Γιάννης Τζώρτζης

Δομές Δεδομένων και Αρχείων

Μέρος 1ο: Επεξεργασία Αρχείων – Αναζήτηση σε Ταξινομημένο Αρχείο

Σκοπός της άσκησης αυτής είναι η εξοικείωση με την απόδοση μεθόδων αναζήτησης στον δίσκο. Γι αυτό ζητείται να υλοποιηθούν 5 διαφορετικοί μέθοδοι αναζήτησης που εφαρμόζονται σε ένα δυαδικό αρχείο και βασίζονται στην λογική της σελιδοποίησης του αρχείου. Αυτές οι μέθοδοι είναι:

- ***Σειριακή Αναζήτηση στο αρχείο για τυχαίο κλειδί:***
  Διαβάζουμε από το αρχείο 256 ακεραίους, τους φορτώνουμε σε έναν πίνακα στην κύρια μνήμη και εφαρμόζουμε σε αυτόν διαδική αναζήτηση. Η ίδια διαδικασία εφαρμόζεται επαναληπτικά μέχρι να ψάξουμε όλα τα στοιχεία του αρχείου. Επειδή ο αριθμός των στοιχείων δια το μέγεθος της σελίδας δεν είναι ακέραιος σημαίνει ότι η τελευταία σελίδα έχει λιγότερα στοιχεία. Οπότε στο πρόγραμμα χρησιμοποιήθηκε ένας πρόσθετος buffer, μικρότερου μεγέθους για την αναζήτηση στην τελευταία σελίδα.
  Όταν το κλειδί που ψάχνουμε είναι μεγάλο τότε η σειριακή αναζήτηση είναι αργή. Αν για παράδειγμα ψάχνουμε ένα κλειδί που ανοίκει στη 500 σελίδα τότε θα έχουμε συνολικά 500 προσβάσεις στον δίσκο. Γρήγορη είναι μόνο όταν μιλάμε για πολύ μικρά κλειδία.

- ***Δυαδική Αναζήτηση στο αρχείο για τυχαίο κλειδί:***
  Για την υλοποίηση αυτής της μεθόδου ακολουθούμε τα βήματα της δυαδικής αναζήτησης που εφαρμόζεται σε έναν απλό πίνακα με ακεραίους, μόνο που τώρα αντί για ακεραίους έχουμε σελίδες δίσκου. Κοιτάμε την μεσαία σελίδα, αν το κλειδί είναι μέσα σε αυτήν τελειώσαμε, αν το κλειδί είναι μικρότερο από το πρώτο στοιχείο της σελίδας αυτής τότε φέρνουμε στη μνήμη τη μεσαία σελίδα του αριστερού μισού του αρχείου, αλλίως αν το κλειδί είναι μεγαλύτερο από το μεγαλύτερο στοιχείο της μεσαίας σελίδας τότε φέρνουμε στη μνήμη τη μεσαία σελίδα του δεξιού μισού του αρχείου.
  Η δυαδική αναζήτηση είναι πολύ πιο γρήγορη από την σειριακή όταν μιλάμε για μεγάλα κλειδιά. Στην δική μας περίπτωση έχουμε έναν μέσο αριθμό προσβάσεων κοντά στο 14. Η δυαδική αναζήτηση μειονεκτεί σε σχέση με τη σειριακή όταν για παράδειγμα το κλειδί που ψάχνουμε είναι στην πρώτη σελίδα. Τότε η σειριακή θα έχει μία πρόσβαση στο δίσκο ενώ η δυαδική περίπου 14 προσβάσεις.

- ***Δυαδική αναζήτηση παρεμβολής στο αρχείο για τυχαίο κλειδί:***
  Η συγκεκριμένη αναζήτηση αποτελεί μια βελτίωση της δυαδικής, γιατί χρησιμοποιεί έναν τύπο με τον οποίο προσπαθεί να μαντέψει σε ποια σελίδα βρίσκεται το κλειδί, σε αντίθεση με την τελευταία που πάντα «σπάει» τις σελίδες στη μέση κτλ. Ο τύπος που χρησιμοποιείται σε αυτή την μέθοδο αναζήτησης περιέχει και το κλειδί που ψάχνουμε.
  Έτσι επειδή στην δική μας περίπτωση η κάθε σελίδα δίσκου περιέχει αρκετά κλειδιά, αυτή η μέθοδος αναζήτησης βρίσκει το κλειδί συνήθως σε μία μόνο πρόσβαση στον δίσκο.

- ***Αναζήτηση με χρήση προσωρινής μνήμης cache:***
  Σε αυτήν την περίπτωση υλοποιούμε μία μνήμη cache, η οποία ουσιαστικά είναι μία ουρά που περιέχει buffers με σελίδες του δίσκου που έχουν διαβαστεί πρόσφατα από το δίσκο. Η λογική της χρήσης της cache είναι να περιορίσουμε τις προσβάσεις στον δίσκο με το να μην διαβάζουμε ξανά σελίδες δίσκου που έχουμε διαβάσει παλιότερα ψάχνοντας κάποιο άλλο κλειδί. Γενικά αυτή η μέθοδος περιμένουμε να έχει θετικά αποτελέσματα όταν το κλειδί, που αναζητούμε, παίρνει τιμές από ένα μικρό εύρος αριθμών έτσι ώστε να κάνουμε χρήση παλιότερων σελίδων.
  Στη δική μας περίπτωση δεν έχει μεγάλη επιτυχία γιατί το εύρος είναι αρκετά μεγάλο, επομένως δεν έχουμε επανάληψη σελίδων.

- ***Αναζήτηση με χρήση αρχείου δεικτών:***
  Σε αυτή την μέθοδο αναζήτησης αποθηκεύουμε σε έναν πίνακα συγκεκριμένους αριθμούς του αρχείου, που βρίσκονται σε θέσεις οι οποίες καθορίζονται από έναν δοσμένο τύπο. Έτσι χωρίζουμε το αρχείο σε πεδία. Με αυτόν τον τρόπο ψάχνουμε αρχικά το πεδίο στο οποίο ανήκει το κλειδί, πράγμα που δεν μας κοστίζει προσβάσεις στον δίσκο. Αφού βρούμε το πεδίο χρησιμοποιούμε δυδική αναζήτηση στις σελίδες που περιλαμβάνει το πεδίο αυτό προκειμένου να βρούμε το κλειδί. Και εδώ χρησιμοποιούμε την cache του προηγούμενου ερωτήματος.
  Γενικά παρατηρούμε μεγάλη βελτίωση στην δυαδική αναζήτηση του 2ου ερωτήματος αφού τώρα έχουμε σχεδόν τις μισές προσβάσεις(6-7).

Όλες οι παραπάνω μέθοδοι αναζήτησης υλοποιήθηκαν όπως περιγράφονται στην εκφώνηση. Σε κάθε περίπτωση υλοποιήθηκαν ξεχωριστές συναρτήσεις ή ρουτίνες που ψάχνουν την τελευταία σελίδα δίσκου, η οποία έχει λιγότερα στοιχεία. Το πρόγραμμα υλοποιήθηκε ώστε να δουλεύει για οποιοδήποτε N και για οποιοδήποτε ταξινομημένο αρχείο. Επίσης υλοποιήθηκαν συναρτήσεις που λειτουργούν σαν handlers για ειδικές συνθήκες, όπως ο έλεγχος τον ορίων του αρχείου κτλ. Το πρόγραμμα συνολικά εμφανίζει τον μέσω αριθμό προσβάσεων στον δίσκο κάθε αναζήτησης. Κάθε μέθοδος έχει υλοποιηθεί σε διαφορετική κλάση και κάθε κλάση περιέχει σχόλια για την επεξήγηση του κώδικα, ο οποίος γενικά απαρτίζεται από όσο το δυνατόν μικρές συναρτήσεις για ευκολότερη κατανόηση και χειρισμό λαθών.Όλη η άσκηση υλοποιήθηκε σε γλώσσα προγραμματισμού Java, στο περιβάλλον του Eclipse. Πριν την εφαρμογή των μεθόδων αναζήτησης το πρόγραμμα γεμίζει το δυαδικό αρχείο που χρησιμοποιούμε, το οποίο είναι λίγο χρονοβόρο.

Η δυαδική αναζήτηση σε πίνακα, που χρησιμοποιούμε σε όλη την άσκηση για να ψάξουμε μία σελίδα, βρέθηκε έτοιμη στο διαδίκτυο στο link που φαίνεται παρακάτω. Επίσης ο τύπος, που χρησιμοποιείται στην δυαδική αναζήτηση παρεμβολής, χρειάστηκε κάποιου είδους casting ώστε να λειτουργήσει σωστά, πράγμα που βρέθηκε στο διαδίκτυο, σε link που φαίνεται παρακάτω.

http://algs4.cs.princeton.edu/11model/BinarySearch.java.html

http://data.linkedin.com/blog/2010/06/beating-binary-search

Ακολουθούν στιγμιότυπα από το «τρέξιμο» του προγράμματος. Φαίνεται και η μεγάλη βελτίωση της cache όταν περιορίσουμε το εύρος των κλειδιών μέχρι το 1000:

```
##############################  ******************************  ******************************
# Serial Search       : 19817 #* ----------------- Cache Info  * ----------------- Cache Info
# Binary Search       : 14    #*                               *
# Interpolation Search: 1     #* Cache hits: 0                 * Cache hits: 36
# Search using cache  : 3     #* Cache misses: 40              * Cache misses: 4
# Search using indexes: 6     #* Disk Accesses: 5             * Disk Accesses: 0
##############################* Cache Size: 40                * Cache Size: 4
                                ******************************  ******************************
```