ΟΝΟΜΑ_1: ΒΑΣΙΛΕΙΟΣ_ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΣ_03153

ΟΝΟΜΑ_2: ΧΑΡΑΛΑΜΠΟΣ_ΓΟΥΛΑΣ_03194

ΑΛΓΟΡΙΘΜΟΙ Project

ΨΕΥΔΟΚΩΔΙΚΕΣ

QS1pL->QS1pL.txt

QS1pR->QS1pR.txt

QS1pM->QS1pM.txt

QS2pR->QS2pR.txt

QS3pR->QS3pR.txt

QS2pRPre->QS2pRPre.txt

QS3pRPre->QS3pRPre.txt

ΚΩΔΙΚΕΣ ΣΕ C

QS1pL->QS1pL.c

QS1pR->QS1pR.c

QS1pM->QS1pM.c

QS2pR->QS2pR.c

QS3pR->QS3pR.c

QS2pRPre->QS2pRPre.c

QS3pRPre->QS3pRPre.c

INVARIANTS

QS1pL: Επιλέγει το αριστερότερο στοιχείο του υποπίνακα ως pivot p και τον ταξινομεί με βάση αυτό το στοιχείο. Μετά το τέλος της αναδρομής και πριν ξεκινήσει η επόμενη αναδρομική κλήση το pivot p που διαλέχθηκε έχει τοποθετηθεί στην τελική του θέση και αριστερά του βρίσκονται στοιχεία μικρότερης τιμής ενώ δεξιά του μεγαλύτερης τιμής, άρα ισχύει η σχέση

. . . < p < . . .

Αυτό ισχύει για κάθε αναδρομική κλήση αφού καλείται ο αλγόριθμος για τους υποπίνακες που προκύπτουν έως ότου να ισχύει η παραπάνω σχέση για όλα τα στοιχεία του πίνακα και τότε ο πίνακας είναι πλήρως ταξινομημένος.

QS1pR: Στην αρχή του αλγορίθμου επιλέγει ένα τυχαίο στοιχείο του υποπίνακα ως pivot p, το τοποθετεί στην αριστερότερη θέση του υποπίνακα και από εκεί και πέρα ακολουθεί την ίδια διαδικασία (την ίδια invariant) του **QS1pL.** Σε κάθε ανδρομική κλήση ακολουθείται η ίδια διαδικασία με την παραπάνω.

QS1pM: Στην αρχή του αλγορίθμου επιλέγει τρία τυχαία στοιχεία του υποπίνακα ως υποψήφια pivots p,q,r, επιλέγει τη μεσαία τιμή από τα τρία και το τοποθετεί στην αριστερότερη θέση του υποπίνακα και από εκεί και πέρα ακολουθεί την ίδια διαδικασία(την ίδια invariant) του **QS1pL.** Σε κάθε ανδρομική κλήση ακολουθείται η ίδια διαδικασία με την παραπάνω.

QS2pR: Επιλέγει 2 τυχαία στοιχεία του υποπίνακα ως pivots p,q, τοποθετεί στην αριστερότερη θέση του υποπίνακα το μικρότερο από τα δύο και στην δεξιότερη θέση του υποπίνακα το μεγαλύτερο από τα δύο. Μετά το τέλος της αναδρομής και πριν ξεκινήσει η επόμενη αναδρομική κλήση τα pivot p,q που διαλέχθηκαν έχουν τοποθετηθεί στις τελικές τους θέσεις και για καθένα ισχύει ότι αριστερά του έχει μόνο μικρότερα στοιχεία και δεξιά του μόνο μεγαλύτερα δημιουργώντας τρεις υποπίνακες. Ισχύει η σχέση (έστω p<q)

. . . < p < . . . < q < . . .

Αυτό ισχύει για κάθε αναδρομική κλήση του αλγορίθμου για τους υποπίνακες που προκύπτουν έως ότου να ισχύει η παραπάνω σχέση για όλα τα στοιχεία του πίνακα και τότε ο πίνακας είναι πλήρως ταξινομημένος.

QS3pR: Επιλέγει 3 τυχαία στοιχεία του υποπίνακα ως pivots p,q,r, τοποθετεί στην αριστερότερη θέση του υποπίνακα το μικρότερο από τα τρία, στην αριστερότερη+1 θέση του υποπίνακα το στοιχείο με τη μεσαία τιμή από τα τρία και στην δεξιότερη θέση του υποπίνακα το μεγαλύτερο από τα τρία. Μετά το τέλος της αναδρομής και πριν ξεκινήσει η επόμενη αναδρομική κλήση τα pivot p,q,r που διαλέχθηκαν έχουν τοποθετηθεί στις τελικές τους θέσεις και για καθένα ισχύει ότι αριστερά του έχει μόνο μικρότερα στοιχεία και δεξιά του μόνο μεγαλύτερα δημιουργώντας τέσσερις υποπίνακες. Ισχύει η σχέση (p<q<r)

. . . < p < . . . < q < . . . < r < . . .

Αυτό ισχύει για κάθε αναδρομική κλήση του αλγορίθμου για τους υποπίνακες που προκύπτουν έως ότου να ισχύει η παραπάνω σχέση για όλα τα στοιχεία του πίνακα και τότε ο πίνακας είναι πλήρως ταξινομημένος.

QS2pRPre: Αρχικά πριν την εκτέλεση του αλγορίθμου δημιουργεί έναν πίνακα $\sqrt{n}$ μεγέθους με στοιχεία του πίνακα από τα οποία θα επιλέγουμε κατάλληλα τα pivots p,q(αν υπάρχουν αλλιώς εκτελείται αυτούσιος ο αλγόριθμος **QS2pR**) και από εκεί και πέρα ακολουθεί τον τρόπο ταξινόμησης και διευθέτησης των ορίων(την ίδια invariant) του **QS2pR**, επομένως θα προκύψει πάλι ταξινομημένος πίνακας**.**

QS3pRPre: Αρχικά πριν την εκτέλεση του αλγορίθμου δημιουργεί έναν πίνακα $\sqrt{n}$ μεγέθους με στοιχεία του πίνακα από τα οποία θα επιλέγουμε κατάλληλα τα pivots p,q,r(αν υπάρχουν αλλιώς εκτελείται αυτούσιος ο αλγόριθμος **QS3pR**) και από εκεί και πέρα ακολουθεί τον τρόπο ταξινόμησης και διευθέτησης των ορίων(την ίδια invariant) του **QS3pR**, επομένως θα προκύψει πάλι ταξινομημένος πίνακας**.**

AVERAGE

RUNTIME:

Αλγόριθμοι\N	10000	100000	1000000	10000000
QS1pL	0.00329596	0.03668926	0.4193405	4.68885716
QS1pR	0.00349354	0.03709892	0.42410928	4.90462842
QS1pM	0.00420924	0.04338088	0.48898198	5.49387928
QS2pR	0.00356674	0.0368065	0.41036606	4.7246729
QS3pR	0.00378588	0.04849516	0.54303008	4.83987972
QS2pRPre	0.00382506	0.0375878	0.45175942	5.0455475
QS3pRPre	0.0039883	0.04096744	0.45615194	5.06152138

COMPARISONS:

Αλγόριθμοι\N	10000	100000	1000000	10000000
QS1pL	190125.98	2460775.98	29933972.52	353012491.7
QS1pR	192427.68	2440986.3	29685613.7	350747489
QS1pM	169988.08	2166457.24	26407580.3	312491473.9
QS2pR	151874.36	1940402.2	23855181.62	282022169.3
QS3pR	148727.48	1911901.06	23282868.56	275739413.5
QS2pRPre	215958.1	2683384.48	32874185.6	377651751.3
QS3pRPre	207778.3	2694486.18	32808440	379076207.5

ASSIGNMENTS:

Αλγόριθμοι\N	10000	100000	1000000	10000000
QS1pL	114656.34	1376779.2	16070306.16	183692675
QS1pR	109557.3	1326886.62	15583108.68	178879058.5
QS1pM	110225.64	1340213.16	15767238.66	181196183
QS2pR	177004.5	2180736.48	25946330.52	299199370.3
QS3pR	162665.82	2068167.12	25252396.86	297310223.1
QS2pRPre	177015.78	2181432.78	25797214.98	299454894.1
QS3pRPre	165790.92	2085189.06	25855107.84	299926119.8

VARIANCE

Για τον υπολογισμό της variance χρησιμοποιήθηκε η συνάρτηση του excel VAR.S που βασίζεται στον τύπο $\sum \frac{(x-average(x))^2}{n-1}$.

RUNTIME:

Αλγόριθμοι\N	10000	100000	1000000	10000000
QS1pL	$5.01607*10^{-7}$	$7.50682*10^{-6}$	0.001538655	0.022446582
QS1pR	$5.1536*10^{-7}$	$1.06042*10^{-5}$	0.000118344	0.011209592
QS1pM	$4.25657*10^{-7}$	$1.02644*10^{-5}$	0.000191529	0.004768274
QS2pR	$5.9347*10^{-7}$	$1.5058*10^{-5}$	0.000458904	0.013537192
QS3pR	$4.79359*10^{-7}$	$8.5885*10^{-5}$	0.007160262	0.016494395
QS2pRPre	$7.07899*10^{-7}$	$1.53418*10^{-5}$	0.000331771	0.014707782
QS3pRPre	$4.11548*10^{-7}$	$1.99777*10^{-5}$	0.00049234	0.018052839

COMPARISONS:

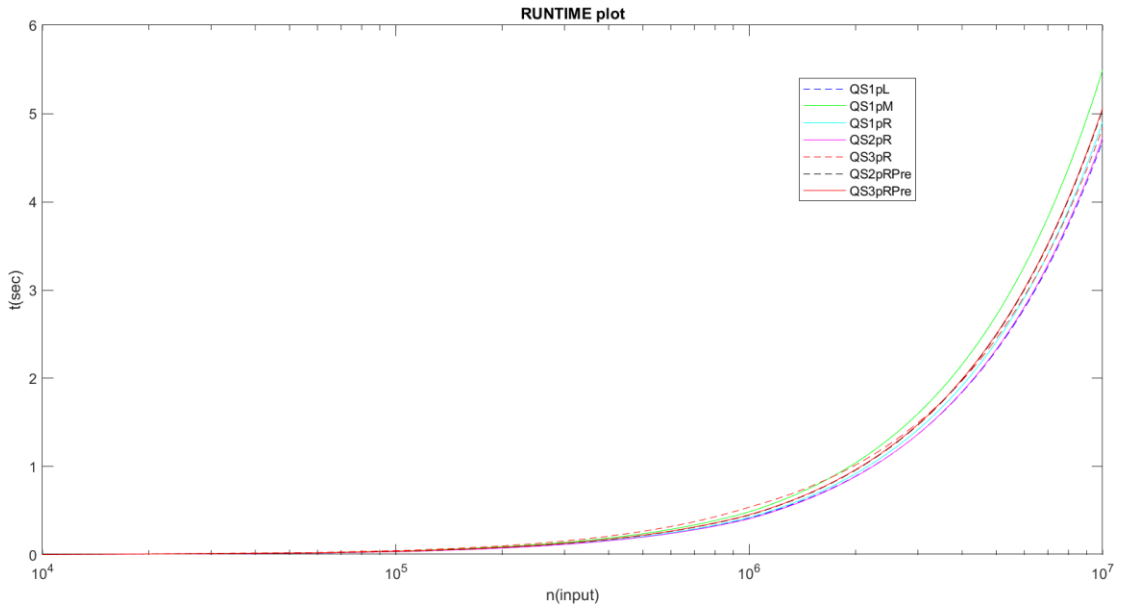
Αλγόριθμοι\N	10000	100000	1000000	10000000
QS1pL	27421858.06	4297729979	$4.30036*10^{11}$	$3.98265*10^{13}$
QS1pR	28647089.9	1941763874	$2.87938*10^{11}$	$2.42491*10^{13}$
QS1pM	8701269.749	984520218.3	92604117394	$8.94678*10^{15}$
QS2pR	25354286.52	1982257490	$3.86419*10^{11}$	$3.2344*10^{13}$
QS3pR	22737835.4	2337443877	$1.67219*10^{11}$	$2.006*10^{13}$
QS2pRPre	184706690.3	25472521376	$2.13673*10^{12}$	$3.7914*10^{14}$
QS3pRPre	195604148.7	30082614641	$2.32438*10^{12}$	$5.40393*10^{14}$

ASSIGNMENTS:

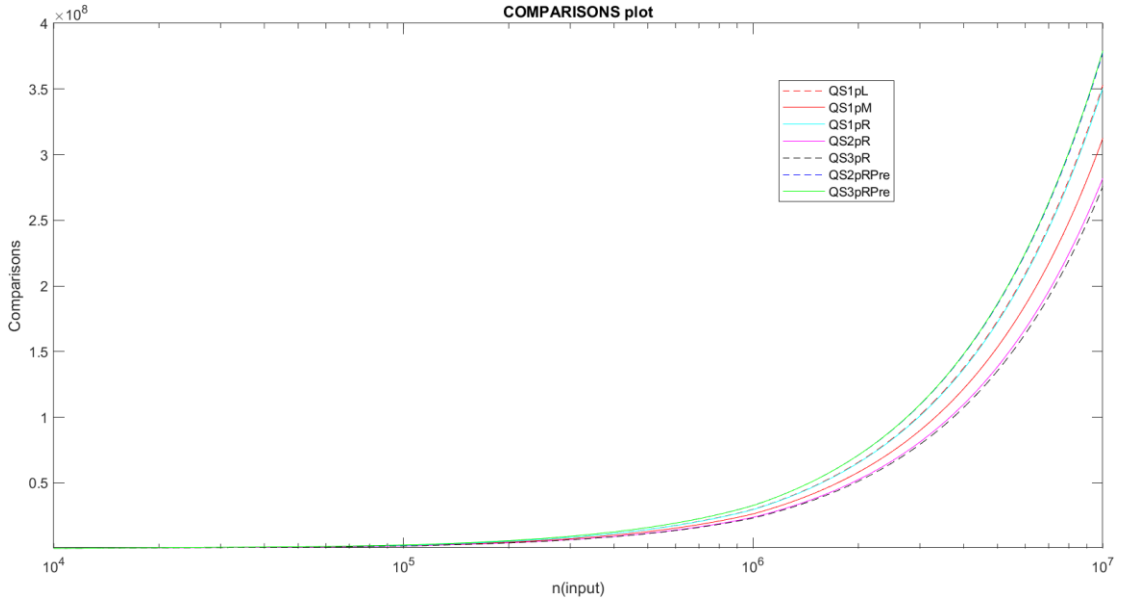
Αλγόριθμοι\N	10000	100000	1000000	10000000
QS1pL	619412.0657	44155409.88	5910014457	$4.28177*10^{11}$
QS1pR	424557.6429	38563415.71	4158022223	$3.95739*10^{11}$
QS1pM	297106.3984	24665603.52	2673747936	$3.76089*10^{11}$
QS2pR	86472211.32	6928600809	$1.09089*10^{12}$	$4.16573*10^{13}$
QS3pR	44406271.99	8819894309	$9.65318*10^{11}$	$8.21686*10^{13}$
QS2pRPre	90057200.99	9535017876	$9.81553*10^{11}$	$1.33315*10^{14}$
QS3pRPre	100625758.8	13933878360	$1.10573*10^{12}$	$1.05558*10^{14}$

PLOTS

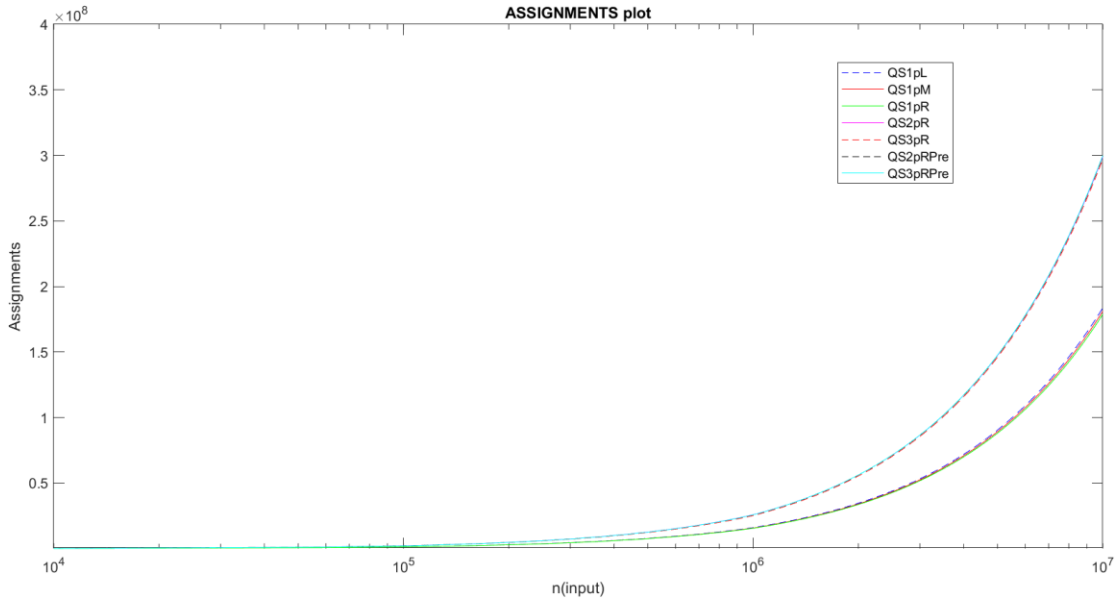
RUNTIME:



COMPARISONS:



ASSIGNMENTS:



ΕΡΜΗΝΕΙΑ ΑΠΟΔΟΣΗΣ

RUNTIME:

Processors: 2

Microprocessor: 0x86

Model name: 11th Gen Intel® $Core^{TM}$ i7-1165G7

Cpu speed: 2803.199MHz=2.80GHz

Cache size: 12288 kB

Memory size: 1999332 kB=1.9 GiB of RAM

Memory available: 1064816 kB

Kernel version: Kubuntu 20.04 5.4.0-113-generic

OS type: 6-bit

Σχολιασμός:

1ο)Αρχικά τα προγράμματα εκτελέστηκαν από laptop χωρίς να είναι συνδεδεμένο στην φόρτιση(παρατηρήθηκε ότι όταν το laptop ήταν συνδεμένο στην φόρτιση έτρεχε πιο γρήγορα όλους τους αλγορίθμους).

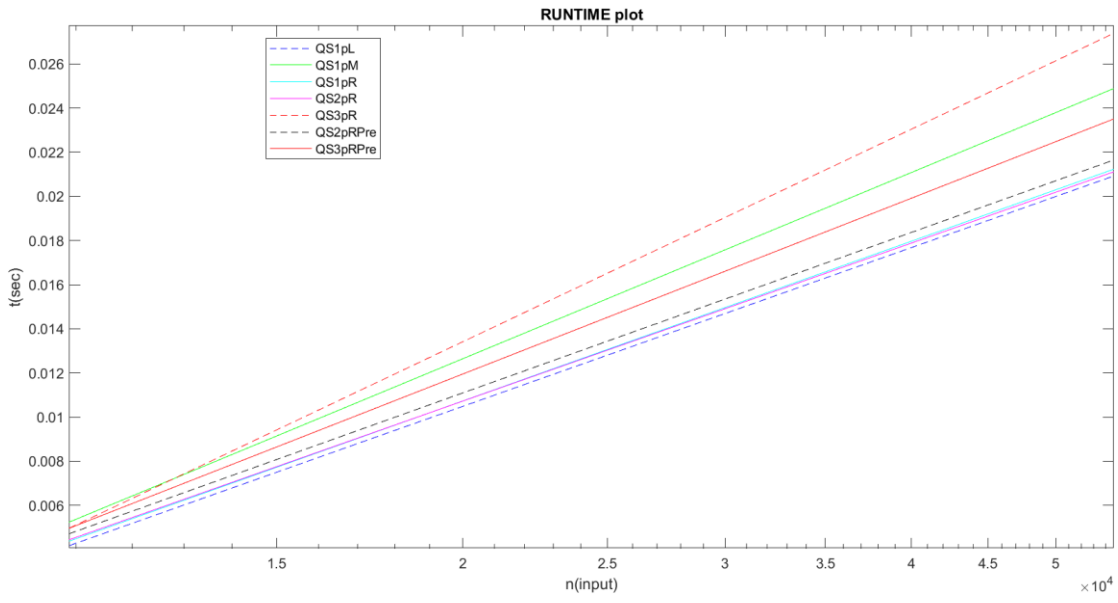
2ο)Οι χρόνοι μπορεί να μην είναι συγκρίσιμοι γιατί δεν γνωρίζουμε τι άλλες διεργασίες εκτελούσε ο υπολογιστής ώστε να γνωρίζουμε την «επιβάρυνση» της CPU εκείνη την χρονική στιγμή. Επίσης η στάθμη της μπαταρίας διέφερε στην εκτέλεση του κάθε αλγόριθμου(πχ ο QS1pL εκτελέστηκε με 75%+ μπαταρία ενώ ο QS3pRPre με 20%-).

3ο)Οι αλγόριθμοι εκτελέστηκαν μέσω του VM(virtual machine) οπότε σίγουρα δεν επιτεύχθηκε η βέλτιστη εκτέλεση του αλγορίθμου από πλευράς χρόνου.

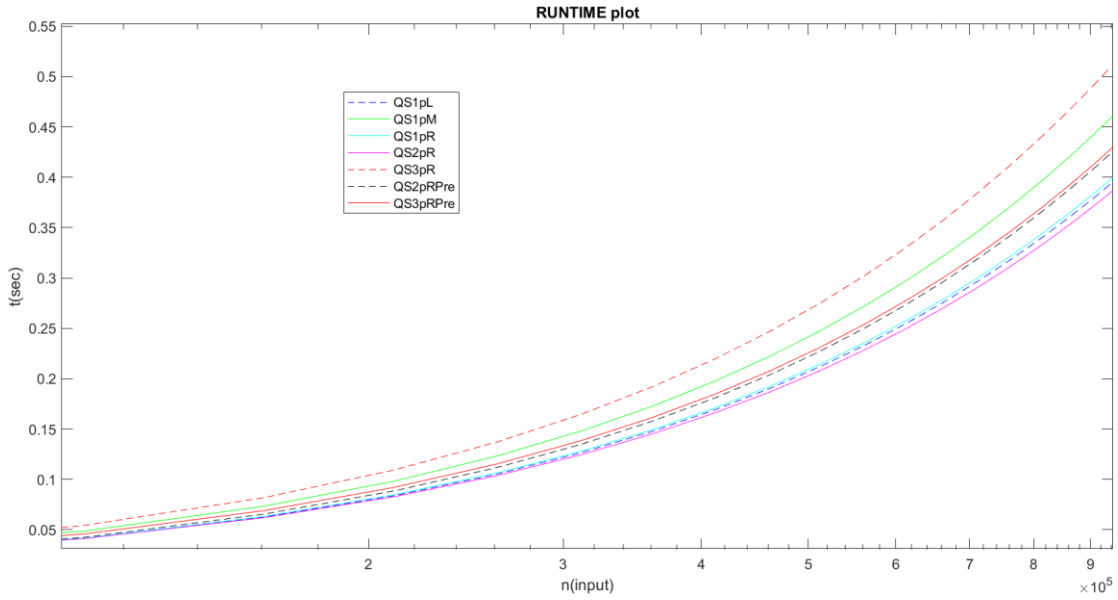
4ο)Παρατηρήθηκε ότι όταν μειώθηκε η RAM στο VM(virtual machine) από 4 GiB σε 2 GiB οι χρόνοι εκτέλεσης των αλγορίθμων μειώθηκαν δηλαδή οι αλγόριθμοι ήταν ταχύτεροι.

5ο)Στις μετρήσεις που έγιναν χρησιμοποιήθηκαν κατά την εκτέλεση οι μετρητές των comparisons και assignments επιφέροντας μια καθυστέρηση στους χρόνους εκτέλεσης καθώς όταν δεν χρησιμοποιούνταν οι αλγόριθμοι έτρεχαν ταχύτερα.

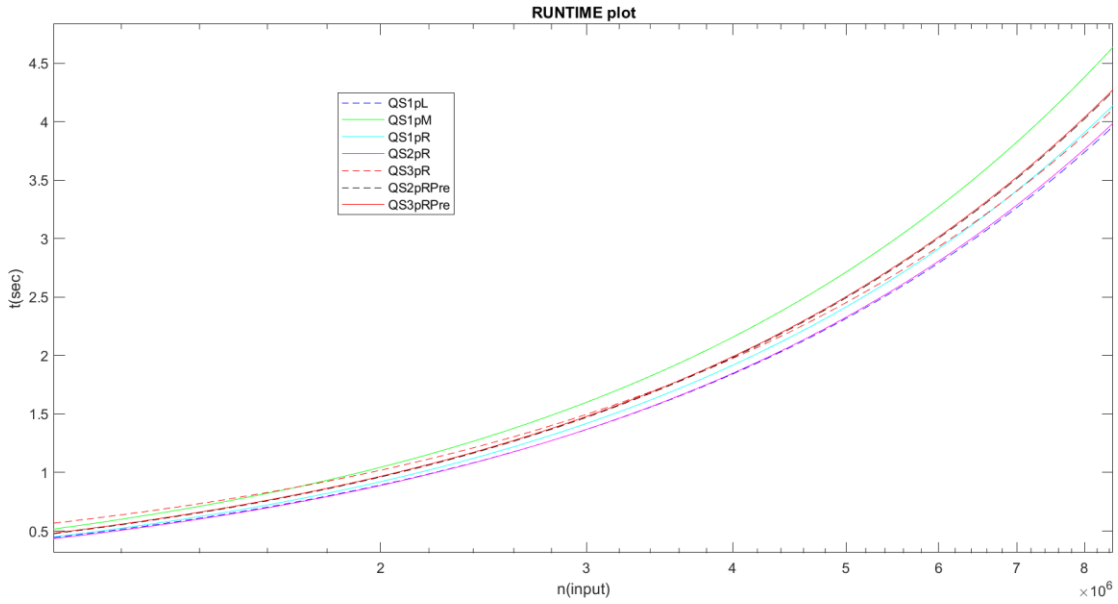
Plot για το διάστημα (10^4,10^5):



Plot για το διάστημα (10^5,10^6):



Plot για το διάστημα (10^6,10^7):



QS1pL: Η καμπύλη που μας ενδιαφέρει είναι η διακεκομμένη σκούρο μπλε γραμμή.

Ο αλγόριθμος αυτός φαίνεται πως είναι από τους πιο γρήγορους για όλα τα παραπάνω διαστήματα, βέβαια τα δεδομένα αφορούν τυχαίες ακολουθίες αριθμών. Όμως για ταξινομημένους και ανάποδα ταξινομημένους πίνακες ο αλγόριθμος αυτός είναι από τους χειρότερους διότι η πολυπλοκότητα του αλγορίθμου από $O(n * \log_2 n)$ γίνεται $O(n^2)$ (αυτό δεν συμβαίνει στους άλλους ανταγωνιστές διότι η επιλογή του pivot δεν είναι προκαθορισμένη αλλά τυχαία και γι' αυτό δεν μπορούν να αιτιολογηθούν όλες οι διαφορές με τους άλλους ανταγωνιστές σχετικά με τον χρόνο εκτέλεσης).

QS1pR: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής γαλάζια γραμμή.

Ο αλγόριθμος αυτός φαίνεται πως αποτελεί έναν από τους γρήγορους για όλα τα παραπάνω διαστήματα με εξαίρεση όταν το μέγεθος των δεδομένων γίνει πολύ μεγάλο της τάξης του 10^7 αυξάνει αρκετά ο χρόνος εκτέλεσης του αλγορίθμου. Το πλεονέκτημα του αλγορίθμου είναι ότι δεν επηρεάζεται πολύ από την είσοδο των δεδομένων(πχ ταξινομημένα-μη ταξινομημένα δεδομένα). Επίσης αυτή η αύξηση στον χρόνο εκτέλεσης για μεγάλο αριθμό δεδομένων δείχνει ότι η ύπαρξη περισσότερων pivots βελτιώνει το χρόνο εκτέλεσης, γι' αυτό τόσο ο QS2pR και ο QS3pR είναι πιο γρήγοροι από τον QS1pR.

QS1pM: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής πράσινη γραμμή.

Ο αλγόριθμος αυτός φαίνεται πως αποτελεί έναν από τους χειρότερους για όλα τα παραπάνω διαστήματα. Σε αυτό οφείλεται η επαναληψιμότητα επιλογής των τριών υποψήφιων pivots, διότι όταν το δεύτερο ή το τρίτο υποψήφιο pivot είναι ίδιο με κάποια από τα προηγούμενα πρέπει να επιλεχθεί εκ νέου, γεγονός που συμβαίνει πολύ συχνά για πολύ μικρούς σε μέγεθος υποπίνακες που υπάρχουν για όλα τα μεγέθη εισόδων. Επίσης συγκριτικά με τους ανταγωνιστές που εμφανίζουν το ίδιο πρόβλημα σε μεγάλο όγκο δεδομένων(επιλογή πολλών pivots,δηλαδή ο QS2pR και QS3pR) είναι χειρότερος ο QS1pM διότι χρησιμοποιεί λιγότερα pivots(1) από ό,τι οι άλλοι(2,3 αντίστοιχα).

QS2pR: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής ροζ γραμμή.

Ο αλγόριθμος αυτός φαίνεται πως αποτελεί έναν από τους γρηγορότερους για όλα τα παραπάνω διαστήματα. Σε αυτό οφείλεται η χρήση 2 pivots randomly για την ταξινόμηση των στοιχείων. Επίσης ο ανταγωνιστής αυτός έχει λιγότερες συγκρούσεις στην επιλογή των pivots σε σχέση με τον ανταγωνιστή QS3pR(το ένα επιπλέον pivot θα δημιουργεί περισσότερες επαναλήψεις στην επιλογή των pivots) γι' αυτό τον λόγο παρατηρείται διαφορά στους χρόνους εκτέλεσης των δυο αυτών ανταγωνιστών με τον QS2pR να είναι πιο γρήγορος από τον QS3pR.

QS3pR: Η καμπύλη που μας ενδιαφέρει είναι η διακεκομμένη κόκκινη γραμμή.

Ο αλγόριθμος αυτός φαίνεται πως αποτελεί έναν από τους χειρότερους για όλα τα παραπάνω διαστήματα με εξαίρεση για μεγάλο όγκο δεδομένων όπου παρουσιάζει ραγδαία βελτίωση. Αρχικά για μικρό όγκο δεδομένων ο αλγόριθμος αυτός είναι ο χειρότερος διότι οδηγεί σε πολλές επαναλήψεις επιλογής του $2^{ου}$ και $3^{ου}$ pivot. Για μεγάλο όγκο δεδομένων παρατηρείται τεράστια μείωση στο χρόνο εκτέλεσης καθώς από χειρότερος γίνεται ο $3^{ος}$ ταχύτερος αλγόριθμος, διότι εκμεταλλεύεται στο έπακρο την ταξινόμηση με τα 3 pivots, γι' αυτό τόσο ο QS1pM όσο και ο QS1pR είναι τελικά πιο αργοί παρότι αρχικά ήταν πιο γρήγοροι από τον QS3pR.

QS2pRPre: Η καμπύλη που μας ενδιαφέρει είναι η διακεκομμένη μαύρη γραμμή.

Ο αλγόριθμος αυτός φαίνεται πως είναι γρήγορος για μικρό όγκο δεδομένων, γεγονός που ανατρέπεται καθώς αυξάνεται ο όγκος των δεδομένων. Μια κύρια αιτία αυτού του φαινομένου αποτελεί ένα προγραμματιστικό χαρακτηριστικό του κώδικα που περιλαμβάνει την realloc και την μετατόπιση των στοιχείων στον πίνακα των προεπιλεγμένων pivots προς τα πίσω και παρατηρείται εντονότερα για μεγάλο όγκο δεδομένων καθώς δημιουργούνται μεγαλύτεροι πίνακες προεπιλεγμένων pivots. Αναμένεται πως ο ανταγωνιστής αυτός θα είχε καλύτερο χρόνο εκτέλεσης από τον QS2pR γιατί επιλέγοντας pivots για τον QS2pRPre από τον πίνακα των προεπιλεγμένων pivots διαμερίζεται καλύτερα ο υποπίνακας αλλά αυτό δεν συμβαίνει πιθανότατα λόγω της προαναφερθείσας αιτίας.

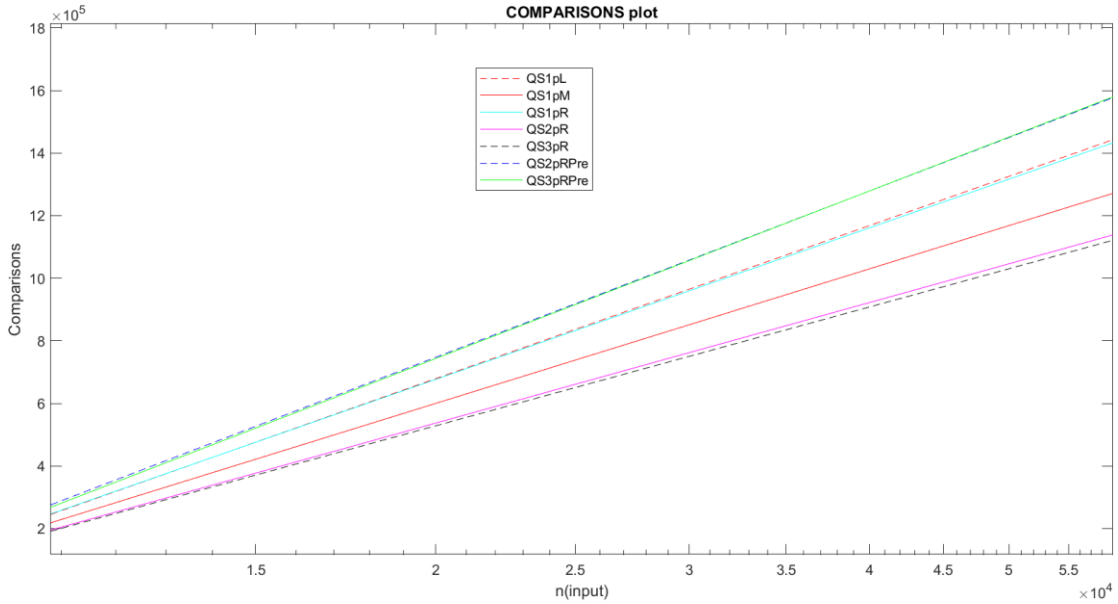
QS3pRPre: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής κόκκινη γραμμή.

Ο αλγόριθμος αυτός φαίνεται πως είναι γρήγορος για μικρό όγκο δεδομένων, γεγονός που ανατρέπεται καθώς αυξάνεται ο όγκος των δεδομένων και ταυτίζεται με τον QS2pRPre αντιμετωπίζοντας το ίδιο πρόβλημα με την realloc. Αρχικά για μικρό όγκο δεδομένων ο QS3pR είναι πιο αργός από τον QS3pRPre διότι εκμεταλλεύεται τον καλύτερο διαμερισμό του υποπίνακα που προκαλεί η προεπιλογή των pivots και δεν επηρεάζεται σε μεγάλο βαθμό από το προγραμματιστικό χαρακτηριστικό του κώδικα με την realloc(γιατί αδειάζει πιο γρήγορα τον πίνακα των προεπιλεγμένων pivots σε σχέση με τον QS2pRPre). Σε μεγάλο όγκο δεδομένων αυτό ανατρέπεται διότι το πρόβλημα που δημιουργεί η realloc φαίνεται εντονότερα κάνοντας τον QS3pR πιο γρήγορο από τον QS3pRPre.

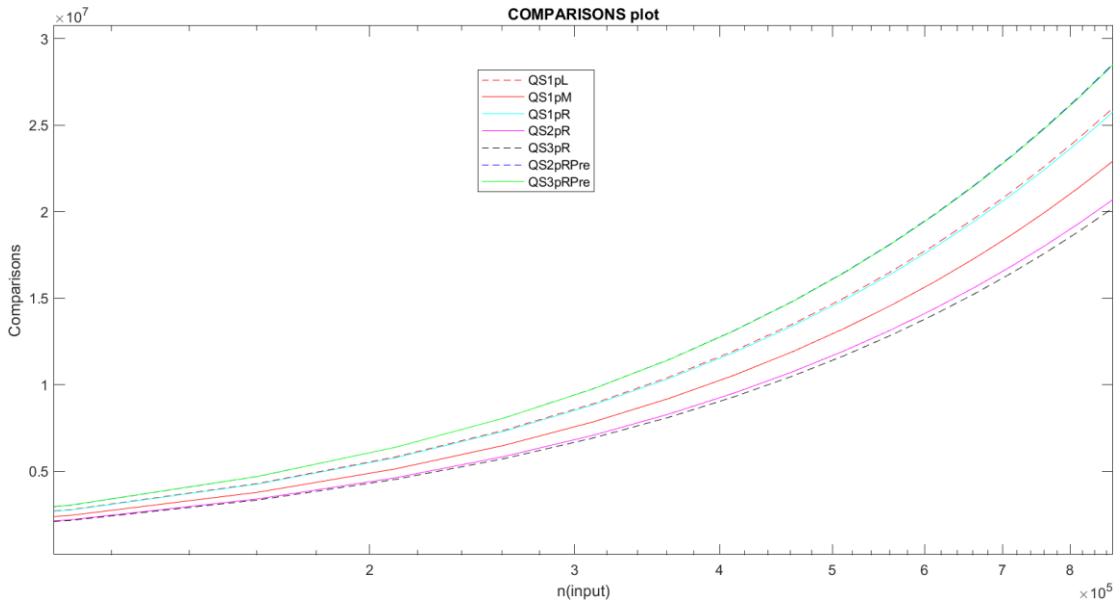
COMPARISONS:

Σημείωση: Τα comparisons αποτελούνται μόνο από τις συγκρίσεις μεταξύ στοιχείων του πίνακα(πχ array[i]<array[j]) και όχι οποιαδήποτε άλλη όπως σύγκριση θέσεων(πχ i<j).

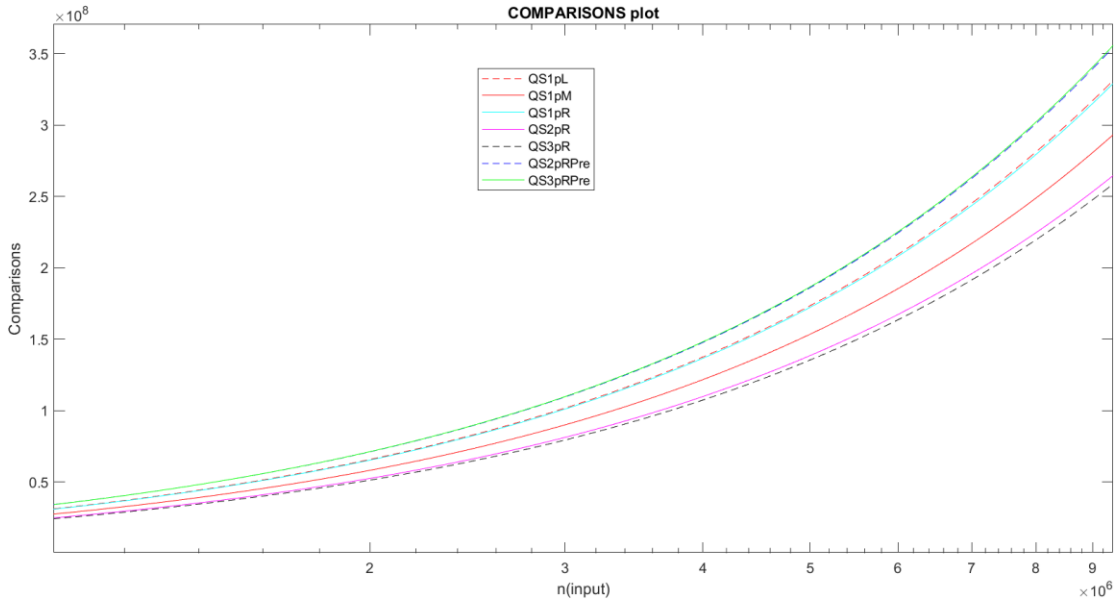
Plot για το διάστημα (10^4,10^5):



Plot για το διάστημα (10^5,10^6):



Plot για το διάστημα (10^6,10^7):



<u>QS1pL: Η καμπύλη που μας ενδιαφέρει είναι η διακεκομμένη κόκκινη γραμμή.</u>

<u>QS1pR: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής γαλάζια γραμμή.</u>

Οι αλγόριθμοι αυτοί σταθερά σε όλα τα διαστήματα έχουν πολλές συγκρίσεις αφού χρησιμοποιούν ένα pivot για την ταξινόμηση των στοιχείων και έχουν να αντιμετωπίσουν μεγάλους υποπίνακες. Επίσης στο γράφημα παρουσιάζουν την ίδια συμπεριφορά στις συγκρίσεις, διότι αποτελούνται από τυχαίες ακολουθίες αριθμών και όχι ταξινομημένες/ανάποδα ταξινομημένες όπου ο QS1pR θα ήταν καλύτερος στις συγκρίσεις από τον QS1pL.

<u>QS1pM: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής κόκκινη γραμμή.</u>

Ο αλγόριθμος αυτός σταθερά σε όλα τα διαστήματα παρουσιάζει καλή συμπεριφορά στις συγκρίσεις καθώς είναι ο 3^{ος} καλύτερος. Αυτό συμβαίνει διότι επιλέγει τρία υποψήφια pivots και από αυτά διαλέγει το μεσαίο γεγονός που οδηγεί σε καλύτερο διαμερισμό του πίνακα σε σχέση με τους QS1pL και QS1pR με αποτέλεσμα να έχει λιγότερες συγκρίσεις.

QS2pR: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής ροζ γραμμή.

QS3pR: Η καμπύλη που μας ενδιαφέρει είναι η διακεκομμένη μαύρη γραμμή.

Οι αλγόριθμοι QS2pR και QS3pR σταθερά σε όλα τα διαστήματα έχουν τις λιγότερες συγκρίσεις καθιστώντας τους τούς καλύτερους στον τομέα των συγκρίσεων αφού χρησιμοποιούν δύο pivots και τρία pivots αντίστοιχα για την ταξινόμηση των στοιχείων διαμερίζοντας καλύτερα τους υποπίνακες. Γι' αυτό τον λόγο έχουν λιγότερες συγκρίσεις από τους ανταγωνιστές που χρησιμοποιούν ένα pivot(QS1pL,QS1pR,QS1pM) για την ταξινόμηση.

QS2pRPre: Η καμπύλη που μας ενδιαφέρει είναι η διακεκομμένη μπλε γραμμή.

QS3pRPre: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής πράσινη γραμμή.

Οι αλγόριθμοι QS2pRPre και QS3pRPre σταθερά σε όλα τα διαστήματα έχουν τις περισσότερες συγκρίσεις καθιστώντας τους τούς χειρότερους στον τομέα των συγκρίσεων διότι για την εύρεση των ορίων μέσα στον πίνακα των προεπιλεγμένων pivots(στα όρια που πρέπει να επιλεχθούν τα pivots για την ταξινόμηση) καθώς και η αντιστοίχιση των τελικών pivots στις θέσεις τους στον κανονικό πίνακα έχουν σαν αποτέλεσμα πολλές παραπάνω συγκρίσεις. Γι' αυτό τον λόγο υστερούν όλων των άλλων ανταγωνιστών και κυρίως των QS2pR και QS3pR.

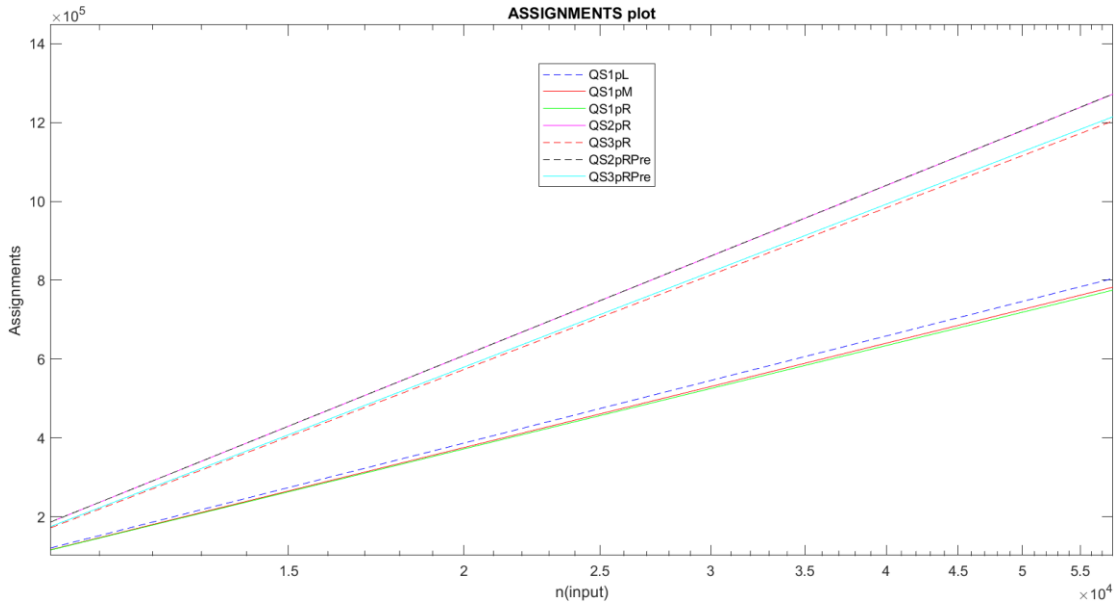
ASSIGNMENTS:

Σημείωση:

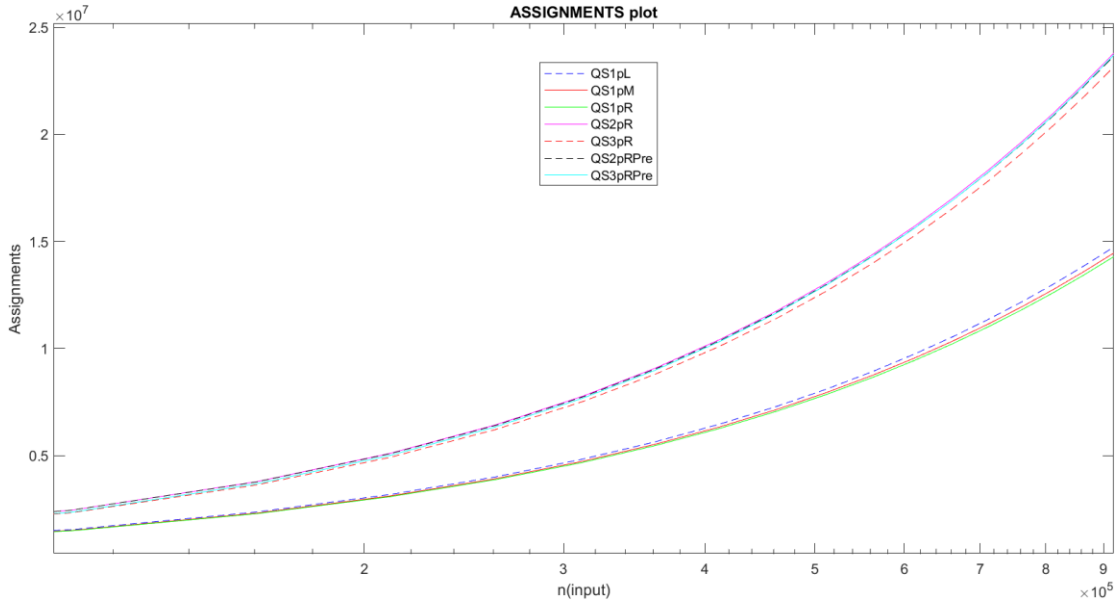
1ο)Θα μπορούσε να χρησιμοποιηθεί η εντολή στην swap "array[i]=array[i]+array[j]-(array[j]=array[i])" που θα μείωνε τις αναθέσεις από τρεις σε δύο και συνολικά θα μείωνε κατά πολύ τον συνολικό αριθμό των αναθέσεων σε όλους τους ανταγωνιστές, όμως επειδή παρουσίαζε ένα warning στο gcc -Wall δεν χρησιμοποιήθηκε.

2ο)Τα assignments αφορούν μόνο τα swaps που γίνονται μεταξύ στοιχείων του πίνακα και **όχι** τις αναθέσεις σε μεταβλητές ή σε βοηθητικούς πίνακες όπως ο πίνακας προεπιλεγμένων pivots(QS2pRPre,QS3pRPre). Η έλλειψη μέτρησης των αναθέσεων στην δημιουργία του πίνακα των προεπιλεγμένων pivots έχει σαν αποτέλεσμα ο QS2pR με τον QS2pRPre και ο QS3pR με τον QS3pRPre να ταυτίζονται μεταξύ τους αντίστοιχα σε επίπεδο αναθέσεων.

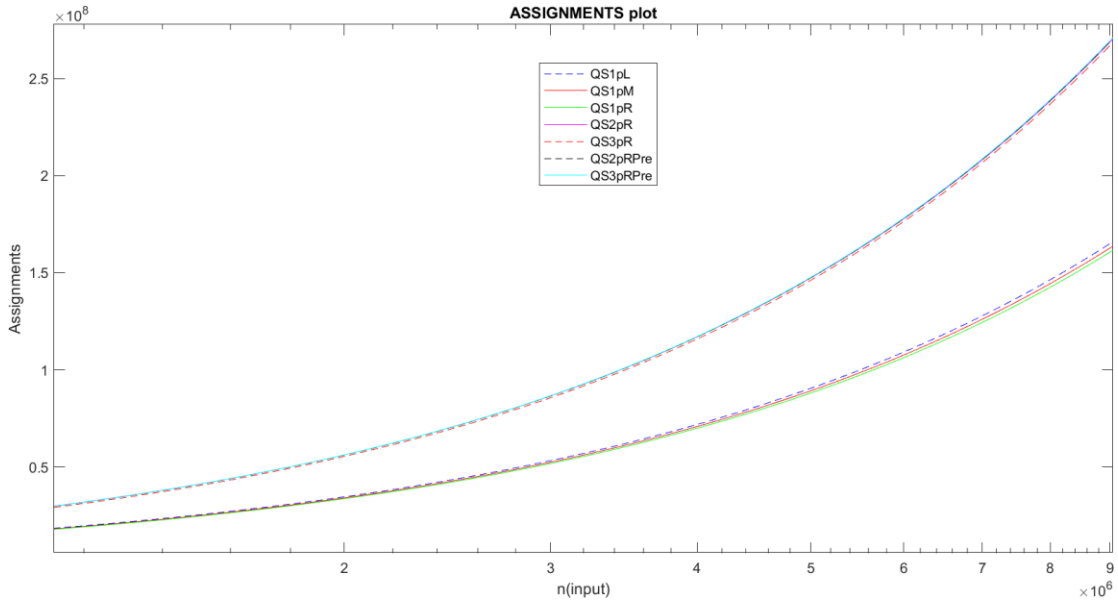
Plot για το διάστημα (10^4,10^5):



Plot για το διάστημα (10^5,10^6):



Plot για το διάστημα (10^6,10^7):



<u>QS1pL: Η καμπύλη που μας ενδιαφέρει είναι η διακεκομμένη μπλε γραμμή.</u>

<u>QS1pR: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής πράσινη γραμμή.</u>

<u>QS1pM: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής κόκκινη γραμμή.</u>

Και οι τρεις αυτοί αλγόριθμοι παρουσιάζουν ίδια συμπεριφορά με τις λιγότερες αναθέσεις καθιστώντας τους τούς καλύτερους στον τομέα των αναθέσεων. Αυτό συμβαίνει διότι δεν χρειάζεται πολλαπλές αναθέσεις πριν την ταξινόμηση των υποπινάκων και γίνονται αποκλειστικά κατά τη διάρκεια της ταξινόμησης των υποπινάκων.

<u>QS2pR: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής ροζ γραμμή.</u>

<u>QS2pRPre: Η καμπύλη που μας ενδιαφέρει είναι η διακεκομμένη μαύρη γραμμή.</u>

Οι αλγόριθμοι αυτοί είναι οι χειρότεροι στον τομέα των αναθέσεων καθώς σε όλα τα διαστήματα παρουσιάζουν πολλές αναθέσεις. Είναι χειρότεροι από τους ανταγωνιστές που χρησιμοποιούν ένα pivot για την ταξινόμηση των στοιχείων(QS1pL,QS1pR,QS1pM), διότι οι QS2pR και QS2pRPre χρειάζονται πολλαπλές αναθέσεις πριν την ταξινόμηση των στοιχείων γιατί θα πρέπει τα τυχαία pivots που επιλέχθηκαν να τοποθετηθούν στις σωστές θέσεις(στην αριστερότερη και δεξιότερη θέση του υποπίνακα) προσθέτοντας με αυτόν τον τρόπο πολλές αναθέσεις.

QS3pR: Η καμπύλη που μας ενδιαφέρει είναι η διακεκομμένη κόκκινη γραμμή.

QS3pRPre: Η καμπύλη που μας ενδιαφέρει είναι η συνεχής γαλάζια γραμμή.

Οι αλγόριθμοι αυτοί δεν είναι καλοί στον τομέα των αναθέσεων καθώς σε όλα τα διαστήματα παρουσιάζουν πολλές αναθέσεις. Είναι χειρότεροι από τους ανταγωνιστές που χρησιμοποιούν ένα pivot για την ταξινόμηση των στοιχείων(QS1pL,QS1pR,QS1pM), διότι οι QS3pR και QS3pRPre χρειάζονται πολλαπλές αναθέσεις πριν την ταξινόμηση των στοιχείων γιατί θα πρέπει τα τυχαία pivots που επιλέχθηκαν να τοποθετηθούν στις σωστές θέσεις(στην αριστερότερη και δεξιότερη θέση του υποπίνακα) προσθέτοντας με αυτόν τον τρόπο πολλές αναθέσεις. Είναι ελάχιστα καλύτεροι από τους QS2pR και QS2pRPre, διότι αποφεύγονται κάποιες ελάχιστες αναθέσεις για πολύ μικρού μεγέθους υποπίνακες 4 έως 7 στοιχείων περίπου γιατί μπορεί τα pivots να είναι τέτοια ώστε να ταξινομήσουν και τα υπόλοιπα στοιχεία του υποπίνακα στις τελικές τους θέσεις εκτός από τα pivots. Αυτό είναι πιο έντονο στους ανταγωνιστές αυτούς από τους QS2pR και QS2pRPre και παρατηρείται πιο έντονα για μικρό όγκο δεδομένων ενώ σε μεγάλο όγκο δεδομένων αυτή η διαφορά είναι αμελητέα γι' αυτό τείνουν και οι τέσσερις ανταγωνιστές(QS2pR, QS2pRPre, QS3pR, QS3pRPre) στην ίδια συμπεριφορά στον τομέα των αναθέσεων.

ΠΗΓΕΣ

Χρησιμοποιήθηκε ο κώδικας σε C dual pivot χωρίς random επιλογή και τροποποιήθηκε σε QS2pR και QS2pRPre από το παρακάτω link: https://www.geeksforgeeks.org/dual-pivot-quicksort/

Χρησιμοποιήθηκε ο ψευδοκώδικας της partition του 3 pivot quicksort και υλοποιήθηκε σε C αποτελώντας κομμάτι του QS3pR και του QS3pRPre από το παρακάτω link(σελίδα 8 του pdf): https://cs.uwaterloo.ca/~skushagr/multipivotQuicksort.pdf