Όνοματα : Κωνσταντίνου Ευστάθιος – Νικόλαος Κολλάτος
ΑΕΜ : 03256 - 03353
Έτος 2ο
Τελική Εργασία 2022

Γενικές Σημειώσεις:

- Τα αρχεία από τις υλοποιήσεις των διαφόρων παραλαγών του quicksort βρίσκονται στο συμπιεσμένο αρχείο που στάλθηκε και στον εξης υπερσύνδεσμο: https://github.com/stathiskon40/algo-project
- Για να τρέξουν τα οι διάφορες παραλλαγες αρκεί να εκτελεστεί το αρχείο project.py ή εναλλακτικά μπορεί να εκτελεστει η εντολή (bash script.sh) ώστε να εγκατασταθούν τα κατάλληλα modules που χρησιμοποιούνται για τον κώδικα
- Αφού ολοκληρωθεί το project.py (υπάρχουν μπάρες που υποδηλώνουν την πρόοδο της κάθε παραλλαγής για ταξινόμηση 50 τυχαίων πινάκων) παράγονται δύο αρχεία (final.csv,mean_variance.txt).
- Το αρχείο final.csv παράγει τον αναλυτικό πίνακα με τους χρόνους, τις αναθέσεις και τις συγκρίσεις που έχουν γίνει για καθένα από τα διαφορετικά μεγέθη για καθένα από τους διαφορετικούς πίνακες και για καθένα από τους διαφορετικούς τρόπους.
- Το αρχείο mean_variance.txt περιέχει τον μέσο χρόνο και την διασπορά για τις αναθέσεις, συγκρίσεις και αναθέσεις για κάθε μέθοδο.
- Οι ψευδοκώδικες – κώδικες βρίσκονται σε καθένα από τα αρχεία του project καθώς έχει επιλεγεί ως γλώσσα η python ώστε να είναι εύκολα κατανοητή και ξεκάθαρη περί των αλγορίθμων που έχουμε υλοποιήσει (χωρίς να μπλεκουμε με low level γλώσσες που μπορεί να είχαν καλύτερες επιδόσεις αλλά θα ήταν λιγότερο κατανοητές ως προς τους αναγνώστες του κώδικα).

Λίγα λόγια για τις μεθόδους που χρησιμοποιήθηκαν:

- QS1pL: Κλασσική μέθοδος Quicksort όπου επιλέγεται ως pivot το αριστερότερο στοιχείο του πίνακα.
- QS1pR: Μέθοδος όπου επιλέγεται ένα τυχαίο index για pivot (μέσω γεννήτριας ψευδοτυχαίων αριθμών). Αφού επιλεχθεί το pivot γίνεται swap με το αριστερότερο στοιχείο του πίνακα και έπειτα εφαρμόζεται η τεχνική QS1pL αφού πλέον το pivot μας έχει αλλάξει σε αυτό που επιλέξαμε με ψευδοτυχαίο τρόπο.
- QS1pM: Μέθοδος στην οποία επιλέγονται τρία index (μέσω γεννήτριας ψευδοτυχαίων αριθμών) από pivots και ως τελικό επιλέγεται η μεσαία τιμή των τριών. Αφού υπολογίσουμε το τελικό index του pivot ομοίως με την μέθοδο

QS1pR κάνουμε swap του pivot με το αριστερότερο στοιχείο και εφαρμόζουμε QS1pL.

- QS2pR: Μέθοδος στην οποία επιλέγονται 2 pivots (μέσω γεννήτριας ψευδοτυχαίων αριθμών). Σε κάθε αναδρομική κληση του αλγορίθμου όλα τα στοιχεία που είναι μικρότερα από το μικρότερο pivot μπαίνουν σε κάποιον πίνακα (έστω A) αυτά που είναι ανάμεσα στα δυο pivots μπαίνουν σε κάποιο πίνακα (έστω B) και αυτά που είναι μεγαλύτερα από το μεγαλύτερο pivot μπαίνουν σε κάποιον πίνακα (έστω C). Τότε αρκεί να κάνουμε απλά merge των:

  [πίνακας A] + [μικρότερο pivot] + [πίνακας B] + [μεγαλύτερο pivot] + [πίνακας C]

  Στην κάθε επόμενή αναδρομική κλήση περνάμε ως παράμετρο στην συνάρτηση καθέναν από τους πίνακες A,B,C

- QS3pR: Μέθοδος παρόμοια με την QS2pR. Συγκεκριμένα σε κάθε αναδρομική σχέση ισχύει:

  Έστω ότι οι τιμές των pivots είναι η εξής: $p_1 < p_2 < p_3$
  - Τα στοιχεία k για τα οποία ισχύει $k < p_1$ μπαίνουν σε ένα πίνακα A
  - Τα στοιχεία k για τα οποία ισχύει $p_1 < k < p_2$ μπαίνουν σε ένα πίνακα B
  - Τα στοιχεία k για τα οποία ισχύει $p_2 < k < p_3$ μπαίνουν σε ένα πίνακα C
  - Τα στοιχεία k για τα οποία ισχύει $k > p_3$ μπαίνουν σε ένα πίνακα D

  Τότε αρκεί να κάνουμε merge των:

  [πίνακας A] + p_1 + [πίνακας B] + p_2 + [πίνακας C] + p_3 + [πίνακας D]

  Στην κάθε επόμενή αναδρομική κλήση περνάμε ως παράμετρο στην συνάρτηση καθέναν από τους πίνακες A,B,C,D

- QS2pRPre: Μέθοδος η οποία βασίζεται στην μέθοδο QS2pR. Πιο συγκεκριμένα επιλεγουμε κάποια indexes απο υποψήφια pivots στην αρχή του αλγορίθμου πλήθους $\sqrt{n}$.

  Αφού αυτά επιλεγούν γίνεται η τελική τους τοποθέτηση στον πίνακα χωρίς ποτέ να παραβιάζεται η συνθήκη του Quicksort (τα πιο μικρά στοιχεία κάθε pivot βρίσκονται αριστερά και τα πιο μεγάλα στοιχεία δεξιά του πίνακα). Έτσι βγάζουμε το συμπέρασμα πως γίνεται μία μερική αναδιάταξη του πίνακα και τα pivots που έχουν επιλεγεί τοποθετούνται στις τελικές τους θέσεις (στις θέσεις που θα είχαν σε έναν ταξινομημένο πίνακα).

  Αφού έχουμε τοποθετήσει τα pivots στις σωστές τους θέσεις έχουμε διαμερίσει τον πίνακα σε $\sqrt{n}$ υποπίνακες. Ως υποπίνακα ορίζουμε έναν πίνακα απο την αρχή πίνακα που δουλεύουμε μέχρι το πρώτο ταξινομημένο pivot και

μετά από το ένα ταξινομημένο pivot μέχρι το επόμενο (χωρίς να συμπεριλαμβάνονται τα pivots).

Έπειτα ταξινομούμε κάθε υποπίνακα βάση της μεθόδου QS2pR

- QS3pRPre: Ίδια μεθοδος με την QS2pRPre απλά εδώ ταξινομούμε κάθε υποπίνακα βάση της μεθόδου QS3pR.

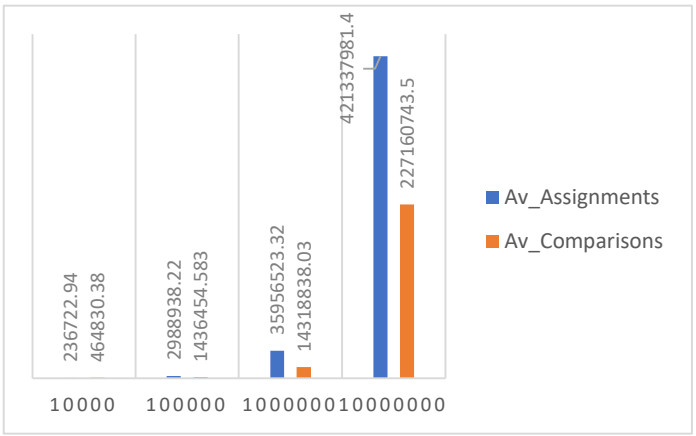
Σύγκριση των επιδόσεων των διαφορετικών μεθόδων:

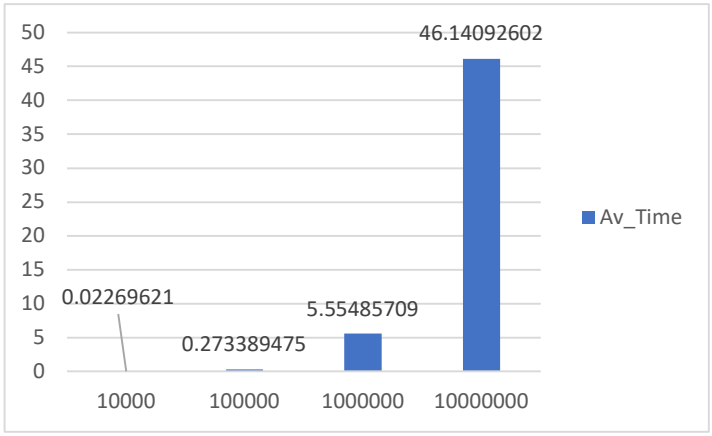
Σημείωση: Τα παρακάτω μετρικά αντλήθηκαν μετά από εκτέλεση όλων των μεθόδων στο εξής σήστημα:

- CPU : AMD Ryzen 3700X 8-Core @4.2GHz
- RAM : 32GB 3600 MHz
- Operating System: Windows 10 Pro
- Cache Size: 32MB

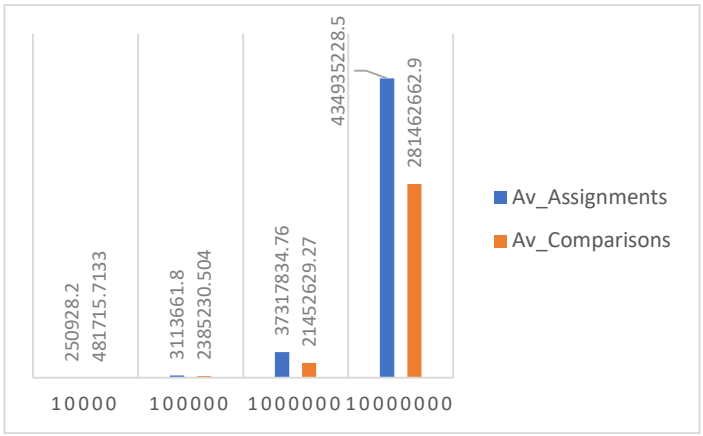
Παρακάτω βλέπουμε τα διαγράμματα των μέσων αναθέσεων-συγκρίσεων και χρόνου (σε s) για καθεμία από τις μεθόδους

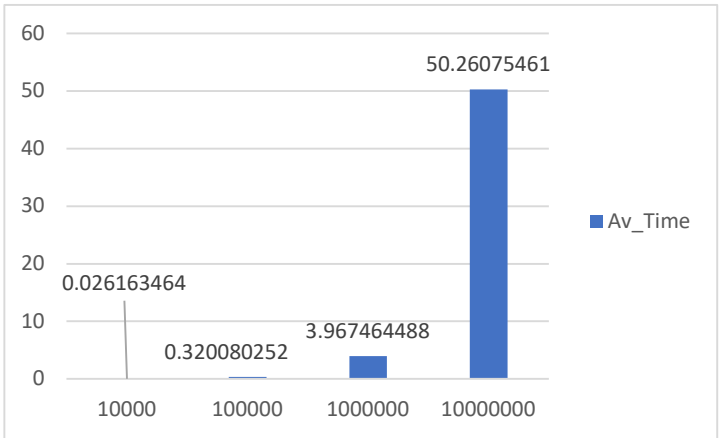
- QS1pL:



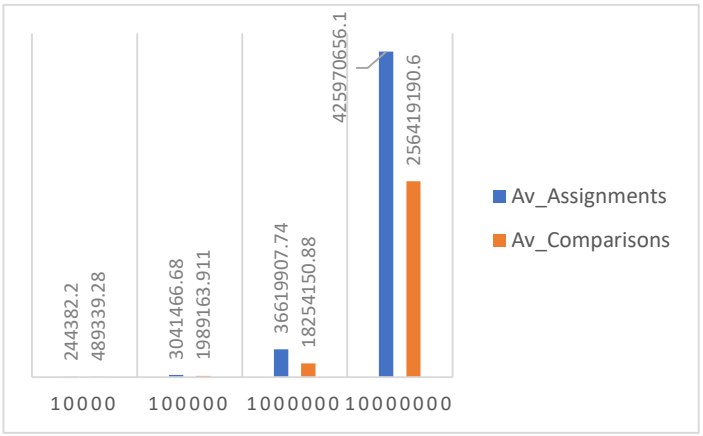


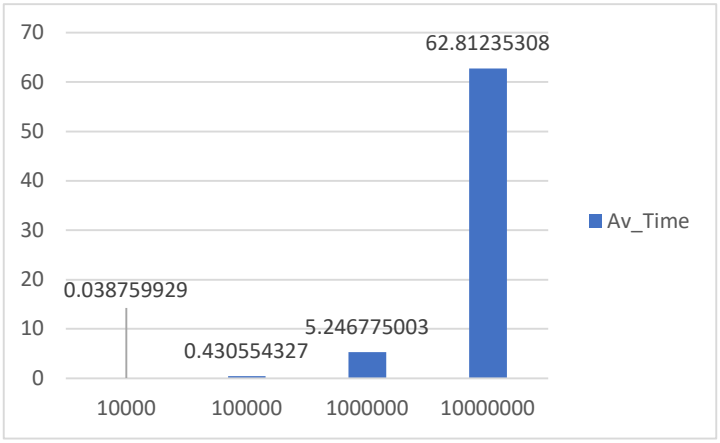
- QS1pR:



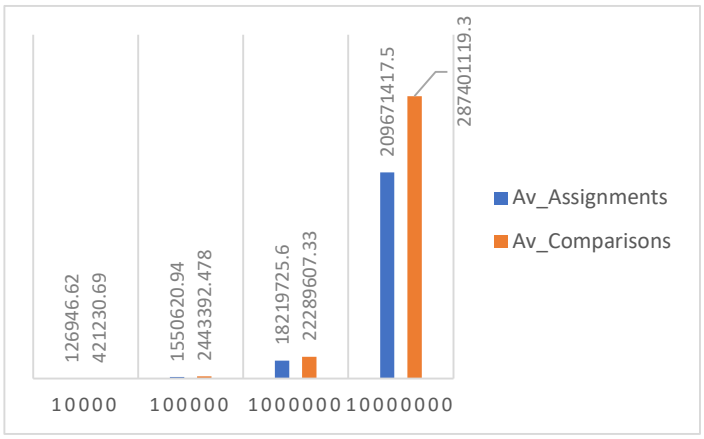


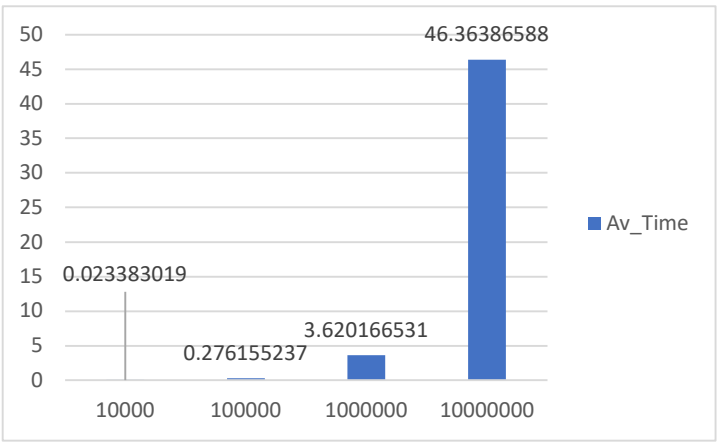
QS1pM:



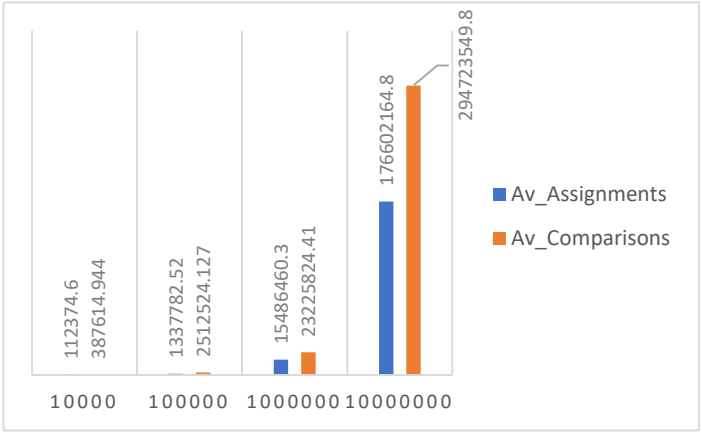


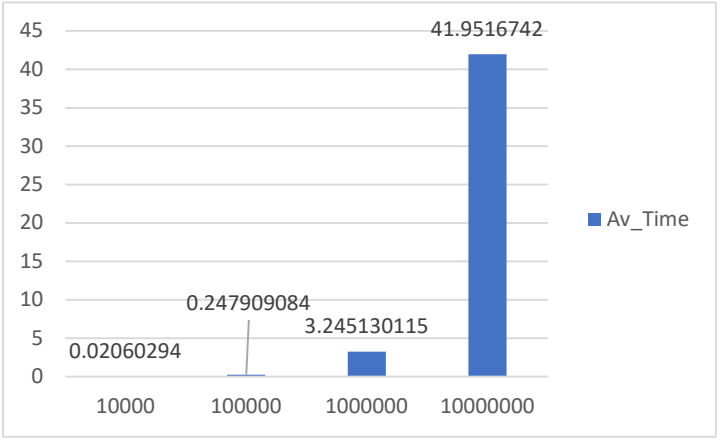
QS2pR:



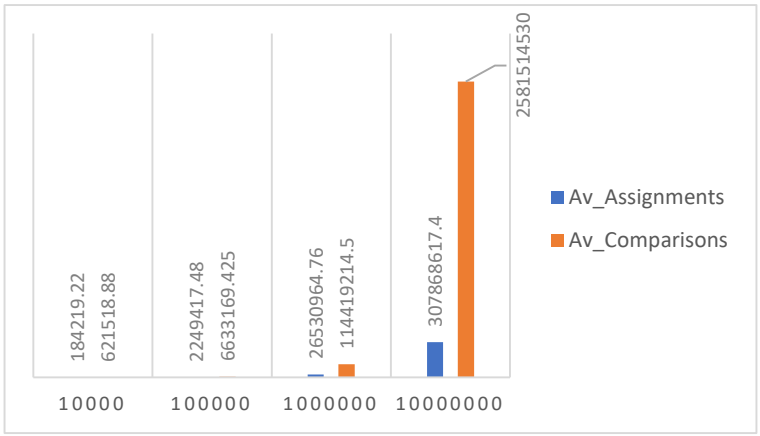


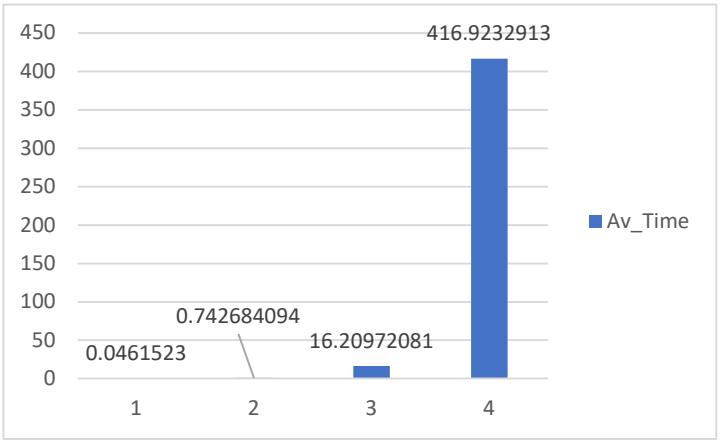
QS3pR:



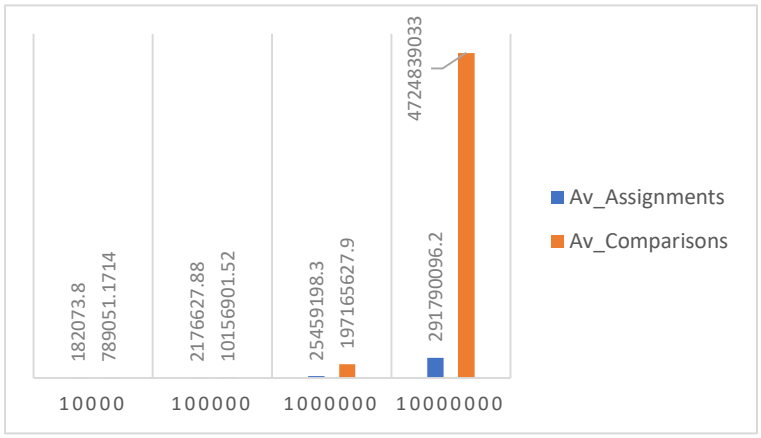


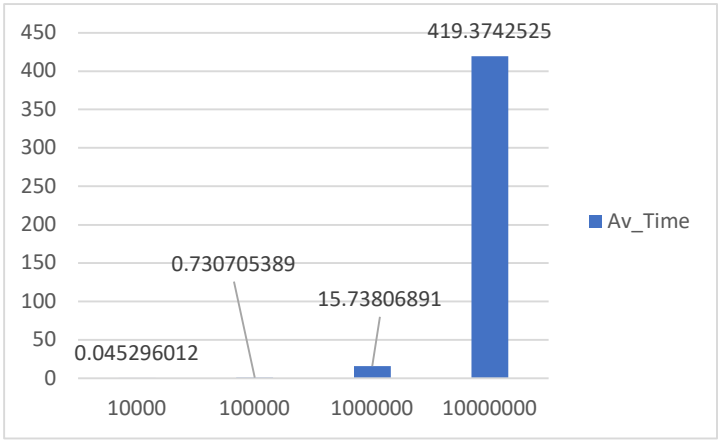
QS2pRPre:



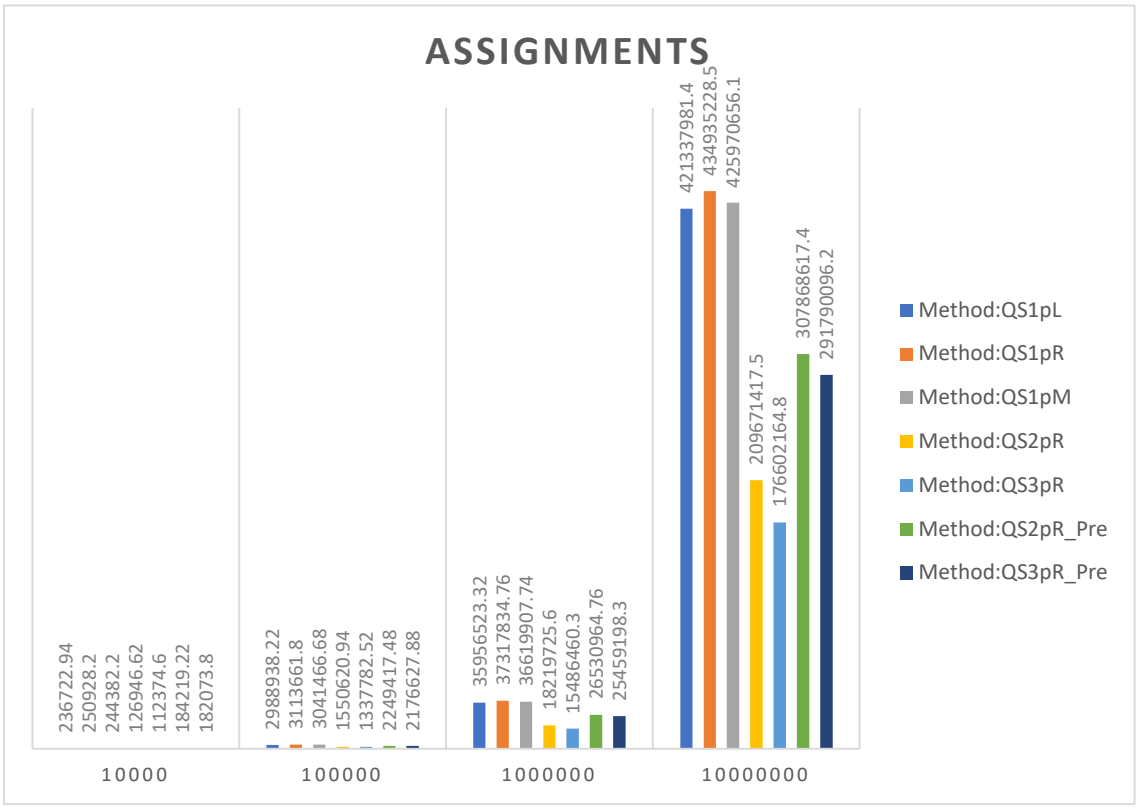


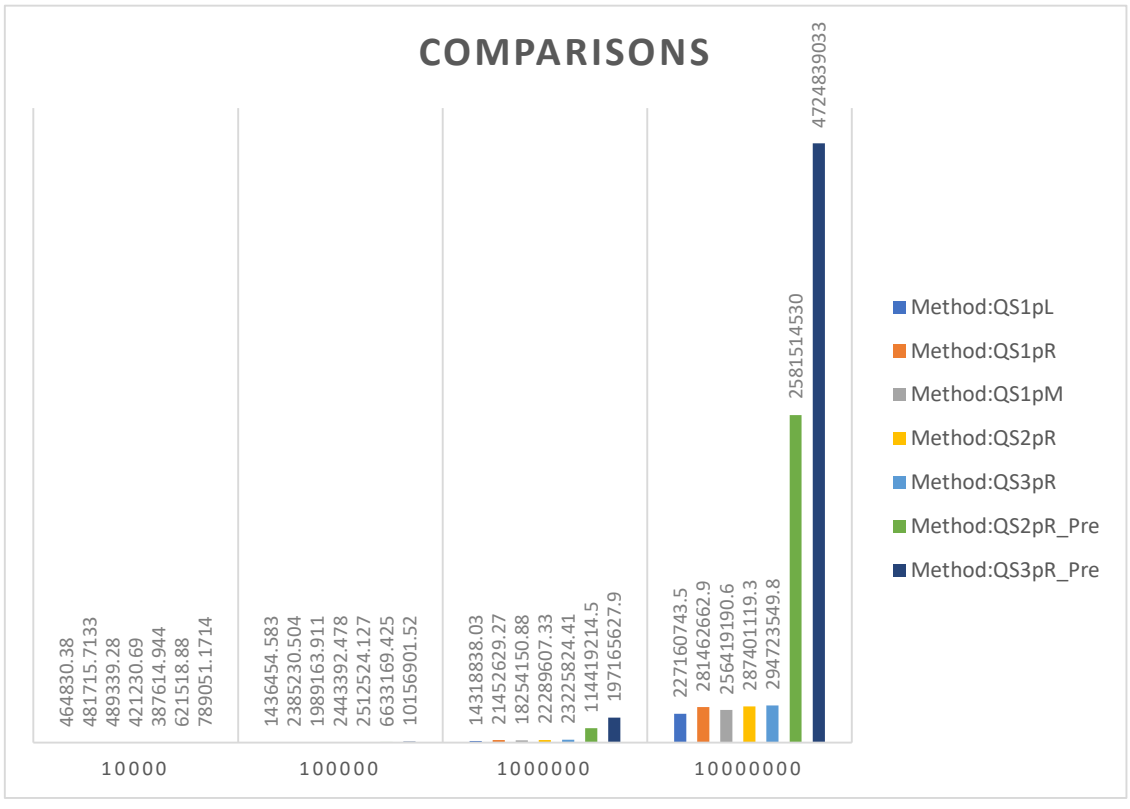
QS3pRPre:

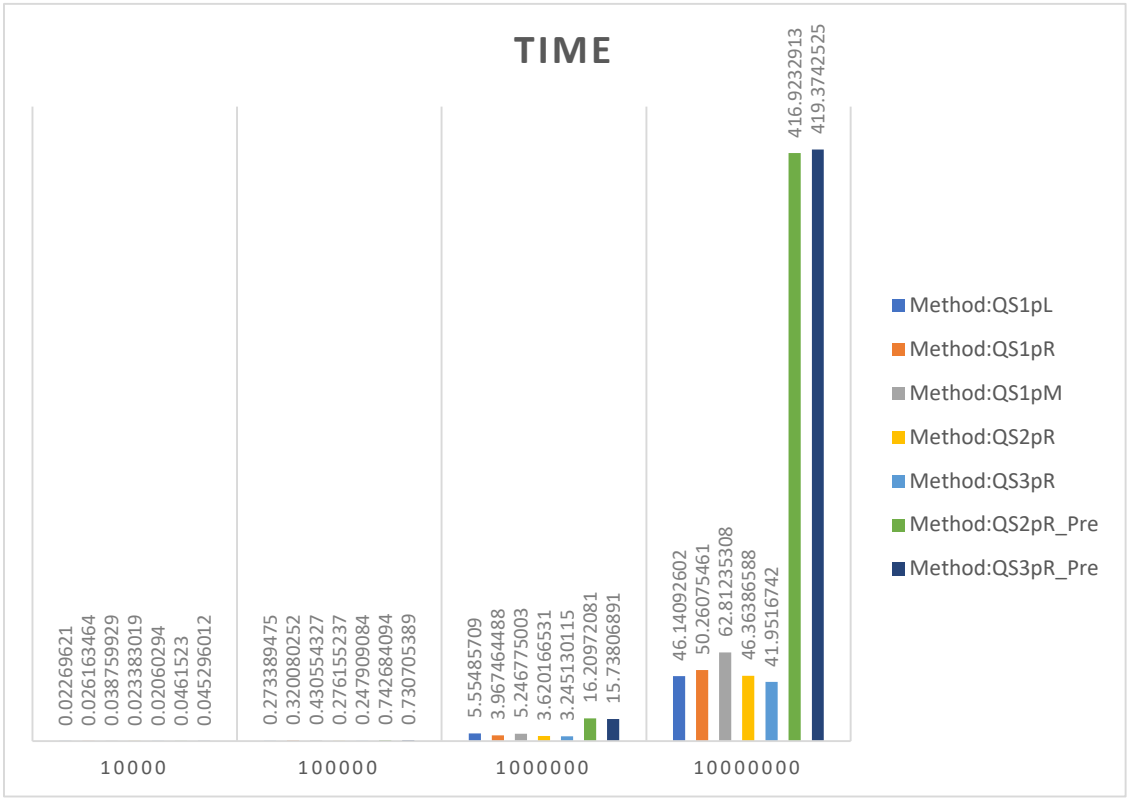




Τελικά 3 διάγράμματα που παρουσιάζουν τον μέσο όρο των συγκρίσεων ,των χρόνων και μεταθέσεων για όλους τους ανταγωνιστές είναι:


ASSIGNMENTS
236722.94
250928.2
244382.2
126946.62
112374.6
184219.22
182073.8
2988938.22
3113661.8
3041466.68
1550620.94
1337782.52
2249417.48
2176627.88
35956523.32
37317834.76
36619907.74
18219725.6
15486460.3
26530964.76
25459198.3
421337981.4
434935228.5
425970656.1
209671417.5
176602164.8
307868617.4
291790096.2
10000
100000
1000000
10000000
Method:QS1pL
Method:QS1pR
Method:QS1pM
Method:QS2pR
Method:QS3pR
Method:QS2pR_Pre
Method:QS3pR_Pre


COMPARISONS
464830.38
481715.7133
489339.28
421230.69
387614.944
621518.88
789051.1714
1436454.583
2385230.504
1989163.911
2443392.478
2512524.127
6633169.425
10156901.52
14318838.03
21452629.27
18254150.88
22289607.33
23225824.41
114419214.5
197165627.9
227160743.5
281462662.9
256419190.6
287401119.3
294723549.8
2581514530
4724839033
10000
100000
1000000
10000000
Method:QS1pL
Method:QS1pR
Method:QS1pM
Method:QS2pR
Method:QS3pR
Method:QS2pR_Pre
Method:QS3pR_Pre



<u>Κάποια σχόλια σχετικά με τις επιδόσεις των αλγορίθμων:</u>

- Παρατηρούμε έναν τεράστιο αριθμό συγκρίσεων στους αλγορίθμους QS2pRPre και QS3pRPre. Αυτό μπορούμε να το αποδώσουμε σε 3 διαφορετικούς παράγοντες:
    - Προσπαθήσαμε να ακολουθήσουμε αυστηρά το πλαίσιο της άσκησης. Στην αρχή του αλγορίθμου επιλέξαμε τα index των pivots και όχι τις τιμές των pivots ώστε να είμαστε απόλυτα συγκεκριμένοι στο ποια στοιχεία πρόκειται να ταξινομήσουμε. Αυτό σημαίνει πως για οποιαδήποτε αλλαγή πραγματοποιούταν στον πίνακα θα έπρεπε να ανανεώσουμε και τον πίνακα οπου διατηρούσε την τωρινή θέση των pivots έχοντας ως αποτέλεσμα την πολλαπλή διάτρεξη του πίνακα που διατηρούσε τα pivots.
    - Μετά απο πειραματική μελέτη καταλήξαμε στο συμπέρασμα πως οι υποπίνακες που φτιάχνονται μετά την ταξινόμηση είναι μερικώς ταξινομημένοι και εμπεριέχουν πολλά ίδια στοιχεία. Αυτό σημαίνει πως παρόλο που δεν έχουμε πολλές μεταθέσεις (όπως φαίνεται και στα διαγράμματα) έχουμε πολλαπλές συγκρίσεις καθώς γνωρίζουμε πως όσο πιο ταξινομημένος είναι ο πίνακας τόσο χειρότερη επίδοση έχει ο Quicksort που οφείλετε στις πολλαπλές συγκρίσεις το ίδιο ισχύει και για τα ίδια στοιχεία.
    - Η προταξινόμηση κάνει αλλαγές στον πίνακα καθώς έχουμε πάντα στον νου μας ότι υλοποιούμε μία παραλλάγή του Quicksort. Επομένως οι

συνθήκες που ορίζουν τον αλγόριθμο δεν θα πρέπει να παραβιάζονται (στοιχεία μικρότερα του pivot αριστερά απο το pivot, στοιχεία μεγαλύτερα του pivot δεξιά του pivot).

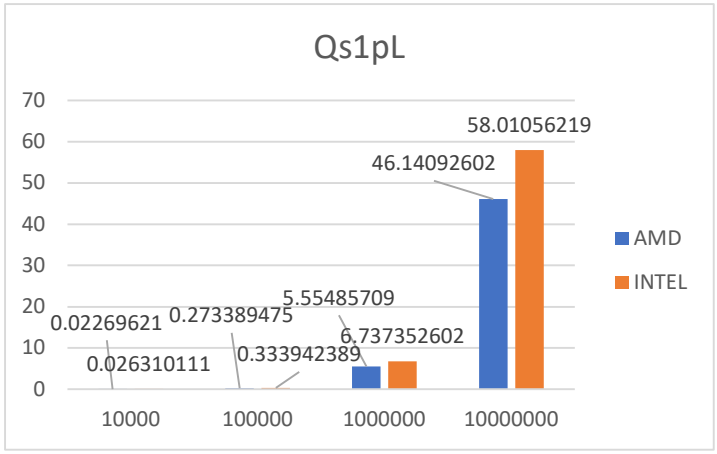
- Κάποιοι τρόποι ώστε να βελτιώσουμε τα παραπάνω προβλήματα με τον κίνδυνο όμως εξτρά πολυπλοκότητας (ανάλογα με τον τρόπο υλοποίησης) είναι:
    - Πρίν ταξινομηθεί ο πίνακας να γίνεται ένα shuffle των στοιχείων του υποπίνακα ώστε να μπορέσουμε να αναιρέσουμε την μερική ταξινόμηση που υπάρχει
    - Μια πιο χαλαρή επιλογή (ίσως με την τιμή και όχι με το index) των αρχικών pivots που πρόκειται να προταξινομηθούν (με τον κίνδυνο των πολλαπλών ίδιων τιμών που μπορεί να βρίσκονται στον πίνακα) ώστε να αποφύγουμε τις πολλαπλές διατρέξεις του πίνακα με τα pivots
    - Ο αρχικός πίνακας να παραμένει ο ίδιος και απλά τα pivots να αφαιρούνται και να τοποθετούνται σε άλλο πίνακα στις σωστές θέσεις. Έπειτα να ταξινομήσουμε τον εναπομείναντα πίνακα και να κάνουμε merge των δύο πινάκων.
- Ως καλύτερο ανταγωνιστή ορίζουμε την μέθοδο QS3pR καθώς έχει τον μικρότερο αριθμό αναθέσεων (που στοιχίζουν περισσότερο σε σχέση με τις συγκρίσεις) και έναν σχετικά καλό αριθμό συγκρίσεων επιτυγχχάνοντας παράλληλα πολύ καλούς χρόνους.
- Στην μεθοδο QS1pL παρατηρούμε μία κλασική nlog(n) επίδοση η οποία είναι δεδομένη λόγω του τυχαίου πίνακα και του μεγέθους του πίνακα
- Στην μεθοδο QS1pR παρατηρούμε μία σχετικά πιο αυξημένη πολυπλοκότητα σε σχέση με τον «πιο απλό» αλγόριθμου που επιλέγει τυφλά το αριστερότερο στοιχείο. Αυτό όμως αναιρείται πλήρως στην χειρότερη περίπτωση του quicksort (ανάποδα ταξινομημένο ή ταξινομημένο πίνακα). Εκεί ο απλός αλγόριθμος έχει έως και δεκαπλάσιες φορές χειρότερη επίδοση. Επίσης εκτώς απο την χειρότερη επίδοση (χρονικά) χρειάζεται και πολύ περισσότερο χώρο στην μνήμη λόγω του μεγαλύτερου αναδρομικού δένδρου που χρειάζεται να χτιστεί προσωρινά στην μνήμη.
- Η μέθοδος QS1pM έχει χειρότερη επίδοση από την QS1pR καθώς προστίθεται μία έξτρα πολυπλοκότητα λόγω της ανάγκης υπολογισμού του mid στοιχείου και της τυχαίας επιλογής 3 στοιχείων έναντι του 1. Γι αυτό τον λόγο επιλέγουμε την QS1pR καθώς έτσι και αλλιως το στοιχείο επιλέγεται με τυχαίο τρόπο οπότε δεν μπορούμε να δούμε ουσιώδη διαφορά ώστε να επιλέξουμε την QS1pM σε κάποια περίπτωση.
- Ως καλύτερες περιπτώσεις μπορούμε να διακρίνουμε τις QS3pR και την QS2pR και καταλήγουμε στο συμπέρασμα ότι οι πιο αποδοτικές παραλλαγές του αλγορίθμου μπορούν να προσεγγιστούν μέσω των multi-pivot algorithms.

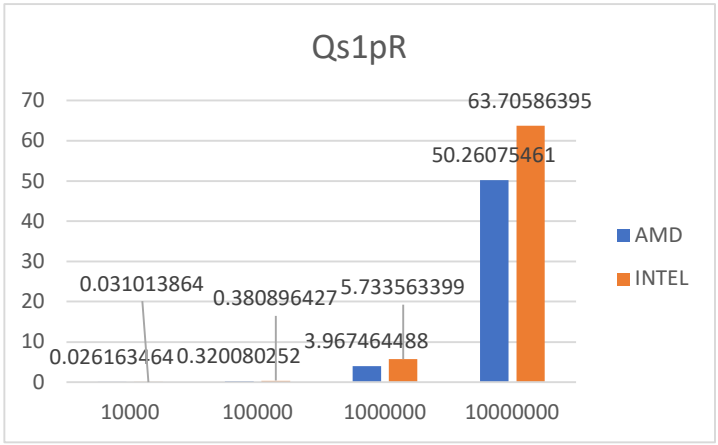
Σύγκριση των αλγορίθμων με διαφορετικό hardware:

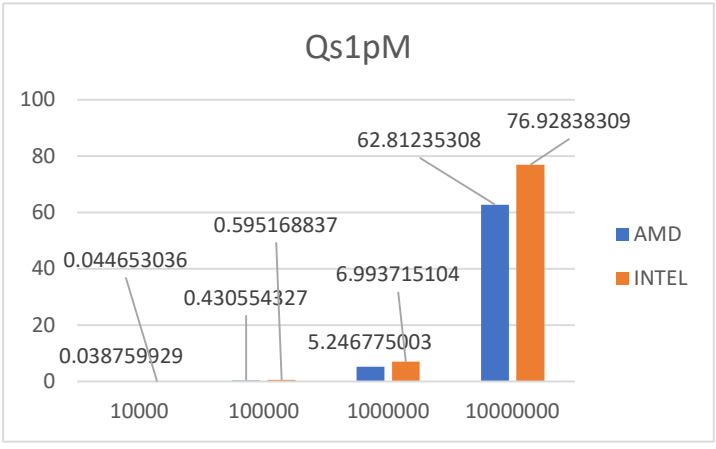
Εκτώς από το παραπάνω hardware οι αλγόριθμοι εκτελέστηκαν και σε διαφορετικό σύστημα με τα εξής specs:

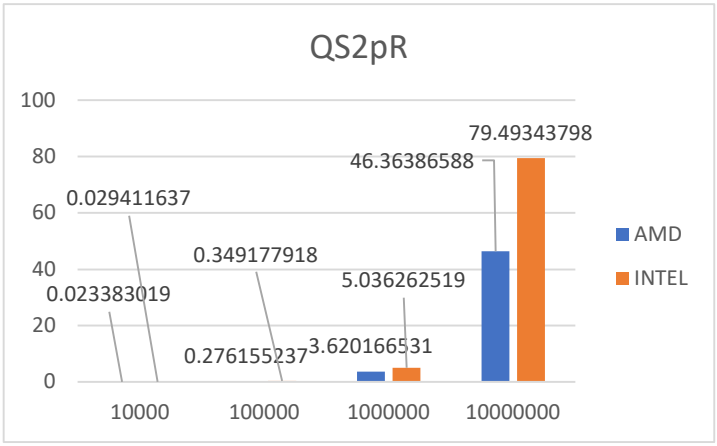
- CPU: Intel Core i5-10210U @1.6GHz
- RAM : 8Gb 2400MHz
- Windows 10 Home
- Cache size: 6MB

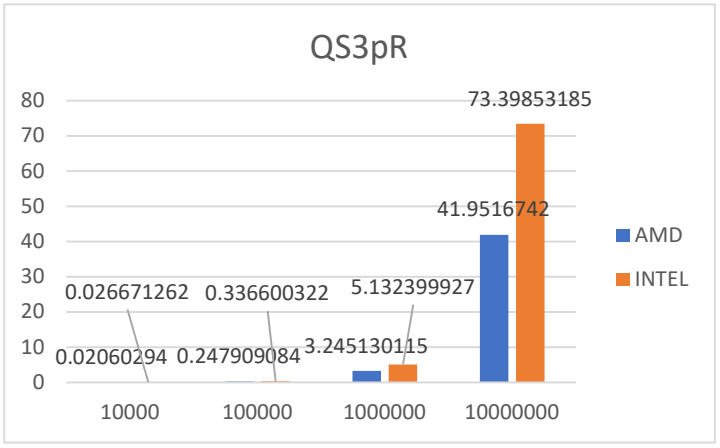
Παρακάτω φαίνονται τα διαγράμματα των δύο ανταγωνιστών κρινόμενοι βάσει του μέσου χρόνου εκτέλεσης των αλγορίθμων μιας που οι υπόλοιπες μετρικές έχουν καταλήξει στα ίδια αποτελέσματα.

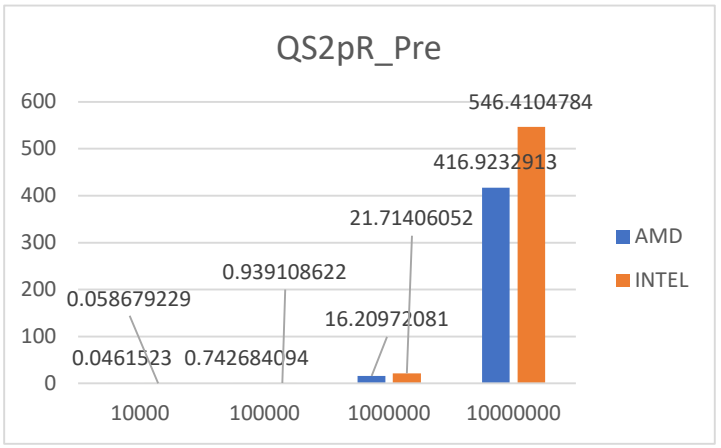


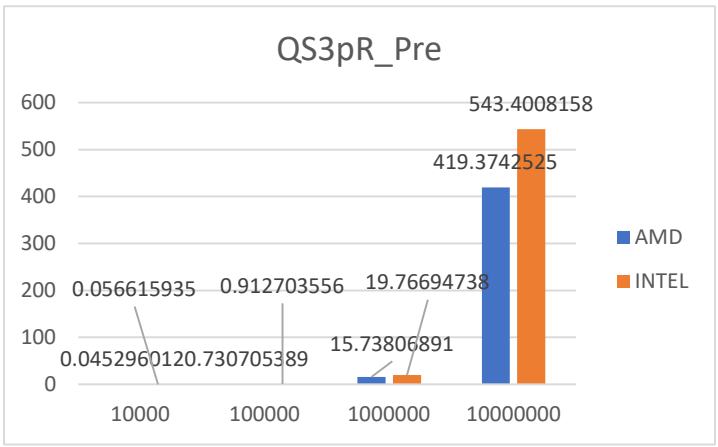












Παρατηρούμε ότι ο επεξεργαστής της Intel έχει αρκετά πιο χαμηλές επιδόσεις σε σχέση με τον επεξεργαστή της AMD. Αυτό αφενός οφείλεται στον χαμηλότερο χρονισμό του επεξεργαστή της Intel και αφετέρου στην μικρότερη χωρητικότητα της cache (μιας που η cache είναι κρίσιμη στην διάτρεξη των πινάκων).

Το δεύτερο σύστημα το οποίο τρέξαμε τους αλγορίθμους έχει σαφώς χαμηλότερα specs και όσον αφορά τον επεξεργαστή και όσον αφορά την RAM. Παρόλα αυτά μετά από recording του performance καταλήξαμε στο γεγονός πως κρίσιμο σημείο για την γρήγορη εκτέλεση των αλγορίθμων είναι ο χρονισμός της RAM παρά από την χωρητικότητα της RAM καθώς δεν είναι διεργασίες που απαιτούν ιδιαίτερα μεγάλο χόρο στην προσωρινή μνήμη (τουλάχιστον στην μέση περίπτωση).