ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ
Εργασία 2

ΜΑΓΚΟΣ ΡΑΦΑΗΛ-ΓΕΩΡΓΙΟΣ 3100098

ΜΠΟΓΔΑΝΟΣ ΜΙΧΑΗΛ 3100123

Επεξήγηση των κοινών κλάσεων των 2 αλγορίθμων.

1)Τάξη PrintJob (θεωρήθηκε σωστό να προσθεθεί πεδίο arrivalTime για διευκόλυνση των υπολογισμών)

-Private μεταβλητές id,size,waitingTime,priority,arrivalTime που περιέχουν τα αντίστοιχα πεδία ID,μέγεθος αρχείου,χρόνος αναμονής,προτεραιότητα και χρόνος άφιξης που πρέπει να περιέχει κάθε αντικειμένο τύπου PrintJob.Επίσης γίνεται χρήση μιας static μεταβλητής nextIdγια την ανάθεση μοναδικών ID σε κάθε αντικείμενο.

-Κατασκευαστής που καλεί τις αντίστοιχες μεθόδους set για την ανάθεση τιμών(οι οποίες θα ελεγχθούν για την εγκυρότητά τους) στις αντίστοιχες μεταβλητές του αντικειμένου.Μόνο η ανάθεση του ID γίνεται απευθείας με την χρήση της static μεταβλητής.

-Μέθοδοι set/get για την ανάθεση ,εντός των ορίων, τιμών και ανάκτηση των μεταβλητών αντίστοιχα.Σε περίπτωση ανάθεσης τιμής εκτός ορίων το πρόγραμμα τερματίζεται.

-Μέθοδος resetId() που επαναφέρει την τιμή του nextId στη τιμή 0 για μελλοντική χρήση.

-Μέθοδος compareTo(PrintJob p) συγκρίνει δύο αντικέιμενα PrintJob με βάση την προτεραιότητά τους.Εάν το αντικέιμενο this έχει μικρότερη προτεραιότητα από το όρισμα p τότε επιστρέφεται -1.Εάν το this έχει μεγαλύτερη προτεραιότητα από το p τότε επιστρέφεται 1.Σε περίπτωση ισότητας καλείται η μέθοδος compareSamePriority(PrintJob p) όπου συγκρίνει το this και το όρισμα με βάση το χρόνο άφιξης.Εάν το this έφτασε νωρίτερα επιστρέφει 1.Εάν το this έφτασε αργότερα επιστρέφει -1 και αν φτάσαν μαζί επιστρέφει 0.Σκοπός της μεθόδου είναι η αποφυγή της παραμονής στην ουρά αρχείων με ίδιο priority αλλά διαφορετικό χρόνο άφιξης.

2)Τάξη PrintJobComparator απλώς καλεί την μέθοδο compareTo της τάξης PrintJob.

3)Τάξη MaxPQ η υλοποίηση της ουράς προτεραιότητας όπως παρουσιάστηκε στο εργαστήριο 5.

4)Τάξη LinkedList η υλοποίηση της συνδεδεμένης λίστας όπως παρουσιάστηκε στο εργαστήριο 2.

Επεξήγηση της υλοποίησης του αλγορίθμου Α

Μεταβλητές:

-Private static int :

- time : ο καθολικός μετρητής χρόνου.
- maxWT : μέγιστος χρόνος αναμονής.
- completedPJ : πλήθος ολοκληρωμένων εκτυπώσεων(για εξαγωγή μέσου όρου).

-Private static double totalWT: συνολικός χρόνος αναμονής (άθροισμα όλων των επιμέρους χρόνων).

-Private static String maxMsg : καταγράφει από ποιο αρχείο προήλθε το maximum waiting time.

-Protected static String :

- inputFile : το αρχείο που περιέχει τα έγγραφα προς εκτύπωση.
- outputFile : το αρχείο στο οποίο θα τυπωθεί η αναφορά.

-Private static LinkedList<PrintJob> *pjs:* συνδεδεμένη λίστα που θα κρατάει τα αρχεία προς εκτύπωση.

- Private static MaxPQ<PrintJob> *pq:* συνδεδεμένη λίστα που θα κρατάει τα αρχεία προς εκτύπωση.

Μέθοδοι:

-private static void reset() : Αρχικοποιεί εκ νέου τις static μεταβλητές για μελλοντική χρήση του αλγορίθμου.

-private static void readFile() : Διαβάζει το αρχείο εισόδου,”κόβει” κατάλληλα την γραμμή που διαβάστηκε (με διαχωριστικό το κενό) και κατασκευάζει αντικείμενα PrintJob τα οποία και τοποθετεί στην λίστα.Η προτεραιότητα που θα έχει το PrintJob καθορίζεται από την συνάρτηση y=128-x όπου x το μέγεθος του αρχείου.Προφανώς όσο το x αυξάνει τόσο μειώνεται η προτεραιότητα και αντιστρόφως.Εάν υπάρχει χρονική ανακολουθία στις αφίξεις τότε το πρόγραμμα τερματίζεται.

-private static void clockTicks(int n) : Αυξάνει τον καθολικό μετρητή του χρόνου κατά n.

-private static void sendToBuffer () : Καλείται για να στείλει στην ουρά τα αρχεία που έχουν αφιχθεί.Με ένα βρόγχο while και όσο ο χρόνος του καθολικού μετρητή είναι μεγαλύτερος από τον χρόνο άφιξης του πρώτου στοιχείου της λίστας,αφαιρούμε το στοιχείο αυτό,ανανεώνουμε τον χρόνο αναμονής του και το τοποθετούμε στην ουρά.Εάν φτάσουμε στο σημείο που δεν υπάρχουν άλλα αρχεία προς εκτύπωση(η λίστα είναι άδεια) τότε ο βρόγχος σπάει.Σε περίπτωση που δεν έχουν αφιχθεί αρχεία,γίνεται αύξηση του καθολικού μετρητή ώστε να δείχνει στην στιγμή άφιξης του επόμενου (πρώτου στη λίστα) αρχείου.

-private static void doPrint() : Καλείται για να εκτυπώσει τα αρχεία που περιέχονται στην ουρά.Όσο η ουρά περιέχει αρχεία,αφαιρούμε το πρώτο (η remove() επιστρέφει το αρχείο με τη μεγαλύτερη προτεραιότητα) αρχείο , ανανεώνουμε τον χρόνο αναμονής του , προσθέτουμε τον χρόνο αναμονής τους στον συνολικο , ελέγχουμε εάν ο χρόνος αναμονής του ήταν μέγιστος και τέλος το στέλνουμε ως όρισμα στην συνάρτηση ολοκλήρωσης της εκτύπωσης completeJob.Εάν μετά την εκτύπωση του αρχείου (ο καθολικός χρονομετρητής έχει προχωρήσει όσο ήταν το μέγεθος του αρχείου) έχουμε άφιξη νέου(ων) αρχείου(ων) τότε σπάμε τον βρόγχο.

-private static void completeJob(PrintJob p) : Η συνάρτηση αυτή απλώς αυξάνει τον χρονομετρητή σύμφωνα με το μέγεθος του αρχείου.Ακολούθως στέλνει ως όρισμα το p στην συνάρτηση writeReport όπου θα γραφεί αναφορά για το αρχείο που μόλις εκτυπώθηκε.Επίσης αυξάνεται ο μετρητής completedPJ κατά 1.

-private static void writeReport(PrintJob p) : Η συνάρτηση αυτή γράφει μια αναφορά στο outputFile για το συγκεκριμένο αρχείο p στην ζητούμενη μορφή.Εάν φτάσαμε στο τέλος των εκτυπώσεων(η λίστα και η ουρά είναι άδειες) τότε καταγράφουμε και τα στατιστικά στοιχεία(μέσος/μέγιστος χρόνος αναμονής).

-public static void runA() : Η βασική συνάρτηση του αλγορίθμου.Κατασκευάζει αρχικά την συνδεδεμένη λίστα και τοποθετεί , με χρήση της μεθόδου readFile() , τα αρχεία προς εκτύπωση σε αυτήν.Ακολούθως κατασκευάζει την ουρά προτεραιότητας με μέγεθος ίδιο με το μέγεθος της λίστας. Ο καθολικός χρονομετρητής αυξάνεται μέχρι την στιγμή άφιξης του πρώτου αρχείου.Με χρήση ενός βρόγχου και συνθήκη τερματισμού εάν η λίστα είναι άδεια , καλούμε τις μεθόδους sendToBuffer και doPrint.

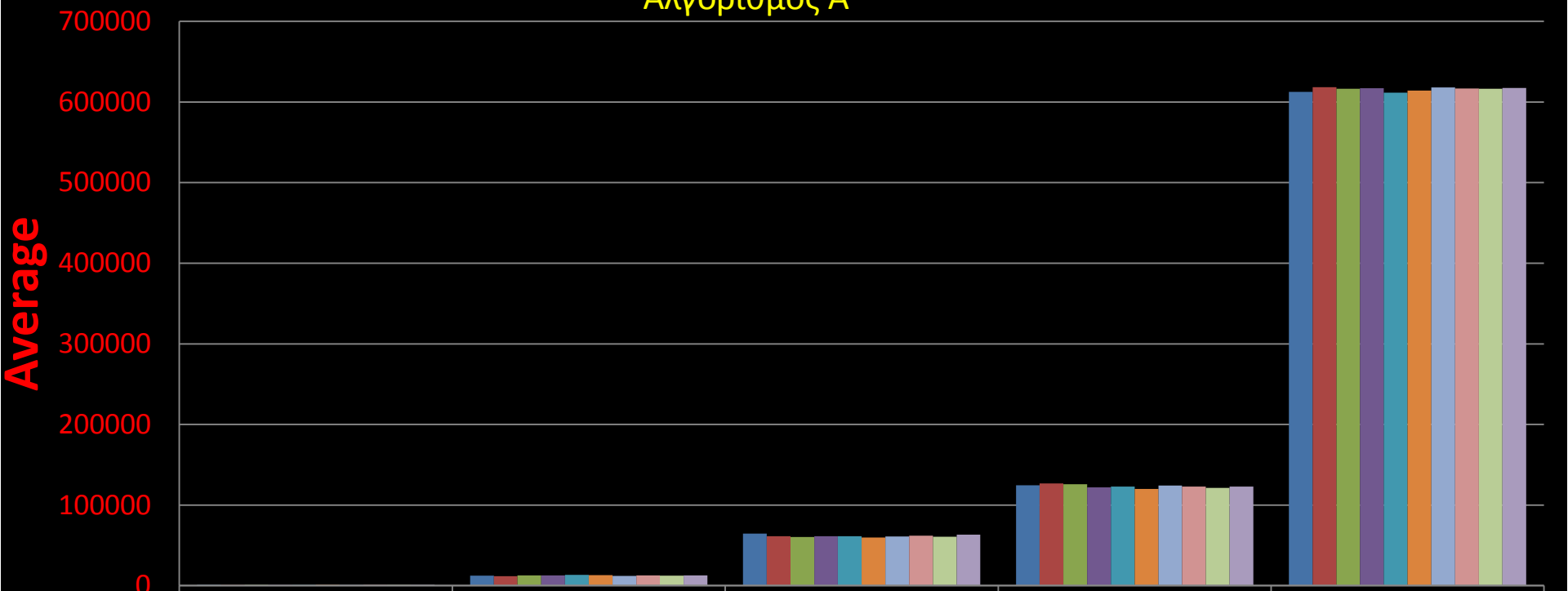
Επεξήγηση της υλοποίησης του αλγορίθμου Β

Ο αλγόριθμος Β υλοποιήθηκε εντελώς ανάλογα όπως ο αλγόριθμος Α.Οι διαφορές τους βρίσκονται:

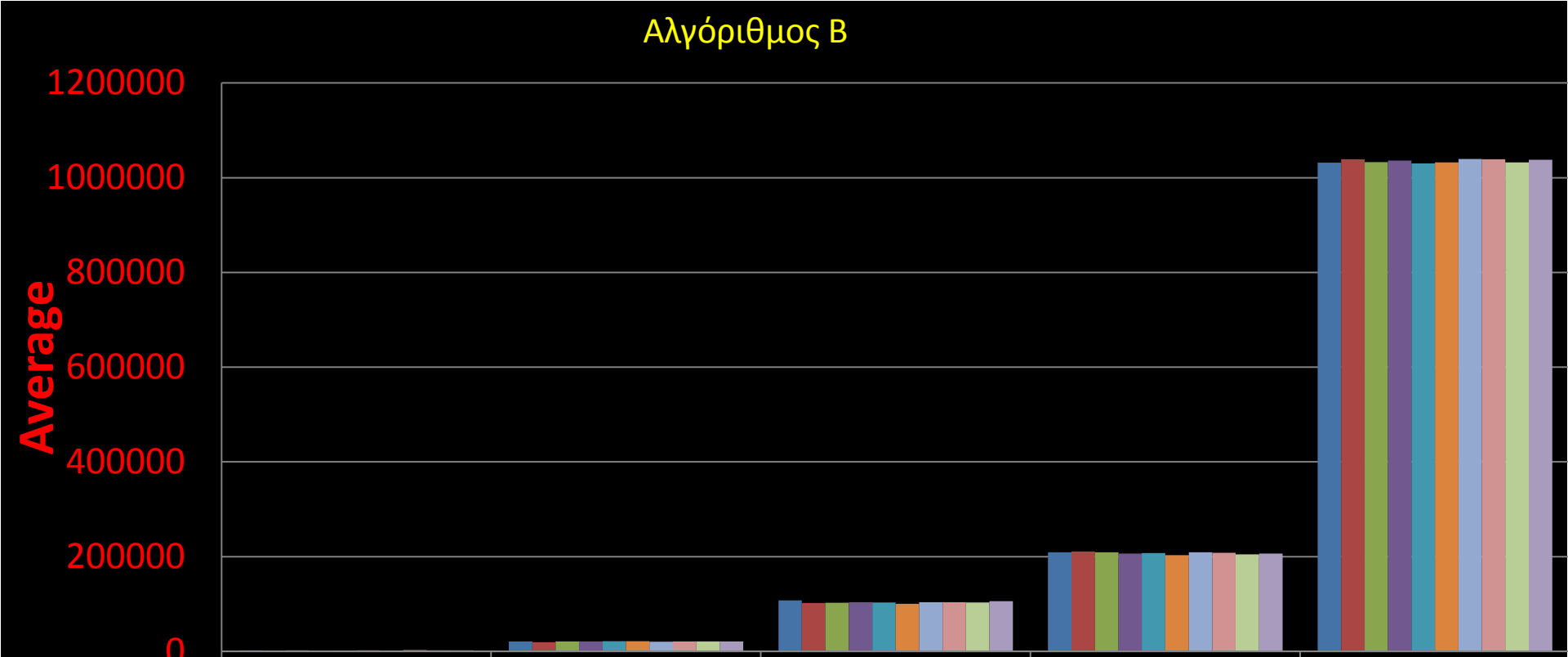
-Υλοποίηση μεθόδου private static void update() που ανανεώνει τις προτεραιότητες των αρχείων της ουράς σύμφωνα με τον τύπο priority = min(127, priority + waitingTime) που δίνεται.Για να επιτευχθεί η ανανέωση αφαιρούμε ένα στοιχείο της ουράς σε κάθε επανάληψη,αποθηκεύουμε προσωρινά το PrintJob , ανανεώνουμε τα πεδία waitingTime και priority του αντικειμένου PrintJob που αφαιρέσαμε και τέλος το προσθέτουμε σε μια προσωρινή λίστα.Επαναλαμβάνουμε το παραπάνω μέχρι να αδειάσει η ουρά.Ακολούθως αφαιρούμε ένα-ένα τα στοιχεία της λίστας και τα προσθέτουμε πάλι στην ουρα μέχρι να αδειάσει η λίστα.Σκοπός της παραπάνω διαδικασίας είναι να ταξινομηθεί σωστά η ουρά προτεραιότητας με τις καινούργιες προτεραιότητες.

-Αλλαγή υλοποίησης μεθόδου private static void clockTicks(int n) ώστε τα δευτερόλεπτα που προσθέτονται στον καθολικό χρονομετρητή να προσθέτονται 1 σε κάθε επανάληψη του βρόγχου μέχρι να προσθεθούν και τα n.Σκοπός αυτής της αλλαγής είναι να γίνεται σωστά η κλήση της μεθόδου update,δηλαδή ανα 15 δευτερόλεπτα.

Σύγκριση μέσου όρου αναμονής των 2 αλγορίθμων:

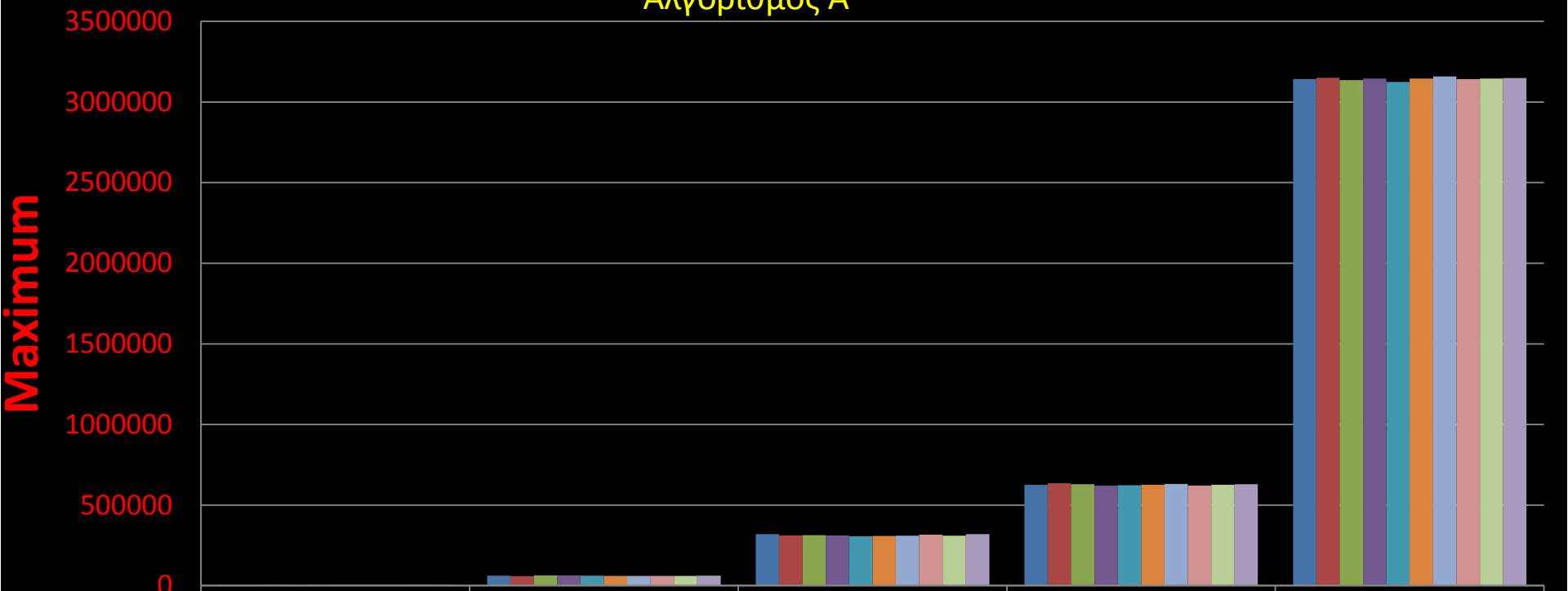


	100	1000	5000	10000	50000
Average1	1340,97	12476,16	64538,93	124513,63	612576,34
Average2	1025,56	11685,31	61229,38	126853,68	618287,04
Average3	1495,02	12712,85	60330,72	125727,26	616404,87
Average4	1112,91	12640,52	61500,49	122102,91	617120,51
Average5	1192,96	13407,01	61410,6	122940,61	611671,21
Average6	1303,91	12953,5	59723,08	120135,47	614079,04
Average7	1053,88	12230,69	61186,54	124119,5	618066,03
Average8	1506,05	12665,29	61900,12	123014,64	616589,21
Average9	1396,8	12619,72	60813,6	121302,22	616497,7
Average10	1450,62	12827,86	63351,28	123017,49	617259,02

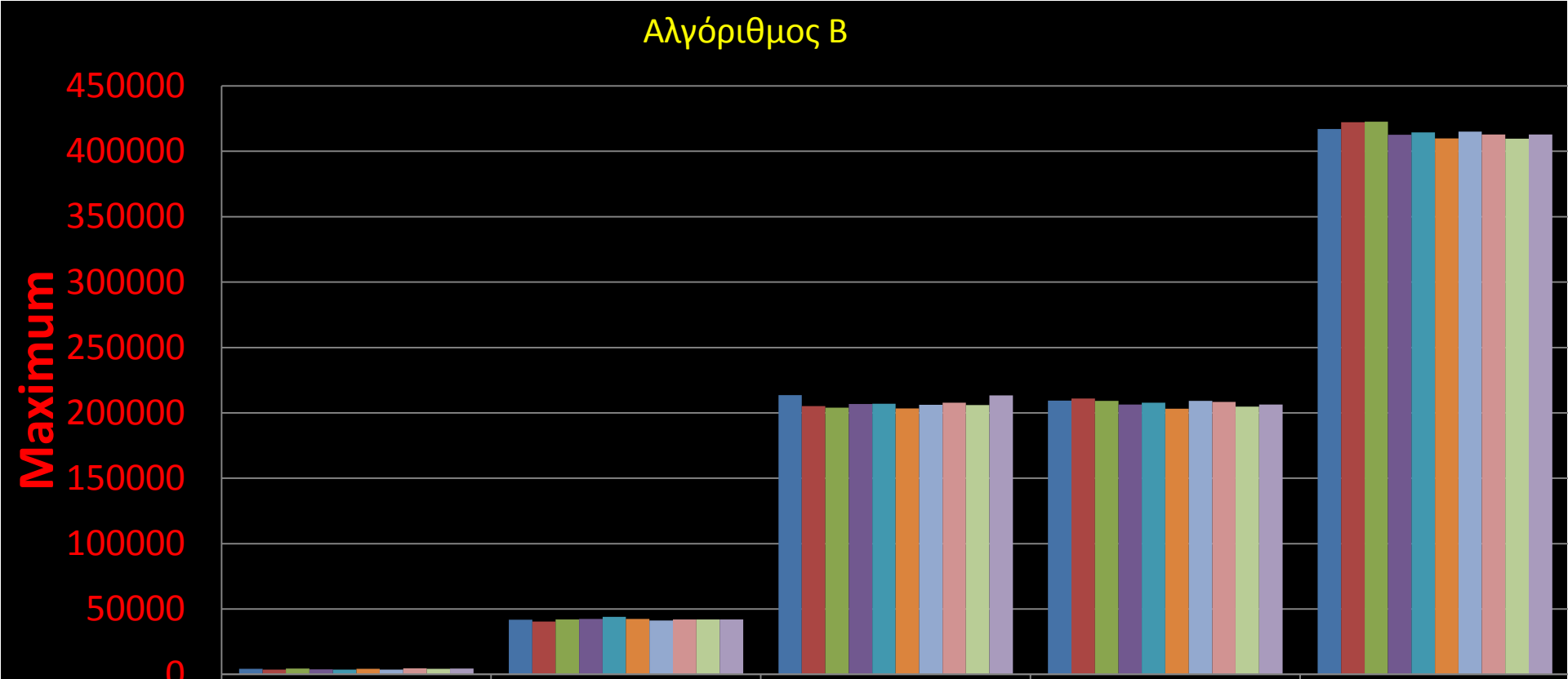


	100	1000	5000	10000	50000
Average1	2161,83	20910,32	107555,09	209310,02	1031582,48
Average2	1807,62	19383,31	102148,06	211060,74	1039002,86
Average3	2308,22	20832,21	102531,1	209264,2	1032754,42
Average4	1847,88	21242,45	103704,61	206418,66	1036140,94
Average5	1767,27	21784,7	103097,71	207686,25	1029988,49
Average6	2311,14	21758,08	100483,2	203273,31	1032506,06
Average7	1859,37	20669,59	103942,15	209143,77	1039297,3
Average8	2503,64	21062,81	103626,68	208397,81	1038841,11
Average9	2230,44	20729,03	103196,39	204734,06	1032482,46
Average10	2245,13	21156,36	105722,46	206280,12	1037829,98

Σύγκριση μέγιστου χρόνου αναμονής των 2 αλγορίθμων:



	100	1000	5000	10000	50000
Maximum1	6131	63084	320579	626109	3142057
Maximum 2	4881	58476	312174	635617	3150130
Maximum 3	6246	63161	312912	629694	3135067
Maximum 4	5491	63675	311867	621652	3144933
Maximum 5	5197	62862	307358	622978	3124407
Maximum 6	5956	61195	308891	625531	3144857
Maximum 7	5032	61368	309573	630248	3158034
Maximum 8	6444	61475	315989	621056	3141563
Maximum 9	5678	60275	309903	625902	3144475
Maximum 10	6150	63039	320121	628762	3148831



	100	1000	5000	10000	50000
Average1	4304	41733	213589	209310,02	417061
Average2	3660	40436	205235	211060,74	422194
Average3	4438	42043	203994	209264,2	422571
Average4	3837	42469	206817	206418,66	412659
Average5	3762	43918	207052	207686,25	414398
Average6	4250	42508	203421	203273,31	409908
Average7	3696	41197	206220	209143,77	415110
Average8	4686	41955	207866	208397,81	412928
Average9	4363	41997	206066	204734,06	409654
Average10	4522	42018	213304	206280,12	412934

Παρατηρούμε ότι ο αλγόριθμος Α έχει εμφανώς καλύτερο μέσο χρόνο αναμονής από τον αλγόριθμο Β , όμως εμφανίζει μεγαλύτερο μέγιστο χρόνο αναμονής απ’ότι ο αλγόριθμος Β.