Κ24: Προγραμματισμός Συστήματος – 1η Εργασία
Εαρινό Εξάμηνο 2017-2018

ΕΥΑΓΓΕΛΙΟΥ ΠΑΝΑΓΙΩΤΗΣ ΑΜ:1115201500039

Το πρόγραμμα λειτουργεί *σωστά*, και υλοποιεί *όλες τις ενέργειες* που περιγράφονται στην εκφώνηση. Δεν υπάρχουν memory leaks σύμφωνα με τον valgrind.

Στον παραδοτέο φάκελο υπάρχουν 4 φάκελοι καθένας από τους οποίους περιλαμβάνει και την υλοποίηση μια δομής, 1 φάκελος με όνομα build στον οποίο θα δημιουργηθεί το εκτελέσιμο πρόγραμμα, 1 φάκελος με όνομα main που περιλαμβάνει την συνάρτηση main και 1 φάκελος με όνομα ManageFuns που περιλαμβάνει τις βοηθητικές συναρτήσεις που καλεί η main.

Εντολή για την μεταγλώττιση του προγράμματος:

- Όντας στον αρχικό φάκελο (PanagiotisEvangeliouProject1/) εκτελούμε την εντολή **make** που έχει σαν αποτέλεσμα την δημιουργία του εκτελέσιμου προγράμματος με όνομα minisearch στον φάκελο build/ .

Εντολή για την εκτέλεση του προγράμματος:

- Το εκτελέσιμο πρόγραμμα βρίσκεται μέσα στην τοποθεσία PanagiotisEvangeliouProject1/build/minisearch. Οπότε για να το τρέξουμε όντας στον αρχικό φάκελο κάνουμε το εξής:
  - **./build/minisearch -i docfile -k K**

Όπου:
- docfile είναι το όνομα του αρχείου που περιλαμβάνει τα κείμενα που θα αποθηκεύσει η εφαρμογή
- K είναι η παράμετρος που λέει στην εφαρμογή πόσα το πολύ αποτελέσματα να γυρίσει για κάθε query

Οι παράμετροι που δίνονται κατά την εκτέλεση των προγραμμάτων μπαίνουν με οποιαδήποτε σειρά. Η χρήση της παραμέτρου που δίνει το όνομα του αρχείου είναι υποχρεωτική. Αν δεν δοθεί η παράμετρος K τότε η default τιμή της είναι 10.

Δομές που χρησιμοποιήθηκαν:

- DocMap
  - Χρησιμοποιείται για την αποθήκευση των κειμένων, καθώς επίσης και για την εκτύπωση τους.
  - Παρατηρήσεις:
    - Στον constructor της class Words όσο και στον constructor της class DocMap, θα πρέπει να δοθεί αποδεκτός αριθμός μεγέθους πίνακα δηλαδή τιμή > 0. Στις υπόλοιπες μεθόδους υπάρχουν οι κατάλληλοι έλεγχοι για τον έλεγχο του index που δίνεται.
    - Για την εκτύπωση του κειμένου έχει γίνει η παραδοχή ότι το πλάτος του terminal θα είναι τόσο ώστε να χωράει τουλάχιστον η ειδική πληροφορία καθώς επίσης και η μεγαλύτερη λέξη ώστε να μην σπάσει.
    - Λαμβάνοντας υπόψιν την παραπάνω παραδοχή, γίνονται οι απαραίτητες ενέργειες ώστε κατά την εναλλάξ εκτύπωση των λέξεων του κειμένου και της γραμμής υπογράμμισης, αν μια λέξη δε χωράει, να γράφεται στην επόμενη γραμμή ώστε να μην χρειαστεί να σπάσει.

- Heap
  - Χρησιμοποιείται για την δημιουργία σωρού χρησιμοποιώντας τον τελικό πίνακα που περιέχει τα scores των ids που σχετίζονται με το query. Η πολυπλοκότητα δημιουργίας του σωρού είναι O(n), όπου n το πλήθος των ids που σχετίζονται με το query. Συνολικά ο αλγόριθμος ταξινόμησης έχει πολυπλοκότητα O (nlogn).

- Παρατηρήσεις:
  - Η ιδέα του αλγορίθμου βασίστηκε στις σημειώσεις του κύριου Ζησιμόπουλου από το μάθημα Αλγόριθμοι και Πολυπλοκότητα (κεφάλαιο 3) του 2ου έτους.

- PostingList
  - Χρησιμοποιείται για την αναπαράσταση των posting lists των λέξεων που περιλαμβάνονται στο Trie.
  - Παρατηρήσεις:
    - Η συγκεκριμένη υλοποίηση της λίστας έχει δείκτη και στον τελικό κόμβο (δηλαδή σε αυτόν που εισήχθηκε την τελευταία φορά) , διότι τα ids τα διαβάζουμε με την σειρά, οπότε κάθε φορά που διαβάζουμε ένα κείμενο, αν αυτό υπάρχει ήδη στην posting list θα είναι στον τελευταίο κόμβος της λίστας (άρα θα αυξήσουμε το termFrequency κατευθείαν του last node), αλλιώς αν δεν υπάρχει θα εισαχθεί στο τέλος της λίστας. Έτσι πετυχαίνουμε να έχουμε πολυπλοκότητα εισαγωγής ή ενημέρωσης O(1).

- Trie
  - Χρησιμοποιείται για την αποθήκευση των λέξεων από όλα τα κείμενα και για την εύρεση διαφόρων πληροφοριών που αφορούν την κάθε λέξη, όπως το document frequency ή το term frequency, πληροφορίες που μπορούμε να πάρουμε από την posting list κάθε λέξης.
  - Παρατηρήσεις:
    - Έγινε χρήση **κληρονομικότητας και virtual μεθόδων**, ώστε **να αποφύγουμε** την σπατάλη του να έχουμε **δείκτη σε posting list σε κάθε κόμβο**, γιατί στους περισσότερους κόμβους ο δείκτης θα είναι NULL, οπότε αυτό αποτελεί μεγάλη σπατάλη μνήμης μια και κάθε κόμβος αναπαριστά και από ένα γράμμα μιας λέξης.
    - Οι λέξεις στο Trie είναι **ταξινομημένες** από το μικρότερο ASCII code προς το μεγαλύτερο, ώστε συνήθως η αναζήτηση μιας λέξης να *σταματάει πιο γρήγορα* εάν η λέξη δεν βρίσκεται μέσα στο Trie καθώς επίσης και για να μπορεί να χρησιμοποιηθεί το Trie στο μέλλον σαν κάποια *μορφή λεξικού*.
    - Το παραπάνω μας επιτρέπει να εκτυπώσουμε το **document frequency vector με ταξινομημένες** λέξεις *όπως αναφέρεται στην εκφώνηση*.
    - Για την εισαγωγή της πρώτης λέξης αν η λέξη αποτελείται από 1 γράμμα, τότε φτιάχνουμε μόνο έναν τελικό κόμβο εκεί που δείχνει ο δείκτης first του Trie και εισάγουμε το id στην posting list του, αλλιώς φτιάχνουμε μη τελικούς κόμβους για τα εσωτερικά γράμματα και τελικό κόμβο για το τελικό γράμμα.
    - Για την εισαγωγή μιας λέξης (πέραν της πρώτης λέξης) :
      - Αν η λέξη έχει μόνο 1 γράμμα σημαίνει ότι έχουμε μόνο έναν τελικό κόμβο ο οποίος θα πρέπει να εισαχθεί στο πρώτο “επίπεδο” των παιδιών. Οπότε κάνοντας τις απαραίτητες ενέργειες βρίσκουμε αν υπάρχει ήδη κόμβος με το ίδιο γράμμα. Αν υπάρχει και είναι τελικός τότε απλώς κάνουμε τις απαραίτητες ενέργειες στην λίστα του, ενώ αν υπάρχει και είναι μη τελικός τον μετατρέπουμε σε τελικό. Αν δεν υπάρχει καθόλου, δημιουργούμε εξαρχής έναν τελικό κόμβο.
      - Αν η λέξη έχει παραπάνω από 1 γράμματα, τότε βρίσκουμε την τοποθεσία του πρώτου γράμματος και κάνουμε τις απαραίτητες ενέργειες ώστε να προχωρήσουμε στα επόμενα παιδιά ή αν δεν υπάρχουν επόμενα παιδιά τότε να δημιουργήσουμε μη τελικούς κόμβους για τα εσωτερικά γράμματα και τελικό κόμβο για το τελευταίο γράμμα. Όταν βρούμε ότι δεν υπάρχει ήδη κόμβος για κάποιο εσωτερικό γράμμα τότε δημιουργούμε μη τελικούς κόμβους για τα γράμματα από το συγκεκριμένο γράμμα μέχρι και το προτελευταίο και τελικό κόμβο για το τελευταίο γράμμα.
      - Για την εύρεση του τελικού παιδιού χρησιμοποιείται διαφορετική συνάρτηση, διότι το τελικό παιδί θα είναι και τελικός κόμβος, οπότε και χρειαζόμαστε τον προηγούμενο κόμβου του τελικού παιδιού ώστε να ελέγξουμε αν ο κόμβος που αντιστοιχεί στο τελικό παιδί πρέπει να εισαχθεί ή πρέπει να μετατραπεί σε τελικό κόμβο αν υπάρχει ήδη.

Γενικές Παρατηρήσεις

- Το αρχείο διαβάζεται 2 φόρες, οπού την πρώτη ελέγχουμε αν η σειρά των ids (ξεκινώντας από id = 0) είναι σωστή καθώς επίσης μετράμε και τον αριθμό των ids. Οι κενές γραμμές παραλείπονται.
- Αν δοθεί id και δίπλα του δεν δοθεί κάποια λέξη, τότε θεωρείται error και το πρόγραμμα σταματάει.
- Για τον υπολογισμό των scores το avgdl θεωρείται double, όλες οι μεταβλητές είναι double, και ως term frequency θεωρείται το πόσες φορές εμφανίζεται η λέξη στο κείμενο. Ο λογάριθμος που χρησιμοποιείται είναι με βάση το 10.
- Για την δημιουργία και την εκτύπωση της ειδικής πληροφορίας μπροστά από κάθε κείμενο, γίνονται οι απαραίτητες ενέργειες ώστε για την ευθυγράμμιση των αποτελεσμάτων , να **χρησιμοποιούνται όσο τον δυνατόν λιγότερα κενά**, ανάλογα με το πλήθος των ψηφίων του μέγιστους αύξοντα αριθμού, του μέγιστου σκορ και του μέγιστου αριθμού id, **τα οποία προσαρμόζονται ανάλογα με το κάθε query**.
- Οι επιλογές που υποστηρίζονται είναι της μορφής /df είτε \df.
- Γίνονται οι απαραίτητοι έλεγχοι κάθε φορά που δεσμεύουμε μνήμη στο σωρό, και **αν κάποιο new επιστρέψει NULL τότε το πρόγραμμα τερματίζει ακαριαία**.
- Για τον υπολογισμό των scores, αρχικά χρησιμοποιείται προσωρινά ένας πίνακας μεγέθους όσα είναι όλα τα id του DocMap, ώστε να έχουμε O(1) εισαγωγή και ενημέρωση Score για κάθε id που σχετίζεται με το query. Έπειτα δημιουργούμε έναν μικρότερο πίνακα που περιλαμβάνει μόνο τα τελικά scores των ids που σχετίζονται με το κείμενο και διαγράφουμε τον αρχικό μεγάλο πίνακα. Αυτός ο τελικός πίνακας είναι που θα δοθεί στον constructor του Heap και θα δημιουργθεί ο σωρός για την εύρεση των top k αποτελεσμάτων. Έτσι μπορεί να έχουμε προσωρινή σπατάλη μνήμης όμως έχουμε γρήγορο υπολογισμό των scores και εύρεση των top k αποτελεσμάτων.
- Στο docfile μπορεί να δοθεί αυθαίρετος αριθμός από κενές γραμμές καθώς επίσης και id της μορφής 000000000015 είτε να υπάρχουν whitespaces πριν από το id.

Περισσότερες λεπτομέρειες και περιγραφή για τον κώδικα του κάθε αρχείου (.cc) μπορούν να βρεθούν στην αρχή του κάθε αρχείου καθώς επίσης και στα διάφορα σχόλια που υπάρχουν ενδιάμεσα στον κώδικα.