ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

(ΜΙΧΟΠΟΥΛΟΣ ΣΟΦΟΚΛΗΣ - ΙΩΑΝΝΗΣ, ΦΡΑΝΤΖΗΣ ΑΝΔΡΕΑΣ-ΣΥΜΕΩΝ)

Σχεδιαστικές επιλογές βελτιστοποιήσεων Vamana:

- **Βελτιστοποίηση 1:** Αρχικοποίηση medoid στον Vamana με τυχαία σημεία.
  Στον αλγόριθμο Vamana του πρώτου παραδοτέου υπολογίζαμε το medoid για όλα τα σημεία του αρχείου κάτι το οποίο ήταν αρκετά χρονοβόρο. Στον βελτιωμένο Vamana επιλέγουμε τυχαία (συνόλο σημείων / 10) σημεία και υπολογίζουμε το medoid σε αυτά κάτι το οποίο κοστίζει λιγότερο σε χρόνο χωρίς να μειώνει το ποσοστό των αποτελεσμάτων.
- **Βελτιστοποίηση 2:** Υπολογισμός αποστάσεων (και πιθανή προσωρινή αποθήκευσή τους) με χρήση pthreads.
  Στον βελτιωμένο αλγόριθμο Vamana για κάθε σημείο xq που εκτελείται ο Vamana, πρωτού εκτελεστουν οι GreedySearch και RobustPrune για το xq, υπολογίζουμε και αποθηκέυουμε προσωρινά τις αποστάσεις του xq απο όλα τα υπόλοιπα σημεία με χρήση pthreads που εκτελούνται παράλληλα για μείωση χρόνου. Αυτό έχει ώς αποτέλεσμα την αποφυγή του υπολογισμού της απόστασης 2 συγκεκριμένων σημείων περισσότερες απο μια φορές. Με το ίδιο τρόπο υπολογίζουμε και αποθηκέυουμε τις αποστάσεις των γειτόνων του xq απο όλα τα σημεία για την δέυτερη εκτέλεση του RobustPrune.

Πειράματα αλγορίθμου Vamana:

Για παραμέτρους : R = 10, L = 12, a = 1.2

Vamana **UpdatedVamana**

Κατασκευή γράφου

```
real    1m4.382s
user    0m59.960s
sys     0m0.030s
```

```
real    0m35.064s
user    0m56.948s
sys     0m3.658s
```

Υπολογισμός ποσοστού αποτελεσμάτων

```
ΠΟΣΟΣΤΟ: 92%

real    0m2.275s
user    0m2.150s
sys     0m0.035s
```

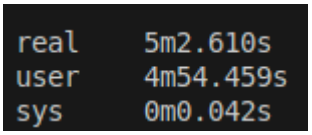
```
ΠΟΣΟΣΤΟ: 91%

real    0m2.234s
user    0m2.215s
sys     0m0.018s
```

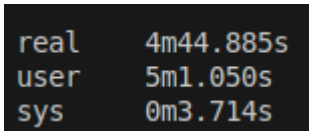
Παρατηρούμε ότι η κατασκευή του γράφου είναι αρκετά ταχύτερη στον βελτιωμένου Vamana και το ποσοστό των αποτελεσμάτων δεν αλλάζει ουσιαστικά.

Για παραμέτρους : R = 50, L = 75, a = 1.2

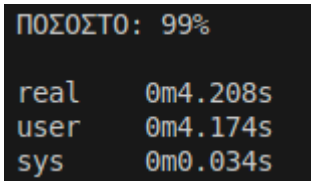
Vamana **UpdatedVamana**

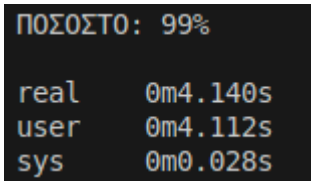
Κατασκευή γράφου





Υπολογισμός ποσοστού αποτελεσμάτων



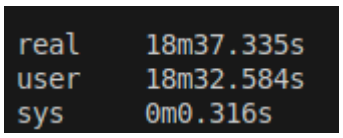


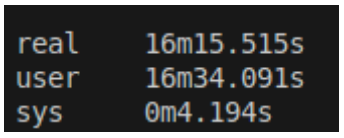
Παρατηρούμε ότι και για μεγαλύτερα ορίσματα η κατασκευή του γράφου είναι αρκετά ταχύτερη στον βελτιωμένου Vamana και το ποσοστό των αποτελεσμάτων δεν αλλάζει ουσιαστικά. Παρ' ολα αυτά η αναλογία της διαφοράς των χρόνων είναι αρκετά μικρότερη σε σχέση με τα παραπάνω μικρότερα ορίσματα κάτι το οποίο δείχνει πως οι διαφορές των χρόνων οφείλονται κυρίως στην πρώτη βελτιστοποίηση δηλαδή αυτή του medoid.

Για παραμέτρους : R = 100, L = 150, a = 1.2

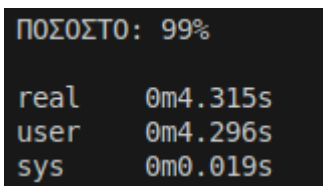
Vamana **UpdatedVamana**

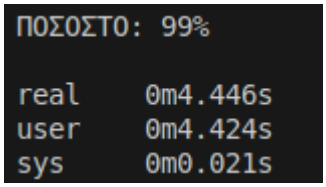
Κατασκευή γράφου





Υπολογισμός ποσοστού αποτελεσμάτων





Παρατηρούμε ότι για ακόμα μεγαλύτερα ορίσματα η αναλογία της διαφοράς των χρόνων μεγαλώνει κάτι το οποίο μας δείχνει πως η δέυτερη βελτιστοποίηση έχει περισσότερο νόημα για μεγαλύτερα ορίσματα. Αυτό το συμπέρασμα έρχεται σε αντίθεση με το συμπέρασμα του προηγούμενου πειράματος.

Σχεδιαστικές επιλογές βελτιστοποιήσεων FilteredVamana:

- **Βελτιστοποίηση 1:** Αρχικοποίηση γράφων με τυχαίες ακμές.
  Στον αλγόριθμο FilteredVamana του δέυτερου παραδοτέου ο γράφος που αρχικοποιείται πρίν την εκτέλεση του αλγορίθμου δεν περιέχει ακμές. Στον βελτιωμένου FilteredVamana πέρα απο τους κόμβους, εισάγουμε στο γράφο τυχαίες ακμές έτσι ώστε να μην δημιουργούνται ανεξάρτητοι υπογράφοι για κάθε φίλτρο αλλά ένας καλά συνδεδεμένος γράφος.
- **Βελτιστοποίηση 2:** Παράλληλη εκτέλεση του FilteredVamana με ένα thread για κάθε φίλτρο. Επειδή στο FilteredVamana συνδέονται σημεία με ίδιο φίλτρο επιχειρήσαμε να τρέξουμε παράλληλα τον αλγόριθμο για κάθε φίλτρο ξεχωριστά ώστε να μειωθεί ο χρόνος εκτέλεσης μέσω pthreads. Έτσι δημιουργούνται ταυτόχρονα οι ανεξάρτητοι υπογράφοι του συνολικού γράφου του ευρετηρίου του αλγορίθμου.

*Οι δύο παραπάνω βελτιστοποιήσεις είναι αδύνατον να δουλέψουν μαζί καθώς άν βάλουμε τυχαίες ακμές μεταξύ σημείων με διαφορετικό φίλτρο, δεν μπορούμε να εκτελέσουμε τη δεύτερη βελτιστοποίηση γιατί δεν θα υπάρχουν ανεξάρτητοι υπογράφοι. Για τον λόγο αυτό όταν εκτελείται ο αλγόριθμος ο χρήστης έχει την δυνατότητα να επιλέξει μια απο τις δύο βελτιστοποιήσεις.

Πειράματα αλγορίθμου FilteredVamana:

*UpdatedFilteredVamana1 → Βελτιστοποίηση 1
*UpdatedFilteredVamana2 → Βελτιστοποίηση 2

Για παραμέτρους : R = 10, L = 12, a = 1.2

FilteredVamana UpdatedFilteredVamana1 UpdatedFilteredVamana2

Κατασκευή γράφου

```
real    0m31.340s
user    0m28.936s
sys     0m0.105s
```

```
real    0m26.142s
user    0m16.436s
sys     0m0.020s
```

```
real    0m16.555s
user    0m28.904s
sys     0m0.061s
```

Υπολογισμός ποσοστού αποτελεσμάτων

```
ΠΟΣΟΣΤΟ Filtered Queries: 98%
ΠΟΣΟΣΤΟ Unfiltered Queries:69%
ΠΟΣΟΣΤΟ όλων των Queries: 83%

real    3m14.052s
user    3m13.914s
sys     0m0.078s
```

```
ΠΟΣΟΣΤΟ Filtered Queries: 97%
ΠΟΣΟΣΤΟ Unfiltered Queries:75%
ΠΟΣΟΣΤΟ όλων των Queries: 86%

real    9m48.672s
user    9m48.574s
sys     0m0.078s
```

```
ΠΟΣΟΣΤΟ Filtered Queries: 98%
ΠΟΣΟΣΤΟ Unfiltered Queries:68%
ΠΟΣΟΣΤΟ όλων των Queries: 83%

real    3m20.725s
user    3m20.661s
sys     0m0.062s
```

Παρατηρούμε πώς ο χρόνος κατασκευής του ευρετηρίου μειώνεται και στις δύο βελτιστοποιήσεις λιγότερο στην πρώτη και περισσότερο στη δέυτερη. Τα αποτελέσματα των ποσοστών δεν έχουν σημαντικές διαφορές όσο αναφορά τον αρχικό FilteredVamana και την δέυτερη βελτιστοποίηση αλλά αργούν σημαντικά στην πρώτη βελτιστοποίηση που οφείλεται στο ότι ο γράφος έχει περισσότερες ακμές.

Για παραμέτρους : R = 50, L = 75, a = 1.2

FilteredVamana UpdatedFilteredVamana1 UpdatedFilteredVamana2

Κατασκευή γράφου

```
real    4m54.918s
user    4m45.029s
sys     0m0.039s
```

```
real    3m47.401s
user    3m22.958s
sys     0m0.046s
```

```
real    2m48.694s
user    5m35.685s
sys     0m0.121s
```

Υπολογισμός ποσοστού αποτελεσμάτων

```
ΠΟΣΟΣΤΟ Filtered Queries: 99%
ΠΟΣΟΣΤΟ Unfiltered Queries:76%
ΠΟΣΟΣΤΟ όλων των Queries: 88%

real    6m18.392s
user    6m18.274s
sys     0m0.078s
```

```
ΠΟΣΟΣΤΟ Filtered Queries: 99%
ΠΟΣΟΣΤΟ Unfiltered Queries:85%
ΠΟΣΟΣΤΟ όλων των Queries: 92%

real    26m34.510s
user    26m33.191s
sys     0m0.712s
```

```
ΠΟΣΟΣΤΟ Filtered Queries: 99%
ΠΟΣΟΣΤΟ Unfiltered Queries:76%
ΠΟΣΟΣΤΟ όλων των Queries: 88%

real    7m0.555s
user    7m0.205s
sys     0m0.164s
```

Παρατηρούμε πώς ο χρόνος κατασκευής του ευρετηρίου μειώνεται και στις δύο βελτιστοποιήσεις λιγότερο στην πρώτη και περισσότερο στη δέυτερη. Τα αποτελέσματα των ποσοστών δεν έχουν σημαντικές διαφορές πέρα απο τα Unfiltered Queries των οποίων το ποσοστό έχει αυξυθεί στην πρώτη βελτιστοποίηση. Επίσης παρατηρούμε πως ο χρόνος του ελέγχου των αποτελεσμάτων αυξάνεται σημαντικά στην πρώτη βελτιστοποίηση πράγμα που είναι λογικό αφου ο γράφος που ελέγχεται έχει περισσότερες ακμες. Το συμπέρασμα της δεύτερης βελτιστοποίησης είναι πως θυσιάζουμε χρόνο για να αυξήσουμε την ακρίβεια.

Για παραμέτρους : R = 100, L = 150, a = 1.2

FilteredVamana UpdatedFilteredVamana1 UpdatedFilteredVamana2

Κατασκευή γράφου

```
real    11m4.944s
user    11m1.053s
sys     0m0.129s
```

```
real    7m0.555s
user    7m0.205s
sys     0m0.164s
```

```
real    6m8.994s
user    12m5.271s
sys     0m0.202s
```

Υπολογισμός ποσοστού αποτελεσμάτων

```
ΠΟΣΟΣΤΟ Filtered Queries: 99%
ΠΟΣΟΣΤΟ Unfiltered Queries:76%
ΠΟΣΟΣΤΟ όλων των Queries: 88%

real    6m58.056s
user    6m57.754s
sys     0m0.181s
```

```
ΠΟΣΟΣΤΟ Filtered Queries: 99%
ΠΟΣΟΣΤΟ Unfiltered Queries:87%
ΠΟΣΟΣΤΟ όλων των Queries: 93%

real    28m10.355s
user    28m9.717s
sys     0m0.276s
```

```
ΠΟΣΟΣΤΟ Filtered Queries: 99%
ΠΟΣΟΣΤΟ Unfiltered Queries:76%
ΠΟΣΟΣΤΟ όλων των Queries: 88%

real    7m1.384s
user    7m1.163s
sys     0m0.116s
```

Τέλος, παρατηρούμε πως ακόμα και για μεγαλύτερα ορίσματα ισχύουν τα παραπάνω συμπεράσματα. Η δέυτερη βελτιστοποίηση βελτιώνει κατα πολύ τον χρόνο και η πρώτη δεν βελτιώνει τόσο τον χρόνο αλλά συμπεριφέρεται καλύτερα σε σχέση με το ποσοστό των αποτελεσμάτων.

Σχεδιαστικές επιλογές βελτιστοποιήσεων StitchedVamana:

- **Βελτιστοποίηση 1:** Αρχικοποίηση medoid στον StitchedVamana με τυχαία σημεία. Στον αλγόριθμο StitchedVamana του δέυτερου παραδοτέου υπολογίζαμε το medoid για όλα τα σημεία του αρχείου για κάθε φίλτρο κάτι το οποίο ήταν αρκετά χρονοβόρο. Στον βελτιωμένο StitchedVamana επιλέγουμε τυχαία το medoid σημείων με το ίδιο φίλτρο κάτι που κοστίζει λιγότερο σε χρόνο χωρίς να μειώνει το ποσοστό των αποτελεσμάτων.
- **Βελτιστοποίηση 2:** Παράλληλη εκτέλεση του StitchedVamana με ένα thread για κάθε γράφο που αντιπροσωπέυει ένα φίλτρο. Επειδή στο StitchedVamana συνδέονται σημεία με ίδιο φίλτρο επιχειρήσαμε να τρέξουμε παράλληλα τον αλγόριθμο για κάθε φίλτρο ξεχωριστά ώστε να μειωθεί ο χρόνος εκτέλεσης μέσω pthreads. Έτσι δημιουργούνται ταυτόχρονα οι ανεξάρτητοι υπογράφοι του συνολικού γράφου του ευρετηρίου του αλγορίθμου.

Πειράματα αλγορίθμου StitchedVamana:

Για παραμέτρους : R = 10, L = 12, a = 1.2

StitchedVamana **UpdatedStitchedVamana**

Κατασκευή γράφου

```
real    0m13.918s
user    0m10.950s
sys     0m0.048s
```

```
real    0m7.003s
user    0m10.517s
sys     0m0.060s
```

Υπολογισμός ποσοστού αποτελεσμάτων

```
ΠΟΣΟΣΤΟ: 99%

real    0m48.850s
user    0m48.297s
sys     0m0.066s
```

```
ΠΟΣΟΣΤΟ: 99%

real    0m48.819s
user    0m48.260s
sys     0m0.063s
```

Παρατηρούμε πως η βελτιστοποιημένη έκδοση μειώνει τον χρόνο χωρίς να υπάρχουν σημαντικές αλλαγές στα ποσοστά των αποτελεσμάτων.

Για παραμέτρους : R = 50, L = 75, a = 1.2

StitchedVamana **UpdatedStitchedVamana**

Κατασκευή γράφου

```
real    3m11.837s
user    3m8.073s
sys     0m0.062s
```

```
real    1m48.905s
user    3m47.572s
sys     0m0.144s
```

Υπολογισμός ποσοστού αποτελεσμάτων

```
ΠΟΣΟΣΤΟ: 99%

real    1m12.974s
user    1m12.796s
sys     0m0.087s
```

```
ΠΟΣΟΣΤΟ: 99%

real    1m16.037s
user    1m15.739s
sys     0m0.130s
```

Παρόμοια συμπεριφορά παρουσιάζεται και εδώ. Παρατηρούμε πως τα ποσοστά των αποτελεσμάτων δεν μεταβάλλονται αλλά ο χρόνος κατασκευής του ευρετηρίου μειώνεται αρκετά.

Για παραμέτρους : R = 100, L = 150, a = 1.2

StitchedVamana **UpdatedStitchedVamana**

Κατασκευή γράφου

```
real    11m53.346s
user    11m49.557s
sys     0m0.530s
```

```
real    6m21.040s
user    14m21.853s
sys     0m0.627s
```

Υπολογισμός ποσοστού αποτελεσμάτων

```
ΠΟΣΟΣΤΟ: 99%

real    1m22.126s
user    1m22.074s
sys     0m0.046s
```

```
ΠΟΣΟΣΤΟ: 99%

real    1m21.576s
user    1m21.335s
sys     0m0.105s
```

Τέλος παρατηρούμε πως ακόμα και για μεγαλύτερα ορίσματα ο χρόνος κατασκευής του ευρετηρίου μειώνεται κατα πολύ και το ποσοστό παραμένει σταθερό.

Αρχείο Groundtrouth

Εκτός απο τα αρχεία που αφορούν την δημιουργία του ευρετηρίου, έχει δημιουργηθεί και ένα αρχείο το οποίο είναι υπέυθυνο για τον υπολογισμό του ποσοστού των αποτελεσμάτων. Η δομή του αρχείο Groundtrouth.cpp χωρίζεται σε τρία κομμάτια. Στο πρώτο κομμάτι ελέγχεται η αποδοτικότητα των γράφων που έχουν δημιουργηθεί απο τον αλγόριθμο του FilteredVamana, στο δεύτερο κομμάτι ελέγχεται η αποδοτικότητα των γράφων που δημιουργούνται απο τον αλγόριθμο StichedVamana και στο τρίτο και τελευταίο κομμάτι ελέγχεται η αποδοτικότητα των γράφων που έχουν δημιουργηθεί απο τον απλό αλγόριθμο Vamana. Ο χρήστης καλείται να επιλέξει τον αλγόριθμο που θέλει να ελέγξει δίνοντας τα κατάλληλα ορίσματα πρίν την εκτέλεση του προγράμματος.

Οδηγίες μεταγλώττισης και εκτέλεσης προγραμμάτων

Σε κάθε φάκελο κάθε αλγορίθμου υπάρχει ένα Makefile το οποίο με την εντολή make μεταγλωττίζει τα αρχεία που χρειάζεται κάθε αλγόριθμος. Με make run εκτελούνται οι μη βελτιστοποιημένοι αλγόριθμοι και με make runUpdated εκτελούνται οι πλέον βελτιστοποιημένοι αλγόριθμοι. Η παραπάνω εκτέλεση είναι interactive, δηλαδή ο χρήστης καλείται να δώσει τα ορίσματα ένα ένα. Επίσης οι αλγόριθμοι μπορούν να εκτελεστούν και με command line options με τους παρακάτω τρόπους.

Vamana: ./Vamana ../Data/siftsmall/siftsmall/siftsmall_base.fvecs
../Data/siftsmall/siftsmall/siftsmall_query.fvecs
../Data/siftsmall/siftsmall/siftsmall_groundtruth.ivecs R L a
UpdatedVamana: ./UpdatedVamana ../Data/siftsmall/siftsmall/siftsmall_base.fvecs
../Data/siftsmall/siftsmall/siftsmall_query.fvecs
../Data/siftsmall/siftsmall/siftsmall_groundtruth.ivecs R L a

FilteredVamana: ./FilteredVamana R L a
UpdatedFilteredVamana: ./UpdatedFilteredVamana R L a

StitchedVamana: ./StitchedVamana R L a
UpdatedStitchedVamana: /UpdatedStit chedVamana R L a

Groundtrouth:

Για Vamana: ./CheckGroundtrouth 3 graph_name 100
Για FilteredVamana: ./CheckGroundtrouth 1 graph_name k
Για StitchedVamana: ./CheckGroundtrouth 2 ./graph_folder k

Περιεχόμενα

./Data → φάκελος με τα δεδομένα που χρησιμοποιούνται
./FilteredVamana →υλοποίηση αλγορίθμου FilteredVamana
./StitchedVamana →υλοποίηση αλγορίθμου StitchedVamana
./Vamana →υλοποίηση αλγορίθμου Vamana
./Groundtrouth →υλοποίηση προγράμματος υπολογισμού ποσοστών των παραπάνω αλγορίθμων
./Test → αρχείο που περιέχει τα test για τις συναρτήσεις και τις δομές που χρησιμοποιούνται