ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΡΧΕΙΩΝ

ΚΑΣΦΙΚΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
LAB 21124888

Μέρος 1ο: Επεξεργασία Αρχείων

ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΑΣΚΗΣΗΣ.

Σκοπός της άσκησης αυτής είναι η εξοικείωση με την απόδοση μεθόδων αναζήτησης στο δίσκο. Συγκεκριμένα ζητείται η δημιουργία ενός δυαδικού αρχείου, το οποίο θα περιέχει Ν πλήθος ακεραίων σε δυαδική μορφή. Το αρχείο αυτό περιέχει «σελίδες» μεγέθους 512 bytes. Το αρχείο αρχικοποιείται με Ν πλήθος σελίδων. Η πλήρωση του αρχείου γίνεται διαδοχικά για τα στοιχεία 1 έως Ν. Κάθε στοιχείο κατατάσσεται σε συγκεκριμένη σελίδα του αρχείου ανάλογα με μία συνάρτηση H(key) όπου key το στοιχείο. Η αναζήτηση υλοποιείται με την εισαγωγή ενός τυχαίου αριθμού με δύο διαφορετικούς τρόπους.

1ος τρόπος: Βρίσκουμε σε ποια σελίδα δίσκου πρέπει να βρίσκεται το στοιχείο βάσει της συνάρτησης H(key) και ψάχνουμε διαδοχικά την σελίδα αυτή και overflow σελίδες της, μέχρι να βρούμε το συγκεκριμένο στοιχείο.

$2^{ος}$ τρόπος: Ξεκινάμε από αρχή του δυαδικού αρχείου, διαβάζοντας σειριακά σελίδα προς σελίδα, μέχρι να βρούμε την σελίδα του στοιχείου που μας ενδιαφέρει.

ΥΛΟΠΟΙΗΣΗ.

Εισαγωγή στοιχείων:
Για την ολοκλήρωση της άσκησης ήταν απαραίτητη η δημιουργία ενός αρχείου με συγκεκριμένο πλήθος σελίδων, οπότε δημιουργείται αρχικά ένα αρχείο το οποίο περιέχει Ν αριθμό σελίδων και τα 127 (από 128) στοιχεία της κάθε σελίδας είναι 0 (κενά) ενώ το τελευταίο είναι -1, επειδή δεν υπάρχουν overflow σελίδες. Για την αρχικοποίηση του αρχείου χρησιμοποιείται η μέθοδος InitializeFile() και AddPage() της κλάσης File. ΄Επειτα για την εισαγωγή των Ν στοιχείων στο αρχείο καλείται η συνάρτηση FillFile() και HKey().

- AddPage():
  Προσθέτει μία σελίδα στο αρχείο μετά την τελευταία ή αν το αρχείο δεν περιέχει σελίδες δημιουργεί την πρώτη σελίδα, η οποία περιέχει 127 στοιχεία με τιμή 0, ενώ το $128^{ο}$ έχει τιμή -1.
- InitializeFile():
  Δημιουργεί ένα αρχείο τυχαίας προσπέλασης, το οποίο το ονομάζουμε project.bin, ελέγχει εάν το αρχείο αυτό υπάρχει ήδη στο δίσκο (εάν υπάρχει το διαγράφει) και γράφει το αρχείο στο δίσκο.

- Στην συνέχεια για την αρχιοποίηση του αρχείου καλείται η συνάρτηση AddPage() Μ φορές, όπου Μ το πλήθος των σελίδων.
- FillFile():
  Σκοπός της είναι η εισαγωγή Ν στοιχείων στο δυαδικό αρχείο, στην κατάλληλη σελίδα κάθε φορά. Για να επιτευχθεί αυτό χρησιμοποιείται η συναρτηση HKey(), η οποία δείχνει σε ποια σελίδα πρέπει να αποθηκευθεί το εκάστοτε στοιχείο. Η FillFile() δημιουργεί όταν χρειάζεται overflow σελίδες χρησιμοποιώντας την μέθοδο AddPage(). Ετσι το κάθε στοιχείο αποθηκευεται στη κατάλληλη σελίδα με βάση την Hkey(), ή εάν αυτή η σελίδα είναι ήδη γεμάτη, ψάχνει να βρεί overflow σελίδες, που να μην έχουν πληρωθεί ακόμη.
- HKey():
  Επιστρέφει τον αριθμό της σελίδας που πρέπει να αποθηκευθεί το εκάστοτε στοιχείο.
- IntegerToByte():
  Μετατρέπει έναν ακέραιο αριθμό στο 32bit binary του, ο οποίος είναι αποθηκευμένος σε έναν πίνακα byte τεσσάρων στοιχείων με Little Endian. Η IntegerToByte() χρησιμοποιείται βοηθητικά από την FillFile().
- ByteToInteger():
  Μετατρέπει έναν Little Endian πίνακα τεσσάρων byte σε ακέραιο.
- CloseFile():
  Κλείνει το αρχείο.

Για την εγγραφή του αρχείου στο δίσκο δημιουργείται στην Main() ένα instance της κλάσης File και στην συνέχεια καλούνται διαδοχικά η InializeFile() και η FillFile(), ενώ μετά από κάθε επεξεργασία και αναζήτηση στο αρχεί καλείται η CloseFile(). Η Main() βρίσκεται σε μία κλάση η οποία περιέχει μόνο στατικές μεθόδους που ονομάζεται MainClass.

1η Μέθοδος Αναζήτησης:
Η 1^{η} μέθοδος αναζήτησης επιδιώκει την εύρεση ενός τυχαίου αριθμού – κλειδιού στο υπάρχον δυαδικό αρχείο με την χρήση της συνάρτησης HKey() ώστε να βρεθεί η σελίδα του δυαδικού αρχείου, στην οποία το κλειδί αριθμός θα έπρεπε να έχει καταχωρηθεί. Αρχικά βρίσκουμε τον δείκτη της σελίδας που μας ενδιαφέρει χρησιμοποιώντας την μέθοδο HKey() της κλάσης File. Στην συνέχεια διαβάζουμε την σελίδα αυτή στην μνήμη, την επεξεργαζόμαστε ψάχνοντας ένα ένα τα στοιχεία της, και εάν η σελίδα δεν περιέχει τον αριθμό κλειδί, διαβάζουμε διαδοχικά και τις overflow σελίδες της, μέχρι να βρούμε τον αριθμό κλειδί. Η διαδικασία αυτή πραγματοποιείται μέσω της μεθόδου Search() της κλάσης FirstSearchMethod.

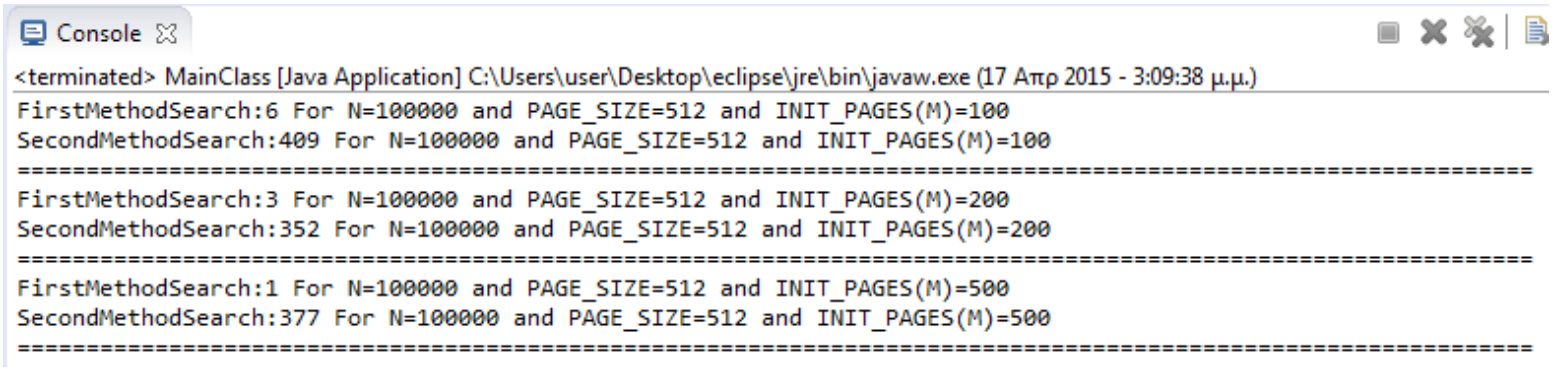
2η Μέθοδος Αναζήτησης:
Η 2^{η} μέθοδος επιδιώκει την εύρεση ενός τυχαίου αριθμού κλειδιού στο δυαδικό αρχείο διαβάζοντας μια μια τις σελίδες του αρχείου, επεξεργαζόμενη την κάθε σελίδα, ψάχνοντας για τον αριθμό κλειδί. Η αναζήτηση αυτή πραγματοποιείται μέσω της μεθόδουSearch() της SecondSearchMethod.

Τόσο η FirstSearchMethod όσο και η SecondSearchMethod αποτελούν υποκλάσεις της Abstract κλάσης Search. Η κλάση Search περιέχει μια αφηρημένη μέθοδο Search η λειτουργία της οποίας ορίζεται ανάλογα και τις παρακάτω μεθόδους οι οποίες χρησιμοποιούνται και στις δύο υποκλάσεις:

- GetDiskAccesses():
  Ουσιαστικά μετά την ολοκλήρωση είτε της πρώτης είτε της δεύτερης μεθόδου, επιστρέφει την τιμή της μεταβλητής DiskAccesses, η οποία αποθηκεύει τον αριθμό των προσβάσεων στο δίσκο που πραγματοποίησε το πρόγραμμα, προκειμένου να βρεθεί ο τυχαίος αριθμος κλειδί σε κάθε μέθοδο.
- ByteToInteger():
  Μετατρέπει έναν Little Endian πίνακα τεσσάρων byte σε ακέραιο.
- Read():
  Περιέχει την εντολή της βιβλιοθήκης java.io read και αυξάνει τον ακέραιο diaskaccesses κατά 1.

Οι μέθοδοι Search() των κλάσεων FirstSearchMethod και SecondSearchMethod καλούνται στη Main() της κλάσης MainClass, ενώ τα αποτελέσματά τους εμφανίζονται στο console.

ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΜΕΘΟΔΩΝ ΑΝΑΖΗΤΗΣΗΣ



Τα αποτελέσματα των δύο μεθόδων φαίνονται στην παραπάνω εικόνα, για N = 100.000 και για M = 100, M = 200, M = 500 (M: αριθμός σελίδων). Ενώ κανονικά το N θα έπρεπε να είναι από την εκφώνηση της άσκησης 10^7 είναι 100.000 καθώς για N = 10^7, το «γέμισμα» του αρχείου απαιτεί πολύ χρόνο. Όπως φαίνεται, λοιπόν, η πρώτη μέθοδος είναι εμφανώς πιο αποτελεσματική από την δεύτερη καθώς χρειάζεται σημαντικά λιγότερες προσβάσεις στο δίσκο προκειμένου να φέρει ένα αποτέλεσμα.
Οσον αφορά τον αριθμό των αρχικών σελίδων του αρχείου με την απόδοση των δύο μεθόδων, ενώ η δεύτερη μέθοδος δεν επηρεάζεται αισθητά, αυξάνεται η αποτελεσματικότητα της πρώτης μεθόδου καθώς μειώνεται ο αριθμός των overflow σελίδων.
Έστω ότι το αρχείο που δημιουργούμε περιέχει είτε αύξοντα είτε φθίνοντα ταξινομημένα στοιχεία. Μια δυαδική αναζήτηση στο αρχείο αυτό θα ήταν σαφώς αποτελεσματικότερη από την σειριακή αναζήτηση και για πεπερασμένο αριθμό

στοιχείων θα ήταν αποτελεσματικότερη και από την πρώτη. Ωστόσο η λογική της πρώτης μεθόδου είναι αποτελεσματικότερη για μικρό σχετικά στοιχείων.