ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
TECHNICAL UNIVERSITY OF CRETE

Δομές Δεδομένων και Αλγόριθμοι

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Άσκηση 1 - Επεξεργασία αρχείων

Όνομα: Ρούλι Ντάνιελ
ΑΜ: 2018030019

22.03.2023

1 Περιγραφή Κώδικα

1.1 Ανάλυση

Ο κώδικας χωρίζεται σε τρία τμήματα:

1. model: Περιέχει την κύρια λειτουργικότητα του προγράμματος, η οποία είναι η υλοποίηση αλγόριθμων αναζήτησης σε αρχεία. Οι μέθοδοι αναζήτησης περιέχονται στην κλάση RFile, η οποία αντιπροσωπεύει μια αφαίρεση ενός αρχείου με λειτουργικότητες γραμμικής και δυαδικής αναζήτησης, διαχωρισμού του κύριου αρχείου σε αρχείο κλειδιών και θέσεων, και ταξινόμησής του ως προς τα κλειδιά. Για την υλοποίηση των παραπάνω δημιουργήθηκαν οι συναρτήσεις:
   - searchFile(int key) για γραμμική αναζήτηση στο αρχείο δεδομένων και στο αρχείο κλειδιών
   - binarySearch(int key) και binarySearch2(int key) για δυαδική αναζήτηση στο ταξινομημένο αρχείο κλειδιών. Η διαφορά της πρώτης σε σχέση με τη δεύτερη είναι ότι η πρώτη, σε κάθε βήμα του αλγορίθμου διαβάζει ένα κλειδί (4 bytes από το αρχείο), το συγκρίνει με το κλειδί που έχει δοθεί ως όρισμα και αν είναι μεγαλύτερο ή μικρότερο, ο δείκτης της θέσης του αρχείου μετακινείται αριστερά ή δεξιά κατά 8 bytes αντίστοιχα. Η δεύτερη υλοποίηση σε κάθε βήμα του αλγορίθμου διαβάζει μία σελίδα (256 bytes), η οποία αποθηκεύεται σε μια λίστα στην κεντρική μνήμη, και συγκρίνει το δοσμένο κλειδί με το πρώτο και το τελευταίο στοιχείο της λίστας. Αν το κλειδί είναι μεγαλύτερο από το τελευταίο στοιχείο της λίστας τότε ο δείκτης του αρχείου μετακινείται στη θέση της επόμενης για να τη διαβάσει, αν το κλειδί είναι μικρότερο από το πρώτο στοιχείο της λίστας μετακινείται στην προηγούμενη σελίδα, και τέλος αν βρίσκεται στο εύρος μεταξύ του πρώτου και του τελευταίου και δε βρεθεί τότε επιστρέφει. Διευκρινίζεται ότι για τη διεξαγωγή των μετρήσεων χρησιμοποιήθηκε η δεύτερη υλοποίηση.
   - split() η οποία δημιουργεί το index file του αρχείου δεδομένων
   - sort() για την ταξινόμηση του index file.
2. modelTesting: Περιέχει τις εξή κλάσεις:
   - App : Περιέχει τη main() και μεθόδους που δημιουργούν τα αρχεία, παίρνουν τις ζητούμενες μετρήσεις και τις τυπώνουν στην κονσόλα.
   - DataGenerator : Γεννήτρια δεδομένων η οποία δημιουργεί N τυχαία δεδομένα με τη generateRecords(), μεγέθους 55 ή 27 bytes τα οποία αποθηκεύονται στο εκάστοτε αρχείο με τη fillFile()
   - RandomString : Κλάση που δημιουργεί τυχαία αλφαριθμητικά
   - TestGenerator : Γεννήτρια μετρήσεων η οποία περιέχει τις μεθόδους¨randomSearchCounter(), randomSearchCounterIndexFile και randomBinarySearchCounter() που κάνουν αναζητήσεις τυχαίων κλειδιών μέσω των συναρτήσεων searchFile(int key) και binarySearch2(int key) και επιστρέφουν τον μέσο αριθμό προσπελάσεων στον δίσκο ανά αναζήτηση και τον μέσο χρόνο ανά αναζήτηση.
   - TestStructure : κλάση αποθήκευσης μετρήσεων
3. utils: Περιέχει μερικές χρήσιμες μεθόδους, η λειτουργικότητα των οποίων περιγράφεται σε javadoc.

1.2 Διευκρινήσεις

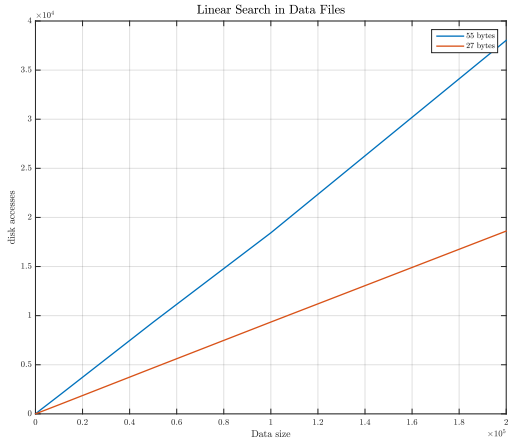
Η κλάση TestGenerator κάνει implement τη διεπφή Runnable και για κάθε N δημιουργείται ένα ξεχωριστό νήμα το οποίο δημιουργεί δεδομένα τα οποία αποθηκεύονται σε ένα ξεχωριστό αρχείο, το οποίο είναι μοναδικό για κάθε N. Αυτή η σχεδιαστική επιλογή είχε ως αποτέλεσμα τη μείωση του συνολικού χρόνου εκτέλεσης του προγράμματος, καθώς κάθε νήμα προσπελάζει διαφορετικούς πόρους.

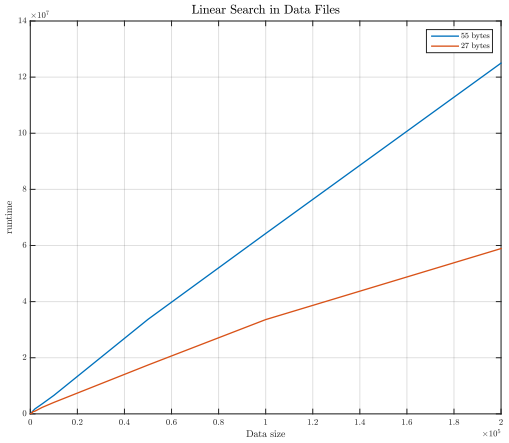
2 Σύγκριση μεθόδων και μετρήσεις

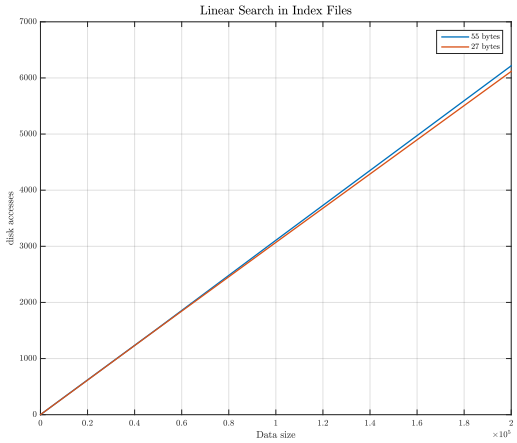
Τα αποτελέσματα των μετρήσεω φαίνονται στους παρακάτω πίνακες:

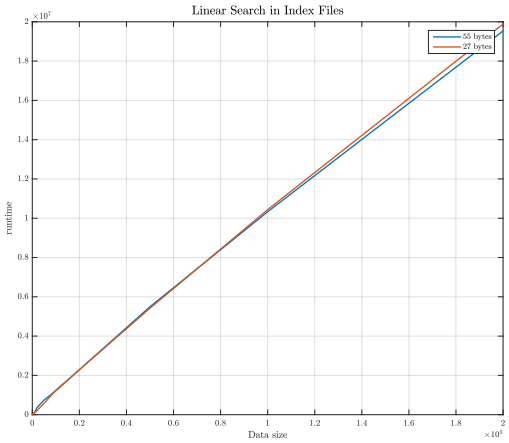
Αριθμός προσπελάσεων στον δίσκο ανά αναζήτηση						
	Τρόπος Α		Τρόπος Β		Τρόπος C	
N	55 bytes	27 bytes	55 bytes	27 bytes	55 bytes	27 bytes
50	10.1	5.3	2.0	2.0	1.5	1.5
100	18.8	9.7	4.0	3.9	1.7	1.6
200	38.4	18.8	7.0	6.9	1.8	1.8
500	94.2	46.0	15.9	15.8	2.7	2.6
800	149.0	74.9	24.8	24.5	3.0	3.0
1000	184.6	93.9	31.8	31.3	3.3	3.2
2000	384.3	186.9	62.4	61.9	3.9	3.8
5000	934.8	461.6	155.3	154.7	4.9	4.7
10000	1858.2	926.7	310.7	307.4	5.3	5.3
50000	9332.1	4679.6	1543.2	1539.4	6.7	6.9
100000	18432.6	9348.2	3107.3	3069.1	7.9	7.9
200000	38032.6	18611.3	6216.5	6116.4	8.6	8.5

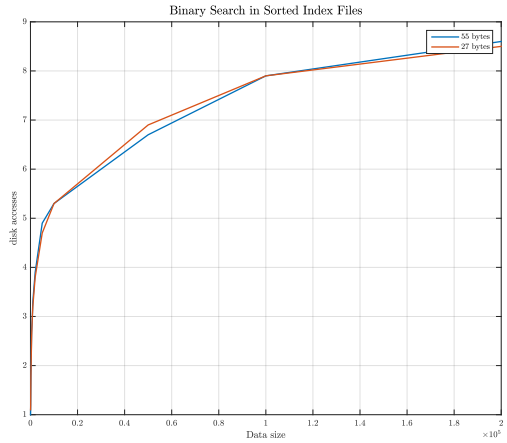
Χρόνος ανά αναζήτηση (ns)						
	Τρόπος Α		Τρόπος Β		Τρόπος C	
N	55 bytes	27 bytes	55 bytes	27 bytes	55 bytes	27 bytes
50	34190	28415	8990	8840	5736	6630
100	65637	33403	22237	21438	5913	6750
200	163987	68069	28256	28322	9567	8252
500	322158	208336	57118	53782	12375	11492
800	736117	279278	94969	99066	10562	13834
1000	918381	764489	121685	112150	12843	15377
2000	1791863	964744	357626	203193	19924	16968
5000	3533965	2287782	744633	580226	21817	21675
10000	6572887	4073538	1219120	1245728	24233	22267
50000	33679282	17400777	5496308	5418635	27795	30203
100000	64270896	33592680	10330172	10437649	27509	28636
200000	124998907	58907292	19531447	19883408	30361	31537

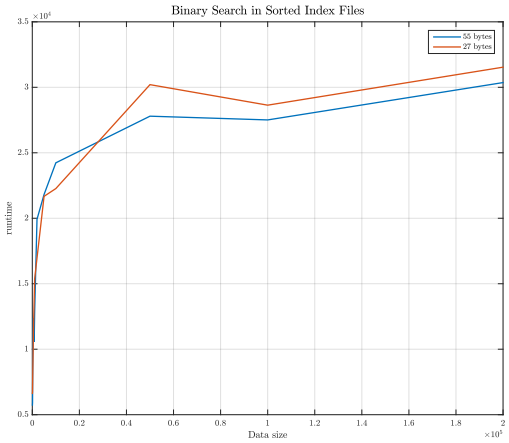












3 Ανάλυση μετρήσεων

3.1 Προσπελάσεις στον δίσκο ανά αναζήτηση

Από τις παραπάνω γραφικές παραστάσεις φαίνεται ότι η σειριακή αναζήτηση κλειδιών σε ένα αρχείο δεδομένων έχει γραμμική συμπεριφορά, δηλαδή όσο αυξάνεται το μέγεθος του αρχείου, τόσο αυξάνονται και οι προσπελάσεις στον δίσκο. Παρατηρούμε ότι η προσπέλαση στο αρχείο με τα δεδομένα μεγέθους 27 bytes έχει μικρότερη κλίση σε σχέση με τις προσπελάσεις στο αρχείο με 55 bytes δεδομένων, το οποίο είναι αναμενόμενο αφού σε μια σελίδα δίσκου χωρούν περισσότερα δεδομένα των 27 bytes σε σχέση με τα 55 bytes.
Η γραμμική αναζήτηση στο αρχείο κλειδιών έχει, όπως και η γραμμική αναζήτηση στο αρχείο δεδομένων, έχουν πολυπλοκότητα O(n). Η διαφορά της σε σχέση με την αναζήτηση στο αρχείο δεδομένων είναι ότι πλέον δεν υπάρχει διαφορά στην κλίση των δυο ευθειών αφού και για τα 27 και για τα 55 bytes το index File περιέχει μόνο τα κλειδιά και τις θέσεις τους στο αρχικό αρχείο που έχουν σταθερό μέγεθος 8 bytes στο αρχείο.
Η δυαδική αναζήτηση στο ταξινομημένο αρχείο έχει λογαριθμική πολυπλοκότητα $(O(\log n))$, το οποίο μπορεί να διαπιστωθεί και από την αντίστοιχη γραφική παράσταση.

3.2 Μέσος χρόνος ανά αναζήτηση

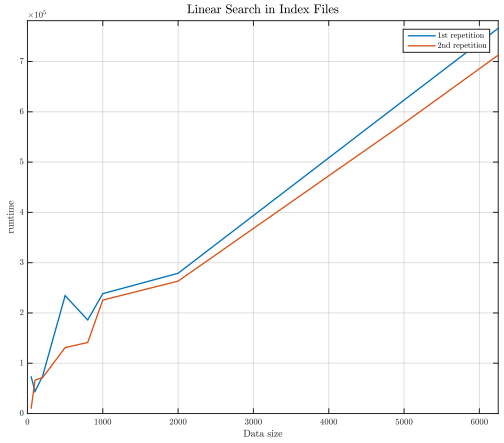
Παρότι είναι εμφανές ότι οι καμπύλες μέσου χρόνου αναζήτησης είναι παρόμοιες με τον αριθμό προσπελάσεων δίσκου ανά αναζήτηση, υπάρχει μια κρίσιμη διαφορά. Μπορεί να παρατηρηθεί ότι στη γραμμική αναζήτηση οι γραμμές είναι τεθλασμένες και για μεγάλο αριθμό δεδομένων ο μέσος χρόνος αναζήτησης στο αρχείο είναι πιο μικρός από ότι θα προβλέπαμε. Το ίδιο είναι εμφανές και στην περίπτωση της δυαδικής αναζήτησης, όπου το φαινόμενο είναι αρκετά πιο έντονο. Μια πιθανή εξήγηση για αυτό είναι ότι το λει-

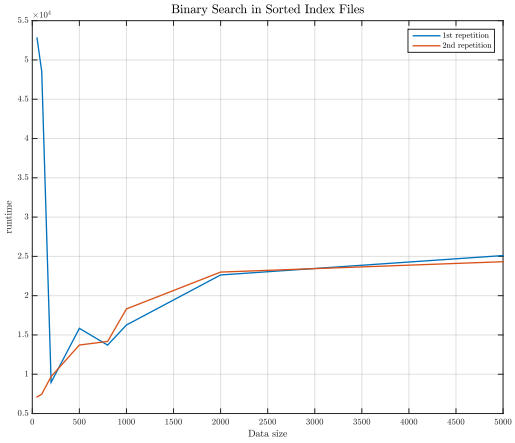
τουργικό σύστημα αποθηκεύει τα δεδομένα που πρόκειται να επαναχρησιμοποιηθούν σε σύντομο χρονικό διάστημα (στην προκειμένη περίπτωση τις σελίδες δίσκου που η randomSearch θα χρειαστεί να ξαναδιαβάσει) στην κεντρική μνήμη έτσι ώστε να διαβαστούν πιο γρήγορα.

3.3 Περαιτέρω ανάλυση

Κατά την εκτέλεση του κώδικα παρατηρήθηκε το εξής: ο μέσος χρόνος εκτέλεσης της γραμμικής αναζήτησης στο αρχείο κλειδιών, καθώς και της δυαδικής αναζήτησης, και ειδικότερα για μικρό αριθμό κλειδιών είναι ασυνήθιστα μεγάλος. Μια πιθανή ερμηνεία για αυτό το φαινόμενο είναι το γεγονός ότι το Class Loading στο Java Virtual Machine είναι εξαιρετικά αργό την πρώτη φορά που γίνεται κλήση μιας κλάσης. Αυτό έχει ως αποτέλεσμα, οι μέθοδοι που καλούν μια κλάση ή μια μέθοδο αυτής της κλάσης (ακόμη και αν δεν κάνουμε new), να κάνουν παραπάνω χρόνο να εκτελεστούν. Αυτό είναι ιδιαίτερα εμφανές στις περιπτώσεις όπου ο χρόνος εκτέλεσης της μεθόδου είναι κατά πολύ μικρότερος του class loading μιας συγκεκριμένης κλάσης. Στον παρακάτω πίνακα παρατίθενται μετρήσεις που έγιναν σε διαδοχικές επαναλήψεις των τυχαίων αναζητήσεων. Όπως είναι εμφανές, στη δεύτερη επανάληψη των τυχαίων αναζητήσεων οι μεταβολές είναι αρκετά πιο ομαλές σε σχέση με την πρώτη επανάληψη.

Χρόνος ανά αναζήτηση (ns)				
	Τρόπος Β		Τρόπος C	
N	1η εκτέλεση	2η εκτέλεση	1η εκτέλεση	2η εκτέλεση
50	72823	10872	52797	7106
100	43986	66384	48551	7465
200	73950	71281	8973	9705
500	234388	131106	15841	13719
800	186155	141318	13709	14174
1000	238352	225593	16266	18314
2000	278916	263312	22635	23001
5000	623253	577416	25104	24322
10000	1194880	1118776	23208	25150
50000	5267581	5551185	29347	35069
100000	11171241	10573279	30960	29393
200000	20304513	20082975	31780	32501





4 Οδηγίες εκτέλεσης προγράμματος

Για την εκτέλεση του προγράμματος είναι απαραίτητη η έκδοση 14 του Java Development Kit καθώς στο πρόγραμμα χρησιμοποιούνται records. Το πρόγραμμα δε θα τρέξει σε Linux, μάλλον λόγω διαφορών που υπάρχουν στον ορισμό των paths των αρχείων, μεταξύ Windows και Linux, όταν γίνεται new File("path/to/file/"+filename), αλλά δεν είμαι βέβαιος. Δεν υπάρχουν παραπάνω λειτουργικότητες στο πρόγραμμα, πέραν αυτών που αναφέρθηκαν κατά την ανάλυση. Ο αναμενόμενος χρόνος εκτέλεσης του προγράμματος είναι περίπου 5 λεπτά για 1000 τυχαίες αναζητήσεις και για όλες τις μετρήσεις, με αποκλίσεις που είναι σχετικές με την ταχύτητα του επεξεργαστή.