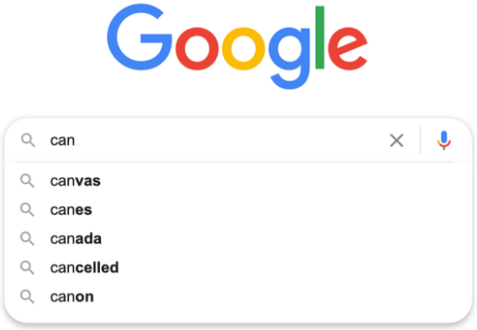
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ομαδική Εργασία – Δεντρικές Δομές Δεδομένων

Προθεσμία Παράδοσης: 02/04@013:30

Η εργασία αυτή επικεντρώνεται στην ανάπτυξη μιας εφαρμογής που υλοποιεί δύο εκδοχές της δομής δεδομένων **Trie** για την αποθήκευση λέξεων σε ένα λεξικό, με σκοπό την αυτόματη πρόταση εναλλακτικών λέξεων. Στόχος είναι η συγκριτική ανάλυση των δύο υλοποιήσεων ως προς την αποδοτικότητά τους στη μνήμη και η διερεύνηση της χρησιμότητας της δομής Trie σε μηχανισμούς που βασίζονται στην πρόβλεψη λέξεων, όπως η αυτόματη συμπλήρωση κειμένου σε μηχανές αναζήτησης.



Μέρος 1ο: Εφαρμογή Πρότασης Εναλλακτικών Λέξεων με Χρήση Δομής Trie

1. Υλοποίηση Λεξικού με Δομή Trie και Πίνακα Κατακερματισμού Robin Hood

Αρχικά, η εφαρμογή θα υλοποιήσει ένα λεξικό με χρήση δομής **Trie** ενσωματώνοντας σε αυτή έναν πίνακα κατακερματισμού τύπου **Robin Hood**, όπως αυτός εξηγήθηκε και υλοποιήθηκε στα πλαίσια του εργαστηρίου Κάθε κόμβος της δομής Trie θα περιέχει μία μεταβλητή **importance**, η οποία αναπαριστά τη σημαντικότητα της λέξης, με τη σημασία να αυξάνεται κάθε φορά που η λέξη εμφανίζεται σε ένα επιπρόσθετο αρχείο κειμένου.

2. Ανάγνωση και Επεξεργασία Αρχείων Εισόδου

Το πρόγραμμά σας θα δέχεται σαν όρισμα (από τη γραμμή εντολής) δύο αρχεία:

- Αρχείο Λεξικού: Περιέχει μια λέξη ανά γραμμή, χωρίς σημεία στίξης ή άλλους χαρακτήρες. Κάθε λέξη του αρχείου θα εισάγεται στο Trie
- Αρχείο Σημαντικότητας: Περιέχει κείμενο (με τη γενική μορφή ενός αρχείου κειμένου) από το οποίο θα αντληθεί η συχνότητα εμφάνισης των λέξεων από το λεξικό. Κάθε φορά που μια λέξη του λεξικού εντοπίζεται στο κείμενο, η τιμή της σημαντικότητας της λέξης αυξάνεται. Οι λέξεις του αρχείου θα πρέπει να διαχωρίζονται από τα σημεία στίξης πριν αυτές ελεγχθούν κατά πόσο υπάρχουν στο λεξικό.

Με το πέρας της επεξεργασίας του αρχείου σημαντικότητας, κάθε λέξη του λεξικού θα έχει μια τελική τιμή σημαντικότητας, η οποία αντιστοιχεί στον αριθμό εμφανίσεων της λέξης στο κείμενο.

3. Πρόταση Εναλλακτικών Λέξεων

Η εφαρμογή σας θα περιλαμβάνει έναν μηχανισμό, ο οποίος θα προσφέρεται επαναληπτικά, για την αναζήτηση και πρόταση εναλλακτικών λέξεων. Θα δέχεται από τον χρήστη μία λέξη και έναν αριθμό k, ο οποίος καθορίζει το πλήθος των εναλλακτικών λέξεων που θα εμφανιστούν ως προτάσεις. Οι λέξεις που προτείνονται πληρούν ένα ή περισσότερα από τα παρακάτω κριτήρια ομοιότητας με τη λέξη εισόδου:

- Πρόθεμα: Η λέξη εισόδου είναι πρόθεμα της προτεινόμενης λέξης.

Παράδειγμα δοθείσας λέξης: plan. Προτεινόμενες λέξεις: plant, plane, plans, planet, plank, planning.

- Ίδιο μήκος με διαφορά χαρακτήρων: Η προτεινόμενη λέξη έχει το ίδιο μήκος με τη λέξη εισόδου και διαφέρει από αυτήν το πολύ κατά δύο χαρακτήρες.

  Παράδειγμα δοθείσας λέξης: plan. Προτεινόμενες λέξεις: play, clan, plum, span.

- Διαφορετικό μήκος: Η προτεινόμενη λέξη έχει μήκος είτε έως και δύο χαρακτήρες μεγαλύτερο από τη λέξη εισόδου, είτε ένα χαρακτήρα λιγότερο. Σε περίπτωση που το μήκος είναι μεγαλύτερο, θα υπάρχει πιθανότητα η λέξη να μην εμφανίζεται αυτούσια στην προτεινόμενη λέξη (επιτρέπεται η εμφάνιση χαρακτήρων ενδιάμεσα των χαρακτήρων της λέξης).

  Παράδειγμα δοθείσας λέξης: plan. Προτεινόμενες λέξεις: pan, lan, planet, planner, aplans, plains.

Για την αποθήκευση των εναλλακτικών λέξεων, θα χρησιμοποιήσετε έναν σωρό ελαχίστων (min-heap), ο οποίος διατηρεί τις k λέξεις με τη μεγαλύτερη τιμή σημαντικότητας ανάμεσα στις εναλλακτικές προτάσεις.

4. Βελτιστοποίηση και Απόδοση

Η υλοποίηση σας θα πρέπει να αποδοτική από άποψη χρόνου και μνήμης. Επίσης θα πρέπει να μπορεί να χειριστεί διαφορετικά μεγέθη λεξικών και αρχείων σημαντικότητας χωρίς να απαιτείται αλλαγή στον κώδικα ή στις δομές που χρησιμοποιήσατε.

Μέρος 2ο: Πειραματική Αξιολόγηση των Δύο Υλοποιήσεων Trie ως προς την Κατανάλωση Μνήμης

Στο δεύτερο μέρος, η εργασία επικεντρώνεται στη συγκριτική ανάλυση των υλοποιήσεων της δομής Trie ως προς την κατανάλωση μνήμης που απαιτείται για την αποθήκευση του λεξικού. Οι δύο εκδοχές είναι:

- Υλοποίηση με χρήση στατικού πίνακα: Σε αντίθεση με την υλοποίηση που έχουμε δει στα πλαίσια του εργαστηρίου, κάθε θέση του πίνακα στον κόμβο του Trie θα αποτελείται από το στοιχείο το οποίο φυλάσσεται και τον κόμβο του Trie στον οποίο παραπέμπει η συγκεκριμένη θέση. Κάθε κόμβος του Trie θα περιέχει και τη σημαντικότητα της λέξης που τερματίζεται εκεί. Δηλαδή η θέση του χαρακτήρα προς εισαγωγή δε θα υπολογίζεται πλέον με τη χρήση του πίνακα ASCII.
- Υλοποίηση με χρήση πίνακα κατακερματισμού τύπου Robin Hood: Όπως αυτή υλοποιήθηκε στο μέρος 1 της εργασίας.

1. Μεταβλητές Πειράματος:

Κατά την πειραματική ανάλυση, θα παρατηρήσετε πώς η κατανάλωση μνήμης επηρεάζεται από τις παρακάτω συνθήκες. Τα λεξικά θα παράγονται τυχαία, δεν είναι υποχρεωτική η χρήση πραγματικών λεξικών.

- **Πλήθος των λέξεων** στο λεξικό (n). Εξετάστε διαφορετικές τιμές του n, π.χ., 1000, 5000, 10000, 100000 κτλ.
- **Μήκος των λέξεων**: Δοκιμάστε δύο διαφορετικά σενάρια για το μήκος των λέξεων.
  - **Ίσο μήκος**: Όλες οι λέξεις του λεξικού έχουν το ίδιο μήκος.
  - **Διάφορα μήκη**: Οι λέξεις του λεξικού ποικίλλουν σε μήκος. Το μήκος των λέξεων δεν πρέπει να είναι παράγεται τυχαία και ομοιόμορφα σε ένα εύρος μήκους (πχ 1 – 20 χαρακτήρες), αλλά θα πρέπει να βασίζεται σε μια κατανομή η οποία να λαμβάνει υπόψιν την κατανομή λέξεων σε ένα πραγματικό λεξικό (πχ οι λέξεις με μήκος 18 ή 21 γράμματα θα είναι πολύ λιγότερες από τις λέξεις με 7 χαρακτήρες). Από που αντλήσατε τις πηγές σας για αυτή την κατανομή;

2. Διαδικασία Πειράματος:

- Για κάθε τιμή του n και για τις διαφορετικές περιπτώσεις μήκους λέξεων, φορτώστε τις λέξεις στα λεξικά και μετρήστε τη μνήμη που χρησιμοποιείται.
- Για κάθε μέγεθος λεξικού, εκτελέστε την εφαρμογή με διαφορετικά σύνολα λέξεων, λαμβάνοντας τον **μέσο όρο** της μνήμης που χρησιμοποιείται για το συγκεκριμένο μέγεθος λεξικού.

3. Παρουσίαση Αποτελεσμάτων:

Παρουσιάστε τα αποτελέσματα υπό μορφή γραφημάτων, όπου ο άξονας x θα αναπαριστά τις διαφορετικές τιμές του n (το μέγεθος του λεξικού) και ο άξονας y τη μέση κατανάλωση μνήμης ανά υλοποίηση. Για κάθε διάγραμμα, γράψτε μία σύντομη ανάλυση (μέχρι 8 γραμμές) των παρατηρήσεών σας.

Αξιολόγηση Εργασίας

Η προγραμματιστική αυτή άσκηση είναι ομαδική και η μέγιστη δυνατή βαθμολογία είναι 100. Για την αξιολόγηση της άσκησης, θα ληφθούν οι πιο κάτω παράμετροι:

1. **Σχόλια και Δομή Προγράμματος:** Γράφετε κατανοητά σχόλια που να εξηγούν την λειτουργία της κάθε κλάσης/πεδίου/μεθόδου. Προτρέπεται όπως το πρόγραμμα σας σχεδιάζεται από πάνω προς τα κάτω, αποφασίζοντας δηλαδή πρώτα τις κλάσεις και διασυνδέσεις τους, στη συνέχεια τις μεταβλητές και μεθόδους και τέλος υλοποιώντας το σώμα των συναρτήσεων.
2. **Σχολιασμός και Δομή Προγράμματος**: Σωστή οργάνωση, σχεδίαση και τεκμηρίωση των κλάσεων, μεθόδων και συναρτήσεων. Θα πρέπει να υπάρχει η κατάλληλη στοίχιση του κώδικα και κατάλληλη ονοματολογία σε κλάσεις του προγράμματος.
3. **Ορθότητα Λειτουργίας (50%):** Η εφαρμογή πρέπει να καλύπτει πλήρως τις προδιαγραφές όπως αυτές περιγράφονται πιο πάνω και να επιστρέφει τη σωστή απάντηση για όλα τα στιγμιότυπα του πεδίου ορισμού του προβλήματος που λύνει.
4. **Πειραματική Αξιολόγηση Δομών Trie (50%):** Ποιοτική συλλογή και ανάλυση των αποτελεσμάτων της πειραματικής ανάλυσης.
5. Σε περίπτωση που το πρόγραμμά σας δεν επιστρέφει αποτέλεσμα ή παρουσιάζει σφάλματα, η μέγιστη βαθμολογία της εργασίας σας θα είναι 20. Σε περίπτωση που παραδώσετε μόνο πρόγραμμα χωρίς αναφορά, η μέγιστη βαθμολογία της εργασίας σας θα είναι 40.
6. Η εργασία θα τύχει αξιολόγησης κατά τη διάρκεια των εργαστηρίων στις 02/04. Στην περίπτωση αυτή, θα πρέπει όχι μόνο να είστε σε θέση να απαντήσετε σε ερωτήσεις που αφορούν την υλοποίησή σας, αλλά και να αλλάξετε τον κώδικά σας έτσι ώστε να ικανοποιεί τις νέες απαιτήσεις που θα σας δοθούν κατά τη διάρκεια της εξέτασης, εάν αυτό ζητηθεί. Αποτυχία στην απάντηση των ερωτήσεων θα οδηγήσει στον μηδενισμό της εργασίας. Σε περίπτωση μη παρουσίασης στην αξιολόγηση δηλώνει ότι η εργασία σας έχει κάποιο από τα προβλήματα του σημείου 6 και η μέγιστη βαθμολογία της εργασίας σας θα είναι 20.
7. Σε περίπτωση αξιοποίησης εργαλείων Τεχνητής Νοημοσύνης (ΤΝ) κατά την εκπόνηση εργασιών, ο/η φοιτητής/τήτρια θα πρέπει να αναφέρει **ρητά και επακριβώς το εργαλείο ΤΝ** που χρησιμοποίησε και **με ποιον τρόπο το αξιοποίησε. Η αυτούσια αντιγραφή από εργαλεία ΤΝ απαγορεύεται.**

Οδηγίες Υποβολής

Η εργασία θα πρέπει να παραδοθεί ηλεκτρονικά στο BlackBoard ως συμπιεσμένο αρχείο .zip ή .rar, το οποίο θα περιλαμβάνει τον κώδικα και την αναφορά. Δώστε στο αρχείο σας το όνομα των φοιτητικών ταυτοτήτων σας και ακολουθήστε τη μορφοποίηση package ID1#.ID2# (πχ **ID768213. ID128412**).

ΚΑΛΗ ΕΠΙΤΥΧΙΑ Απάντηση στην ερώτηση για καλ. μνήμης και για τον αφήναμε το Trie με το ASCII: Σημαντικό για το πείραμα. Στο trie, σε κάθε θέση το πίνακα εκτός από το στοιχείο, θεωρώ ότι κρατώ και την θέση που πάω για να είναι δίκαιο με το Robin Hood. Πρέπει να έχω και πα 2 το importance για να είμαι δίκαιος.

Θα μετρήσαμε την κατανάλωση μνήμης MANUALLY

Απάντηση για random Generator:

Ναι θα παράγονται τυχαία, 1) ίδιο μήκος 2) κατανομή.
Δεν μας ενδιαφέρει να βγάζουν νόημα οι λέξεις.
Να κάναμε script για να πάρουμε τα αποτελέσματα

Απάντηση για ερώτηση file reading και string:

Όχι να χρησιμοποιείτε functions της java αρκεί να μην επηρεάζουν τις δομές

Απάντηση για Min Heap:

Όχι για το πείραμα αλλά για την εφαρμογή, ουσιαστικά θα έχαμε MENU, ο χρήστης δείνει λέξη και k και θέλει να του εμφανίζεις τις πρώτες k λέξεις σε θέμα importance που πληρούν τα κριτήρια. Οι λέξεις αυτές θα φυλάγονται στο min Heap.

Απάντηση για Importance:

Το Importance έχει να κάνει με την εφαρμογή και όχι το πείραμα. Κεντρική ιδέα: Φορτώναμε το λεξικό στο Trie Node. Έπειτα μετά την φόρτωση τα κείμενα δυναμικότητας, κάθε φορά που βρίσκαμε μια λέξη πα

υπάρχει στο λεξικό αυξάναμε το importance της μέσα στο Trie Node.

Γιατί είναι σημαντικό το importance:

Για την εφαρμογή. Μετά την φόρτωση τα λέξεια και τα αρχεία σημαντικότητας ο χρήστης θα δείνει μια λέξη και ένα k. Βάση της λέξης, θα βρίσκαμε τις πρώτες k λέξεις με το μεγαλύτερο importance που θα ικανοποιούν τα κριτήρια, θα τις βάζαμε μέσα στο Heap και θα τις εμφανίζαμε.

=>ΖΗΤΟΥΜΕΝΑ

Υλοποίηση του Trie με χρήση Robin Hood όπου υλοποιεί όλα τα παραπάνω ζητούμενα ώστε να δουλεύει η εφαρμογή με το μενού.

Υλοποίηση των μεθόδων για μνήμη στο static Trie και Robin Hood Trie για σωστό υπολογισμό μνήμης με τα πιο πάνω κριτήρια

Υλοποίηση μετρήσεων και γραφικών