



**ΠΟΛΥΤΕΧΝΕΙΟ
ΚΡΗΤΗΣ**

*[ΠΛΗ 201] ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ
ΚΑΙ ΑΡΧΕΙΩΝ*

1^Η ΑΣΚΗΣΗ

ΚΑΤΣΙΜΠΑΣ ΠΕΤΡΟΣ

2016030038

File Manager

Υλοποιήθηκαν όλες οι συναρτήσεις όπως ζητάει η εκφώνηση. Τα .JAVA αρχεία βρίσκονται στο (*default package*), δηλαδή το *FileManager.java* με τις διπλανές λειτουργίες και το αρχείο *Main.java* όπου και τρέχει το πρόγραμμα.

```
public byte[] FileHandling(String fileName) throws IOException {}

public int CreateFile(int filePos, int pageNum) throws FileNotFoundException, IOException {}

public int OpenFile(String filename) throws FileNotFoundException {}

public int ReadBlock(String filename, int pageNum) throws IOException {}

public int ReadNextBlock(String filename, int pageNum) throws IOException {}

public int ReadPrevBlock(String filename, int pageNum) throws IOException {}

public int WriteBlock(String filename, int filePos, byte buffer[]) throws IOException {}

public int WriteNextBlock(String filename, int filePos, byte buffer[]) throws IOException {}

public int AppendBlock(String filename, byte buffer[]) throws IOException {}

public int deleteBlock(String filename, int pageNum) throws FileNotFoundException, IOException {}

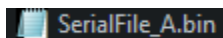
public int CloseFile(String filename, byte[] bufinfo) throws FileNotFoundException, IOException {}
```

Οργάνωση πληροφορίας σε σειριακό αρχείο και απόδοση αναζήτησης

Το δεύτερο μέρος της άσκησης έχει υλοποιηθεί στο *package.organi*, το οποίο περιέχει το *SerialFile.java* με τις συναρτήσεις του προγράμματος και τη *MainClass* (στο τέλος), η βοηθητική κλάση *Node.java*.



Α' ΤΡΟΠΟΣ

Για τον πρώτο τρόπο οργάνωσης αρχείου δημιουργήσαμε ένα αρχείο 320.000 bytes (40.000 bytes για τα τυχαία κλειδιά και 280.000 bytes για την τυχαία πληροφορία, παραγώμενα με τη βοήθεια του φροντηστηριακού υλικού). Όπως φαίνεται στη συνάρτηση γεμίζαμε ένα buffer 128 bytes, δηλαδή 4 εγγραφές – 1 σελίδα, μέχρι να φτλασουμε τον αριθμό των 10.000 κλειδιών. Εν τέλει, με την χρήση βιβλιοθηκών της JAVA δημιουργούσαμε το αρχείο



SerialFile_A.bin. Στο τέλος της δημιουργίας, εξάγαμε τον αριθμό προσβάσεων του δίσκου, ο οποίος είναι σταθερός καθώς μιλάμε για σειριακή οργάνωση.



B' ΤΡΟΠΟΣ

Για τον δεύτερο τρόπο χρησιμοποιήσαμε το αρχικό παραγώμενο αρχείο και με την βιβλιοθήκη *RandomAccessFile* και συγκεκριμένα με την συνάρτηση *.seek* τροποποιούσαμε τον *FilePointer* με αποτέλεσμα να εξάγουμε μόνο τα *keys* και επίσης να τοποθετούμε τον αριθμό σελίδας στο παραγώμενο αρχείο. Έτσι, δημιουργούνται τα αρχεία  *SerialFile_B.bin* αλλά και το αρχείο με την υπόλοιπη πληροφορία  *SerialFile_B_info.bin* με μέγεθος 80.000 για το πρώτο (40.000 keys και 40.000 page number (int)) και 280.000 για το δεύτερο. Στο τέλος της δημιουργίας, εξάγαμε τον αριθμό προσβάσεων του δίσκου, ο οποίος είναι σταθερός καθώς μιλάμε για σειριακή οργάνωση.

ΑΝΑΖΗΤΗΣΕΙΣ

Υλοποιήθηκαν συναρτήσεις 20 αναζητήσεων και για το Α' αρχείο και για το Β'. Η αναζήτηση ξεκινάει από την εύρεση ενός ήδη υπάρχον τυχαίου κλειδιού με σκοπό την αναζήτηση, η οποία γίνεται σειριακά και για κάθε σελίδα που προσπελάσουμε προσθέτουμε μία πρόσβαση στο δίσκο (συν άλλη μία πρόσβαση στο δίσκο για το αρχείο Β, το οποίο για να πάρουμε την πληροφορία πρέπει να μεταβούμε σε συγκεκριμένη σελίδα του αρχείου Β.info).

External Sorting of 2 Files

Για την εξωτερική ταξινόμηση χρειαστήκαμε τις λειτουργίες των ArrayList αλλά και το δοθέν override του φροντηστηριακού υλικού. Πετύχαμε με αυτό το τρόπο να έχουμε έναν ταξινομημένο πίνακα βάσει του κλειδιού με αύξουσα σειρά, το οποίο γράψαμε και δημιουργούσαμε τα αρχεία  *ExternalFile_A.bin* και  *ExternalFile_B.bin*. Δυστυχώς, όμως, με τη χρήση του ArrayList λόγω επιπλέον πληροφοριών που απαιτούνται για τη δημιουργία του, αυξήθηκαν τα bytes/Node. Όπως και πριν, καταμετρήσαμε τις προσβάσεις στον δίσκο που χρειάστηκε η λειτουργία αυτή.

Binary Searches in 2 Files

Για τις δυαδικές αναζητήσεις υλοποιήσαμε 2 συναρτήσεις ώστε να πετύχουμε την αναδρομή. Η συνάρτηση, δυστυχώς, δεν δουλεύει σωστά καθώς «σκάει» σε κάποια αλλαγή του File Pointer. Αρά δεν έχουμε την εκτίμηση της απόδοσης στα δύο αυτά αρχεία.

RESULTS

Στην παρακάτω εικόνα φαίνονται οι τιμές των προσβάσεων στον δίσκο. Η τιμή για την δημιουργία του A και B αρχείου είναι σταθερές καθώς μιλάμε για σειριακά αρχεία.

Παρατηρούμε εύκολα, ωστόσο, ότι η οργάνωση του αρχείου B απαιτεί 6.5 φορές παραπάνω προσβάσεις σε σχέση με το A.

Στην αναζήτηση, όμως, της πληροφορίας το αρχείο B υπερτερεί όσο αναφορά την απόδοση σε αρκετά μεγάλο βαθμό.

Στην εξωτερική κατανομή φαίνονται οι τιμές και βλέπουμε ότι η απόδοση για το B είναι καλύτερη αφού είναι αντιστρόφως ανάλογη της τιμής των προσβάσεων στον δίσκο.

```
Counter of Organization ( File A ) :5000  
  
Counter of Organization ( File B ) :32500  
  
Counter Of The Average of 20 Searches (File A) :1201  
  
Counter Of The Average of 20 Searches (File b) :291  
  
Counter of External Sorting ( File A ) :5000  
  
Counter of External Sorting ( File B ) :3125
```

ΕΥΧΑΡΙΣΤΩ ΠΟΛΥ ΓΙΑ ΤΟΝ ΧΡΟΝΟ ΣΑΣ.