Χρήστος Ζήσκας

Δομές Δεδομένων & Αρχείων

Επεξεργασία Αρχείων

Σκοπός της άσκησης είναι η εξοικείωση με την επεξεργασία αρχείων και της απόδοσης τους, αποθηκεύοντας τις λέξεις που εμπεριέχονται σε αρχεία, αλλά και τις εγγραφές τους, αρχικά στην κεντρική τιμή και έπειτα στο δίσκο. Ως εγγραφές θεωρούνται το όνομα του φακέλου στο οποίο βρίσκεται η λέξη αλλά και την απόσταση σε bytes της λέξης από την αρχή της σελίδας.

Η άσκηση αποτελείται από τα εξής μέρη:

- Κατασκευή της Δομής Δεδομένων στο δίσκο
- Κατασκευή της Δομής αρχείου
- Αναζήτηση στη δομή δεδομένων

Κατασκευή της Δομή Δεδομένων στο δίσκο:

Για την κατασκευή της δομής στο δίσκο κατασκευάζουμε την class createDomes:

readFiles: Αρχικά , ζητείται από τον χρήστη να δηλώσει τον αριθμό των αρχείων που επιθυμεί να δέχεται το πρόγραμμα και στην συνέχεια εισάγει τα ονόματα των αρχείων δημιουργώντας έτσι μια δομή πίνακα που αποτελείται από δεδομένα τύπου File(Η δομή είναι array με μέγεθος τον αριθμό που δίνει ο χρήστης).

createLexiko:

Έπειτα χρησιμοποιείται μια μεταβλητή τύπου BufferedReader για ανάγνωση του κειμένου κάθε αρχείου και μεταβλητές με όνομα line,line_rest για να καταχωρούνται μια-μια προτάσεις του κάθε κειμένου και χρησιμοποιώντας πίνακα token τοποθετούνται οι λέξεις για κάθε πρόταση απομακρύνοντας τα σημεία στίξης(εκτελείται η μέθοδος split της class String).Μετατρέπουμε τα κεφαλαία γράμματα κάθε λέξης σε μικρά(με την μέθοδο toLowerCase) ώστε να θεωρούνται ίδιες οι λέξεις ανεξάρτητα κεφαλαίων ή μικρών. Στη συνέχεια ελέγχουμε για κάθε λέξη του tokens να μην υπάρχει στη δομή του λεξικού ώστε να μην υπάρχουν διπλές καταχωρήσεις λέξεων. Για αυτό χρησιμοποιείται η Boolean flag.(flag=false σημαίνει ότι η λέξη δεν έχει βρεθεί στο λεξικό.) Η δομή αρχικά είναι άδεια άρα η πρώτη καταχώρηση γίνεται χωρίς έλεγχο για επανάληψη λέξης).Αφού γίνει η εκχώρηση της λέξης το λεξικό ταξινομείται χρησιμοποιώντας τη βιβλιοθήκη java.Collections , ενημερώνεται ο δείκτης για την επόμενη πρόταση στο κείμενο , αυξάνεται το counter που δείχνει το startbyte κάθε λέξης ενώ παράλληλα δημιουργούμαι και μια δομή τύπου wordItem που περιέχει το κάθε word, το filename και το startByte(χρησιμοποιείται για να δημιουργήσουμε το ευρετήριο).Η ίδια διαδικασία επαναλαμβάνεται για όλο το κείμενο, όλων των αρχείων.

createEurethrio:

Για τη δημιουργία του ευρετηρίου, εφαρμόζουμε δυο loop. Το πρώτο αφορά κάθε λέξη του λεξικού. Αναζητούμε για κάθε λέξη του λεξικού ,την λέξη στον πίνακα wordArray ώστε να εμφανίσουμε για κάθε λέξη τις καταχωρήσεις της για κάθε αρχείο(fileName,startByte).

(Δεν έχει ιδιαίτερη σημασία το ευρετήριο σαν δομή καθώς τα στοιχεία του wordArray είναι περισσότερο πολύτιμα)

Κατασκευή της Δομής Αρχείου:

Για την υλοποίηση της δομής αρχείου κατασκευάζουμε την class BufferTools

writeEurethrio:

Αφού δημιουργούμε το αρχείο, κατασκευάζουμε ένα stream από bytes που αφορά δεδομένα προς έξοδο και το εμφωλεύουμε γύρω από ένα γενικού τύπου stream τύπου Data. Έπειτα για κάθε λέξη που βρίσκεται στο λεξικό αναζητούμε τις καταχωρήσεις του στον πίνακα .Θεωρώντας ότι τα στοιχεία κάθε καταχώρησης αποθηκεύονται σε πίνακα byte[21] και επομένως ο buffer χωράει μέχρι 4 εγγραφές. Άρα αποθηκεύουμε σε μορφή byte μια-μια τις εγγραφές(filename,startbyte) κάθε λέξης και έπειτα τις ξανά αντιγράφουμε σε πίνακα byte[] dst και ακολούθως τις εγγράφουμε στο data stream . Αν για μια λέξη υπάρχουν περισσότερες από 4 καταχωρήσεις ,καθίσταται αναγκαίο αντιγραφή του stream στον buffer(128 byte) ,μετά σε σελίδα(128 byte), σύνδεση της σελίδας με την επόμενη(Γράφουμε τον αριθμό της επόμενης σελίδας(arithmosSelidas+1)),άδειασμα του buffer ,εγγραφή των στοιχείων της λέξης στο dst και έπειτα στο stream ώστε να αποφεύγεται η πιθανότητα για κάποια λέξη να μην εγγραφούν όλα τα στοιχεία της. Αν υπάρχουν λιγότερες από 4 καταχωρήσεις για μια λέξη φορτώνουμε τον buffer με ό,τι περιέχει το data stream και δημιουργείται σελίδα ενώ στο τέλος γράφουμε έναν αρνητικό αριθμό στο stream για να δείξουμε ότι τελείωσαν οι καταχωρήσεις για την ίδια λέξη(απαιτούνται 4 ακόμα bytes λόγω της σύνδεσης.(21+4)*4+4)=104).Τέλος κρατάμε σε πίνακα για κάθε λέξη την σελίδα από την οποία ξεκινούν οι καταχωρήσεις για κάθε λέξη. Επαναλαμβάνοντας την ίδια διαδικασία για όλες τις λέξεις του λεξικού και ελέγχοντας τες με αυτές που βρίσκονται στο πίνακα wordArray(εκεί υπάρχουν και λέξεις παραπάνω από μια φορά) ολοκληρώνομαι το αρχείο με όλα τα στοιχεία.

writeLexiko:

Δημιουργούμε νέο αρχείο για να αποθηκεύσουμε την δομή του λεξικού στο αρχείο. Κατασκευάζουμε ξανά ένα stream από bytes που αφορά δεδομένα προς έξοδο και το εμφωλεύουμε γύρω από ένα γενικού τύπου stream τύπου Data.Η δημιουργία του λεξικού είναι πιο απλή καθώς το μόνο που απαιτείται για να δημιουργείται μια νέα σελίδα είναι να γεμίζεται ο buffer με 5 λέξεις και μαζί με κάθε λέξη να αποθηκεύεται η σελίδα στην οποία ξεκινούν οι εγγραφές της συγκεκριμένης στο ευρετήριο. Θεωρώντας ότι τα στοιχεία κάθε καταχώρησης αποθηκεύονται σε πίνακα byte[20] και επομένως ο buffer χωράει μέχρι 5 εγγραφές((20+4)*5)=120) . Άρα αποθηκεύουμε σε μορφή byte μια-μια τις λέξεις και έπειτα τις αντιγράφουμε σε πίνακα byte dst και ακολούθως τις εγγράφουμε στο data stream . Μόλις γράψουμε 5 λέξεις στο stream και τους 5 αριθμούς(την σελίδα στο ευρετήριο για τις εγγραφές τους) που απαιτούνται για κάθε λέξη, κατασκευάζουμε ένα buffer(128 byte)

αντιγράφουμε το stream στην σελίδα, αυξάνουμε τον αριθμό σελίδων, κατασκευάζουμε ξανά ένα stream από bytes και αποθηκεύομαι τις επόμενες λέξεις .Επαναληπτικά δημιουργούνται νέες σελίδες και ολοκληρώνεται το αρχείο.

Αναζήτηση στη δομή δεδομένων:

Για την κατασκευή της αναζήτησης στη δομή δεδομένων κατασκευάζουμε την κλάση Binary Search.

BinarySearchInDic:

Βασισμένη στη μέθοδο της δυαδικής αναζήτησης(binary search) αναζητούμε την λέξη που επιθυμεί ο χρήστης στο λεξικό και εμφανίζουμε τις εγγραφές του στα διάφορα αρχεία.

Πρώτα βρίσκουμε την πρώτη λέξη της μεσαίας σελίδας και την συγκρίνουμε με την ζητούμενη αφού έχουμε διαβάσει την σελίδα (θυμόμαστε ότι το αρχείο είναι δυαδικό άρα για να διαβάσουμε χρειαζόμαστε πίνακα τύπου byte[] ώστε να αποθηκεύουμε όλη τη σελίδα και να παίρνουμε κάθε λέξη της-έχουμε δημιουργήσει τα απαιτούμενα streams).

Χρησιμοποιούμε επίσης και μια μεταβλητή filePointer τύπου int ώστε να κατευθυνόμαστε στις επόμενες λέξεις της σελίδας αυξάνοντας την κατά 24(20 για το string kai 4 int). Κάθε διάβασμα νέας σελίδας θεωρείται εισροή στο δίσκο και κρατάμε αυτόν τον αριθμό. Αν η ζητούμενη λέξη δεν βρίσκεται στην σελίδα τότε χωρίζουμε τον πίνακα και διαλέγουμε για αναζήτηση εκείνο το τμήμα του πίνακα που μας υποδεικνύει η μεταβλητή compare.(compare<0 σημαίνει ότι παίρνουμε το αριστερό τμήμα του πίνακα ενώ >0 για το δεξιό τμήμα).Η διαδικασία αυτή επαναλαμβάνετε μέχρι να βρεθεί η λέξη η να οδηγηθούμε στο τέλος του αρχείου. Θυμόμαστε ότι κάθε σελίδα διαθέτει 5 λέξεις και 5 αριθμούς. Διαθέτουμε επίσης μια Boolean μεταβλητή flag. Αν το flag παραμείνει false μέχρι το τέλος της αναζήτησης σημαίνει ότι δεν βρέθηκε η λέξη. Αλλιώς διαβάζουμε τον αριθμό που βρίσκεται μπροστά από την λέξη και αποτελεί την σελίδα που περιέχονται οι εγγραφές για αυτήν την λέξη στο ευρετήριο. Έτσι αναζητούμε το ζητούμενο σημείο(*128 λόγω ότι ψάχνομε την σελίδα στην οποία βρίσκεται) στο αρχείο του ευρετηρίου αυξάνοντας τον diskAccess Counter όταν αλλάζουμε σελίδα(αλλαγή σελίδα μετά από 4 λέξεις). Διαβάζουμε τις εγγραφές για την λέξη και ελέγχουμε την συνθήκη να μην βρούμε κάποιον αριθμό startByte <0(αυτό σημαίνει ότι έχουμε δώσει αρνητική τιμή στη σύνδεση με νέα σελίδα οπότε τελείωσαν οι εγγραφές για αυτή την λέξη).Ως επιστρεφόμενη τιμή προορίζεται η τιμή του discAccess.

Οι μέθοδοι για την υλοποίηση του προγράμματος εν μέρει συνάδουν με την περιγραφή της εκφώνησης με την διαφορά ότι θεωρείται ότι το string που δέχεται ο πίνακας τύπου byte, για αποθήκευση των εγγραφών και της λέξης διαφέρει(η εκφώνηση θεωρεί λέξη με μέγιστο 12 byte+ 4 τον int και στο ευρετήριο 8 το όνομα του αρχείου και 4 το startByte. Το πρόγραμμα θεωρεί 21 το filename και 20 το word).Όπως και να έχει έχουν υλοποιηθεί ξεχωριστές μέθοδοι και ρου τίνες για κάθε τμήμα της άσκησης. , Το πρόγραμμα είναι κατασκευασμένο ώστε να λειτουργεί για οποιοδήποτε αριθμό λέξεων και καταγραφών . Επίσης υλοποιήθηκαν try-catch blogs και ειδικές συνθήκες για να αποφεύγονται λάθη όπως ,το όριο του μεγέθους του πίνακα με τα αρχεία, αλλά και να δίνεται κατάλληλη μορφή

απαντήσεων του χρήστη για την ομαλή λειτουργία του προγράμματος. Το πρόγραμμα εμφανίζει τις προσβάσεις που πραγματοποιήθηκαν στο δίσκο. Αν η λέξη βρέθηκε ,τότε εμφανίζει και τις καταχωρήσεις που βρέθηκαν στο ευρετήριο. Κάθε κλάση περιέχει σχόλια για την επεξήγηση του κώδικα. Πρώτα δημιουργούνται οι δομές στην κεντρική μνήμη , έπειτα αποθηκεύονται στο δίσκο και τέλος υλοποιείται η αναζήτηση. Κατά την εγγραφή των αρχείων παρατηρείται μια ελαφρά καθυστέρηση που είναι κατανοητή.

Κώδικας που χρησιμοποιήθηκε από ιστοσελίδες του διαδίκτυο θεωρείται η δυαδική αναζήτηση στην πρωτότυπη μορφή της. Το τμήμα κώδικα που αφορά το διάβασμα των λέξεων από τα αρχεία κειμένου να σημειωθεί ότι βρέθηκε από σημειώσεις του μαθήματος.

Επίσης, η ενημέρωση για την χρήση της βιβλιοθήκης BufferedReader μπορεί να θεωρηθεί ως πηγή(το scanner εμφάνιζε λάθη)

Παρακάτω δίνονται τα link:

http://interactivepython.org/runestone/static/pythonds/SortSearch/TheBinarySearch.html

https://docs.oracle.com/javase/7/docs/api/java/io/BufferedReader.html

---

Στιγμιότυπα του project!

```
Epiloges
1)Anazitisi lexis
2)Exodos programmatos
Dialexte apantisi:1
Grapste ti lexi pou thelete na anazitisete:liberty

IndexPointer is: 1055

The give word:liberty found below:
1)File Name:MartinLutherKing.txt Start Byte:2029
2)File Name:MartinLutherKing.txt Start Byte:12494
3)File Name:Obama.txt Start Byte:17636
4)File Name:Obama.txt Start Byte:20187
5)File Name:Kennedy.txt Start Byte:2214
Pragmatopoithikan 9 prosvaseis sto disko
```

```
Epiloges
1)Anazitisi lexis
2)Exodos programmatos
Dialexte apantisi:1
Grapste ti lexi pou thelete na anazitisete:week

IndexPointer is: 2189

The give word:week found below:
1)File Name:Obama.txt Start Byte:5930
Pragmatopoithikan 9 prosvaseis sto disko
```

```
Arithmos arxeion: 3
Enter your file name: MartinLutherKing.txt
Enter your file name: Obama.txt
Enter your file name: Kennedy.txt

Epithxhs dhmiourgia dictionary
Epityxhs dhmiourgia index

Epiloges
1)Anazitisi lexis
2)Exodos programmatos
Dialexte apantisi:1
Grapste ti lexi pou thelete na anazitisete:wake
H lexh pou anazhtate dn vrethike!
Pragmatopoithikan 8 prosvaseis sto disko
```

```
1)Anazitisi lexis
2)Exodos programmatos
Dialexte apantisi:1
Grapste ti lexi pou thelete na anazitisete:America

IndexPointer is: 96

The give word:America found below:
1)File Name:MartinLutherKing.txt Start Byte:2358
2)File Name:MartinLutherKing.txt Start Byte:3198
3)File Name:MartinLutherKing.txt Start Byte:3556
4)File Name:MartinLutherKing.txt Start Byte:4861
5)File Name:MartinLutherKing.txt Start Byte:12609
6)File Name:Obama.txt Start Byte:967
7)File Name:Obama.txt Start Byte:2396
8)File Name:Obama.txt Start Byte:2752
9)File Name:Obama.txt Start Byte:5118
10)File Name:Obama.txt Start Byte:6175
11)File Name:Obama.txt Start Byte:12506
12)File Name:Obama.txt Start Byte:15284
13)File Name:Obama.txt Start Byte:20628
14)File Name:Obama.txt Start Byte:21305
15)File Name:Obama.txt Start Byte:22222
16)File Name:Kennedy.txt Start Byte:13531
17)File Name:Kennedy.txt Start Byte:13645
Pragmatopoithikan 13 prosvaseis sto disko
```