# Δομές Δεδομένων και Αρχείων

# Εργαστηριακή Άσκηση 1,Επεξεργασία αρχείων


# Μιχαήλ Μαρκετάκης(2017030165)


## Μέρος Α:

Σκοπός του πρώτου μέρους της άσκησης ήταν η επεξεργασία δεδομένων στην κεντρική μνήμη.

## Α1)

Στο Α1 μέρος πιο συγκεκριμένα το ζητούμενο ήταν να δημιουργηθεί μια συνδεδεμένη λίστα,όπου θα εχει τις μεθόδους της εισαγωγής και την αναζήτησης ενός σημείου.Λόγω της απλότητας του ερωτήματος,χρησιμοποιήθηκε η έτοιμη δομή LinkedList από την βιβλιοθήκη util της Java.Η συνδεδεμένη μου αυτή λίστα κάνει δυναμικη δέσμευση μνήμης,και αρκετά εύκολα υλοποιήθηκαν οι συναρτήσεις εισαγωγής και αναζήτησης του στοιχείου.Όλα γίνανε βάση της εκφώνησης ,αφού η λίστα υποστηρίζεται από δείκτη στην αρχή όσο και στο τέλος της,για ευκολότερη εισαγωγή

Average for 1000 Points: 49.5

Average for 10.000 Points: 49.5

Average for 30.000 Points:49.5

Average for 50.000 Points:49.5

Average for 70.000 Points:49.5

Average for 100.000 Points:49.5

Ο Μέσος αριθμός συγκρίσεων είναι σταθερος.

### Α2)

Σκοπός του ερωτήματος αυτού ήταν η δημιουργία ενός πίνακα από συνδεδεμένες λίστες του ερωτήματος Α1,στην Κεντρική Μνήμη.Και σε αυτό το ερώτημα η εισαγωγή και η αναζήτηση ενός σημείου ήταν σε μεγάλο βαθμό απλές συναρτήσεις .

Για να έχω έναν δείκτη σε κάθε θεση του πίνακα(σε κάθε λιστα δηλαδή),χρησιμοποίησα ακόμη έναν πίνακα από ακεραίους ,ArrayOfPointers,με μεγεθος ίδιο με το μέγεθος του Hash Table.

Κάθε φορα που γίνεται εισαγωγή στοιχείου πχ στην θέση Κ,η αντίστοιχη τιμη του ArrayOfPointers[K] αυξανόταν κατά 1.

Average for 1000 Points: 466

Average for 10.000 Points: 5090

Average for 30.000 Points: 16185

Average for 50.000 Points: 25662

Average for 70.000 Points: 34889

Average for 100.000 Points: 50151

## Μερος Β:

Σκοπός του Μέρους Β της άσκησης ήταν η επεξεργασία των δεδομένων στον δίσκο αυτή την φορά και όχι στην ram.

**Β1)**

Στο πρώτο ερώτημα της επεξεργασίας στον δίσκο δημιουργήθηκε η συνάστηρη για την εισαγωγή σημείου στο δυαδικό αρχείο και για την αναζήτηση.

Στην εισαγωγή ενός στοιχείου η υλοποίηση μου αποτελείται από έναν WriteBuffer 256bytes,μέγεθος ίδιο με αυτό της Σελίδας δίσκου.Κάθε φορά που ο συγκεκριμένος buffer γέμιζε(32 στοιχεία) γράφεται στο αρχείο.

Η αναζήτηση γίνεται με όμοιο τρόπο,αυτή την φορά διαβάζοντας ένα ένα τα Pages του Αρχείου(ReadBuffer[256]) μέχρι να βρεθεί το σημείο προς αναζήτηση.Η συνάρτηση επιστρέφει το πόσες προσβάσεις έγιναν στο αρχείο(κλήση της read) μέχρι να βρεθεί το σημείο προς αναζήτηση.

Average for 1000 Points: 1

Average for 10.000 Points: 1

Average for 30.000 Points:1

Average for 50.000 Points:1

Average for 70.000 Points:1

Average for 100.000 Points:1

Ο Μέσος αριθμός συγκρίσεων είναι σταθερος.

**B2)**

Το ερώτημα αυτό θα μπορούσαμε να το χαρακτηρίσουμε ως το πιο απαιτητικό της συγκεκριμένης άσκησης καθώς και το πιο χρονοβόρο,αφου απαιτούσε πλήρη κατανόηση αλλα και αρκετή ακρίβεια.

Η υλοποίηση μου είναι αρκετά απλή:

Αρχικα εχουμε την κλαση **DiskHash** η οποία αντιπροσωπεύει το **HashTable**.Αποτελείται από δύο τιμές **Head,Tail**.

Για λόγους ευκολίας ,δημιουργήθηκαν ακόμα δύο κλάσεις,**DataPage**,όπου αντιπροσοπεύει την Σελίδα δίσκου,και **PageList**,όπου αποτελείται από μια *LinkedList από DataPages.*

**InsertPoint(Point A):**

Η υλοποίηση του ερωτήματος Β2 έγινε στην κλάση **DiskHashList**.Παραλληλα με το ArrayHash[M] υπάρχει ακόμα ενας πίνακας ArrayOfListsOfPages[M](πινακας από PageList).Αν πχ η Hash Function μας δώσει αποτέλεσμα **H**,θα δημιουργήσουμε στην αντίστοιχη θέση του πίνακα ArrayOfListsOfPages[H] την πρώτη σελίδα(DataPage[0]),ώστε να γραφτεί το συγκεκρίμενο σημείο.Αν η συγκεκριμένη σελίδα γεμίσει(31 σημεία) τότε θα δημιουργηθεί η 2η σελίδα για την συγκεκριμένη θέση(DataPage[1]).Αν προστεθούν ακόμα περισσότερα στοιχεία στην συγκεκριμένη θέση και γεμίσει το Page τότε θα δημιουργούνται συνεχώς νέες σελίδες στο τέλος του αρχείου (σελίδες υπερχείλισης).

Σε κάθε σελίδα τα **4 πρώτα Bytes** δείχνουν τον αριθμό των σημείων της συγκεκριμένης σελίδας,αφου πολλές από τις σελίδες μπορεί και να μην είναι γεμάτες.

‘Επειτα,γράφονται όλα τα σημεία της συγκεκριμένης σελίδας(8 byte κάθε σημείο,ένα -ένα σημείο ανά κλήση της συνάρτησης).Στην περίπτωση που η σελίδα γεμίσει(31 σημεία)τοτε στα **4 τελευταία bytes** της γράφουμε την διεύθυνση της σελίδας υπερχείλισης(τέλος του αρχείου).

*Προφανώς κάτι το οποίο ήθελε ιδιαίτερη προσοχή,ήταν ότι ενδιάμεσα θα μπορούσε φυσικά να μπουν στοιχεία σε άλλη θέση του HashTable,έτσι λοιπόν κατά την δημιουργία της επόμενης σελίδας για την θέση Η,πρέπει να ενημερώσουμε κατάλληλα την τιμή NextPage(4 τελευταία byte σελίδας) της προηγούμενης σελίδας.*

Ο πίνακας HashTable,δείχνει κάθε φορα με την μεταβλητή Head,την αρχη,δηλαδή την διεύθυνση της πρώτης σελίδας για την συγκεκριμένη θέση Η,καθώς επίσης η μεταβλητή Tail δείχνει το τέλος της αλυσίδας από Pages για την συγκεκριμένη θέση.

**Η συνάρτηση μου InsertPoint λειτουργεί με μεγάλο βαθμό ακρίβεια αφού έγιναν πολλές δοκιμές.**

**Βοηθητικές Μεταβλητές**:

**NumOfPages**:Counter για να ξέρουμε πόσες σελίδες έχουμε στο αρχείο γενικότερα.

**ArrayOfListOfPages[H].NumOfPages**: Counter για να ξέρουμε πόσες σελίδες έχουμε στην συγκεκριμένη θέση Η.

Όσο αναφορά την αναζήτηση ενός στοιχείου,αυτή την φορά τα πράγματα είναι αρκετά απλά αφου όλη η προεργασία έχει γίνει στην InsertPoint,με τις κατάλληλες ενημερώσεις σχετικά με τις διευθύνσεις των σελίδων υπερχείλισης,η συναρτηση SearchPoint ήταν σε μεγάλο βαθμό απλή.

Η **SearchPoint** αποτελείται από ένα loop,το οποίο τρέχει από το 0 έως την τιμή ArrayOfListOfPages[H].NumOfPages,δηλαδή για όλες τις σελίδες της συγκεκριμένης θέσης Η.

Διαβάζει μία μία τις σελίδες για την συγκεκριμενη θέση Η,σε έναν ReadBuffer μήκους 256bytes.

Επιστρέφει το πλήθος των προσβάσεων στο αρχείο μέχρι να βρεθεί το στοιχείο προς αναζήτηση.

Average for 1000 Points: 50

Average for 10.000 Points: 50

Average for 30.000 Points:50

Average for 50.000 Points:50

Average for 70.000 Points:50

Average for 100.000 Points:101