ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Άσκηση: Σύγκριση Αλγορίθμων Αναζήτησης

Παύλος Λουκαρέας

Α.Μ. 1046970

Εισαγωγή

Στην εργασία αυτή γίνεται η εύρεση η εύρεση των κοινών αριθμών μιας πρώτης ομάδας σε αυτούς της δεύτερης, με τέσσερεις διαφορετικούς τρόπους. Οι τρόποι που χρησιμοποιήθηκαν είναι πλήρης αναζήτηση, η δυαδική αναζήτηση και η αναζήτηση με πίνακα κατακερματισμού που υλοποιήθηκε με δύο τρόπους (open addressing & chaining). Κάθε υλοποίηση είχε τα πλεονεκτήματα και τις ιδιαιτερότητές της, ενώ τα προγράμματα μοιάζουν αρκετά μεταξύ τους ως προς την δομή του κώδικα. Το προγράμματα υλοποιήθηκαν σε γλώσσα Python και οι αναγκαίες συναρτήσεις καλούνται μέσα από τη συνάρτηση `main()`.

Η συνάρτηση `main()` έχει όπως πάντα το ρόλο της κλήσης των υπολοίπων συναρτήσεων, της διαβίβασης των αποτελεσμάτων που αυτές παράγουν και της παρουσίασης του αποτελέσματος που ζητείται. Γίνεται δημιουργία των δύο συνόλων ακεραίων και έπειτα καλείται ανάλογα την υλοποίηση η αντίστοιχη συνάρτηση αναζήτησης του ενός συνόλου μέσα στο άλλο.

Η αρχικοποίηση, ο ορισμός των global μεταβλητών του προγράμματος και η μέτρηση του χρόνου εκτέλεσης γίνεται εξωτερικά της `main()`.

Από εκεί και πέρα κάθε πρόγραμμα διαφοροποιείται με την υλοποίησή του, με το τελικό αποτέλεσμα να είναι ίδιο για όλα τα προγράμματα, ενώ υπάρχει διαφορετικός χρόνος εκτέλεσης λόγω της διαφορετικής πολυπλοκότητας των υλοποιήσεων.

Πολυπλοκότητα

Για το πρώτο πρόγραμμα γίνεται πλήρης επιλογή των στοιχείων του ενός πίνακα και στη συνέχεια σειριακή αναζήτηση στον δεύτερο πίνακα. Γνωρίζοντας ότι η σειριακή αναζήτηση απαιτεί O(N) πολυπλοκότητα και ότι θα εκτελεστεί N φορές θα έχουμε N*O(N) = $O(N^2)$ πολυπλοκότητα.

Για το δεύτερο πρόγραμμα γίνεται πλήρης επιλογή των στοιχείων του ενός πίνακα και στη συνέχεια δυαδική αναζήτηση στον δεύτερο πίνακα. Γνωρίζοντας ότι η δυαδική αναζήτηση απαιτεί O(logN) πολυπλοκότητα και ότι θα εκτελεστεί N φορές θα έχουμε N*O(logN) = O(NlogN) πολυπλοκότητα. Για την εκτέλεση της δυαδικής αναζήτησης απαιτείται ταξινόμηση του δεύτερου πίνακα, η οποία γίνεται μέσω του αλγορίθμου Quicksort που έχει επίσης μέση περίπτωση O(NlogN), άρα δεν επηρεάζει το τελικό αποτέλεσμα της πολυπλοκότητας.

Για το τρίτο και τέταρτο πρόγραμμα γίνεται αρχικά η δημιουργία του πίνακα κατακερματισμού με τις στρατηγικές open addressing & chaining για κάθε στοιχείο του πρώτου πίνακα. Η διαδικασία αυτή απαιτεί N*O(1) = O(N) πολυπλοκότητα. Στη συνέχεια για την πρώτη περίπτωση μετά το mapping των στοιχείων του δεύτερου πίνακα στα στοιχεία του πρώτου εξάγεται η συχνότητα της εμφάνισης του στοιχείου αυτού (επίσης O(N) πολυπλοκότητα). Για τη δεύτερη περίπτωση γίνεται ακριβώς το ίδιο, μόνο που το ρόλο της συχνότητας τον έχει το μήκος της λίστας των αριθμών. Και για τις δυο περιπτώσεις θα έχουμε O(N) + O(N) = O(N) πολυπλοκότητα. Τα αποτελέσματα επιβεβαιώνουν τη θεωρία.

Αποτελέσματα

Για τα τέσσερα προγράμματα θα έχουμε (N = 1000):

```
complete_search.py
The common numbers between the two are 631
0.06604576110839844
binary_search.py
The common numbers between the two are 631
0.015010356903076172
open_address_search.py
The common numbers between the two are 631
0.0030012130737304688
chaining_search.py
The common numbers between the two are 631
0.0029993057250976562
```