Κ23α - Ανάπτυξη Λογισμικού Για Πληροφοριακά Συστήματα

Χειμερινό Εξάμηνο 2021– 2022

Καθηγητής Ι. Ιωαννίδης

Άσκηση 2 – Παράδοση: Κυριακή 19 Δεκεμβρίου 2021

Ανεστραμμένη Μηχανή Αναζήτησης (inverted search engine)

Μονονηματική υλοποίηση (single-threaded implementation)

Εισαγωγή

Σε αυτήν την άσκηση, θα υλοποιηθεί η πλήρης λειτουργικότητα της μηχανής αναζήτησης χωρίς παραλληλισμό (χρησιμοποιώντας ένα thread). Παρακάτω παρουσιάζεται ο κώδικας που δείχνει τον τρόπο χρήσης της μηχανής. Αρχικά καλείται η συνάρτηση `InitializeIndex()`, η οποία θα πρέπει να αρχικοποιεί της απαραίτητες δομές δεδομένων που θα χρησιμοποιηθούν κατά την λειτουργία της ανεστραμμένης μηχανής αναζήτησης. Στη συνέχεια, καταχωρούνται τα ερωτήματα, τα οποία μπορεί να είναι πολλά σε πλήθος (Q >> 1000), αλλά υποθέτουμε ότι χωράνε στην μνήμη.

```
InitializeIndex();
// Start queries
for 1..Q {
     StartQuery( ... );
}
// Match documents in batches
for 1..B {
     // Issue documents
     for 1..D {
          MatchDocument( ... );
     }
     // Get results
     for 1..D {
          GetNextAvailRes( ... );
     }
}
DestroyIndex();
```

Κατά την βασική λειτουργία, τα κείμενα έρχονται σε ομάδες μεγέθους D. Για κάθε ομάδα, πρώτα καταχωρούνται όλα τα κείμενα της ομάδας χρησιμοποιώντας την συνάρτηση `MatchDocument()`, και στην συνέχεια ζητούνται τα αποτελέσματα με την συνάρτηση `GetNextAvailRes()`. Τέλος, όταν επεξεργαστούμε όλα τα κείμενα, καλείτε η συνάρτηση `DestroyIndex()` η οποία πρέπει να κάνει αποδέσμευση των πόρων που έχουν δεσμευτεί κατά την διάρκεια της λειτουργίας. Παρακάτω περιγράφονται αναλυτικά οι συναρτήσεις της διεπαφής που χρειάζεται να υλοποιηθούν.

Περιγραφή Διεπαφής

`ErrorCode InitializeIndex()`
Καλείται μία μόνο φορά στην έναρξη της δοκιμής και υλοποιεί τις απαραίτητες αρχικοποιήσεις. Επιστρέφει κατάλληλο μήνυμα λάθους.

`ErrorCode DestroyIndex()`
Καλείται μια μόνο φορά στο τέλος της δοκιμής. Μπορεί να χρησιμοποιηθεί για την απελευθέρωση μνήμης που σχετίζεται με τις δομές/ευρετήρια που χρησιμοποιείτε. Επιστρέφει κατάλληλο μήνυμα λάθους.

```
ErrorCode StartQuery (QueryID query_id, const char * query_str,
MatchType match_type, unsigned int match_dist )
```
Προσθέτει ένα νέο (συγκεκριμένου τύπου) ερώτημα στο σύνολο ερωτημάτων. Επιστρέφει κατάλληλο μήνυμα λάθους. ΣΗΜΕΙΩΣΗ: Όλα τα ερωτήματα θα είναι γνωστά εξ αρχής (δεν θα καλείται η add μετά την πρώτη κλήση της `matchDocument()`).

```
ErrorCode EndQuery(QueryID query_id);
```
Αφαιρεί το ερώτημα από το σύνολο των ενεργών ερωτημάτων.

```
ErrorCode MatchDocument (DocID doc_id, const char * doc_str)
```
Η ρουτίνα αυτή προσθέτει ένα καινούριο κείμενο προς επεξεργασία. Επιστρέφει κατάλληλο μήνυμα λάθους.

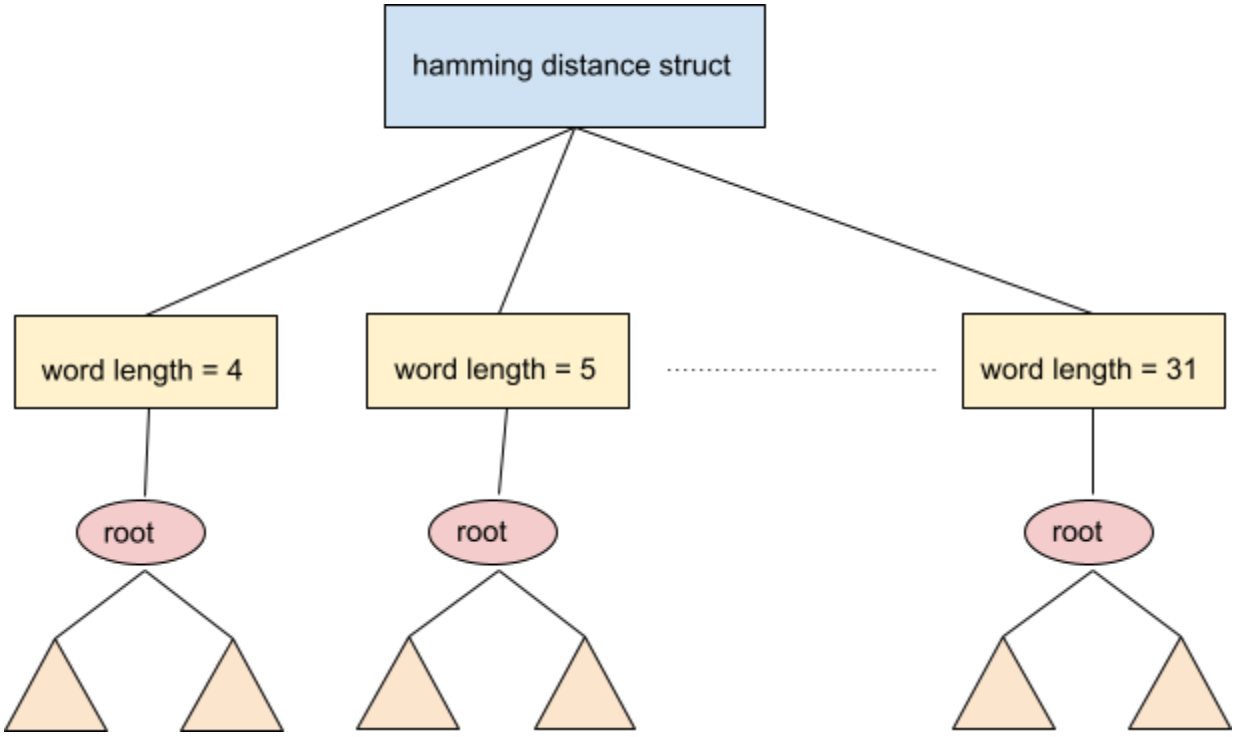
```
ErrorCode GetNextAvailRes (DocID * p_doc_id, unsigned int * p_num_res,
QueryID **  p_query_ids)
```
Η ρουτίνα αυτή επιστρέφει το επόμενο διαθέσιμο σύνολο ερωτημάτων που απαντούν σε κάποιο από τα κείμενα (*p_doc_id) που έχουν ήδη δοθεί, ταξινομημένα βάση των Query Ids. Επιστρέφει κατάλληλο μήνυμα λάθους.

Περισσότερες λεπτομέρειες μπορείτε να δείτε στο παρακάτω σύνδεσμο: https://sigmod.kaust.edu.sa/task-details.html

Ευρετήρια

Στο επίπεδο αυτό θα χρησιμοποιηθεί κατάλληλα το ευρετήριο που υλοποιήθηκε στην προηγούμενη άσκηση. Το ευρετήριο θα κρατάει τις λέξεις των ερωτημάτων. Οι μέθοδοι ταιριάσματος που πρέπει να υποστηριχθούν είναι οι exact, edit, και hamming.

- Για τη μέθοδο exact, θα χρησιμοποιηθεί ένα ευρετήριο στατικού κατακερματισμού (static hashing). Στον στατικό κατακερματισμό υπάρχει μια συνάρτηση η οποία παίρνει ως όρισμα μια λέξη και επιστρέφει μια τιμή που δηλώνει τον κουβά στο οποίο θα πρέπει να αναζητηθεί η λέξη. Επειδή ο στατικός κατακερματισμός έχει ένα σταθερό σύνολο από κουβάδες (buckets), είναι πιθανόν περισσότερες από μία λέξεις να αντιστοιχούν στον ίδιο κουβά.
- Για την edit, θα χρησιμοποιηθεί ακριβώς ένα BKTree χρησιμοποιώντας όλες τις διαφορετικές λέξεις των ερωτημάτων και υπολογίζοντας την απόσταση επεξεργασίας (edit distance). Αυτό το δέντρο θα χρησιμοποιηθεί για να βρεθούν οι λέξεις των ερωτημάτων με τις οποίες ταιριάζει μια λέξη του κειμένου.
- Για τη hamming, θα χρησιμοποιηθεί πάλι το ευρετήριο των BKTrees. Η διαφορές σε σχέση με την edit είναι ότι
    - η υπολογιζόμενη απόσταση μεταξύ λέξεων θα είναι η hamming και όχι η distance, και
    - θα πρέπει να δημιουργηθεί ένα δέντρο για κάθε μήκος λέξης στο διάστημα [4, 31]. Για παράδειγμα, έστω ότι έχουμε τα εξής ερωτήματα: 1. {'vacation', 'island'} και 2. {'work', 'deadline', 'incomplete'}. Θα χρειαστεί να δημιουργηθούν 4 BKTrees, ένα στο σύνολο {'work'}, ένα στο {'island'}, ένα στο {'vacation', 'deadline'}, και το τέταρτο στο σύνολο {'incomplete'}. Ο λόγος δημιουργίας της πολυπλοκότερης δομής είναι ότι αφενός η hamming distance δεν ορίζεται σε λέξεις διαφορετικού μήκους και αφετέρου ότι επιταχύνει τη μέθοδο της αναζήτησης.



Εύρεση Αποτελεσμάτων

Τα ευρετήρια θα κατασκευαστούν πάνω στις λέξεις των queries, και πιο συγκεκριμένα πάνω σε Entries. Κάθε Entry αποτελείται από μια λέξη και ένα payload το οποίο θα είναι μια δομή που θα κρατάει τα ids των ερωτημάτων (queries) στα οποία υπάρχει η λέξη. Στα ευρετήρια θα πραγματοποιούνται ερωτήσεις για τις λέξεις των κειμένων και θα επιστρέφεται λίστες από Entries. Στο τέλος, οι ερωτήσεις (queries) που θα είναι επιτυχείς, θα είναι αυτές για τις οποίες έχουμε ταιριάξει όλες τις λέξεις τους για τον τύπο ταιριάσματος και επιθυμητή απόσταση.

Επειδή ο τύπος ταιριάσματος και η μέγιστη απόσταση διαφέρει για την ίδια λέξη μεταξύ queries, θα πρέπει να σχεδιαστεί και ο κατάλληλος και αποδοτικός τρόπος αντιμετώπισης του συγκεκριμένου θέματος.

Χρήση κώδικα διαγωνισμού

Μπορείτε να χρησιμοποιήσετε τον κώδικα του διαγωνισμού [1]. Συγκεκριμένα πρέπει να υλοποιήσετε ακριβώς το interface όπως ορίζεται στο `include/core.h`

Παράδοση εργασίας

Προθεσμία παράδοσης: 19/12/2021
Γλώσσα υλοποίησης: C / C++ χωρίς χρήση stl.
Περιβάλλον υλοποίησης: Linux (gcc > 5.4+).

Παραδοτέα: Η παράδοση της εργασίας θα γίνει με βάση το τελευταιο commit πριν την προθεσμία υποβολής στο git repository σας. **Η χρήση git είναι υποχρεωτική**.
Επιπλέον, εκτός από τον πηγαίο κώδικα, θα παραδώσετε μια σύντομη αναφορά, με τις σχεδιαστικές σας επιλογές καθώς και να εφαρμόσετε ελέγχους ως προς την ορθότητα του λογισμικού με τη χρήση ανάλογων βιβλιοθηκών (Software testing).

Παραπομπές

[1] https://sigmod.kaust.edu.sa/files/sigmod2013contest-final-testdriver.tar.gz κώδικας για τον έλεγχο του προγράμματος του διαγωνισμού