε αυτή την μικρή πόλη των 3619 ατόμων, θα ήταν αδύνατο για τον επιτιθέμενο να βρει εύκολα ένα αρχικό γνωστό άτομο στο κλαδεμένο δέντρο. Εάν όμως υπήρχε το σχεδόν απίθανο σενάριο όπου θα «αναγνώριζε» κάποιον από αυτούς, τότε όπως αναφέρθηκε πιο πάνω, για μεγάλα n ταυτοποιείται ένα μεγάλο μέρος του πληθυσμού γύρω στο 40-50% των αρχικών ατόμων.

7.2 Ανοικτά Ζητήματα

Μια σημαντική βελτίωση στον αλγόριθμο αποανωνυμοποίησης της παρούσας διπλωματικής θα ήταν η χρήση μεθόδων βασισμένων στον ισομορφισμό.
Για παράδειγμα, η χρήση μιας επιπλέον συνάρτησης μέσα στον αλγόριθμο, που εκτός από τους προαναφερθέντες ελέγχους, θα μπορούσε να συγκρίνει το κλαδεμένο με το αρχικό δέντρο θα μπορούσε να οδηγήσει και στην περαιτέρω ταυτοποίηση περισσοτέρων ατόμων. Εάν, ένα κομμάτι του αρχικού δέντρου ήταν ίδιο η ελαφρώς αλλοιωμένο με κάποιο του κλαδεμένου δέντρου τότε θα υπήρχε λόγος για πιθανόν ταυτοποίηση της ρίζας αυτού του κομματιού και μετέπειτα των απογόνων της. Αυτό, θα μπορούσε επίσης να χρησιμεύσει στο να βρεθεί ένας αρχικός κόμβος με εφαρμογή μόνο τεχνικών αποανωνυμοποίησης και να μην επιλέγεται τυχαία ένα οποιοδήποτε άτομο του δέντρου, κάτι που θα ήταν πιο κόντα στην πραγματικότητα στην συγκεκριμένη περίπτωση.

Επίσης, θα μπορούσαν να χρησιμοποιηθούν άλλες μεθόδοι κλαδέματος του γενεαλογικού δέντρου, είτε περισσότερο δραστικές είτε λιγότερο, και να μελετηθεί η δική τους απόδοση όσον αφορά την επίθεση πάνω σε αυτές με τεχνικές αποανωνυμοποίησης.

Μια επιπλέον προσθήκη στα προηγούμενα, θα ήταν σε κάθε στάδιο ανωνυμοποίησης να λαμβάνονται και τα στατιστικά αποτελέσματα για αυτό. Να φαίνεται δηλαδή η επιρροή της κάθε τεχνικής πάνω στα ψευδοαναγνωριστικά της βάσης και πάνω στο κλαδεμένο γράφο (στους κόμβους και ακμές του) γενικότερα.

7.3 Επίλογος

Η έρευνα για τη διπλωματική αυτή οδήγησε στην εξαγωγή σημαντικών συμπερασμάτων αλλά και πληροφοριών «μελλοντικά επεξεργάσιμων». Το αντικείμενο της, άκρως σημαντικό στην σύγχρονη εποχή βοήθησε στο να επιλεχθεί εξαρχής αφού κεντρίζει το ενδιαφέρον του καθενός από εμάς. Η ανωνυμοποίηση πρέπει να αποτελέσει σημαντική προσθήκη στο σήμερα εώς ότου να μπορούμε να ζούμε σε μια κοινωνία όπου δεν υπάρχει διαρροή πληροφοριών για εκπλήρωση προσωπικών σκοπών και η προστασία των προσωπικών δεδομένων του κάθε ατόμου να τυγχάνει της απαραίτητης προσοχής. Καθημερινά όλο και διαφορετικές τεχνικές αποανωνυμοποίησης πιθανόν να αναπτύσσονται κάποιες αποδοτικές και κάποιες όχι.
Παρ'όλ'αυτά, όπως φάνηκε, ακόμη και οι πιο απλές τεχνικές ανωνυμοποίησης μπορούν να φανούν χρήσιμες έστω και αν δεν αποδίδουν στον μέγιστο βαθμό όταν επέμβει ο επιτιθέμενος. Κλείνοντας, κατά τη διάρκεια εκπόνησης της διπλωματικής, μελετήθηκαν χρήσιμες έννοιες και τεχνικές που θα αποτελέσουν σημαντικά προσωπικά εφόδια για το μέλλον.

Παράρτημα Ι

Προγράμματα σε Python

Όλα τα παρακάτω προγράμματα είναι πρωτότυπα.

1) FindTree.py

```
1. #function that returns the parent-child relation written in a file
2. def family(k,co0,co1,a):
3.     #a is a list with the ids i've checked
4.     #k is a list with the ids i will check
5.     a.append(k[0])
6.     #open my output file in append mode
7.     fout=open("id29.txt", "a")
8.     #search for all the parent-child
9. relations for the first element of k list (k[0])
10.          for line in range(0,len(co0)):
11.              #if the id i want to check is a parent and
12.      his child is not in the ids(list a) i've already checked
13.              if (k[0]==co0[line] and co1[line] not in a):
14.                  #write the parent-child relation in the output file
15.                  fout.write(str(co0[line]))
16.                  fout.write(" ")
17.                  fout.write(str(co1[line]))
18.                  fout.write('\n')
19.                  if (co1[line] not in k):
20.                      k.append(co1[line])
21.              #if the id i want to check is a child and
22.       his parent is not in the ids(list a) i've already checked

23.              elif (k[0]==co1[line] and co0[line] not in a):
24.                  fout.write(str(co0[line]))
25.                  fout.write(" ")
26.                  fout.write(str(co1[line]))
27.                  fout.write('\n')
28.                  if (co0[line] not in k):
29.                      k.append(co0[line])
30.
31.          #delete the k[0] element = delete the id i've checked
32.          del k[0]
33.          return k
34.          return a
35.
36.      fin=open("relations-anon.txt", "r")
37.      fout=open("id29.txt", "w")
38.      c0=[]
39.      c1=[]
40.      with fin:
41.          for i in fin:
42.              c0.append(i.split()[0])
43.              c1.append(i.split()[1])
```

```
44.       #co0: left column in the relations-anon file (parent id)
45.       co0=list(map(int,c0))
46.       #co1: right column in the relations-anon file (child id)
47.       co1=list(map(int,c1))
48.       #s=int(input("Enter a parent id:"))
49.       #k=[s]-->faster when i put the id directly k=[34]
50.
51.       #k=[the id for whom i want to find his family tree]
52.       #in our research we use the id 2
53.       k=[29]
54.       #a is a list that will be holding the ids i'm gonna check
55.       a=[]
56.       #calling the function
57.       family(k,co0,co1,a)
58.
59.       #keep checking for family members until k list is empty
60.       for i in range(0,len(co0)):
61.           if (k!=[] and k[0] not in a):
62.               family(k,co0,co1,a)
63.
64.       print("family tree ready")
65.       fin.close()
66.       fout.close()
```

Περιγραφή Findtree.py:

Αρχικά, ανοίγει το αρχείο relations-anon.txt ως fin, σε μορφή ανάγνωσης "r" **(γραμμή 33)**. Έπειτα, ανοίγει το αρχείο που θα περιέχει τις σχέσεις πατέρα-παιδιού για το γενεαλογικό δέντρο, ως fout σε μορφή εγγραφής "w" **(γραμμή 34)**. Δημιουργούνται 2 λίστες c0,c1 **(γραμμες 35-36)** και χρησιμοποιώντας το αρχείο fin **(γραμμή 37)** , διατρέχοντας το, για κάθε γραμμή του αρχείου **(γραμμή 38)** αποθηκεύεται στην λίστα c0 ο πατέρας, δηλαδή την αριστερή στήλη του relations-anon.txt **(γραμμή 39)** και στην λίστα c1 το αντίστοιχο παιδί του, δηλαδή την δεξιά στήλη **(γραμμή 40)**. Μετατρέπονται οι δύο λίστες σε λίστες ακεραίων co0 και co1 αντίστοιχα, για πιο εύκολη χρήση τους **(γραμμές 42+44)**. Δημιουργείται μια λίστα k η οποία περιέχει αρχικά τον αριθμό (id) του ατόμου γύρω από το οποίο επιθυμείται να «κτιστεί» το γενεαλογικό του δέντρο με τους προγόνους και απογόνους του. Μετά από αρκετές δοκιμές με διάφορα id, το άτομο με id=29 λόγω του μεγάλου πλήθους συγγενικών σχέσεων που βρέθηκαν μέσω του προγράμματος **(γραμμή 50)** είναι και αυτό που θα χρησιμοποιηθεί ως αρχικό πρόσωπο. Επίσης, δημιουργείται ακόμη μια λίστα a **(γραμμή 52)**, η χρήση της οποίας θα περιγραφεί πιο κάτω. Στη συνέχεια, καλείται η συνάρτηση family με ορίσματα τα k, co0,co1 και a **(γραμμή 54)**. Προχωρώντας τώρα στη συνάρτηση **(γραμμή 2)**, προστίθεται το πρώτο στοιχείο της λίστας k, στη συγκεκριμένη περίπτωση τον αριθμό 29, στην λίστα a **(γραμμή 5)**. Στην λίστα αυτή θα αποθηκεύονται οι απόγονοι και πρόγονοι του 29, για να εξεταστούν στη συνέχεια και να βρεθούν οι δικοί τους απόγονοι και πρόγονοι , κ.ο.κ εώς ότου συμπληρωθεί το γενεαλογικό δέντρο και θα χρησιμοποιείται επίσης για να αποφευχθεί η τυχόν επανάληψη κάποιας σχέσης στο τελικό αρχείο. Με την χρήση λοιπόν της a, κάθε σχέση πατέρα-παιδιού θα γράφεται μόνο μια φορά στο τελικό αρχείο. Συνεχίζοντας, διατρέχονται οι λίστες co0,co1 (**γραμμή 9**- χρειάζεται μόνο η δήλωση της μιας εκ των 2 αφου το εύρος τιμών τους είναι ίδιο), και γίνονται οι παρακάτω ελέγχους:

- **Γραμμή 11**: Εάν, το πρώτο στοιχείο της λίστας k, εδώ το 29, βρίσκεται στην λίστα co0 (δηλαδή συμπεριφέρεται ως γονιός) , και το αντίστοιχο στοιχείο/παιδί της λίστας co1 δεν

υπάρχει στην λίστα a, δηλαδή δεν έχουν ήδη βρεθεί και γραφεί στο αρχείο οι συγγενικές του σχέσεις, τότε γράφεται στο αρχείο εξόδου (id29.txt) ο γονιός **(γραμμή 13)** και δίπλα το παιδί **(γραμμή 15)**. Επίσης, εάν το παιδί δεν βρίσκεται στην λίστα k, προστίθεται για να βρεθούν στη συνέχεια οι δικοί του απόγονοι/πρόγονοι **(γραμμές 17-18)**.

- **Γραμμή 20**: Αντίστοιχα με πριν, ελέγχεται εάν, το πρώτο στοιχείο της λίστας k, εδώ το 29, βρίσκεται στην λίστα co1 (δηλαδή συμπεριφέρεται ως παιδί) , και το αντίστοιχο στοιχείο/γονιός της λίστας co0 δεν υπάρχει στην λίστα a, δηλαδή δεν έχουν ήδη βρεθεί και γραφεί στο αρχείο οι συγγενικές του σχέσεις, τότε γράφεται στο αρχείο εξόδου (id29.txt) ο γονιός **(γραμμή 21)** και δίπλα το παιδί **(γραμμή 23)**. Επίσης, εάν ο γονιός δεν βρίσκεται στην λίστα k, τότε προστίθεται για να βρεθούν στη συνέχεια οι δικοί του απόγονοι/πρόγονοι **(γραμμές 25-26)**.

Με λίγα λόγια, με αυτό τον τρόπο, θα γίνει ο βρόγχος for για όλα τα στοιχεία των co0,co1 και όπου βρίσκεται το στοιχείο 29 σαν γονιός/παιδί, θα γράφεται στο αρχείο μαζί με τον αντίστοιχο παιδί/γονιό του εώς ότου βρεθούν όλες οι σχέσεις που περιλαμβάνουν τον 29.
Στη συνέχεια, θα διαγραφεί το άτομο 29 από την λίστα k **(γραμμή 29)** και θα επιστραφούν οι νέες λίστες k,a **(γραμμές 30-31)**. Επιστρέφοντας στο κυρίως πρόγραμμα, εφόσων η λίστα k δεν είναι κενή και το πρώτο της στοιχείο δεν βρίσκεται ήδη στην λίστα a **(γραμμή 58)** ξανακαλείται η συνάρτηση **(γραμμή 59)**.
Εάν η πρώτη σχέση που είχε βρεθεί και γραφεί στο αρχείο πριν ήταν για παράδειγμα η 29(γονιός) 879065(παιδί), τώρα θα βρεθούν οι απόγονοι/πρόγονοι του 879065. Έτσι δημιουργούνται όλες οι σχέσεις που χρειάζονται για το γενεαλογικό δέντρο, και αποθηκεύονται στο αρχείο “id29.txt”.

2) Numberofpeople.py

```
1. fin=open("id29.txt","r")
2. fout=open("id29people.txt","w")
3. text=fin.read().split()
4. ftree=list(map(int,text))
5. counter=0
6. id=[]
7.
8. for i in range(0,len(ftree)):
9.     if (ftree[i] not in id):
10.             id.append(ftree[i])
11.             counter+=1
12.       id.sort()
13.
14.       for r in range(0,len(id)):
15.           fout.write(str(id[r]))
16.           fout.write('\n')
17.
18.       print("the number of people in id29.txt is:",counter)
19.
20.       fin.close()
21.       fout.close()
```

Περιγραφή Numberofpeople.py:

Το παραπάνω πρόγραμμα, χρησιμοποιώντας το αρχείο που δημιουργήθηκε πριν (id29.txt) το διαβάζει **(γραμμή 1)** , παίρνει το κάθε id από το αρχείο ξεχωριστά **(γραμμή 3)** και δημιουργεί μια λίστα ακεραίων ftree με όλα τα id που περιέχονται στο αρχείο **(γραμμή 4)**. Δηλώνει ένα μετρητή counter αρχικοποιημένο στο 0 που θα αποθηκεύσει το πλήθος των ατόμων που αποτελούν το δέντρο **(γραμμή 5)** και μια **λίστα** id που θα αποθηκεύει τα άτομα αυτά **(γραμμή 6)**. Διασχίζοντας την λίστα ftree **(γραμμή 8)**, εάν το άτομο δεν υπάρχει ήδη στην λίστα id **(γραμμή 9)** τότε προστίθεται **(γραμμή 10)** και ο μετρητής αυξάνεται κατά 1 **(γραμμή 11)**, αλλιώς προχωρά στο επόμενο στοιχείο της λίστας. Η λίστα ταξινομείται **(γραμμή 12)** και τυπώνεται στο αρχείο "id29people.txt" **(γραμμές 14-16).** Στην κονσόλα τυπώνεται ο μετρητής **(γραμμή 18)**.

3) Findprofiles.py

Βρίσκει τα 75 πεδία/ιδιότητες για το καθένα εκ των 3619 ατόμων του γενεαλογικού δέντρου και τα τυπώνει στο αρχείο profiles.txt

```
1. import glob
2. for filename in glob.glob('x000*'):
3.     #open the file with the final parent-child relations
4.     fin=open("id29people.txt","r")
5.     #open all splitted files with profile's data one at a time
6.     fpro=open(filename, encoding="utf8")
7.     #output file that contains data for all the profiles that we need
8.     fout=open("profiles.txt", "a" ,encoding="utf8")
9.
10.          text=fin.read().split()
11.          ftree=list(map(int,text))
12.
13.          #id2 is a list with all the ids
14.          id2=[]
15.          with fpro:
16.              for line in fpro:
17.                  id2.append(line.split()[0])
18.
19.          id=list(map(int,id2))
20.
21.          fpro=open(filename, encoding="utf8")
22.          tfpro=fpro.readlines()
23.
24.          for k in range(0,len(ftree)):
25.              for j in range(0,len(id)):
26.                  #if we find matching id
27.                  if (id[j]==ftree[k]):
28.                      #write the data of the specific id
29.                      fout.write(tfpro[j])
30.
31.          fin.close()
32.          fout.close()
33.          fpro.close()
```

Περιγραφή Findprofiles.py

Όπως αναφέρθηκε στην παράγραφο 3.3, χρησιμοποιήθηκε το GitBash για το σπάσιμο του αρχείου "profiles-anon.txt" σε 86 μικρότερα με ονόματα x00000, x000001,x00002…x00086. Στο παραπάνω πρόγραμμα, ανοίγεται ένα-ένα το καθένα εκ των 86 αρχείων που δημιουργήθηκαν **(γραμμή 2)** . Χρησιμοποιώντας τα μοναδικά id, που βρέθηκαν και γράφηκαν στο αρχείο "id29people.txt" **(γραμμή 4)** , προστίθενται σε μια λίστα ακεραίων ftree **(γραμμές 10-11)**. Στη συνέχεια δημιουργείται μια λίστα id2 **(γραμμή 14)**, στην οποία αποθηκεύεται η πρώτη στήλη του αρχείου x000*, δηλαδή ο μοναδικός αριθμός του κάθε ατόμου **(γραμμές 15-17)**. Μετατρέπεται η λίστα id2 σε λίστα ακεραίων ονόματος id **(γραμμή 19)**. Ανοίγει το αρχείο x000* **(γραμμή 21)** και σε μια μεταβλητή tfpro τοποθετούνται οι γραμμές του αρχείου **(γραμμή 22)**. Διασχίζοντας τις 2 λίστες ftree και id **(γραμμές 24-25)**, εάν βρεθεί id που υπάρχει και στις 2 **(γραμμή 27)**, τότε τυπώνεται η αντίστοιχη γραμμή από το αρχείο x000* που περιέχει όλες τις πληροφορίες (δηλαδή και τα 75 πεδία) του συγκεκριμένου ατόμου **(γραμμή 29)**.

4) Tags.py

```
1. fin=open("profiles.txt","r")
2.
3. gender=[]
4. byear2=[]
5. dyear2=[]
6. birthloc=[]
7.
8. with fin:
9.     for line in fin:
10.               gender.append(line.split()[1])
11.               byear2.append(line.split()[14])
12.               dyear2.append(line.split()[29])
13.               birthloc.append(line.split()[9])
14.
15.  byear=list(map(int,byear2))
16.  dyear=list(map(int,dyear2))
17.
18.  fout=open("info.txt","w")
19.
20.  for i in range(0,3618):
21.      fout.write(str(gender[i]))
22.      fout.write("")
23.      fout.write(str(byear[i]))
24.      fout.write("")
25.      fout.write(str(dyear[i]))
26.      fout.write("")
27.      fout.write(str(birthloc[i]))
28.      fout.write('\n')
29.
30.  fin.close()
31.  fout.close()
```

Περιγραφή προγράμματος Tags.py:

Το παραπάνω πρόγραμμα, ανοίγει το αρχείο "profiles.txt" **(γραμμή 1)**. Δημιουργεί 4 λίστες gender, byear2, dyear2, birthloc **(γραμμές 3-6)** στις οποίες θα αποθηκευτούν τα στοιχεία gender (φύλο), Birth Year (χρονολογία γέννησης) , Death Year (χρονολογία θανάτου) , Birth Location (τοποθεσία γέννησης) αντίστοιχα. Με τη χρήση της εντολής split και των αριθμών των πεδίων που χρειάζονται, βρίσκονται οι τιμές των επιθυμητών ετικετών και τυπώνονται στο αρχείο εξόδου με όνομα info.txt.

Από το αρχείο "profiles-field-list.txt" οι αντιστοιχίες για τα επιθυμητά πεδία είναι:
Gender – πεδίο 2 => $2^{η}$ στήλη στο αρχείο profiles.txt
Birth Year - πεδίο 15 => $15^{η}$ στήλη στο αρχείο profiles.txt
Death Year - πεδίο 30 => $30^{η}$ στήλη στο αρχείο profiles.txt
Birth Location - πεδίο 10 => $10^{η}$ στήλη στο αρχείο profiles.txt

- **SEX**
  Η τιμή της ετικέτας φύλου βρίσκεται στην **γραμμή 10** όπου με την εντολή split επιλέγονται οι τιμές της $2^{ης}$ στήλης (i=0,1,… άρα ο αριθμός 1 αντιπροσωπεύει την $2^{η}$ στήλη)
  και η στήλη αυτή τυπώνεται στο αρχείο εξόδου (γραμμή 21).

- **BIRT**
  Δήλωση γέννησης, η χρονολογία και ημερομηνία της οποίας θα φανούν στις επόμενες 2 ετικέτες.

- **DATE**
  Η τιμή της ετικέτας «ημερομηνία γέννησης» βρίσκεται στην **γραμμή 11** όπου με την εντολή split επιλέγονται οι τιμές της $15^{ης}$ στήλης και τυπώνονται στο αρχείο εξόδου (γραμμή 23).

- **PLAC**
  Η τιμή της ετικέτας «τοποθεσία» βρίσκεται στην **γραμμή 13** όπου με την εντολή split επιλέγονται οι τιμές της $10^{ης}$ στήλης και τυπώνονται στο αρχείο εξόδου (γραμμή 27).

- **DEAT**
  Δήλωση θανάτου, η χρονολογία του οποίου θα φανεί στην επόμενη ετικέτα.

- **DATE**
  Η τιμή της ετικέτας «τοποθεσία» βρίσκεται στην **γραμμή 12** όπου με την εντολή split επιλέγονται οι τιμές της $30^{ης}$ στήλης και τυπώνονται στο αρχείο εξόδου (γραμμή 25).

5) CreateGed.py

```
1. with open('id.txt', 'r') as file1, open('husb.txt', 'r') as file2, open
   ('wife.txt','r') as file3,open('famc.txt', 'r') as file4, open('sex.txt
   ', 'r') as file5, open('byear.txt','r') as file6, open('dyear.txt', 'r'
   ) as file7, open('birthloc.txt','r') as file8:
2.     id=list(map(int,file1.read().split()))
3.     husb=list(map(int,file2.read().split()))
4.     wife=list(map(int,file3.read().split()))
5.     famc=list(file4.read().split())
6.     sex=list(file5.read().split())
7.     byear=list(file6.read().split())
8.     dyear=list(file7.read().split())
9.     birthloc=list(file8.read().split())
10.
11.       fout=open("Family.ged","w")
12.       j=0
13.
14.       fout.write("0 HEAD\n1 GEDC\n2 VERS 5.5.1\n2 FORM LINEAGE-
   LINKED\n1 CHAR UTF-8\n")
15.
16.       for i in range(0,3619):
17.           fout.write("0 @I%s@ INDI\n1 NAME %s //\n" % (i,id[i]))
18.           if(sex[i]!="*"):
19.               fout.write("1 SEX %s\n" % (sex[i]))
20.           if(byear[i]!="*" or birthloc[i]!="*"):
21.               fout.write("1 BIRT\n")
22.           if(byear[i]!="*"):
23.               fout.write("2 DATE %s\n" % (byear[i]))
24.           if(birthloc[i]!="*"):
25.               fout.write("2 PLAC %s\n" % (birthloc[i]))
26.           if(dyear[i]!="*"):
27.               fout.write("1 DEAT \n2 DATE %s\n" % (dyear[i]))
28.           fout.write("1 FAMS @")
29.           for t in range(0,len(famc)):
30.               if(id[i]==husb[t] or id[i]==wife[t]):
31.                   fout.write(str(famc[t]))
32.                   break
33.           fout.write("@\n")
34.           if(i<=2902):
35.               fout.write("1 FAMC @%s@\n" % (famc[j]))
36.               j+=1
37.       #opou elegxw gia =1 (p.x line 65,70)
38.       #einai ekei pou den iparxei pateras i mitera
39.       #apla to ethesa me ton arithmo 1 gia eukolia
40.       #sto perasma twn dedomenwn se listes
41.       fc=[]
42.       for k in range(0,len(famc)):
43.           if(famc[k] not in fc):
44.               fout.write("0 @%s@ FAM\n" % (famc[k]))
45.               fc.append(famc[k])
46.               for m in range(0,len(id)):
47.                   if(husb[k]!=1 and husb[k]==id[m]):
48.                       fout.write("1 HUSB @I%s@\n" % (m))
49.                   if(wife[k]!=1 and wife[k]==id[m]):
50.                       fout.write("1 WIFE @I%s@\n" % (m))
```

```
51.                 for n in range(0,len(famc)):
52.                     if(famc[k]==famc[n]):
53.                         fout.write("1 CHIL @I%s@\n" % (n))
54.
55.         fout.write("0 TRLR")
```

Περιγραφή του CreadeGed.py

Άνοιγμα και διάβασμα αρχείων (γραμμές 1-11):

Το πρόγραμμα ξεκινά διαβάζοντας κάποια αρχεία εισόδου τα οποία είναι τα εξής:
id.txt - το αρχείο με το μοναδικό αριθμό (id) του κάθε ατόμου του δέντρου
famc.txt - το αρχείο με τις οικογένειες "F0,F1,..." στις οποίες κάθε id συμπεριφέρεται ως παιδί
sex.txt - το αρχείο με το φύλο του κάθε ατόμου στο αρχείο id
birthloc.txt - το αρχείο με την τοποθεσία γέννησης του κάθε ατόμου(id)
byear.txt - το αρχείο με την ημερομηνία γέννησης του κάθε ατόμου(id)
dyear.txt - το αρχείο με την ημερομηνία θανάτου του κάθε ατόμου(id)
husb.txt - το αρχείο με όλους τους συζύγους γένους αρσενικού
wife.txt - το αρχείο με όλους τους συζύγους γένους θηλυκού
Καθώς και το αρχείο εξόδου στο οποίο θα τυπωθεί το Gedcom αρχείο: Family.ged

Έπειτα γράφονται στο αρχείο εξόδου τα δεδομένα στοιχεία από το Gedcom Standard όπως οι ετικέτες:
HEAD,GEDC,VERS,FORM,CHAR **(γραμμή 14)**

1ο κομμάτι ενός Gedcom αρχείου - οι πληροφορίες για κάθε άτομο ξεχωριστά (γραμμές 16-36):

Με τη βοήθεια ενός μετρητή i που το εύρος του είναι από 0 εώς 3619 που είναι τα συνολικά άτομα του δέντρου, αρχίζει να τυπώνεται στο αρχείο το κάθε άτομο ξεχωριστά μαζί με της πληροφορίες του:
Αρχίκα τυπώνεται ο μοναδικός αύξων αριθμός του και το όνομα του ή αλλιώς το id του **(γραμμή 17)** και στη συνέχεια ελέγχεται εάν υπάρχει τιμή για το φύλο του (εάν δηλαδή δεν είναι ίση με "*") και τυπώνεται η αντίστοιχη τιμή **(γραμμές 18-19)**.
Με τον ίδιο τρόπο τυπώνονται οι ημερομηνίες γέννησης **(γραμμές 20-23)**, τοποθεσία γέννησης **(γραμμές 24-25)** και ημερομηνία θανάτου **(γραμμές 26-27)**.
Έπειτα, διατρέχοντας την λίστα famc, εάν βρεθεί πατέρας (husb) ή μητέρα (wife) που είναι ίση με το id του τρέχον ατόμου, τότε η τιμή του FAMS είναι ίση με την τιμή FAMC του συγκεκριμένου husb ή wife **(γραμμές 28-33)**.
Όσον αφορά την ετικέτα FAMC, επειδή μόνο τα πρώτα 2902 από τα συνολικά άτομα είναι παιδιά σε μια οικογένεια, χρησιμοποιώντας ένα σταθερό μετρητή i και έναν μεταβαλλόμενο j τυπώνεται η τιμή FAMC για αυτά τα άτομα **(γραμμές 34-36)**.

<u>2ο κομμάτι του Gedcom αρχείου - οι πληροφορίες για κάθε οικογένεια ξεχωριστά (γραμμές 41-55) :</u>

Διατρέχοντας την λίστα famc,εφόσων η τιμή famc είναι καινούρια και δεν έχει ήδη εξεταστεί **(γραμμές 41-45)**, αρχίζοντας από την πρώτη τιμή "F0" η τιμή αυτή τυπώνεται **(γραμμή 44)** και προστίθεται σε μια λίστα fc **(γραμμή 45)** στην οποία θα κρατιούνται οι μοναδικές τιμές των famc για να μην γίνει τυχών επανάληψη τους και να τυπωθούν περισσότερες από μια φορές στο τελικό αρχείο.
Έπειτα, διατρέχοντας την λίστα id **(γραμμή 46)**, εάν βρεθεί id που είναι ίσο με τον σύζυγο-πατέρα της τρέχουσας οικογένειας famc τότε τυπώνεται ο αύξων αριθμός του συζύγου **(γραμμές 47-48)**.
Με αντίστοιχο τρόπο, εάν βρεθεί η σύζυγος-μητέρα της τρέχουσας οικογένειας famc, τυπώνεται ο άυξων αριθμός της **(γραμμές 49-50)**.
Τέλος, διατρέχοντας παράλληλα 2η φορά την λίστα famc, όπου βρίσκεται τιμή famc ίση με την τρέχουσα τιμή famc, τυπώνεται ο αντίστοιχος αύξων αριθμός ως παιδί της οικογένειας αυτής **(γραμμές 51-53)**.
Τελευταία τυπώνεται και η ετικέτα TRLR που υποδηλώνει το τέλος του αρχείου Gedcom **(γραμμή 55).**

6) DeleteNodes.py

```
1. def family(r,q,wife,husb,name,checked,n,deleted,notq):
2.     checked.append(r[0])
3.     fout.write("Person to check: %s\n" % (r[0]))
4.
5.     #Find r[0] parents and put them in the lists
6.     for j in range(0,len(name)):
7.         if (r[0]==name[j]):
8.             if (husb[j]!=1 and husb[j] not in q and husb[j] not in notq
   ):
9.                 fout.write("Father: %s\n" % (husb[j]))
10.                    q.append(husb[j])
11.                    r.append(husb[j])
12.                if (wife[j]!=1 and wife[j] not in q and wife[j] not i
   n notq):
13.                    fout.write("Mother: %s\n" % (wife[j]))
14.                    q.append(wife[j])
15.                    r.append(wife[j])
16.
17.         #Put r[0]'s children in a list
18.         children=[]
19.         for i in range(0,len(husb)):
20.             if (r[0]==husb[i]):
21.                 children.append(name[i])
22.             elif (r[0]==wife[i]):
23.                 children.append(name[i])
24.         fout.write("\nNumber of Total Children: %s\n" % (len(children
   )))
25.
26.         #Check if children already identified or aldready deleted
27.         kk=0
```

```
            childrenold=children
            childreninq ,childreninnotq = ([] for i in range(2))
            for i in range(0,len(children)):
                if(children[i] in q):
                    kk+=1
                    childreninq.append(children[i])
                if(children[i] in notq):
                    childreninnotq.append(children[i])
            children=list((set(children)^set(childreninq))^set(childrenin
notq))

            # Delete Random Children
            import random
            if (len(childreninq)>=n):
                children = random.sample(childreninq, k=n)
            elif(len(children)>=n):
                if((n-kk)>0):
                    n=n-kk
                    children = random.sample(children, k=n)
                    children.extend(childreninq)
            else:
                children=list(set(childrenold)^set(childreninnotq))

            deleted = list((set(childrenold)^set(children))^set(childreni
nnotq))
            #Put deleted children in the notq list
            for i in range(0,len(deleted)):
                if(deleted[i] not in notq):
                    notq.append(deleted[i])
            #Put selected children in the identified list and list to che
ck later
            for i in range(0,len(children)):
                if(children[i] not in q and children[i] not in notq):
                    q.append(children[i])
                    r.append(children[i])

            del r[0]
            fout.write("Number of Children Selected: %s\nSelectedChildren
: %s\nDeleted Children: %s\n\nQ: %s\nR: %s\nNotQ: %s\n-----------------
--------\n" % (len(children),children,deleted,q,r,notq))

            return r,q,checked,deleted,notq

        # MAIN PROGRAM
        with open('id.txt', 'r') as file1, open('husb.txt', 'r') as file2
, open('wife.txt','r') as file3,open('children.txt', 'r') as file4, ope
n('childrenofchildren.txt', 'r') as file5, open('parent1.txt','r') as f
ile6, open('parent2.txt', 'r') as file7, open('howmany.txt','r') as fil
e8:
            name=list(map(int,file1.read().split()))
            husb=list(map(int,file2.read().split()))
            wife=list(map(int,file3.read().split()))
            id=list(map(int,file4.read().split()))
            howmanychild=list(map(int,file5.read().split()))
            parent1=list(map(int,file6.read().split()))
            parent2=list(map(int,file7.read().split()))
            howmany=list(map(int,file8.read().split()))
```

```
75.       fout=open("NewTree.txt","w")
76.       #increase recursion limit
77.       import sys
78.       sys.setrecursionlimit(10**6)
79.       # r: list with those to check, q: list with the identified people

80.       #r,q= ([29] for i in range(2))
81.       # checked: list with those who are checked, deleted: list with de
   leted children, notq: children that won't be in final tree
82.       checked, deleted, notq = ([] for i in range(3))
83.
84.       fresults=open("id29people.txt")
85.       fn=list(map(int,fresults.read().split()))
86.       import random
87.       person=random.sample(fn, k=1)
88.       r=person
89.       import copy
90.       q=copy.deepcopy(r)
91.
92.       #n=int(input("Enter number of children to delete: "))
93.       fnumber=open("C:/Users/Rebecca/Desktop/number.txt")
94.       nn=fnumber.read()
95.       n=int(nn)
96.       filename = 'C:/Users/Rebecca/Desktop/DEANON OVER N - Copy/Αποτελέ
   σματα/n%d.txt'%(n,)
97.
98.       #calling the function
99.       family(r,q,wife,husb,name,checked,n,deleted,notq)
100.      for i in range(0,len(name)):
101.          if (r!=[] and r[0] not in checked):
102.              family(r,q,wife,husb,name,checked,n,deleted,notq)
103.
104.      print("Number of people in new tree:", len(q))
105.      #fout.write("Number of people in new tree: %s\n" % (len(q)))
106.
107.      fout2=open("cut_tree.txt","w")
108.      for i in range(0,len(q)):
109.          fout2.write("%s\n" %(q[i]))
110.
111.      fn=open(filename,"a")
112.      fn.write(str(len(q)))
113.      fn.write('\n')
114.
115.      name2=copy.deepcopy(name)
116.      husb2=copy.deepcopy(husb)
117.      wife2=copy.deepcopy(wife)
118.      id2=copy.deepcopy(id)
119.      howmanychild2=copy.deepcopy(howmanychild)
120.      parent12=copy.deepcopy(parent1)
121.      parent22=copy.deepcopy(parent2)
122.      howmany2=copy.deepcopy(howmany)
123.
124.      for i in range(0,len(notq)):
125.          for j in range(0,len(name)):
126.              if(notq[i]==name[j]):
127.                  name2[j]=1
128.                  husb2[j]=1
```

```
129.                 wife2[j]=1
130.             if(notq[i]==id[j]):
131.                 id2[j]=1
132.                 howmanychild2[j]=0
133.         for j in range(0,len(parent1)):
134.             if(notq[i]==parent1[j]):
135.                 if (parent2[j]==1):
136.                     parent12[j]=1
137.                     howmany2[j]=0
138.                 else:
139.                     parent12[j]=1
140.             elif(notq[i]==parent2[j]):
141.                 if(parent1[j]==1):
142.                     parent22[j]=1
143.                     howmany2[j]=0
144.                 else:
145.                     parent22[j]=1
146.
147.     with open('id_2.txt', 'w') as file1, open('husb_2.txt', 'w') as f
    ile2, open('wife_2.txt','w') as file3,open('children_2.txt', 'w') as fi
    le4, open('childrenofchildren_2.txt', 'w') as file5, open('parent1_2.tx
    t','w') as file6, open('parent2_2.txt', 'w') as file7, open('howmany_2.
    txt','w') as file8:
148.         for i in range(0,len(name2)):
149.             file1.write("%s\n" % (name2[i]))
150.             file2.write("%s\n" % (husb2[i]))
151.             file3.write("%s\n" % (wife2[i]))
152.         for i in range(0,len(id2)):
153.             file4.write("%s\n" % (id2[i]))
154.             if(howmanychild2[i]>n):
155.                 file5.write("%s\n" % (n))
156.             else:
157.                 file5.write("%s\n" % (howmanychild2[i]))
158.         for i in range(0,len(parent12)):
159.             file6.write("%s\n" % (parent12[i]))
160.             file7.write("%s\n" % (parent22[i]))
161.             file8.write("%s\n" % (howmany2[i]))
```

Περιγραφή DeleteNodes.py :

Άνοιγμα και διάβασμα αρχείων (γραμμές 65-70):
Το πρόγραμμα ξεκινάει στην **γραμμή 65** όπου και διαβάζει τα παρακάτω αρχεία που θα χρησιμοποιήσουμε στο πρόγραμμα.
id.txt - το αρχείο με το μοναδικό αριθμό (id) του κάθε ατόμου του δέντρου
husb.txt - το αρχείο με όλους τους συζύγους γένους αρσενικού
wife.txt - το αρχείο με όλους τους συζύγους γένους θηλυκού
NewTree.txt - το αρχείο στο οποίο θα γραφούν τα αποτελέσματα
Τα παραπάνω δεδομένα αποθηκεύονται αντίστοιχα στις λίστες:
name,husb,wife,fout

Στις **γραμμές 73-74** αυξάνεται το όριο για αναδρομές

Έπειτα, δημιουργουνται 6 λίστες:
Λίστα r **(γραμμή 76)**: θα αποθηκεύονται τα id των ατόμων που θα πάνε για έλεγχο.
Λίστα q **(γραμμή 76)**: θα αποθηκεύονται τα id των ατόμων που έχουν ταυτοποιηθεί.
Λίστα checked **(γραμμή 78)** : θα αποθηκεύονται τα id των ατόμων που έχουν ελεγχτεί (δηλαδή όσα φεύγουν από την r έρχονται εδώ).
Λίστα deleted **(γραμμή 78)**: θα αποθηκεύονται τα id των παιδιών που διαγράφονται κάθε φορά
Λίστα notq **(γραμμή 78)**: θα αποθηκεύονται τα id όλων στο σύνολο παιδιών που έχουν διαγραφεί

Έπειτα από την κονσόλα ζητείται και διαβάζεται ο ακέραιος αριθμός ν **(γραμμή 80)**.

Στη συνέχεια, καλείται για πρώτη φορά η συνάρτηση newtree με ορίσματα τα: r,q,wife,husb,name,checked,n,deleted,notq **(γραμμή 84)**.

Συνάρτηση newtree (γραμμές 1-63):
Η παραπάνω συνάρτηση κάνει τις ακόλουθες διαδικασίες:
Ανοίγει το αρχείο NewTree.txt και γράφει μέσα το πρώτο άτομο της λίστας r, δηλαδή το άτομο από το οποίο θα αρχίσει μια σειρά ελέγχων για την επιτυχή ανωνυμοποίηση όσο το δυνατόν περισσότερων ατόμων και ταυτόχρονα προστίθεται στην λίστα checked **(γραμμές 2-3)**.

Τοποθέτηση πατέρα και μητέρας στο νέο δέντρο (γραμμές 5-15):
Διατρέχοντας την λίστα name με τα μοναδικά id, εάν βρεθεί name[j] το οποίο είναι ίδιο με το id του ατόμου που ελέγχεται στην παρούσα φάση, δηλαδή το πρώτο στοιχείο της λίστας r, "το r[0]" **(γραμμές 6-7)** τότε:
Εάν το αντίστοιχο στοιχείο της λίστας husb είναι διαφορετικό του 1 (υπενθυμίζεται ότι η τιμή 1 δίνεται σε ένα στοιχείο εφόσων δεν υπάρχει-δηλαδή εδώ ελέγχεται εάν υπάρχει ή όχι πατέρας) και δεν ανήκει στην λίστα q, δηλαδή δεν έχει ήδη ταυτοποιηθεί αλλά ούτε ανήκει στην λίστα με τα διεγραμμένα από το δέντρο άτομα notq **(γραμμή 8)** , τότε στο αρχείο εξόδου γράφεται το όνομα του πατέρα **(γραμμή 9)** και έπειτα το μοναδικό id του αποθηκεύεται στις λίστες q,r **(γραμμές 10-11)**.
Αντίστοιχη διαδικασία ακολουθείται για την εύρεση μητέρας με χρήση της λίστας wife **(γραμμές 12-15)**.

Επιλογή παιδιών (γραμμές 17-58):
Δημιουργείται μια λίστα children **(γραμμή 18**). Διατρέχοντας την λίστα husb , εάν βρεθεί στοιχείο της που να είναι ίδιο με το r[0] **(γραμμές 19-20)** σημαίνει οτι το συγκεκριμένο άτομο που ελέγχουμε είναι σύζυγος γένους αρσενικού και αυτόματα αποθηκεύονται στην λίστα children όλα τα παιδιά του **(γραμμή 21)** αλλιώς εάν βρεθεί στη λίστα wife τότε είναι γένους θηλυκού και πάλι αποθηκεύονται στη λίστα children όλα τα παιδιά της **(γραμμές 22-23)**.
Στο αρχείο εξόδου τυπώνεται το πλήθος των παιδιών **(γραμμή 24)**.
Έπειτα αρχικοποιούμε μια μεταβλητή nq=0 **(γραμμή 27)** η οποία θα μετρά το πλήθος των παραπάνω παιδιών που είναι ήδη στη λίστα q, και δημιουργούνται 3 λίστες:
Λίστα childrenold **(γραμμή 28)**: Αποθηκεύονται όλα τα παιδιά της λίστας children σαν backup
Λίστα childreninq **(γραμμή 29)**: Αποθηκεύονται όσα παιδιά της λίστας children είναι ήδη στο καινούριο δέντρο - βρίσκονται στην q **(γραμμές 30-33).**

Λίστα childreninnotq **(γραμμή 29)**: Αποθηκεύονται όσα παιδιά της λίστας children έχουν ήδη διαγραφεί από το καινούριο δέντρο - βρίσκονται στην notq **(γραμμές 34-35)**.
Έπειτα από τη λίστα children αφαιρούνται όσα παιδιά βρίσκονται στην q και όσα παιδιά βρίσκονται στην notq **(γραμμή 36**) , και γίνονται οι παρακάτω ελέγχοι:

1. Εάν τα παιδιά που βρίσκονται στην λίστα q είναι περισσότερα από αυτά που θέλουμε (περισσότερα από ν) τότε επιλέγουμε τυχαία ν από αυτά **(γραμμές 39-40)** .

2. Εάν δεν ισχύει το (α), εάν τα παιδιά που έμειναν στην λίστα children είναι περισσότερα από ν και τα παιδιά που βρίσκονται στην λίστα q είναι λιγότερα από ν τότε διαλέγουμε ν-nq από αυτά που έμειναν στην λίστα children και προσθέτουμε μετά όλα τα παιδιά της λίστας q **(γραμμές 42-46)** .
   Δηλαδή: Έστω ότι ο Α έχει 5 παιδιά, τα Β,Γ,Δ,Ε,Ζ και έστω ότι τα Β,Γ έχουν ήδη προστεθεί στο δέντρο (λίστα q) και κανένα δεν έχει ακόμη διαγραφεί. Τότε για n=4 θα επιλεχθούν 2 από τα Δ,Ε,Ζ και έπειτα θα προστεθούν τα Β,Γ. Αν δεν γινόταν αυτό και επιλέγονταν 4 από όλους, μπορεί για παράδειγμα να είχαν επιλεχθεί οι Γ,Δ,Ε,Ζ αλλά επειδή ο Β είναι ήδη στο καινούριο δέντρο θα είχαμε στο σύνολο 5 παιδιά που είναι λάθος.

3. Εάν δεν ισχύει ούτε το (α) ούτε το (β) σημαίνει ότι ο αριθμός των παιδιών είναι μικρότερος του ν και έτσι θα επιλεγούν όλα εκτός από αυτά που έχουν ήδη διαγραφεί σε προηγούμενο στάδιο **(γραμμές 47-48).**

Στη συνέχεια αποθηκεύονται τα παιδιά που έχουν διαγραφεί, στην λίστα deleted **(γραμμή 49)** και όσα από αυτά δεν είναι ήδη αποθηκευμένα στην λίστα notq, αποθηκεύονται **(γραμμές 51-53)** .
Επίσης, από τα παιδιά που έμειναν στην λίστα children όσα δεν έχουν προστεθεί στις λίστες q και r, προστίθενται **(γραμμές 55-58).**
Όλα τα παραπάνω στοιχεία τυπώνονται στο αρχείο NewTree.txt **(γραμμή 61)** και για να τελειώσει η συνάρτηση newtree, διαγράφεται ο r[0] που έχει ήδη ελεγχθεί **(γραμμή 60)** για να μπορέσει να προχωρήσει η διαδικάσια με τον επόμενο που βρίσκεται προς έλεγχο, δηλαδή τον r[1] που στην γραμμή 60 θα γίνει r[0].
Η συνάρτηση επιστρέφει τις λίστες r, q, checked, deleted, notq **(γραμμή 63**) και εφόσων η λίστα r δεν είναι κενή και το πρώτο της στοιχείο δεν έχει ήδη ελεγθεί (λίστα checked) τότε η συνάρτηση newtree ξανακαλείται **(γραμμές 83-85).**
Όταν το πρόγραμμα τελειώσει, τυπώνονται το πλήθος των ατόμων που αποτελούν το δέντρο στην κονσόλα **(γραμμή 104)** και στο αρχείο NewTree.txt **(γραμμή 88)** και τα άτομα που αποτελούν το καινούριο δέντρο, δηλαδή το περιεχομένο της λίστας q, σε ένα αρχείο με όνομα cut_tree.txt, **(γραμμές 107-109).**
Έπειτα δημιουργούνται αντίγραφα των λιστών που αντιπροσωπεύουν τα αρχεία εισόδου στις οποία αποθηκέυονται τα καινούρια δεδομένα του δέντρου με τα κομμένα παιδιά για να διαβαστούν στο επόμενο κεφάλαιο για αποανωνυμοποίηση **(γραμμές 115-145)**.
Τέλος, δημιουργούνται 8 αρχεία αντίστοιχα για τις καινούριες λίστες για να αποθηκευτούν τα δεδομένα τους τα οποία θα χρησιμοποιηθούν στο επόμενο κεφάλαιο για αποανωνυμοποίηση, στα οποία τυπώνονται τα νέα αποτελέσματα για το καθένα από αυτά.
Τα αρχεία αυτά είναι τα id_2.txt, husb.txt, wife_2.txt, children_2.txt, childrenofchildren_2.txt, parent1_2.txt, parent2_2.txt, howmany_2.txt **(γραμμές 147-161)** .

7) DeleteEdges.py

```
1. def family(r,q,id,howmanychild,wife,husb,name,checked,parent2,parent1,h
   owmany,n,name2,husb2,wife2,id2,howmanychild2,parent12,parent22,howmany2
   ):
2.     checked.append(r[0])
3.     fout.write("Person to check: %s\n" % (r[0]))
4.
5.     #Find r[0] parents and put them in the lists
6.     for j in range(0,len(name)):
7.         if (r[0]==name[j]):
8.             if (husb[j]!=1 and husb[j] not in q):
9.                 fout.write("Father: %s\n" % (husb[j]))
10.                    q.append(husb[j])
11.                    r.append(husb[j])
12.                if (wife[j]!=1 and wife[j] not in q):
13.                    fout.write("Mother: %s\n" % (wife[j]))
14.                    q.append(wife[j])
15.                    r.append(wife[j])
16.
17.            #Put r[0]'s children in a list
18.            children=[]
19.            for i in range(0,len(husb)):
20.                if (r[0]==husb[i]):
21.                    children.append(name[i])
22.                elif (r[0]==wife[i]):
23.                    children.append(name[i])
24.            fout.write("\nNumber of Total Children: %s\n" % (len(children
   )))
25.
26.            #Check if children already identified
27.            kk=0
28.            childrenold=children
29.            childreninq=[]
30.            for i in range(0,len(children)):
31.                if(children[i] in q):
32.                    kk+=1
33.                    childreninq.append(children[i])
34.            fout.write("Children in q: %s\n" % (childreninq))
35.            children=list(set(children)^set(childreninq))
36.
37.            deleted=[]
38.            import random
39.            if (len(childreninq)>=n):
40.                children = random.sample(childreninq, k=n)
41.            elif(len(children)>=n):
42.                if((n-kk)>0):
43.                    n=n-kk
44.                    children = random.sample(children, k=n)
45.                    children.extend(childreninq)
46.            else:
47.                children=childrenold
48.
49.            deleted = list(set(childrenold)^set(children))
50.            #Put selected children in the identified list and list to che
   ck later
```

```
51.             for i in range(0,len(children)):
52.                 if(children[i] not in q):
53.                     q.append(children[i])
54.                     r.append(children[i])
55.
56.             #Delete parent-child edges
57.             for j in range(0,len(deleted)):
58.                 for k in range(0,len(name)):
59.                     if(deleted[j]==name[k]):
60.                         husb2[k]=1
61.                         wife2[k]=1
62.                         for m in range(0,len(parent1)):
63.                             if(parent1[m]==husb[k] and parent2[m]==wife[k
    ]):
64.                                 howmany2[m]=howmany2[m]-1
65.                                 break
66.             #Put deleted children in the identified list and list to chec
    k later
67.                 if deleted[j] not in q:
68.                     q.append(deleted[j])
69.                     r.append(deleted[j])
70.
71.             del r[0]
72.             fout.write("Number of Children Selected: %s\nSelected Childre
    n: %s\nChildren with cut edges: %s\n\nQ: %s\nR: %s\n-------------------
    ------\n" % (len(children),children,deleted,q,r))
73.
74.             return r,q,checked,deleted,name2,husb2,wife2,id2,howmanychild
    2,parent12,parent22,howmany2
75.
76.         # MAIN PROGRAM
77.         with open('id.txt', 'r') as file1, open('husb.txt', 'r') as file2
    , open('wife.txt','r') as file3,open('children.txt', 'r') as file4, ope
    n('childrenofchildren.txt', 'r') as file5, open('parent1.txt','r') as f
    ile6, open('parent2.txt', 'r') as file7, open('howmany.txt','r') as fil
    e8:
78.             name=list(map(int,file1.read().split()))
79.             husb=list(map(int,file2.read().split()))
80.             wife=list(map(int,file3.read().split()))
81.             id=list(map(int,file4.read().split()))
82.             howmanychild=list(map(int,file5.read().split()))
83.             parent1=list(map(int,file6.read().split()))
84.             parent2=list(map(int,file7.read().split()))
85.             howmany=list(map(int,file8.read().split()))
86.         fout=open("NewTree.txt", "w")
87.
88.         import copy
89.         name2=copy.deepcopy(name)
90.         husb2=copy.deepcopy(husb)
91.         wife2=copy.deepcopy(wife)
92.         id2=copy.deepcopy(id)
93.         howmanychild2=copy.deepcopy(howmanychild)
94.         parent12=copy.deepcopy(parent1)
95.         parent22=copy.deepcopy(parent2)
96.         howmany2=copy.deepcopy(howmany)
97.         import sys
98.         sys.setrecursionlimit(10**6)
```

```
99.      # r: list with those to check, q: list with the identified people

100.     r,q= ([48348395] for i in range(2))
101.     # checked: list with those who are checked
102.     checked=[]
103.     n=int(input("Enter number of children: "))
104.     #calling the function
105.     family(r,q,id,howmanychild,wife,husb,name,checked,parent2,parent1
   ,howmany,n,name2,husb2,wife2,id2,howmanychild2,parent12,parent22,howman
   y2)
106.     for i in range(0,len(name)):
107.         if (r!=[] and r[0] not in checked):
108.             family(r,q,id2,howmanychild2,wife2,husb2,name2,checked,pa
   rent22,parent12,howmany2,n,name2,husb2,wife2,id2,howmanychild2,parent12
   ,parent22,howmany2)
109.
110.     print("Number of people in new tree: ", len(q))
111.     fout.write("Number of people in new tree: %s\n" % (len(q)))
112.
113.     fout2=open("cut_tree.txt","w")
114.     for i in range(0,len(q)):
115.         fout2.write("%s\n" %(q[i]))
116.
117.     with open('id_2.txt', 'w') as file1, open('husb_2.txt', 'w') as f
   ile2, open('wife_2.txt','w') as file3,open('children_2.txt', 'w') as fi
   le4, open('childrenofchildren_2.txt', 'w') as file5, open('parent1_2.tx
   t','w') as file6, open('parent2_2.txt', 'w') as file7, open('howmany_2.
   txt','w') as file8:
118.         for i in range(0,len(name)):
119.             file1.write("%s\n" % (name2[i]))
120.             file2.write("%s\n" % (husb2[i]))
121.             file3.write("%s\n" % (wife2[i]))
122.         for i in range(0,len(id2)):
123.             file4.write("%s\n" % (id2[i]))
124.             if(howmanychild2[i]>n):
125.                 file5.write("%s\n" % (n))
126.             else:
127.                 file5.write("%s\n" % (howmanychild2[i]))
128.         for i in range(0,len(parent12)):
129.             file1.write("%s\n" % (parent12[i]))
130.             file2.write("%s\n" % (parent22[i]))
131.             file3.write("%s\n" % (howmany2[i]))
132.
133.     fout.close()
```

Περιγραφή DeleteEdges.py:

Άνοιγμα και διάβασμα αρχείων (γραμμές 76-86):
Αρχικά διαβάζονται και κάποια άλλα αρχεία εκτός των id.txt, husb.txt, wife.txt.
Διαβάζονται και τα:
children.txt - ίδιο με το id.txt όμως το συγκεκριμένο θα αντιστοιχηθεί με το πιο κάτω αρχείο (γραμμές 171-173)

childrenofchildren.txt - ο αριθμός των παιδιών του κάθε ατόμου του children.txt (γραμμές 175-177)
parent1.txt - ίδιο με husb.txt - αντοιστοιχία με wife.txt και howmany.txt(γραμμές 180-182)
parent2.txt - ίδιο με wife.txt - αντοιστοιχία με husb.txt και howmany.txt (γραμμές 184-186)
howmany.txt - ο αριθμός των παιδιών των parent1.txt, parent2.txt. (γραμμές 188-190)
Δηλαδή, οι parent1[0] και parent2[0] έχουν howmany[0] παιδιά μαζί

Τα παραπάνω δεδομένα αποθηκεύονται αντίστοιχα στις λίστες:
id,howmanychild,parent1,parent2,howmany

Έπειτα δημιουργούνται αντίγραφα όλων των πιο πάνω λιστών στις οποίες θα αποθηκευτούν τα καινούρια δεδομένα του δέντρου με τους κομμένους συνδέσμους μεταξύ κάποιων παιδιών-γονιών **(γραμμές 88-96)**.
Γίνεται αύξηση του ορίου για αναδρόμες **(γραμμές 97-98)**.
Δημιουργούνται οι παρακάτω λίστες:
Λίστα r **(γραμμή 100)**: θα αποθηκεύονται τα id των ατόμων που θα πάνε για έλεγχο.
Λίστα q **(γραμμή 100)**: θα αποθηκεύονται τα id των ατόμων που έχουν ταυτοποιηθεί.
Λίστα checked **(γραμμή 102)** : θα αποθηκεύονται τα id των ατόμων που έχουν ελεγχτεί (δηλαδή όσα φεύγουν από την r έρχονται εδώ).

Στη συνέχεια, από την κονσόλα ζητείται και διαβάζεται ο ακέραιος αριθμός ν **(γραμμή 104)** και καλείται για πρώτη φορά η συνάρτηση newtree με ορίσματα τα:
r,q,wife,husb,name,checked,n,deleted,notq **(γραμμή 106)**.
Στη συνέχεια η συνάρτηση newtree καλείται με ορίσματα όλες τις πιο πάνω λίστες.

<u>Συνάρτηση newtree (γραμμές 1-74):</u>
Όπως και πριν αποθηκεύονται στο αρχείο το άτομο που ελέγχεται (r[0]) και οι γονείς του. Αποθηκεύονται τα παιδιά του στην λίστα children και γίνονται οι ίδιοι ελέγχοι με πριν, με τν διαφορά ότι εδώ δεν υπάρχει λίστα notq αφού δεν διαγράφεται κανένας κόμβος από το δέντρο, παρά μόνο ακμές.
Προχωρώντας στην βασική διαφορά αυτού του αλγόριθμου με τους προηγούμενους **(γραμμές 56-65)** :
Για τα παιδιά που αποθηκεύονται στην λίστα deleted δηλαδή αυτά για τα οποία θα κόψουμε τις ακμές που τα συνδέουν με τους γονείς τους, γίνεται το εξής:
Διατρέχοντας τις λίστες deleted και name παράλληλα **(γραμμές 57-58)**, εάν βρεθεί παιδί από την $1^{η}$ που υπάρχει στην $2^{η}$ τότε αυτόματα στα αντίγραφα των αρχικών λιστών, οι γονείς αυτού του παιδιού γίνονται ίσοι με 1 - δηλαδή σαν να μην υπάρχει γονιός **(γραμμές 60-61)**. Έπειτα, όταν ελεγχθούν όλα τα παιδιά, τοποθετούνται στις λίστες q και r **(γραμμές 67-69)** αφού όπως ειπώθηκε πριν τα παιδιά αυτά δεν διαγράφονται από το δέντρο και τέλος στις λίστες howmanychild2 **(γραμμές 125-126)** και howmany2 **(γραμμές 62-64)** το πλήθος των παιδιών αλλάζει ανάλογα με το πόσα έχουν διαγραφεί. Αυτό γίνεται διότι εάν π.χ κάποιος είχε 15 παιδιά και το n=5, όπως είναι φυσικό, στο τέλος αυτό το άτομο θα είχε 5 παιδιά.
Στη συνέχεια τυπώνονται τα παραπάνω στοιχεία στο αρχείο NewTree.txt **(γραμμή 72)** και η εφόσων η λίστα r δεν είναι κενή, ξανακαλείται με ορίσματα τα αντίγραφα των λιστών αυτή την φορά, δηλαδή με τα καινούρια αποτελέσματα - διεγραμμένες ακμές **(γραμμές 107-109)**.

Τέλος, τυπώνονται τα άτομα της λίστας q στο αρχείο cut_tree.txt **(γραμμές 114-116)** και δημιουργούνται 8 αρχεία αντίστοιχα για τις καινούριες λίστες για να αποθηκευτούν τα δεδομένα τους τα οποία θα χρησιμοποιηθούν στο επόμενο κεφάλαιο για αποανωνυμοποίηση, στα οποία τυπώνονται τα νέα αποτελέσματα για το καθένα από αυτά.
Τα αρχεία αυτά είναι τα id_2.txt, husb.txt, wife_2.txt, children_2.txt, childrenofchildren_2.txt, parent1_2.txt, parent2_2.txt, howmany_2.txt **(γραμμές 118-132)** .

8) GedStats.py

```
1. with open('id.txt', 'r') as file1, open('famc.txt', 'r') as file2, open
   ('husb.txt','r') as file3,open('wife.txt', 'r') as file4:
2.     id=list(map(int,file1.read().split()))
3.     famc=list(file2.read().split())
4.     husb=list(map(int,file3.read().split()))
5.     wife=list(map(int,file4.read().split()))
6.
7. fout=open("id-howmany_2.txt","w")
8. fp1=open("parent1_2.txt","w")
9. fp2=open("parent2_2.txt","w")
10.      fchild=open("howmanychildren_2.txt","w")
11.
12.      fc,idh,idw,idc=([] for i in range(4))
13.      counter=0
14.      max=0
15.      for k in range(0,len(famc)):
16.          counter=0
17.          if(famc[k] not in fc):
18.              fc.append(famc[k])
19.              for m in range(0,len(id)):
20.                  if(husb[k]==id[m] and wife[k]==1):
21.                      hus=id[m]
22.                      idh.append(id[m])
23.                      idw.append(1)
24.                      break
25.                  elif(wife[k]==id[m] and husb[k]==1):
26.                      wif=id[m]
27.                      idw.append(id[m])
28.                      idh.append(1)
29.                      break
30.                  elif(husb[k]==id[m] or wife[k]==id[m] and husb[k]!=1
   and wife[k]!=1):
31.                          hus=husb[k]
32.                          wif=wife[k]
33.                          idh.append(husb[k])
34.                          idw.append(wife[k])
35.                          break
36.              for n in range(0,len(famc)):
37.                  if(famc[k]==famc[n]):
38.                      counter+=1
39.              idc.append(counter)
40.              #id parent-howmany children
41.              fout.write("%s:%s\n%s:%s\n" % (hus,counter,wif,counter))

42.
```

```
43.       for new in range(0,len(idw)):
44.           fp1.write("%s\n" % (idh[new]))
45.           fp2.write("%s\n" % (idw[new]))
46.           fchild.write("%s\n" % (idc[new]))
47.
48.       from itertools import repeat
49.       parents = [[] for i in repeat(None, 17)]
50.       zeroleft,zero=([] for i in range(2))
51.
52.       for j in range(0,len(id)):
53.           counter=0
54.           for k in range(0,len(idh)):
55.               if(id[j]==idh[k]):
56.                   counter=counter+idc[k]
57.                   parent=idh[k]
58.               elif(id[j]==idw[k]):
59.                   counter=counter+idc[k]
60.                   parent=idw[k]
61.
62.           #maximum children
63.           if(counter>max):
64.               max=counter
65.
66.           for i in range(1,17):
67.               if(counter==i):
68.                   parents[i].append(parent)
69.                   zeroleft.append(parent)
70.
71.       fc=open("children_2.txt","w")
72.       fcofc=open("childrenofchildren_2.txt","w")
73.       fstats=open("Statistics_2.txt","w")
74.       fstats.write("0: ")
75.       for q in range(0,len(id)):
76.           if(id[q] not in zeroleft):
77.               zero.append(id[q])
78.               fc.write("%s\n" % (id[q]))
79.               fcofc.write("0\n")
80.       fstats.write("%s\n\n" % (zero))
81.
82.       for q in range(1,17):
83.           fstats.write("%s: %s\n\n" %(q,parents[q]))
84.           for s in range(0,len(parents[q])):
85.               fc.write("%s\n" % (parents[q][s]))
86.               fcofc.write("%s\n" % (q))
```

Περιγραφή GedStats.py:

Διάβασμα αρχείων εισόδου (γραμμές 1-5):
Το πρόγραμμα ξεκινάει στην γραμμή 120 όπου και διαβάζει τα παρακάτω αρχεία που θα χρησιμοποιήσουμε στο πρόγραμμα.
id.txt - το αρχείο με το μοναδικό αριθμό (id) του κάθε ατόμου του δέντρου (γραμμές 120-122)
famc.txt - το αρχείο με τις οικογένειες (F0,F1,F2,...) στις οποίες το κάθε id συμπεριφέρεται σαν παιδί. Δηλαδή το id[0] είναι παιδί στην οικογένεια F0.

husb.txt - το αρχείο με όλους τους συζύγους γένους αρσενικού (γραμμές 124-126)
wife.txt - το αρχείο με όλους τους συζύγους γένους θηλυκού (γραμμές 128-130)

Δημιουργία αρχείων εξόδου (γραμμές 7-10):
Δημιουργούνται τα αρχεία στα οποία θα τυπωθούν τα στατιστικά των οικογενειών.
Τα αρχεία είναι τα εξής:
id-howmany.txt
parent1.txt
parent2.txt
howmanychildren.txt

Στη συνέχεια δημιουργούνται 6 λίστες: countlist,fc,newlist,idh,idw,idc η χρήση των οποίων θα φανεί μετά και αρχικοποιείται ο μετρητής max **(γραμμές 12-14)**.

Διατρέχοντας την λίστα famc **(γραμμή 15)** αρχικοποιείται ο μετρητής counter **(γραμμή 16)** που θα υπολογίζει πλήθος των παιδιών του κάθε ατόμου και στη λίστα fc μπαίνει το τρέχον στοιχείο famc εάν δεν υπάρχει ήδη **(γραμμές 17-18)**. Αυτό γίνεται για να μην υπάρχει άσκοπη επανάληψη των στοιχείων.
Έπειτα διατρέχοντας την λίστα id **(γραμμή 19)**, εάν βρεθεί στοιχείο της λίστας husb που είναι ίσο με το id και η αντίστοιχη τιμή της λίστας wife είναι ίση με 1 **(γραμμή 20)** τότε η μεταβλητή hus γίνεται ίση με το id και στις λίστες idh και idw τοποθετούνται οι τιμές id και 1 αντίστοιχα **(γραμμές 21-24)**.
Με ίδιο τρόπο γίνονται οι αντιστοιχίες για husb=1 και wife=id **(γραμμές 25-29)** και για την περίπτωση όπου και οι 2 εκ των husb και wife έχουν τιμές διάφορες του 1 **(γραμμές 30-35)**.
Στη συνέχεια διατρέχοντας για 2η φορά την λίστα famc, κάθε φορά που βρίσκεται τιμή famc ίση με την αρχική famc σημαίνει ότι υπάρχει κι άλλο παιδί στην οικογένεια αυτή και έτσι αυξάνεται ο μετρητής counter **(γραμμές 36-38)**.
Όταν τελειώσει η πιο πάνω διαδικασία, στην λίστα idc προστίθεται ο μετρητής counter **(γραμμή 39)**.
Έτσι, στις λίστες idh,idw και idc θα υπάρχουν στο τέλος οι τιμές: πατέρας,μητέρα,πλήθος παιδιών των προηγούμενων 2.

Στα 2 αρχεία εξόδου τυπώνονται τα εξής:
id-howmany.txt: τυπώνονται τα στοιχεία γονιός: "πλήθος παιδιών" δημιουργώντας το αρχείο που φαίνεται στην εικόνα... **(γραμμή 41)**
Έπειτα, οι λίστες idh,idw,idc τυπώνονται αντίστοιχα στα αρχεία:
parent1,parent2,howmanychildren **(γραμμές 43-46)**.

Δημιουργούνται μία πολυλίστα parent με 17 υπόλιστες, η χρήση της οποίας θα εξηγηθεί παρακάτω, και δύο λίστες zeroleft και zero για την αποθήκευση όσων δεν έχουν παιδιά **(γραμμές 49-50)**.
Στη συνέχεια, διατρέχοντας τις λίστες idw,idh για κάθε αρχικό id που υπάρχει μετρίεται το πλήθος των παιδιών του το οποίο αποθηκεύεται στον μετρητή counter **(γραμμές 52-53)**, η μεταβλητή parent παίζερι την τιμή του τρέχον γονιού **(γραμμές 55-60)** και γίνονται 2 ελέγχοι: Εάν το πλήθος αυτό είναι μεγαλύτερο από το τρέχον μέγιστο (μεταβλητή max) τότε γίνεται αυτό το μέγιστο **(γραμμές 63-64)**.

Εάν το πλήθος είναι ίσο με την τιμή i όπου i=1...max τότε στην αντίστοιχη θέση i της πολυλίστας parents τυπώνεται ο γονιός αυτός **(γραμμές 66-69)**.
Δηλαδή εάν οι γονιοί X,Y έχουν 3 παιδιά και ο γονιός Z έχει 16 τότε η λίστα parents θα έχει ως εξής:
parents=[[i=1],[i=2],...,[i=16]] = [[],[],[X,Y],[]...[Z]]

Τέλος, στις **γραμμές 71-86** τα παραπάνω στοιχεία τυπώνονται ομαδοποιημένα στο αρχείο Statistics.txt και στο αρχεία children.txt και childrenofchildren.txt.

9) Deanonymize.py

```
1. def isListEmpty(children2):
2.     return children2 == '' or isinstance(children2, list) and (not chil
   dren2 or all(map(isListEmpty, children2)))
3.
4. def deanonymize(r,q,id,howmanychild,wife,husb,name,checked,parent2,pare
   nt1,howmany):
5.     fout.write("-------------------------------------
   \nPerson to check: %s\n" % (r[0]))
6.     checked.append(r[0])
7.     wi,hu,unc,uncw=([] for i in range(4))
8.     bo=2
9.
10.          #Find r[0] parents and put them in the lists
11.          for j in range(0,len(name)):
12.              if (r[0]==name[j]):
13.                  if (husb[j]!=1 and husb[j] not in q):
14.                      fout.write("Father: %s\n" % (husb[j]))
15.                      q.append(husb[j])
16.                      r.append(husb[j])
17.                  if (wife[j]!=1 and wife[j] not in q):
18.                      fout.write("Mother: %s\n" % (wife[j]))
19.                      q.append(wife[j])
20.                      r.append(wife[j])
21.
22.          #Find Spouse/Spouses and Children
23.          children=[]
24.          for i in range(0,len(husb)):
25.              if (r[0]==husb[i]):
26.                  children.append(name[i])
27.                  if (wife[i]!=1 and wife[i] not in q and wife[i] not i
   n wi):
28.                      wi.append(wife[i])
29.                      #he is male
30.                      bo=1
31.              elif (r[0]==wife[i]):
32.                  children.append(name[i])
33.                  if (husb[i]!=1 and husb[i] not in q and husb[i] not i
   n hu):
34.                      hu.append(husb[i])
35.                      #she is female
36.                      bo=0
37.
38.          #if person is male
```

```
39.           if(bo==1):
40.               for b in range(0,len(parent2)):
41.                   for w in range(0,len(wi)):
42.                       if(wi[w]==parent2[b]):
43.                           if(parent1[b]==r[0]):
44.                               if (parent2[b] not in uncw):
45.                                   unc.append(howmany[b])
46.                                   uncw.append(parent2[b])
47.                                   break
48.           #if person is female
49.           elif (bo==0):
50.               for v in range(0,len(parent1)):
51.                   for h in range(0,len(hu)):
52.                       if(hu[h]==parent1[v]):
53.                           if(parent2[v]==r[0]):
54.                               if (parent1[v] not in uncw):
55.                                   unc.append(howmany[v])
56.                                   uncw.append(parent1[v])
57.                                   break
58.
59.           #find the spouse of all his/her spouses with he/she has uniqu
   e number of children
60.           e={}
61.           for x in unc: e[x] = x in e
62.           uniqc=[z for z in e if not e[z]]
63.
64.           if(bo==1):
65.               for s in range(0,len(uniqc)):
66.                   for p in range(0,len(unc)):
67.                       if(unc[p]==uniqc[s] and uncw[p] not in q):
68.                           fout.write("With his wife: %s he has unique n
   umber of children (%s) so she is identified\n" % (uncw[p],uniqc[s]))
69.                           q.append(uncw[p])
70.                           r.append(uncw[p])
71.           if(bo==0):
72.               for s in range(0,len(uniqc)):
73.                   for p in range(0,len(unc)):
74.                       if(uniqc[s]==unc[p] and uncw[p] not in q):
75.                           fout.write("With her husband: %s she has uniq
   ue number of children (%s) so he is identified\n" % (uncw[p],uniqc[s]))

76.                           q.append(uncw[p])
77.                           r.append(uncw[p])
78.
79.           #if with his/her spouse he/she has only one child it can be i
   dentified immediately
80.           for g in range(0,len(uniqc)):
81.               if(uniqc[g]==1):
82.                   for c in range(0,len(husb)):
83.                       if(r[0]==husb[c] or r[0]==wife[c]):
84.                           if(uncw[g]==husb[c] or uncw[g]==wife[c]):
85.                               if(name[c] not in q):
86.                                   fout.write("%s and %s's only child is
   :%s\n" % (r[0],uncw[g],name[c]))
87.                                   q.append(name[c])
88.                                   r.append(name[c])
89.                                   break
```

```
90.
91.          #Call of FindDescendandsTree for spouses
92.          spouse,spouse2=([] for i in range(2))
93.          if (bo==1):
94.              spouse=wi
95.          elif(bo==0):
96.              spouse=hu
97.
98.          for m in range(0,len(spouse)):
99.              if(spouse[m] not in q):
100.                 spouse2.append(spouse[m])
101.
102.         spouse=spouse2
103.         origchildren=spouse
104.
105.         from itertools import repeat
106.         list1 = [[] for i in repeat(None, len(spouse))]
107.         names=list1
108.
109.         for i in range(0,len(spouse)):
110.             list1[i].append(spouse[i])
111.
112.         FindDescendandsTree(list1,names,husb,wife,name,origchildren,q
    ,r,id,howmanychild)
113.
114.         #find the children of the person we check that have unique nu
    mber of children
115.         #children of children
116.         childrenofchildren,nameofchild=([] for i in range(2))
117.         for l in range(0,len(children)):
118.             for m in range(0,len(id)):
119.                 if (children[l]==id[m] and children[l] not in q):
120.                     childrenofchildren.append(howmanychild[m])
121.                     nameofchild.append(children[l])
122.         d={}
123.         for n in childrenofchildren: d[n] = n in d
124.         uniq=[o for o in d if not d[o]]
125.
126.         for s in range(0,len(uniq)):
127.             for p in range(0,len(childrenofchildren)):
128.                 if(uniq[s]==childrenofchildren[p]):
129.                     if(nameofchild[p] not in q):
130.                         fout.write("Child of %s with %s child/childre
    n is:\n" % (r[0],uniq[s],nameofchild[p]))
131.                         q.append(nameofchild[p])
132.                         r.append(nameofchild[p])
133.
134.         #Call of FindDescendandsTree for children
135.         children2=[]
136.         for m in range(0,len(children)):
137.             if(children[m] not in q):
138.                 children2.append(children[m])
139.
140.         children=children2
141.         origchildren=children
142.
143.         from itertools import repeat
```

```
144.          list1 = [[] for i in repeat(None, len(children))]
145.          names=list1
146.
147.          for i in range(0,len(children)):
148.              list1[i].append(children[i])
149.
150.          FindDescendandsTree(list1,names,husb,wife,name,origchildren,q
   ,r,id,howmanychild)
151.
152.          del r[0]
153.          return r,q,checked
154.
155.      def FindDescendandsTree(list1,names,husb,wife,name,origchildren,q
   ,r,id,howmanychild):
156.          length,uniqquantity,uniqnames,count=([] for i in range(4))
157.          from itertools import repeat
158.          names = [[] for i in repeat(None, len(origchildren))]
159.          for i in range(0,len(list1)):
160.              counter=0
161.              for k in range(0,len(list1[i])):
162.                  for j in range(0,len(husb)):
163.                      if(list1[i][k]==husb[j] or list1[i][k]==wife[j]):

164.                          names[i].append(name[j])
165.                          counter+=1
166.
167.          quantity=[[] for i in repeat(None, len(origchildren))]
168.          for a in range(0,len(names)):
169.              for b in range(0,len(names[a])):
170.                  for c in range(0,len(id)):
171.                      if(names[a][b]==id[c]):
172.                          quantity[a].append(howmanychild[c])
173.
174.          if(len(quantity)>1):
175.              for s in range(0,len(quantity)):
176.                  if(quantity[s] not in uniqquantity):
177.                      uniqquantity.append(quantity[s])
178.          for t in range(0,len(uniqquantity)):
179.              qcount=0
180.              for s in range(0,len(quantity)):
181.                  if(uniqquantity[t]==quantity[s]):
182.                      qcount+=1
183.              count.append(qcount)
184.
185.          for k in range(0,len(count)):
186.              if(count[k]<=1):
187.                  for l in range(0,len(origchildren)):
188.                      if(uniqquantity[k]==quantity[l]):
189.                          if(len(origchildren)!=1 and origchildren[l] n
   ot in q):
190.                              fout.write("Based on the number of descen
   dands %s was identified\n" % (str(origchildren[l])))
191.                              q.append(origchildren[l])
192.                              r.append(origchildren[l])
193.                              del origchildren[l]
194.                              del names[l]
195.                              del list1[l]
```

```
196.                         elif(len(origchildren)==1 and origchildren[0]
    not in q):
197.                             fout.write("Based on the number of de
   scendands %s was identified\n" % (str(origchildren[0])))

198.                             q.append(origchildren[0])
199.                             r.append(origchildren[0])
200.
201.         if(not isListEmpty(list1) and len(origchildren)!=1 and origch
   ildren!=[]):
202.             list1=names
203.             FindDescendandsTree(list1,names,husb,wife,name,origchildr
   en,q,r,id,howmanychild)
204.
205.         return q,r
206.
207.     #MAIN PROGRAM
208.     with open('id_2.txt', 'r') as file1, open('husb_2.txt', 'r') as f
   ile2, open('wife_2.txt','r') as file3,open('children_2.txt', 'r') as fi
   le4, open('childrenofchildren_2.txt', 'r') as file5, open('parent1_2.tx
   t','r') as file6, open('parent2_2.txt', 'r') as file7, open('howmany_2.
   txt','r') as file8:
209.         name=list(map(int,file1.read().split()))
210.         husb=list(map(int,file2.read().split()))
211.         wife=list(map(int,file3.read().split()))
212.         id=list(map(int,file4.read().split()))
213.         howmanychild=list(map(int,file5.read().split()))
214.         parent1=list(map(int,file6.read().split()))
215.         parent2=list(map(int,file7.read().split()))
216.         howmany=list(map(int,file8.read().split()))
217.     fout=open("Deanonymized.txt", "a")
218.
219.     import sys
220.     sys.setrecursionlimit(10**6)
221.     # q: list with identified people , r: list with those to check
222.     q,r=([48348395] for i in range(2))
223.     #list with people i've checked
224.     checked=[]
225.
226.     #calling the function
227.     deanonymize(r,q,id,howmanychild,wife,husb,name,checked,parent2,pa
   rent1,howmany)
228.     for i in range(0,len(name)):
229.         if (r!=[] and r[0] not in checked):
230.             deanonymize(r,q,id,howmanychild,wife,husb,name,checked,pa
   rent2,parent1,howmany)
231.
232.     print("deanonymization done")
233.     print("Number of people identified: ", len(q))
234.     print("identified people:", q)
```

Περιγραφή Deanonymize.py:

Διάβασμα αρχείων εισόδου και δημιουργία αρχείου εξόδου (γραμμές 207-217):
Το πρόγραμμα ξεκινάει στην γραμμή 211 όπου και διαβάζει τα καινούρια (από τα προγράμματα 6,7) αρχεία που θα χρησιμοποιήσουμε στο πρόγραμμα.
id_2.txt - το αρχείο με το μοναδικό αριθμό (id) του κάθε ατόμου του δέντρου
husb.txt - το αρχείο με όλους τους συζύγους γένους αρσενικού
wife_2.txt - το αρχείο με όλους τους συζύγους γένους θηλυκού
children_2.txt - ίδιο με το id.txt όμως το συγκεκριμένο θα αντιστοιχηθεί με το πιο κάτω αρχείο
childrenofchildren_2.txt - ο αριθμός των παιδιών του κάθε ατόμου του children.txt
parent1_2.txt - ίδιο με husb_2.txt - αντοιστοιχία με wife_2.txt και howmany_2.txt
parent2_2.txt - ίδιο με wife_2.txt - αντοιστοιχία με husb_2.txt και howmany_2.txt - ο αριθμός των παιδιών των parent1_2.txt, parent2_2.txt.
Δηλαδή, οι parent1_2[0] και parent2_2[0] έχουν howmany_2[0] παιδιά μαζί

deanonymized.txt – δημιουργείται το αρχείο στο οποίο θα γραφούν τα αποτελέσματα
Τα παραπάνω δεδομένα αποθηκεύονται αντίστοιχα στις λίστες:
name,famc,husb,wife,id,howmanychild,parent1,parent2,howmany,fout

Έπειτα, δημιουργουνται 3 λίστες **(γραμμές 222-224)**:
λίστα r: θα αποθηκεύονται τα id των ατόμων που θα πάνε για έλεγχο.
λίστα q: θα αποθηκεύονται τα id των ατόμων που έχουν ταυτοποιηθεί.
λίστα checked: θα αποθηκεύονται τα id των ατόμων που έχουν ελεγχτεί (δηλαδή όσα φεύγουν από την r έρχονται εδώ).

Στη συνέχεια, καλείται για πρώτη φορά η συνάρτηση deanonymize με ορίσματα τα:
r,q,id,howmanychild,wife,husb,name,famc,checked,parent2,parent1,howmany **(γραμμή 227)**

Συνάρητηση deanonymize (γραμμές 4-157):
Η παραπάνω συνάρτηση κάνει τις ακόλουθες διαδικασίες:
Ανοίγει το αρχείο deanonymized.txt και γράφει μέσα το πρώτο άτομο της λίστας r, δηλαδή το άτομο από το οποίο θα αρχίσει μια σειρά ελέγχων για την επιτυχή ανωνυμοποίηση όσο το δυνατόν περισσότερων ατόμων και αύτοματα προστίθεται στην λίστα checked **(γραμμές 5-6)**.
Δημιουργούνται 4 λίστες wi, hu, unc, uncw και μία μεταβλητή bo η χρήση των οποίων θα περιγραφεί στην συνέχεια **(γραμμές 7-8)**.

Εύρεση και ταυτοποίηση πατέρα και μητέρας (γραμμές 16-33):
Διατρέχοντας την λίστα name με τα μοναδικά id, εάν βρεθεί name[j] το οποίο είναι ίδιο με το id του ατόμου που ελέγχεται στην παρούσα φάση, δηλαδή το πρώτο στοιχείο της λίστας r, "το r[0]" **(γραμμές 11-12)** τότε:
Εάν το αντίστοιχο στοιχείο της λίστας husb είναι διαφορετικό του 1 (υπενθυμίζεται ότι η τιμή 1 δίνεται σε ένα στοιχείο εφόσων δεν υπάρχει-δηλαδή εδώ ελέγχεται εάν υπάρχει ή όχι πατέρας) και δεν ανήκει στην λίστα q, δηλαδή δεν έχει ήδη ταυτοποιηθεί **(γραμμή 13)**, τότε στο αρχείο εξόδου γράφεται το όνομα του πατέρα **(γραμμή 14)** και έπειτα το μοναδικό id του αποθηκεύεται στις λίστες q,r **(γραμμές 15-16)**.

Αντίστοιχη διαδικασία ακολουθείται για την εύρεση μητέρας με χρήση της λίστας wife **(γραμμές 17-20)**.

Εύρεση συζύγου/συζύγων και παιδιών (γραμμές 23-61):

Δημιουργείται μια λίστα children **(γραμμή 23)**. Διατρέχοντας την λίστα husb , εάν βρεθεί στοιχείο της που να είναι ίδιο με το r[0] σημαίνει οτι το συγκεκριμένο άτομο που ελέγχουμε είναι σύζυγος γένους αρσενικού και αυτόματα αποθηκεύονται στις λίστες children όλα τα παιδιά του **(γραμμές 24-26)**. Έπειτα, ελέγχεται εάν έχει σύζυγο ο r[0] η οποία δεν έχει ήδη ταυτοποιηθεί (δεν βρίσκεται στη λίστα q) και αν ισχύει, τότε την προσθέτει στη λίστα wi (στη λίστα wi στο τέλος του for loop θα έχουν αποθηκευτεί όλες οι σύζυγοι του r[0]) και η μεταβλητή bo γίνεται ίση με 1 για να δηλώσει ότι ο r[0] είναι γένους αρσενικού **(γραμμές 27-30)**.
Η ίδια διαδικασία ακολουθείται διατρέχοντας την λίστα wife , εάν δηλαδή ο r[0] είναι γένους θηλυκού **(γραμμές 31-36)**.
Στη συνέχεια, εάν ο r[0] είναι άντρας δηλαδή bo=1, τότε διατρέχοντας τις λίστες parent2 και wi, εάν έχει βρεθεί άτομο στη λίστα parent1 ίδιο με τον r[0] και το αντίστοιχο άτομο στη λίστα parent2 είναι ίδιο με το πρώτο άτομο της λίστας wi τότε αποθηκεύεται στη λίστα uncw το id της γυναίκας του και στην λίστασ unc το πλήθος των παιδιών που έχει μαζί της **(γραμμές 38-47)**. Η παραπάνω διαδικάσια γίνεται για όλες τις γυναίκες του - μέχρι το τέλος της λίστας wi, εάν η λίστα wi έχει μόνο ένα στοιχείο τότε ο r[0] έχει μόνο μια γυναίκα.
Η ίδια διαδικασία ακολουθείται εάν ο r[0] είναι γυναίκα **(γραμμές 49-57)**.

Ταυτοποίηση συζύγου/συζύγων (γραμμές 60-77):

Αρχικά, χρησιμοποιώντας την λίστα unc με το πλήθος των παιδιών που έχει με την/τον κάθε σύζυγο, βρίσκουμε ποιοι από αυτούς τους αριθμούς εμφανίζονται στην unc μόνο μια φορά, δηλαδή είναι μοναδικοί.
Η λογική είναι η εξής: Έαν ο r[0] έχει 4 συζύγους uncw={a,b,c,d} και με την κάθε μια έχει αντίστοιχα 3,3,4,3 παιδιά - δηλαδή unc={3,3,4,3} αυτόματα μπορεί η σύζυγος c να ταυτοποιηθεί αφού είναι η μόνη με την οποία έχει 4 παιδιά. Ενώ οι a,b,d μπορεί να είναι οποιοιδήποτε κόμβοι στο δέντρο μας.
Άρα δημιουργούμε μια λίστα uniqc στην οποία αποθηκεύονται οι μοναδικοί αυτοί αριθμοί της λίστας unc.. Εδώ θα είχαμε uniqc={4}
Έπειτα, έαν ο r[0] είναι άντρας τότε διατρέχοντας τις λίστες uniqc και unc, βρίσκουμε ποιο στοιχείο της λίστας unc έχει τον κάθε μοναδικό αριθμό της λίστας uniqc (εδώ θα ήταν uniqc[0]==unc[2]) και εάν η συγκεκριμένη σύζυγος με την οποία έχει τον μοναδικό αυτό αριθμό παιδιών δεν έχει ήδη ταυτοποιηθεί τότε στο αρχείο τυπώνεται το όνομα της και αυτόματα μπαίνει στις λίστες q,r δηλαδή έχει ταυτοποιηθεί και θα εξεταστεί αργότερα **(γραμμές 60-70)**.
Ίδια διαδικασία ακολουθείται έαν ο r[0] ήταν γυναίκα **(γραμμές 71-77)**.

Ταυτοποίηση μοναδικού παιδιού (γραμμές 80-89):

Εάν με κάποια από τις παραπάνω ταυτοποιημένες συζύγους το πλήθος των παιδιών τους είναι 1, δηλαδή έχουν ένα και μοναδικό παιδί, τότε βάσει λογικής αυτό μπορει αμέσως να ταυτοποιηθεί. Αυτό γίνεται εδώ.
Διατρέχοντας την λίστα uniqc εάν βρεθεί στοιχείο της με τιμή 1, τότε εάν ο r[0] έχει ίδια τιμή με κάποιο στοιχείο της λίστας husb-δηλαδή είναι άντρας, ή της λίστας wife - γυναίκα και εάν ο/η αντίστοιχος/η του σύζυγος (από τη λίστα uncw) είναι ο αντίστοιχος του husb ή wife, δηλαδή εάν έχει βρεθεί ουσιαστικά το ζευγάρι που έχει το μοναδικό αυτό παιδί, τότε το παιδί τους βρίσκεται

στην αντίστοιχη θέση της λίστας name. Εάν λοιπόν δεν έχει ήδη ταυτοποιηθεί το παιδί αυτό τότε τυπώνεται στο αρχείο και αποθηκεύεται και στις λίστες q,r.

Ταυτοποίηση περαιτέρω παιδιών (γραμμές 114-132):

Η λογική εδώ είναι ίδια με τη λογική ταυτοποίησης συζύγου. Εάν δηλαδή κάποιο από τα παιδιά του r[0] έχει μοναδικό αριθμό παιδιών τότε αυτό το παιδί ταυτοποιείται αυτόματα.
Δηλαδή: έστω τα παιδιά του r[0], children={a,b,c,d,e,f} και από τη λίστα id έχουν ταυτιστεί τα a,d,e,f δηλαδή nameofchild={a,d,e,f} και τα συγκεκριμένα παιδία έχουν με τις/τους δικές/-όυς τους συζύγους 5,4,4,3 παιδία αντίστοιχα, δηλαδή childrenofchildren={5,4,4,3}. Τότε, αυτόματα οι a,f ταυτοποιούνται στο δέντρο μας.
Από την λίστα children που είχαμε δημιουργήσει, στην οποία αποθηκεύτηκαν όλα τα παιδιά του r[0] ελέγχουμε έαν κάποιο από αυτά ταυτίζεται με στοιχείο της λίστας id και εάν αυτό δεν έχει ήδη ταυτοποιηθεί τότε προστίθεται στη λίστα nameofchild το όνομα του και στη λίστα childrenofchildren το πλήθος των παιδιών του.
Έπειτα δημιουργείται μια λίστα uniq με τα μοναδικά πλήθη παιδιών, εδώ uniq={5,3}. Μετά γίνεται έλεγχος για να βρεθεί ποιο στοιχείο της λίστας childrenofchildren έχει τον μοναδικό αυτό αριθμό παιδιών και εάν το αντίστοιχο παιδί που έχει αυτά τα παιδιά δεν έχει ήδη ταυτοποιηθεί τότε τυπώνεται στο αρχείο και προστίθεται στις λίστες q,r.
Έτσι, για να τελειώσει η συνάρτηση deanonymize , διαγράφεται ο r[0] που έχει ήδη ελεγχθεί για ταυτοποιήσεις **(γραμμή 152)** για να μπορέσει να προχωρήσει η διαδικάσια με τον επόμενο που βρίσκεται προς έλεγχο, δηλαδή τον r[1] που στην γραμμή 152 θα γίνει r[0].
Η συνάρτηση επιστρέφει τις λίστες r,q,checked **(γραμμή 153)** και εφόσων η λίστα r δεν είναι κενή και το πρώτο της στοιχείο δεν έχει ήδη ελεγθεί (λίστα checked) τότε η συνάρτηση deanonymize ξανακαλείται (**γραμμές 228-230)**.

Όταν το πρόγραμμα τελειώσει, τυπώνεται στην κονσόλα ένα μήνυμα πως η αποανωνυμοποίηση έχει τελειώσει και τυπώνεται και το πλήθος των ατόμων που έχουν ταυτοποιηθεί, δηλαδή το μήκος της λίστας q, καθώς και τα στοιχεία της **(γραμμές 232-234)**.

Συνάρτηση FindDescendandsTree (γραμμές 155-205):

Η παραπάνω συνάρτηση καλείται σε δύο σημεία του προγράμματος:

1. Στις **γραμμές 92-112**
2. Στις **γραμμές 134-150**

Στην πρώτη περίπτωση καλείται με ορίσματα τις άγνωστες συζύγους του r[0] και στην δεύτερη περίπτωση καλείται με ορίσματα τα άγνωστα παιδιά του r[0].

Η συνάρτηση αυτή λειτουργεί ως εξής (θα δοθεί ένα απλό παράδειγμα για καλύτερη επεξήγηση):
Έστω ότι ο r[0] έχει 4 παιδιά, 1 εκ των οποίων έχει ήδη ταυτοποιηθεί σε κάποιο άλλο σημείο του προγράμματος. Τότε μένουν τα υπόλοιπα τρία έστω Α,Β,Γ αντίστοιχα. Η συνάρτηση αυτή θα δημιουργήσει τόσες λίστες όσες τα παιδιά του r[0]. Δηλαδή θα δημιουργήσει τις λίστες Α,Β,Γ. Κάθε φορά σε αυτές τις λίστες θα αποθηκεύει το πλήθος των παιδιών που έχουν τα προηγούμενα στοιχεία της λίστας. Σε πρώτη φάση θα αποθηκευτουν δηλαδή:

Λίστα Α= [πλήθος παιδιών Α]=[1], Λίστα Β=[πλήθος παιδιών Β]=[1], Λίστα Γ=[πλήθος παιδιών Γ]=[1]
Σε 3 λίστες Α',Β',Γ' αποθηκεύονται τα ονόματα των παιδιών των Α,Β,Γ.
Έστω:
Λίστα Α'= [Δ], Λίστα Β=[Ε], Λίστα Γ=[Ζ]
Εφόσων οι λίστες Α,Β,Γ είναι ίσες, η συνάρτηση θα προχωράει στην επόμενη γενιά και οι λίστες θα διαμορφωθούν ως εξής:
Λίστα Α= [πλήθος παιδιών Δ]=[2], Λίστα Β=[πλήθος παιδιών Ε]=[1], Λίστα Γ=[πλήθος παιδιών Ζ]=[1]
Εδώ, αυτόματα ξεχωρίζει η λίστα Α αφού είναι ανίση με τις Β,Γ και έτσι ταυτοποιείται ο πρόγονος του Δ, δηλαδή αυτός που είναι παιδί του r[0] => ο Α
Για τα παιδιά Β,Γ η συνάρτηση συνεχίζει για όλες τις επόμενες γενιές εώς ότου βρεθεί διαφορετικό δέντρο απογόνων τους και μπορέσουν να ταυτοποιηθούν.

Η ίδια διαδικασία ακολουθείται και για τις συζύγους του r[0].
Τονίζεται ότι δεν ταυτοποιείται κανένας απόγονος των παιδιών η συζύγων του r[0], παρά μόνο οι ίδιοι. Δηλαδή όλοι οι απόγονοι τους είναι απλά βοηθητικοί στο να βρεθεί διαφορετικό δέντρο σε κάποια γενιά και να ταυτοποιηθεί ο αρχικός «στόχος».

Συνάρτηση isListEmpty (γραμμές 1-2):

Είναι απλά μια συνάρτηση η οποία ελέγχει εάν μια πολυλίστα (Πολυλίστα=[[], [[]], []]) είναι κενή.

Παράρτημα ΙΙ

Αποτελέσματα διαγραφής παιδιών n=1,…,8																
	n=1	n=1 D	n=2	n=2 D	n=3	n=3 D	n=4	n=4 D	n=5	n=5 D	n=6	n=6 D	n=7	n=7 D	n=8	n=8 D
1	17	17	37	35	25	20	568	417	87	40	2361	1286	2916	1544	3178	1612
2	18	18	123	111	49	32	1004	667	89	39	1890	1031	386	157	13	6
3	103	103	56	49	6	3	1264	862	181	89	2224	1244	2683	1409	3207	1634
4	41	41	19	15	22	16	105	68	286	160	1774	1046	2744	1439	3176	1589
5	7	7	140	127	39	27	1108	662	489	257	2388	1319	2490	1324	3113	1624
6	22	22	100	88	548	329	75	20	247	121	2371	1308	2830	1531	9	2
7	40	40	241	217	196	148	1097	713	1440	900	2470	1355	2935	1536	3074	1565
8	20	20	73	67	63	38	1130	603	226	104	2265	1229	2695	1418	3166	1614
9	6	6	35	31	358	273	1058	711	1918	1163	651	306	2810	1497	3123	1625
10	20	20	131	118	652	526	158	90	1950	1204	216	89	2971	1551	3079	1603
11	20	20	35	33	482	350	10	8	263	115	2245	1253	2452	1348	2985	1557
12	34	34	42	38	476	339	1707	1094	225	110	2509	1374	2788	1449	3051	1588
13	14	14	57	49	98	69	9	6	1873	1142	2306	1276	307	123	3095	1593
14	21	21	16	14	701	472	12	10	1621	969	2052	1098	2862	1533	3140	1610
15	4	4	153	126	93	59	1096	755	1973	1101	2163	1184	2769	1485	2953	1531
16	48	48	350	322	501	354	251	153	1805	1097	1833	1052	2769	1504	2972	1545
17	7	7	496	446	514	374	1134	732	168	100	2226	1282	2818	1494	3082	1604
18	15	15	144	126	64	37	1137	761	1339	833	2417	1335	2608	1435	3105	1607
19	41	41	31	27	112	76	146	79	1277	789	1949	1094	2665	1402	3140	1621
20	23	23	212	274	671	538	132	71	1168	733	2093	1154	2950	1517	3122	1590
21	46	46	79	55	553	441	1381	900	2039	1185	2498	1375	2673	1407	3191	1628
22	6	6	144	121	500	400	1099	692	1873	283	2455	1356	2653	1449	3112	1608
23	10	10	332	311	45	32	12	4	1672	1014	2060	1149	47	25	3126	1592
24	16	16	50	38	111	80	149	84	340	189	2487	1372	2543	1357	2989	345
25	29	29	6	4	626	415	29	17	1992	1153	1981	1142	13	4	49	26
26	4	4	95	97	542	455	1210	778	283	126	1789	1008	2675	1407	3186	1618
27	36	34	145	112	33	19	1361	898	2178	1232	2490	1381	7	2	3106	1598
28	16	16	161	129	245	199	28	12	1192	740	2460	1355	2877	1510	3131	1599
29	28	28	8	8	431	320	867	559	1862	1093	2402	1322	2835	1486	2961	1517
30	56	56	12	10	74	53	714	520	1835	1114	2454	1365	2829	1503	3123	1590
31	13	13	54	48	753	530	1416	906	178	96	2420	1299	2538	1394	3130	1606
32	7	7	43	37	91	63	999	654	1954	180	2280	1244	2744	1455	3232	1630
33	34	34	14	12	75	61	862	572	1701	1030	2291	1256	2737	1490	3061	1577
34	8	8	6	4	487	383	1052	655	10	6	2371	1319	2829	1487	2935	1543
35	34	34	19	15	103	42	796	403	2013	1141	2284	1268	2648	1402	3141	383
36	38	38	309	288	138	85	165	87	1622	63	2209	1239	2772	1454	3066	1570
37	40	40	27	21	663	483	514	352	1756	1023	2405	1319	2875	1514	3203	1650
38	23	23	34	32	632	511	102	77	1312	778	2365	1348	2805	1466	3048	1559
39	18	18	20	18	701	554	237	184	1842	1091	2337	1278	2752	301	3053	1572
40	75	75	53	51	603	436	1184	749	1606	962	2442	1357	2836	1515	3174	1619
41	21	21	63	71	190	139	756	508	1693	1053	2212	1254	2668	1424	3202	1634
42	13	13	62	55	439	312	68	51	1023	657	12	6	2859	1522	2922	1556
43	14	14	122	112	121	79	95	59	262	125	2484	1328	2639	1436	3133	1572
44	75	75	17	17	27	22	1269	418	1449	192	2136	1162	2769	1434	3209	1624
45	5	5	26	24	30	25	1359	881	850	521	2420	277	2905	1503	3143	1630
46	21	21	209	190	7	5	951	626	1840	1098	474	231	2894	1516	3047	1572
47	60	60	25	23	4	2	641	460	1861	1014	2332	5	2780	1450	3221	1634
48	75	60	19	17	709	516	1143	725	456	245	627	300	2858	1487	3044	1574
49	32	32	70	62	95	63	799	469	189	97	2140	1214	2850	1496	3167	1612
50	7	7	324	292	449	294	168	85	93	48	2260	1258	2853	1519	3022	1556

Αποτελέσματα διαγραφής παιδιών n=9,…,16																
	n=9	n=9 D	n=10	n=10 D	n=11	n=11 D	n=12	n=12 D	n=13	n=13 D	n=14	n=14 D	n=15	n=15 D	n=16	n=16 D
1	3321	1662	3490	1720	3531	1727	3560	1736	3591	1743	3597	1746	3599	1746	3600	1744
2	3231	1623	3438	1693	3546	1730	3574	1734	3580	1739	3597	1740	3599	1741	3600	1740
3	3358	1684	3438	1701	3545	1724	3576	1739	3584	1732	3597	1741	3599	1734	3600	1738
4	3377	1681	3488	1712	3510	1710	3575	1739	3591	1740	3597	1737	3599	1741	3600	1740
5	3349	1679	3426	1698	3540	1737	3558	1721	3591	1740	3597	1740	3599	1741	3600	1743
6	3340	1654	3378	1690	3510	1728	3573	1740	3575	1723	3597	1740	3599	1737	3600	1740
7	3353	1677	3340	1651	3530	1710	3575	1732	3576	1738	3597	1739	3599	1740	3600	1737
8	3344	1663	3448	1705	3519	1718	3562	1733	3570	1733	3597	1738	3599	1740	3600	1734
9	3290	1665	3451	1715	3511	1717	3544	1725	3584	1733	3597	1740	3599	1738	3600	1741
10	3290	1626	3520	1728	3525	1727	3542	1726	3591	1737	3597	1740	3599	1740	3600	1740
11	3252	1634	3446	1704	3532	1729	3562	1730	3575	1731	3597	1738	3599	1740	3600	1733
12	3328	1661	3420	1696	3529	1719	3556	1732	3591	1741	3597	1737	3599	1738	3600	1740
13	15	4	3442	1717	3531	1726	3550	1729	3574	1733	3597	1737	3599	1739	3600	1742
14	3331	1653	3475	1708	3563	1739	3553	1730	3591	1738	3597	1740	3599	1737	3600	1743
15	3320	1662	3479	1722	3522	1731	3565	1733	3575	1732	3597	1740	3599	1744	3600	1738
16	3305	1643	3429	1685	3526	1714	3553	1730	3591	1741	3597	1740	3599	1741	3600	1734
17	3367	1677	3456	1699	3558	1738	3550	1734	3576	1738	3597	1738	3599	1739	3600	1739
18	3366	1686	3523	1728	3538	1730	3568	1721	3574	1735	3597	1740	3599	1737	3600	1745
19	3295	1659	3407	1690	3512	1725	3573	1733	3575	1732	3597	1741	3599	1740	3600	1741
20	3277	1636	3502	1721	3559	1740	3540	1724	3585	1733	3597	1741	3599	1741	3600	1740
21	3325	1667	3418	1705	3521	1718	3540	1722	3591	1741	3597	1740	3599	1744	3600	1740
22	3382	1680	3472	1706	3543	1738	3548	1720	3570	1735	3597	1741	3599	1738	3600	1741
23	3337	1675	3506	1724	3541	1730	3546	1728	3584	1730	3597	1732	3599	1736	3600	1740
24	3266	1636	3477	1703	3564	1732	3581	1740	3591	1740	3597	1740	3599	1737	3600	1740
25	3219	1579	3380	1672	3555	1730	3532	1716	3569	1730	3597	1738	3599	1736	3600	1743
26	3346	1663	3413	1687	3530	1721	3556	1733	3576	1737	3597	1741	3599	1740	3600	1741
27	3362	1683	3442	1707	3526	1729	3563	1729	3591	1740	3597	1737	3599	1740	3600	1745
28	3305	1646	3426	1697	3519	1722	3579	1732	3574	1739	3597	1740	3599	1741	3600	1740
29	51	25	3470	1715	3540	1727	3578	1743	3591	1740	3597	1740	3599	1739	3600	1741
30	3268	1647	3486	1718	3474	1709	3552	1734	3591	1741	3597	1741	3599	1740	3600	1741
31	3339	1653	3346	1696	3546	1733	3581	1740	3575	1731	3597	1741	3599	1739	3600	1740
32	3159	1602	3496	1719	3537	1730	3579	1737	3591	1738	3597	1741	3599	1740	3600	1742
33	3190	1611	3417	1689	3490	1717	3568	1726	3580	1735	3597	1737	3599	1734	3600	1742
34	3353	1675	3447	1709	3533	1735	12	2	3591	1740	3597	1743	3599	1743	3600	1743
35	3409	1699	3456	1706	3532	1726	3560	1736	3591	1738	3597	1735	3599	1740	3600	1740
36	3297	1673	3453	1702	3556	1726	3555	1735	3591	1744	3597	1741	3599	1740	3600	1741
37	3365	1680	3461	1714	3528	1721	3570	1742	3574	1735	3597	1738	3599	1741	3600	1740
38	3370	1687	3464	1703	3537	1728	3549	1732	3591	1741	3597	1742	3599	1740	3600	1741
39	3277	1642	3428	1704	3552	1726	3567	1735	3591	1741	3597	1741	3599	1738	3600	1744
40	3377	1678	3435	1706	3510	1716	3565	1732	3591	1742	3597	1733	3599	1741	3600	1732
41	3358	1662	3507	1712	3526	1717	3550	1732	3591	1741	3597	1742	3599	1733	3600	1740
42	3329	1670	3444	1712	3530	1715	3550	1730	3584	1734	3597	1740	3599	1738	3600	1739
43	3282	1654	3389	1681	3533	1732	3564	1735	3574	1739	3597	1739	3599	1741	3600	1739
44	3331	1644	3468	1710	3511	1711	3566	1731	3577	1731	3597	1737	3599	1741	3600	1740
45	3335	1650	3449	1711	3528	1728	3581	1744	3591	1745	3597	1734	3599	1737	3600	1737
46	3163	1584	3368	1680	3544	1731	3561	1726	3591	1740	3597	1737	3598	1741	3600	1740
47	3363	1684	3449	1695	3560	1733	3567	1731	3570	1736	3597	1741	3599	1740	3600	1740
48	3368	1669	3449	1710	3538	1731	3543	1735	3584	1731	3597	1740	3592	1735	3600	1740
49	3431	1683	3491	1727	3552	1749	3570	1737	3591	1740	3597	1739	3599	1740	3600	1737
50	3360	1667	3456	1703	3519	1719	3552	1731	3591	1738	3597	1740	3599	1738	3600	1732

Πίνακας 18: Ολοκληρωτικά αποτελέσματα 50 εκτελέσεων για διαγραφή περισσοτέρων των n παιδιών

Αποτελέσματα διαγραφής παιδιών και απογόνων n=1,…,8																
	n=1	n=1 D	n=2	n=2 D	n=3	n=3 D	n=4	n=4 D	n=5	n=5 D	n=6	n=6 D	n=7	n=7 D	n=8	n=8 D
1	17	17	52	36	52	31	561	197	100	1826	992	312	138	3057	1493	17
2	15	15	20	12	84	24	882	981	607	1995	959	31	13	2752	1466	15
3	17	17	36	26	11	7	1202	959	586	11	6	2553	1395	3025	1534	17
4	28	28	17	13	10	6	507	1385	832	1881	1080	2829	1485	3022	5	28
5	4	4	37	33	92	57	25	1092	675	1932	1071	2446	1	2879	1516	4
6	8	8	27	27	264	202	347	979	531	378	199	2654	1396	2971	1535	8
7	29	29	10	5	69	44	202	43	15	1653	865	2612	1421	2629	1382	29
8	26	26	40	36	307	105	606	871	1	1744	980	2629	1393	3001	1550	26
9	7	7	252	234	215	173	18	1131	680	1899	1025	2215	1175	2849	1447	7
10	36	36	62	54	335	250	126	781	525	1867	1015	2632	1382	2940	1463	36
11	8	8	42	38	390	66	147	215	119	7	1	346	146	2942	1465	8
12	9	9	3	3	50	31	558	1411	835	2115	1157	2158	1150	2918	1505	9
13	34	34	170	158	57	39	942	1141	722	1750	1007	13	7	2991	1546	34
14	33	33	183	161	440	354	90	1198	573	2037	1126	2611	1363	3070	1566	33
15	71	40	231	213	31	26	581	1544	916	2109	1130	2743	1390	2896	1510	71
16	123	123	8	6	30	26	862	1490	877	1973	1093	2679	1412	3012	1567	123
17	31	31	39	29	254	141	10	148	81	12	4	1976	1091	3060	1536	31
18	14	14	202	149	6	4	1004	1402	846	2214	1182	2770	1443	3026	1518	14
19	33	33	6	6	404	318	259	24	19	2129	1013	2027	1096	3122	1590	33
20	10	10	32	30	7	4	265	326	223	307	123	2622	1424	3099	1566	10
21	15	15	210	176	662	371	813	116	86	1769	998	281	118	3125	1566	15
22	26	20	71	65	212	168	808	1563	917	99	50	2503	1333	2893	1480	26
23	27	27	73	63	612	451	129	1350	786	1717	944	1938	1081	2227	1234	27
24	55	55	38	34	64	43	280	11	4	2195	1197	2567	1364	2964	1555	55
25	30	30	209	191	372	179	77	116	59	2023	1101	1876	965	2849	1174	30
26	7	7	33	31	80	53	90	465	249	309	132	2583	1418	2960	1541	7
27	3	3	32	24	75	46	70	1240	735	2010	959	2585	1358	2959	1487	3
28	2	2	61	55	484	313	1028	1593	953	10	5	338	138	3091	1598	2
29	5	5	30	28	350	269	359	35	14	2237	1249	2441	1285	2813	1358	5
30	18	18	10	6	464	357	128	1207	735	2321	1215	2731	1456	3205	1648	18
31	16	16	25	23	232	179	303	138	82	1463	845	2765	1408	2888	1507	16
32	17	17	252	232	36	22	897	342	181	216	107	2713	1386	426	166	17
33	17	17	48	42	352	260	67	1228	716	1571	865	2707	1471	2308	1251	17
34	75	40	4	4	386	281	10	72	39	1557	839	2430	1114	2170	1140	75
35	18	18	157	113	43	25	1068	1316	817	1812	984	1985	1085	2965	1522	18
36	5	5	107	96	4	1	101	1689	47	1458	827	2773	1420	3104	1575	5
37	23	23	50	40	455	101	8	454	5	477	319	2105	1175	3094	1560	23
38	7	7	3	1	355	231	722	92	42	1792	1009	1912	1045	1928	1050	7
39	102	102	295	245	99	70	942	1371	821	2146	1161	10	4	2987	1495	102
40	56	56	88	80	549	408	732	21	14	1983	1085	2635	1255	3136	1614	56
41	34	34	375	357	391	307	254	920	546	1653	914	2755	1422	3029	1566	34
42	37	37	196	172	506	377	127	1187	698	2103	1131	2550	1372	9	2	37
43	34	34	27	25	573	397	451	922	569	1838	1051	2245	1197	3092	1581	34
44	44	44	318	224	238	177	56	448	282	1831	1024	2756	1484	2906	1503	44
45	8	8	215	166	424	333	135	139	68	1300	808	2564	1379	2988	1529	8
46	26	26	8	8	22	4	667	1428	886	1886	228	2620	1172	2963	1526	26
47	7	7	8	6	89	59	22	1552	744	2116	1150	2396	1117	2891	1471	7
48	21	21	161	151	119	77	1142	1104	656	1993	1095	2631	1404	2893	1	21
49	29	29	17	11	630	420	554	1229	679	1363	741	2596	1234	2730	1421	29
50	40	40	222	176	60	46	547	1160	711	2235	1241	2604	1382	3161	1577	40

Αποτελέσματα διαγραφής παιδιών και απόονων n=9,…,16																
	n=9	n=9 D	n=10	n=10 D	n=11	n=11 D	n=12	n=12 D	n=13	n=13 D	n=14	n=14 D	n=15	n=15 D	n=16	n=16 D
1	3175	1614	3464	1715	3529	1718	3567	1739	3591	1738	3597	1741	3599	1738	3600	1734
2	3223	1617	3461	1700	3516	1719	3553	1730	3591	1733	3597	1741	3599	1736	3600	1740
3	3278	1591	3356	1655	3538	1729	3575	1738	3590	1738	3597	1740	3599	1740	3600	1740
4	3350	1687	3350	1639	3532	1729	3539	1726	3556	1727	3597	1740	3599	1740	3600	1741
5	3254	1602	3465	1707	3504	1716	3560	1721	3591	1742	3597	1740	3599	1738	3600	1740
6	3211	1567	3475	1714	3533	1726	3529	1724	3591	1741	3597	1737	3599	1746	3600	1739
7	51	25	3508	1713	3525	1722	3577	1736	3585	1738	3597	1743	3599	1740	3600	1741
8	3295	1653	3349	1667	3516	1725	3544	1727	3564	1728	3597	1742	3599	1741	3600	1743
9	3347	1657	3501	1720	3523	1723	3574	1732	3575	1731	3597	1741	3599	1738	3600	1741
10	3293	1654	3365	1652	3468	1711	3565	1733	3585	1738	3597	1740	3599	1737	3600	1740
11	3325	1637	3445	1693	3509	1715	18	7	3585	1735	3597	1740	3599	1741	3600	1744
12	3287	1585	3433	1699	3505	1718	3580	1738	3577	1733	3597	1740	3599	1740	3600	1745
13	3398	1682	3474	1705	3547	1734	3546	1728	3576	1741	3597	1742	3599	1741	3600	1740
14	3326	1654	3465	1712	3489	1696	3577	1732	3584	1733	3597	1732	3599	1740	3600	1737
15	3302	1605	3467	1700	3555	1727	3565	1729	3591	1740	3597	1737	3599	1743	3600	1734
16	3224	1580	3442	1689	3505	1720	3563	1733	3573	1737	3597	1740	3599	1740	3600	1739
17	3194	1615	3414	1689	3510	1720	3554	1722	3577	1732	3597	1740	3599	1740	3600	1738
18	3344	1682	3012	1486	3486	1714	3561	1735	3591	1741	3597	1741	3599	1740	3600	1737
19	3296	1667	3393	1683	3523	1721	3560	1735	3591	1742	3597	1740	3599	1741	3600	1741
20	3291	1614	3456	1700	3528	1729	3573	1729	3591	1753	3597	1741	3599	1741	3600	1742
21	3306	1619	3458	1714	3517	1726	3580	1735	3591	1744	3597	1741	3599	1737	3600	1737
22	3165	1586	3417	1691	3557	1740	3553	1729	3584	1732	3597	1741	3599	1739	3600	1739
23	3336	1651	3411	1709	3533	1724	3564	1734	3576	1737	3597	1739	3599	1740	3600	1741
24	3342	1661	3481	1723	3547	1727	3553	1725	3591	1737	3597	1740	3599	1740	3600	1738
25	3320	1645	3388	1690	3507	1718	3548	1724	3591	1742	3597	1737	3599	1741	3600	1739
26	3247	1597	3493	1721	3526	1723	3575	1739	3591	1744	3597	1740	3599	1741	3600	1741
27	3283	1605	3469	1693	3544	1734	3538	1726	3591	1739	3597	1737	3599	1742	3600	1740
28	3156	1611	3407	1685	3517	1709	3560	1738	3575	1734	3597	1745	3599	1739	3600	1742
29	3136	1536	3436	1684	3531	1721	3573	1735	3584	1732	3597	1741	3599	1738	3600	1740
30	3322	1611	3465	1706	3559	1752	3546	1720	3591	1740	3597	1741	3599	1740	3600	1739
31	3375	1659	3452	1711	3482	1707	3574	1732	3575	1731	3597	1741	3599	1739	3600	1741
32	3287	1636	3504	1725	3527	1717	3542	1727	3576	1735	3597	1739	3599	1741	3600	1737
33	3233	1588	3472	1710	3529	1730	3529	1729	3570	1737	3597	1741	3599	1739	3600	1740
34	3284	1651	3374	1667	3516	1724	3578	1734	3584	1732	3588	1738	3599	1740	3600	1742
35	3259	1632	3443	1712	3528	1733	3540	1725	3575	1732	3597	1741	3599	1744	3600	1740
36	3268	1639	11	2	3538	1720	3520	1715	3591	1741	3597	1739	3599	1737	12	2
37	3275	1631	3460	1703	3541	1735	3561	1732	3591	1744	3597	1742	3599	1734	3600	1740
38	3322	1670	3425	1705	3495	1706	3560	1732	3591	1737	3597	1733	3599	1740	3600	1739
39	3184	1621	3441	1697	3512	1728	3574	1734	3564	1730	3597	1741	3599	1740	3600	1739
40	3285	1641	3399	1690	3542	1740	3540	1724	3591	1736	3597	1740	3599	1741	3600	1737
41	3184	1619	3477	1709	3532	1721	3581	1741	3576	1738	3597	1739	3599	1737	3600	1741
42	3239	1636	3404	1637	3542	1731	3543	1730	3591	1739	3597	1741	3599	1740	3600	1743
43	3319	1665	3418	1691	3529	1731	3550	1723	3591	1745	3597	1741	3599	1739	3600	1740
44	3261	1639	3465	1712	3515	1714	3551	1724	3591	1741	3597	1742	3599	1738	3600	1740
45	2905	1437	3463	1701	3517	1700	3562	1736	3575	1732	3597	1738	3598	1741	3600	1738
46	3242	1620	3431	1702	3540	1725	3542	1726	3577	1731	3597	1738	3599	1741	3600	1741
47	3297	1621	3468	1700	3530	1727	3550	1730	3576	1738	3597	1740	3599	1740	3600	1740
48	3336	5	3433	1702	3536	1725	3579	1737	3591	1740	3597	1742	3599	1738	3600	1737
49	3312	1650	3443	1697	3537	1735	3539	1722	3570	1739	3597	1741	3599	1740	3600	1739
50	3283	1635	3435	1653	3479	1714	3565	1734	3591	1740	3597	1740	3599	1740	3600	1740

Πίνακας 19: Ολοκληρωτικά αποτελέσματα 50 εκτελέσεων για διαγραφή περισσοτέρων των n παιδιών και απογόνων τους

Αποτελέσματα διαγραφής ακμών n=1,…,8																
	n=1	n=1 D	n=2	n=2 D	n=3	n=3 D	n=4	n=4 D	n=5	n=5 D	n=6	n=6 D	n=7	n=7 D	n=8	n=8 D
1	3371	1	3512	11	3558	1	3571	1	3569	5	3585	31	3592	1451	3598	1620
2	3348	1	3449	33	3559	345	3566	34	3573	19	3580	1	3594	1460	3599	4
3	3406	10	3530	9	3513	1	3573	530	3575	58	3585	16	3587	1	3598	1642
4	3370	7	3435	1	3555	11	3577	33	3564	909	3585	1403	3585	1481	3599	1605
5	3361	3	3465	1	3532	1	3519	1	3566	1200	3579	1348	3587	1507	3593	1568
6	3362	5	3497	1	3514	115	3535	632	3571	52	3581	52	3591	1512	3598	1642
7	3389	3	3503	1	3569	1	3528	486	3579	957	3586	5	3586	1492	3593	1622
8	3357	1	3521	14	3502	1	3567	1	3579	217	3580	4	3594	19	3599	1650
9	3370	1	3499	1	312	7	3519	735	3576	1062	3581	7	3587	1471	3598	1559
10	3418	7	3503	2	3515	2	3561	21	3532	898	3587	3	3586	1518	3598	1617
11	3369	3	3479	7	3536	1	3567	635	3580	889	3583	1143	3587	1538	3595	1638
12	3417	1	3542	1	3521	159	3575	3	3578	1229	3582	1271	3588	1532	3598	1565
13	3433	1	3497	14	3523	2	3568	1	3569	56	3586	1387	3588	1511	3599	1616
14	281	2	3442	44	3523	5	3571	4	3526	1189	3583	1381	3593	1567	3594	15
15	23	1	3505	9	3566	1	3530	1	3561	806	3592	1331	3591	1448	3598	27
16	3383	1	3542	29	3563	3	3571	1	3581	9	3586	1411	3586	1492	3595	1622
17	3360	1	3526	5	3572	30	3577	637	3525	1	3577	1209	3585	1497	3599	1623
18	3376	29	3484	4	3478	2	3569	16	3582	1000	3582	1019	3588	1479	3599	17
19	10	4	3425	20	3558	7	3576	2	3580	2	3577	2	3593	1469	3599	1537
20	3437	12	3465	1	3524	144	3566	14	3575	1045	3588	43	3590	1439	3593	1544
21	3405	2	3471	16	3563	4	3576	4	3563	1193	3579	236	3588	1533	3598	1578
22	3417	1	3428	1	3513	3	3567	1	3576	1112	3586	7	3592	1531	3594	1639
23	3368	1	3455	2	3562	36	3558	4	3537	1135	3578	1	3590	1435	12	1
24	3401	1	3507	7	3555	19	3521	1	3559	994	3579	19	3589	1	12	2
25	3357	3	3534	4	3520	7	3570	122	3568	1247	3579	1332	3586	67	3598	1597
26	3397	2	3450	7	3516	10	3570	3	3578	1	3581	1247	3585	5	3593	1548
27	3396	21	3505	112	3556	3	3575	813	3574	1	3581	17	3587	1517	3593	1577
28	3394	14	3501	117	3527	1	3578	736	3575	1059	3587	1102	3592	1421	3599	9
29	3378	5	3463	7	3569	3	3531	976	3576	1096	3584	11	3592	1506	3597	34
30	306	1	3482	90	45	21	3572	884	3576	810	3580	1350	3588	1445	3598	1644
31	3416	1	3426	73	3514	6	3563	5	3572	732	3584	652	3592	1263	3598	10
32	3360	16	3472	30	3511	2	3523	4	3584	7	3584	1252	3588	47	3599	1588
33	3443	11	3535	176	3557	331	3559	1	3568	94	3579	1338	3586	1417	3599	1678
34	3437	1	3476	2	3527	3	3533	140	3575	993	3583	4	3585	1107	3599	1579
35	3426	2	3460	6	3572	14	3498	53	3577	8	3577	1282	3584	1455	3599	1543
36	3382	5	3513	100	3531	20	3569	649	3577	284	3585	1381	3590	1556	3598	1649
37	3363	7	3476	23	12	1	3561	1	3578	1118	3587	448	3587	317	3594	1599
38	3399	5	3489	3	3523	5	3569	47	12	2	3581	1380	3592	1403	3599	1627
39	3425	6	3517	15	3489	16	3572	1	3542	694	3581	959	3593	1453	3599	1551
40	3399	5	3500	278	3565	567	3566	1	3580	945	3588	2	3593	1460	3599	2
41	3367	4	3542	7	3556	129	3528	113	3578	1	3587	1334	3592	1424	3593	1627
42	3373	3	3485	28	3556	11	3563	670	3577	1	3584	1343	3593	1361	3592	1635
43	3369	7	3470	40	3556	111	3526	29	3576	1193	3576	172	3593	9	3594	1605
44	3384	9	3474	1	3564	1	3573	1	3543	5	3585	1302	3592	1537	3599	340
45	3384	1	3531	72	3555	48	3561	1000	3565	16	3587	1285	3593	1575	3592	1580
46	3362	1	3523	12	3559	1	3566	2	3572	23	3579	1206	3587	1414	3600	1559
47	3370	1	3525	1	3517	352	3524	334	3583	71	3579	1028	3593	1347	3598	1592
48	3354	1	3452	61	3563	1	3531	1	3573	1026	3579	43	12	2	3599	14
49	3382	13	3527	7	3511	15	3569	23	3565	1	3585	19	3592	1521	3598	1601
50	3370	1	3466	32	3521	1	3574	2	3536	1	3579	1300	3590	1359	3599	1636

Αποτελέσματα διαγραφής ακμών n=9,…,16																
	n=9	n=9 D	n=10	n=10 D	n=11	n=11 D	n=12	n=12 D	n=13	n=13 D	n=14	n=14 D	n=15	n=15 D	n=16	n=16 D
1	3600	1667	3600	1717	3600	1741	7	1	3600	1741	3600	1741	3600	1743	3600	1741
2	3594	1685	3600	1715	3595	1717	3595	1734	3600	1731	3600	1739	3600	1739	3600	1740
3	3600	1668	3600	1713	3600	1714	3595	1723	3600	1731	3600	1740	3600	1741	3600	1741
4	3594	1661	3600	1719	3595	1728	3600	1728	3600	1740	3600	1741	3600	1738	3600	1743
5	3599	1676	3600	1730	3595	1715	3600	1721	3600	1730	3600	1742	3600	1741	3600	1740
6	3598	1	3600	1710	3599	1723	3600	1730	3600	1743	3600	1741	3600	1738	3600	1739
7	3599	1642	3599	1726	3600	1728	3600	1740	3600	1746	3600	1740	3600	1738	3600	1741
8	3593	1670	3599	1702	3600	1	3600	1737	3600	1734	3600	1741	3600	1741	3600	1740
9	3599	1711	3600	1715	3600	1710	3600	1722	3600	1733	3600	1740	3600	1742	3600	1740
10	3598	1675	3595	1728	3595	1724	3600	1732	3600	1740	3600	1740	3600	1740	3600	1736
11	3594	1707	3600	1713	3600	1738	3600	1734	3600	1737	3600	1740	3600	1746	3600	1744
12	3600	1675	3599	1715	3600	1711	3595	1732	3600	1729	3600	1741	3600	1737	3600	1736
13	3600	1668	3600	1716	12	2	3595	1725	3600	1746	3600	1740	3600	1738	3600	1737
14	3594	1673	3600	1700	3599	1724	3600	1731	3600	1740	3600	1743	3600	1738	3600	1741
15	3600	1663	3600	1725	3600	1729	3600	1726	3600	1735	3600	1744	3600	1742	3600	1740
16	3595	1674	3600	1704	3600	1725	3600	1731	3600	1750	3600	1739	3600	1737	3600	1745
17	3600	1659	3594	1696	3600	1735	3600	1733	3600	1741	3600	1740	3600	1739	3600	1741
18	3599	1602	3600	1722	3600	1728	3600	1736	3600	1734	3600	1740	3600	1741	3600	1737
19	3600	1672	3594	1703	3600	1735	3600	1739	3600	1735	3600	1741	3600	1741	7	2
20	3594	1639	3595	1699	3600	1733	3600	1730	3600	1741	3600	1740	3600	1739	3600	1740
21	3594	1633	3600	1722	3600	1727	3600	1723	3600	1734	3600	6	3600	1741	3600	1740
22	3598	1686	3599	1723	3600	1726	3600	1738	3600	1741	3600	1741	3600	1741	3600	1740
23	3593	1702	3595	1693	3600	1723	3595	1730	3600	1743	3600	1740	3600	1733	3600	1740
24	3595	1675	3600	1716	3600	1736	3600	1732	3600	1732	3600	1745	3600	1738	3600	1741
25	3593	1652	3599	1705	3594	1722	3595	1717	3595	1735	3600	1737	3600	1737	3600	1739
26	3595	1659	3599	1724	3595	1730	3600	1741	3600	1744	3600	1738	3600	1738	3600	1740
27	3600	13	3600	1683	3600	1738	3595	1731	3600	1735	3600	1739	3600	1740	3600	1741
28	3594	1654	3600	1698	3600	1732	3600	1733	3600	1737	3600	1743	3600	1740	3600	1741
29	3599	1692	3595	1708	3600	1721	3600	1727	3600	1736	3600	1739	3600	1739	3600	1741
30	3599	1671	3595	1702	3600	1734	3595	1729	3600	1739	3600	1739	3600	1741	3600	1741
31	3595	1656	3600	1697	3600	1741	3600	1736	3600	1732	3600	1741	3600	1740	3600	1741
32	3600	1605	3600	1724	3600	1727	3600	1737	3600	1737	3600	1741	3600	1734	3600	1740
33	3594	1681	3595	1699	3600	1722	3600	1740	3600	1741	3600	1740	3600	1736	3600	1746
34	3599	1672	3600	1699	3600	1718	3600	1736	3600	1740	3600	1738	3600	1741	3600	1740
35	3599	1680	3595	1688	3600	1719	3595	1734	3600	1730	3600	1742	3600	1739	3600	1738
36	3599	1685	3600	1721	3600	1715	3600	1738	3600	1728	3600	1734	3600	1740	3600	1737
37	3599	1692	3595	3	3600	1721	3595	1729	3600	1735	3600	1740	3600	1739	3600	1733
38	3595	1617	3594	1710	3600	1734	3600	1734	3600	1740	3600	1740	3600	1740	3600	1738
39	3595	1665	3595	1712	3600	1727	3600	1720	3600	1742	3600	1744	3600	1741	3600	1741
40	3599	1675	3600	1713	3600	1729	3595	1728	3600	1733	3600	1740	3600	1740	3600	1745
41	3594	1648	3599	1716	3600	1732	3600	1732	3600	1740	3600	1743	3600	1740	3600	1739
42	3594	1677	7	2	3600	1740	3600	1733	3600	1741	3600	1739	3600	1747	3600	1737
43	3593	1660	3600	1725	3600	1723	3600	1737	3600	1739	3600	1741	3600	1741	3600	1740
44	3599	1676	3595	1697	3595	1737	3600	1734	3600	1737	3600	1746	3600	1742	3600	1739
45	3593	1636	3600	1693	3600	1740	3600	1728	3600	1737	3600	1741	3600	1736	3600	1739
46	3599	1618	3600	1	3600	1728	3600	1726	3600	1742	3600	1740	3600	1740	3600	1738
47	3599	5	3595	1722	3600	1737	3600	1727	3600	1737	3600	1742	3600	1741	3600	1738
48	3594	1642	3599	1701	3600	1719	3600	1740	3600	1732	3600	1740	3600	1741	3600	1740
49	3594	1700	3600	1730	3600	1721	3595	1725	3600	1739	3600	1738	3600	1740	3600	1737
50	3594	1664	3595	1682	3595	1727	3600	1742	3600	1741	3600	1743	3600	1740	3600	1739

Πίνακας 20: Ολοκληρωτικά αποτελέσματα 50 εκτελέσεων για διαγραφή ακμών μεταξύ γονέα-παιδιού

Παράρτημα ΙΙΙ

Τεχνολογίες που χρησιμοποιήθηκαν

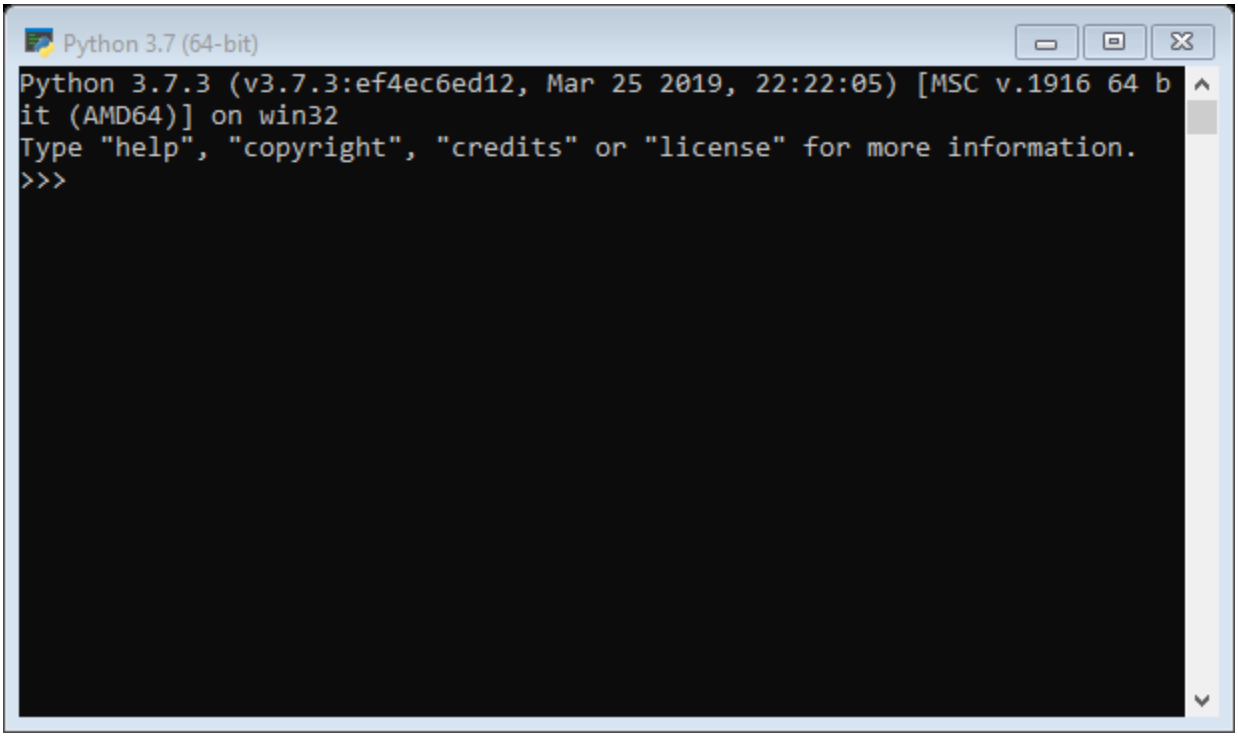
1) Python

Η Python είναι μια δημοφιλής γλώσσα προγραμματισμού. Κάποιες εκ των χρήσεων της, για τις οποίες επιλέχθηκε είναι:

- Διαχείριση μεγάλου όγκου δεδομένων
- Διάβασμα και τροποποίηση αρχείων

Παράλληλα με τα 2 πιο πάνω σημεία, η συγκεκριμένη γλώσσα επιλέχθηκε για αυτή τη διπλωματική και για τους παρακάτω λόγους:

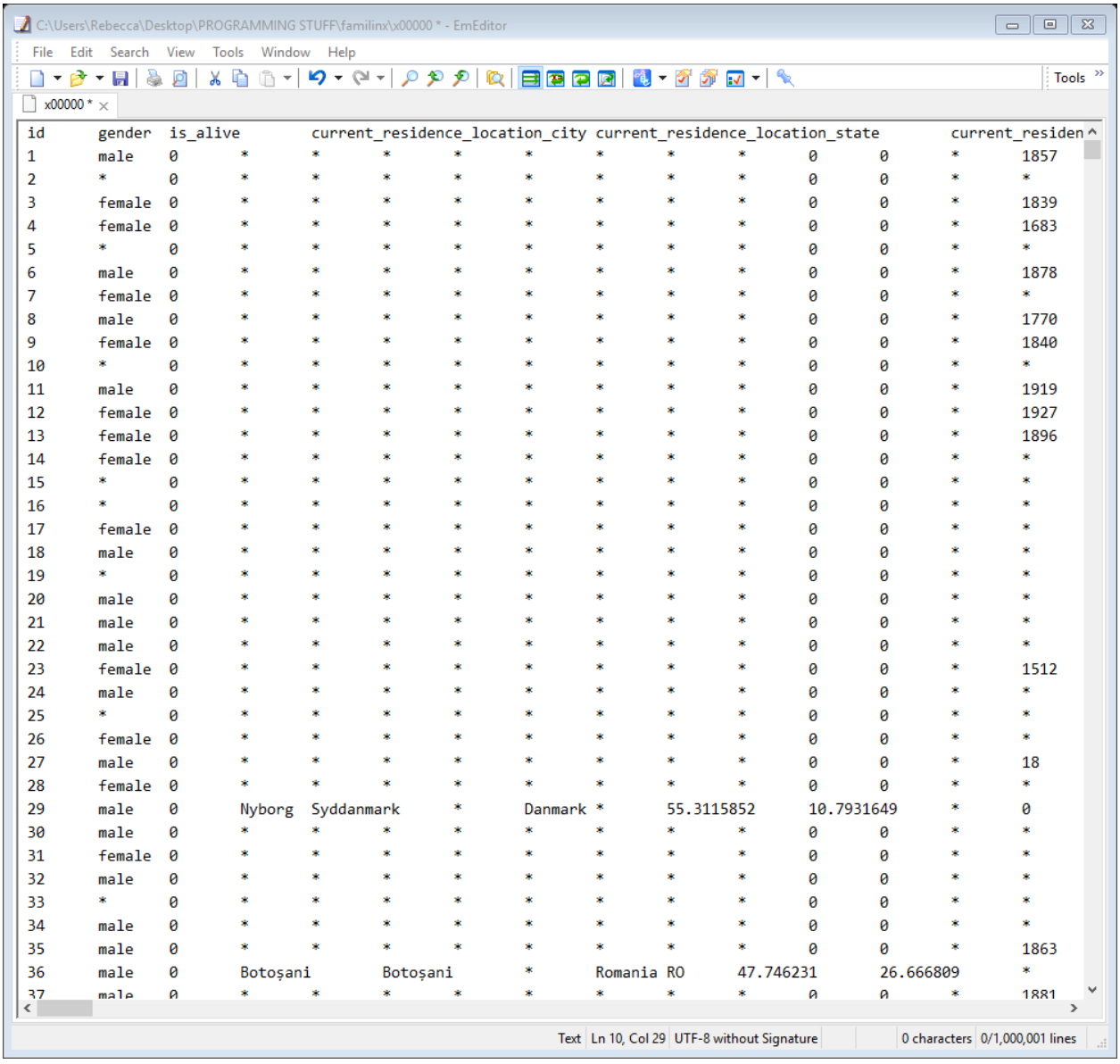
- Λειτουργέι σε όλες τις πλατφόρμες (Windows, Mac, Linux, κτλπ).
- Έχει απλή σύνταξη , παρόμοια με την αγγλική γλώσσα.
- Η σύνταξη της επιτρέπει την γραφή προγραμμάτων σε λιγότερες γραμμές σε σχέση με άλλες γλώσσες.
- Τρέχει σε σύστημα διερμηνέων, δηλαδή ο κώδικας μπορεί να εκτελεστεί μόλις συνταχθεί. Άρα τα πρωτότυπα είναι πολύ γρήγορα.



Εικόνα 18: Γραμμή Εντολών Python

2) EmEditor

Λόγω του μεγέθους του αρχείου «profiles-anon» που είναι 14,6GB, χρειάστηκε να γίνει λήψη ενός editor που να το ανοίγει αφού οι απλοί που είναι εγκατεστημένοι στον υπολογιστή δεν μπορούσαν.



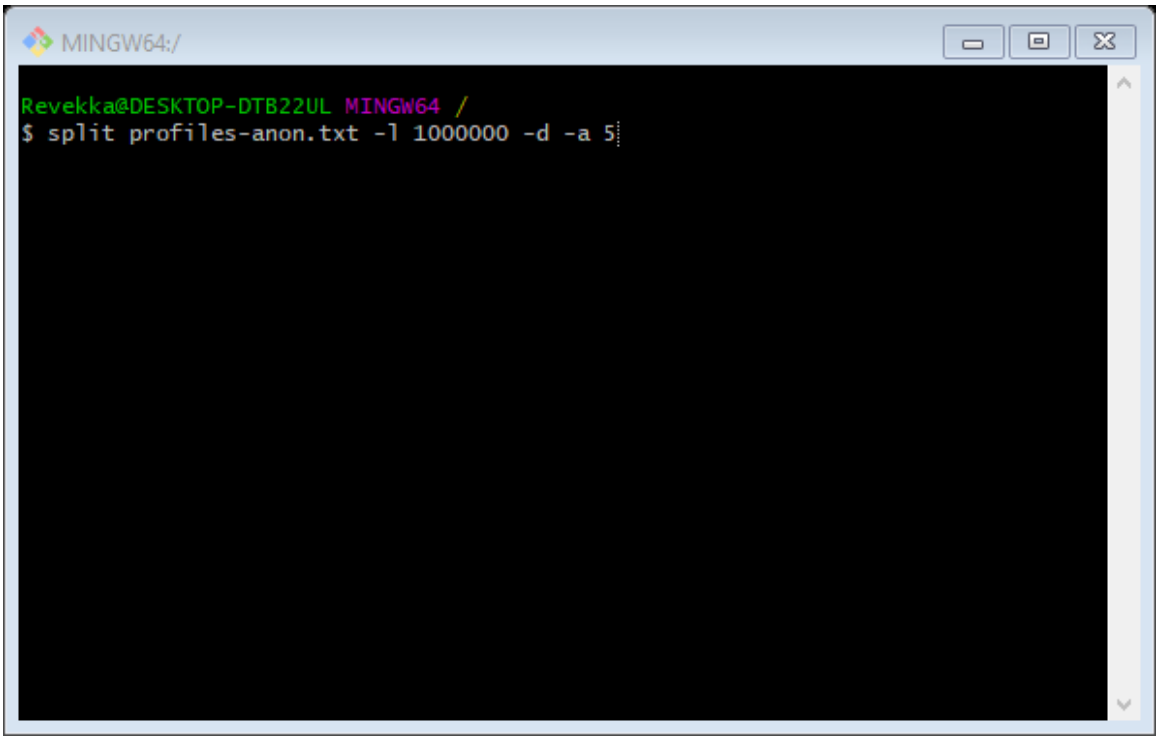
Εικόνα 19: EmEditor

3) GitBash

Πάλι λόγω του μεγέθους του προαναφερθέντος αρχείου, χρησιμοποιώντας το GitBash, έσπασε το «profiles-anon» σε 86 μικρότερα αρχεία περίπου 180MB το κάθε ένα, αφού έτσι θα ήταν πιο εύκολη η χρήση τους μετά στα προγράμματα που θα αναφερθουν. Η εντολή που χρησιμοποιήθηκε για το «σπάσιμο» του αρχείου σε μικρότερα είναι η παρακάτω:
split profiles-anon.txt –l 1000000 –d –a 5
όπου:

- split: η εντολή που χωρίζει το αρχείο σε μικρότερα
- profiles-anon.txt: το αρχείο που θέλουμε να χωρίσουμε
- -l 1000000 : ο αριθμός των γραμμών που θέλουμε να περιέχει το κάθε ένα από τα καινούρια αρχεία
- -d –a 5: τα ονόματα των αρχείων μας θα περιέχουν 5 ψηφία, και θα είναι x00000,x00001,x00002 κοκ.

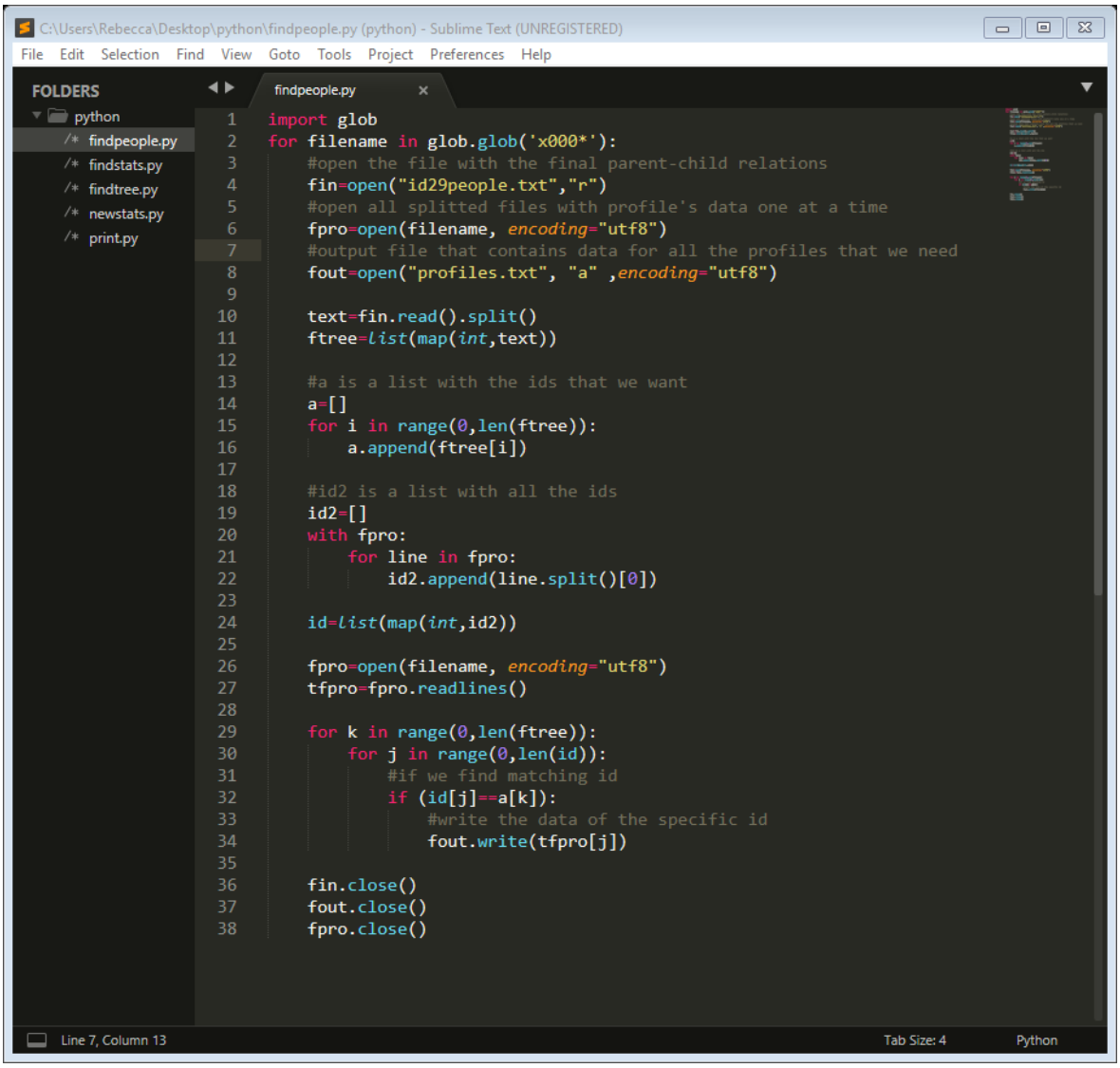
Επίσης χρησιμοποιήθηκε για την εκτέλεση των προγραμμάτων αρκετές φορές με σκοπό την λήψη αρκετών αποτελεσμάτων για την μετέπειτα εισαγωγή τους για εξαγωγή στατιστικών στοιχείων και διαστημάτων εμπιστοσύνης.



Εικόνα 20: GitBash

4) Sublime Text

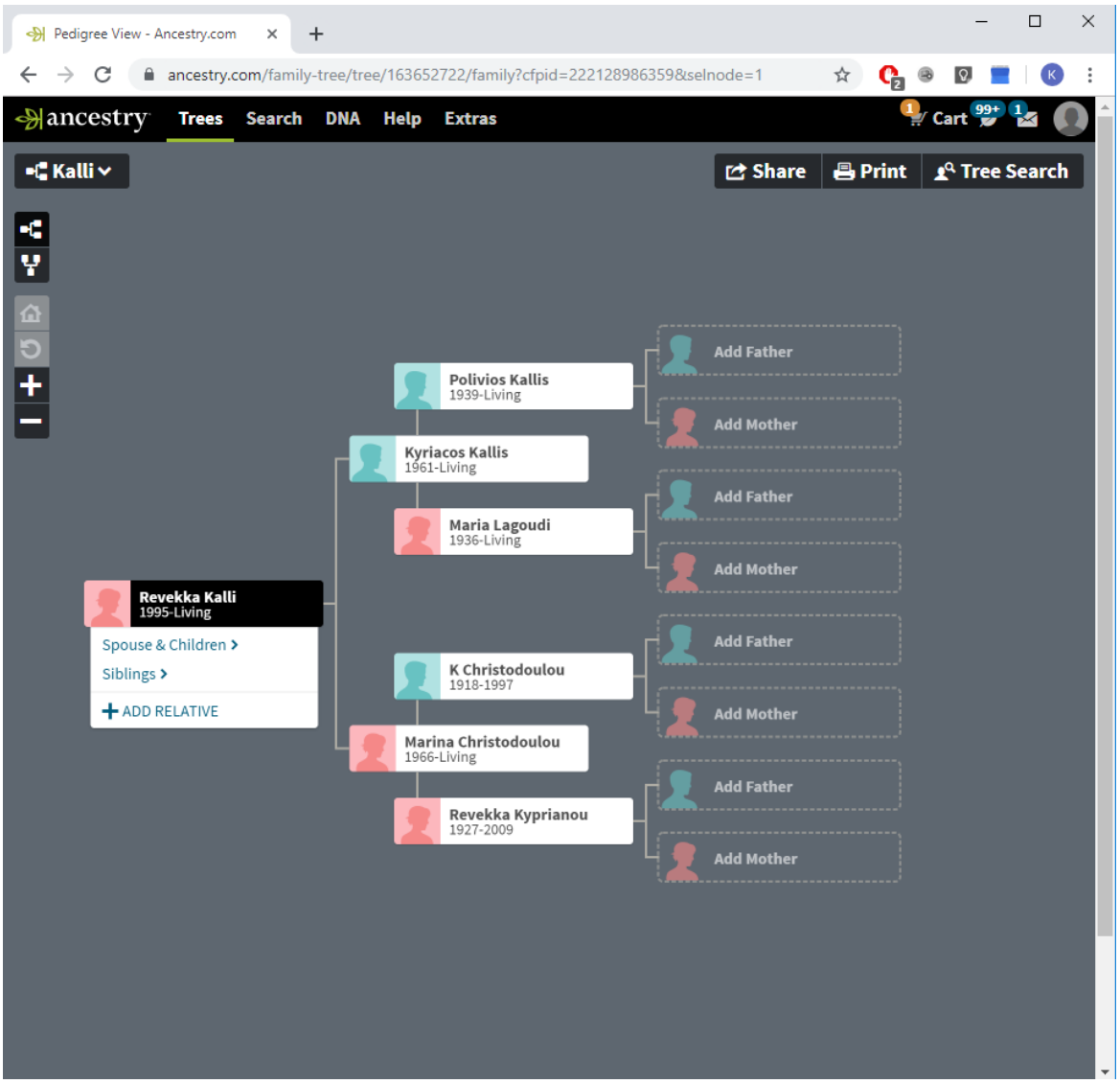
Όλοι οι κώδικες γράφηκαν στο πρόγραμμα Sublime Text το οποίο διατίθεται δωρεάν. Η εκτέλεση του κάθε κώδικα γίνεται από τη γραμμή εντολών του υπολογιστή (command line) ή και απευθείας μέσω της γραμμής εντολών της python.



Εικόνα 21: Sublime Text

5) Ancestry

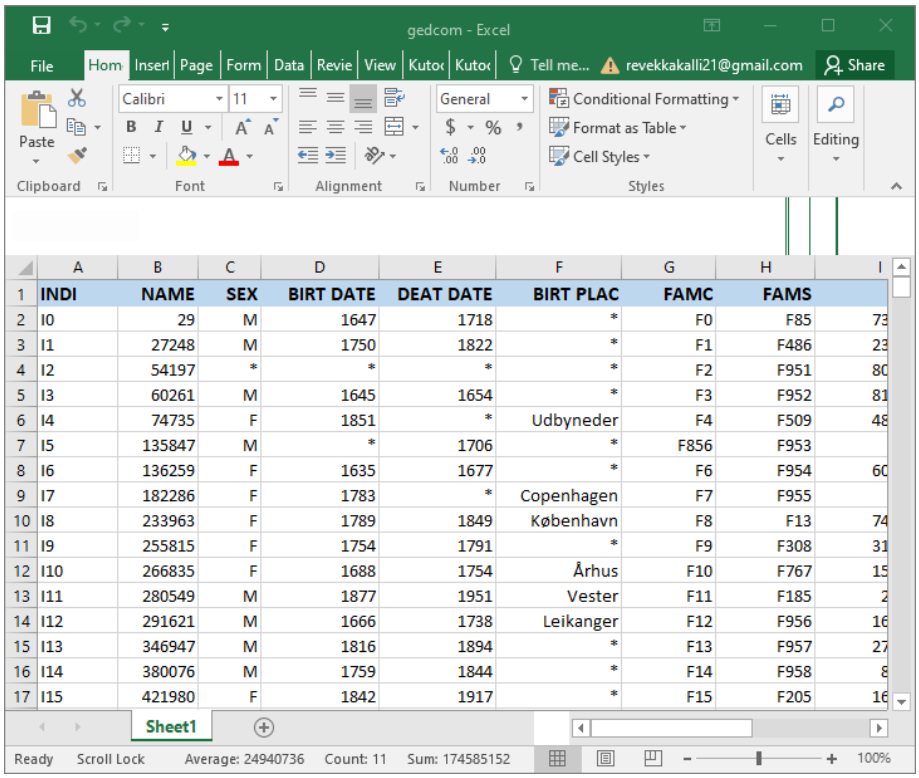
https://www.ancestry.com/ Ιστοσελίδα αναπαράστασης γενεαλογικών δέντρων χρήσιμη για την αναζήτηση/προσθήκη/διαγραφή ενός ατόμου, για την εύρεση των προγόνων και απογόνων του και γενικά για την αναπαράσταση του γενεαλογικού δέντρου.



Εικόνα 22: Ιστοσελίδα https://www.ancestry.com/

6) Excel

Η Excel χρησιμοποιήθηκε για προβολή αποτελεσμάτων, απλή επεξεργασία τους και εύκολη διόρθωση σε συγκεκριμένα δεδομένα κάποιων ατόμων καθώς και για την χρήση πολλών μαθηματικών συναρτήσεων της.



	A	B	C	D	E	F	G	H	I
1	INDI	NAME	SEX	BIRT DATE	DEAT DATE	BIRT PLAC	FAMC	FAMS	
2	I0	29	M	1647	1718	*	F0	F85	73
3	I1	27248	M	1750	1822	*	F1	F486	23
4	I2	54197	*	*	*	*	F2	F951	80
5	I3	60261	M	1645	1654	*	F3	F952	81
6	I4	74735	F	1851	*	Udbyneder	F4	F509	48
7	I5	135847	M	*	1706	*	F856	F953	
8	I6	136259	F	1635	1677	*	F6	F954	60
9	I7	182286	F	1783	*	Copenhagen	F7	F955	
10	I8	233963	F	1789	1849	København	F8	F13	74
11	I9	255815	F	1754	1791	*	F9	F308	31
12	I10	266835	F	1688	1754	Århus	F10	F767	15
13	I11	280549	M	1877	1951	Vester	F11	F185	2
14	I12	291621	M	1666	1738	Leikanger	F12	F956	16
15	I13	346947	M	1816	1894	*	F13	F957	27
16	I14	380076	M	1759	1844	*	F14	F958	8
17	I15	421980	F	1842	1917	*	F15	F205	16

Εικόνα 23: Excel

Βιβλιογραφία

[1] Aho, A. V. . J. E. H. J. D. U. (2014). The Design and Analysis of Algorithms. *Katalog BPS*. https://doi.org/10.1007/s13398-014-0173-7.2

[2] Blackmer, W. S. (2018). EU general data protection regulation. *American Fuel and Petrochemical Manufacturers, AFPM - Labor Relations/Human Resources Conference 2018*. https://doi.org/10.1308/rcsfdj.2018.54

[3] Casas-Roma, J., Herrera-Joancomartí, J., & Torra, V. (2017). A survey of graph-modification techniques for privacy-preserving on networks. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-016-9484-8

[4] Gloor, P. (1997). Elements of Hypermedia Design: Techniques for Navigation & Visualization in Cyberspace. In *Elements of Hypermedia Design: Techniques for Navigation & Visualization in Cyberspace*. https://doi.org/10.1007/978-1-4612-4144-7

[5] Kaplanis, J., Gordon, A., Shor, T., Weissbrod, O., Geiger, D., Wahl, M., … Erlich, Y. (2018). Quantitative analysis of population-scale family trees with millions of relatives. *Science*, *360*(6385), 171–175. https://doi.org/10.1126/science.aam9309

[6] McGuffin, M. J., & Balakrishnan, R. (2005). Interactive visualization of genealogical graphs. *Proceedings - IEEE Symposium on Information Visualization, INFO VIS*. https://doi.org/10.1109/INFVIS.2005.1532124

[7] Rise of the Netflix Hackers | WIRED. (n.d.). Retrieved November 11, 2019, from https://www.wired.com/2007/03/rise-of-the-netflix-hackers/

[8] Smith, D. K., & Jungnickel, D. (1999). Graphs, Networks and Algorithms. The Journal of the Operational Research Society. https://doi.org/10.2307/3010642

[9] UK Parliament. Data Protection Act, 2018 c. 12. , UK law § (2018).

[10] Van Alsenoy, B. (2019). General Data Protection Regulation. In Data Protection Law in the EU: Roles, Responsibilities and Liability. https://doi.org/10.1017/9781780688459.021

[11] Kennedy, M. (1990). GEDCOM-the genealogical standard for data communications. Computers in Genealogy.

[12] El Emam, K., & Dankar, F. K. (2008). Protecting Privacy Using k-Anonymity. Journal of the American Medical Informatics Association. https://doi.org/10.1197/jamia.M2716

[13] Furnas, G. W., & Zacks, J. (1994a). Human Factors in Computing Systems Multitrees : Enriching and Reusing Hierarchical Structure. ACM Conference on Human Factors in Computing Systems (CHI).

[14] Furnas, G. W., & Zacks, J. (1994b). Multitrees: enriching and reusing hierarchical structure. Conference on Human Factors in Computing Systems - Proceedings.

[15] Furnas, G. W., & Zacks, J. (1994c). Multitrees. https://doi.org/10.1145/259963.260396

[16] Kennedy, M. (1990). GEDCOM-the genealogical standard for data communications. Computers in Genealogy.

[17] *Data protection*. n.d. <https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules/eu-data-protection-rules_en>.

[18] Demerjian, Dave. "Rise of the Netflix Hackers." 15 03 07. Wired. <https://www.wired.com/2007/03/rise-of-the-netflix-hackers/>.

[19] Download. n.d. <https://familinx.org/download.html>.

[20] Family History Department, The Church of Jesus Christ of Latter-day Saints. "THE GEDCOM STANDARD , DRAFT Release 5.5.1." (1999). {Formatting Citation}

[21] GEDCOM. n.d. <https://en.wikipedia.org/wiki/GEDCOM>.

[22] General Data Protection Regulation - GDPR. n.d. <https://gdpr-info.eu/>.

[23] Geni. n.d. <https://www.geni.com/>.

[24] "Graph isomorphism." n.d. Wikipedia. <https://en.wikipedia.org/wiki/Graph_isomorphism>.

[25] "Graph isomorphism problem." n.d. Wikipedia. <https://en.wikipedia.org/wiki/Graph_isomorphism_problem#CITEREFAhoHopcroftUllman1974>.

[26] "Tree (graph theory)." n.d. Wikipedia. <https://en.wikipedia.org/wiki/Tree_(graph_theory)>.

[27] Trees. n.d. <https://www.ancestry.com/>.

[28] Williams, Ryan and Manuel Blum. "K-Anonymity." 2007. Carnegie Mellon University, School of Computer Science. <https://www.cs.cmu.edu/~jblocki/Slides/K-Anonymity.pdf>.

[29] —. "Python: 7 Important Reasons Why You Should Use Python." n.d. Medium. <https://medium.com/@mindfiresolutions.usa/python-7-important-reasons-why-you-should-use-python-5801a98a0d0b>.

[30] Syntax Highlight Code In Word Documents. n.d. <http://www.planetb.ca/syntax-highlight-word>.

[31] Machanavajjhala, Ashwin, et al. "ℓ-Diversity: Privacy Beyond k-Anonymity." (n.d.).

[32] Quasi-identifier. n.d. <https://en.wikipedia.org/wiki/Quasi-identifier>.

[33] Lubowicka, Karolina. "The Ultimate Guide to Data Anonymization in Analytics." 13 12 2018. *Piwik PRO.* <https://piwik.pro/blog/the-ultimate-guide-to-data-anonymization-in-analytics/?utm_source=google&utm_medium=cpc&utm_campaign=search_DSA_data-management&gclid=CjwKCAjw1KLkBRBZEiwARzyE7_xJmdnhL0iJdxJWIoGKf5M7iEyLSkMQzZxBLeuikLCV_sXEN5CUdxoCqsoQAvD_BwE>.

[34] Palmer, Danny. "ZDNet." 19 05 2019. *What is GDPR? Everything you need to know about the new general data protection regulations.* <https://www.zdnet.com/article/gdpr-an-executive-guide-to-what-you-need-to-know/>.

[35] *Python*. n.d. <https://www.python.org/>.

[36] *Real Python Tutorials*. n.d. <https://realpython.com/>.

[37] Simi, Ms. M S, Mrs. K Sankara Nayaki and Dr. M. Sudheep Elayidom. "An Extensive Study on Data Anonymization Algorithms Based on K-Anonymity." 2017. *IOPscience.*.

[38] Solutions, Mindfire. "Advantages and Disadvantages of Python Programming Language." n.d. *Medium.* <https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121>.