Χαροκόπειο Πανεπιστήμιο
Τμήμα Πληροφορικής και Τηλεματικής

Δομές Δεδομένων
Εργασία

Διδάσκων: Δημήτρης Μιχαήλ

2020-2021

Σκοπός της εργασίας αυτής είναι η εξοικείωση σας με την υλοποίηση απλών δομών δεδομένων και αλγορίθμων που χειρίζονται αυτές τις δομές. Στην συγκεκριμένη εργασία καλείστε να υλοποιήσετε την κωδικοποίηση Huffman σε γλώσσα Java.

1 Πίνακας Συχνοτήτων

Παρόλο που η Java υποστηρίζει UTF-8 εσείς θα υπολογίσετε την κωδικοποίηση Huffman μόνο για τους χαρακτήρες ASCII από 0 έως και 127. Για να είναι όμως σωστές οι συχνότητες που θα χρησιμοποιήσετε, θα χρειαστεί να τις υπολογίσετε πειραματικά.

Στο πρώτο μέρος της άσκησης καλείστε να γράψετε ένα πρόγραμμα σε Java που να διαβάζει τα παρακάτω αρχεία και να υπολογίζει έναν πίνακα συχνοτήτων για τα γράμματα με ASCII από 0 έως και 127.

- `https://www.gutenberg.org/files/1342/1342-0.txt`
- `https://www.gutenberg.org/files/11/11-0.txt`
- `https://www.gutenberg.org/files/2701/2701-0.txt`

Οποιοσδήποτε άλλος χαρακτήρας, που δεν είναι στο εύρος 0 έως 127, δεν αφορά το συγκεκριμένο πρόγραμμα και μπορεί να αγνοηθεί. Το πρόγραμμα σας θα πρέπει να διαβάζει τα παραπάνω αρχεία κειμένου και να παράγει στην έξοδο ένα αρχείο `frequencies.dat` που να περιέχει ανά γράμμα την συχνότητα εμφάνισης. Μελετήστε το tutorial για I/O στον παρακάτω σύνδεσμο

- `https://docs.oracle.com/javase/tutorial/essential/io/index.html`

για το πως χρησιμοποιούμε αρχεία. Για την μέτρηση της συχνότητας αρκεί να χρησιμοποιήσετε έναν πίνακα ακεραίων με 128 θέσεις, μία για κάθε χαρακτήρα ASCII.

2 Υπολογισμός Δέντρου Huffman

Στο δεύτερο μέρος της άσκησης καλείστε να υλοποιήσετε ένα πρόγραμμα που να διαβάζει το αρχείο με τον πίνακα συχνοτήτων που υπολογίσατε στον πρώτο μέρος και να παράγει ένα δέντρο Huffman.

- Θα χρειαστεί να ορίσετε μία κλάση για τους κόμβους του δέντρου. Οι κόμβοι περιέχουν μία συχνότητα, ένα γράμμα σε περίπτωση που είναι φύλλα και δύο παιδιά.
- Θα χρειαστεί να ορίσετε επίσης μία κλάση για το δέντρο Huffman.
- Θα πρέπει να υλοποιήσετε τον αλγόριθμο του Huffman που να παράγει ένα δέντρο χρησιμοποιώντας ως είσοδο τον πίνακα συχνοτήτων.

Κατά την διάρκεια του αλγορίθμου θα χρειαστεί να διατηρήσετε μία συλλογή από ρίζες (δέντρων) και να μπορείτε να βρείτε τις δύο ρίζες με την μικρότερη συχνότητα. Για μία βασική λύση, μπορείτε να χρησιμοποιήσετε την λίστα της Java (`https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html`) και να ψάχνετε με επανάληψη. Θα δωθεί όμως bonus σε όποιους υλοποιήσουν την διαδικασία αυτή με ουρά προτεραιότητας, χρησιμοποιώντας για παράδειγμα τον δυαδικό σωρό που φτιάξαμε στο εργαστήριο.

Μετά το τέλος του αλγορίθμου θα πρέπει να αποθηκεύσετε το δέντρο σε ένα αρχείο με όνομα `tree.dat`. Εδώ μπορείτε να χρησιμοποιήσετε κάποια δική σας αναπαράσταση για το δέντρο ή μπορείτε για ευκολία να γράψετε απευθείας το αντικείμενο του δέντρου σε αρχείο χρησιμοποιώντας Java Object Serialization. Δείτε εδώ `https://docs.oracle.com/javase/tutorial/essential/io/objectstreams.html` για περισσότερες λεπτομέρειες. Συνήθως αρκεί να υλοποιήσουν όλες οι κλάσεις σας το interface Serializable και να έχουν όλες οι κλάσεις default constructor.

3 Υπολογισμός Κωδικοποίησης

Στο τρίτο μέρος της εργασίας καλείστε να υλοποιήσετε ένα πρόγραμμα που να διαβάζει το δέντρο Huffman από το αρχείο `tree.dat` και να υπολογίζει τον κώδικα για κάθε γράμμα. Το αποτέλεσμα θα πρέπει να το γράψετε σε ένα αρχείο `codes.dat` όπου θα πρέπει να περιέχει ανά γράμμα την κωδικοποίηση με 0 και 1. Για απλότητα τα 0 και 1 μπορείτε να τα συμβολίσετε με χαρακτήρες. Με άλλα λόγια το αρχείο θα πρέπει να περιέχει ανά γράμμα μία συμβολοσειρά με 0 και 1.

Για τον υπολογισμό της κωδικοποίησης θα πρέπει να υλοποιήσετε μία διάσχιση του δέντρου. Κατά την διάρκεια της διάσχισης θα πρέπει να χρησιμοποιήσετε μία στοίβα ώστε να θυμάστε ανά πάσα στιγμή την κωδικοποίηση του μονοπατιού από την ρίζα μέχρι και τον κόμβο που είναι η διάσχιση. Στην περίπτωση που η διάσχιση φτάσει σε κάποιο φύλλο, θα πρέπει μέσω της στοίβας να βρείτε την αναπαράσταση του γράμματος που αντιστοιχεί στο φύλλο αυτό.

Για στοίβα μπορείτε να χρησιμοποιήσετε είτε την κλάση που φτιάξαμε στο εργαστήριο είτε για ευκολία την κλάση ArrayDeque της Java. Στην περίπτωση που χρησιμοποιήσετε την κλάση του εργαστηρίου θα πρέπει να την εμπλουτίσετε με μία μέθοδο που να σας επιστρέφει όλα τα περιεχόμενα της στοίβας για παράδειγμα σε μία λίστα. Η ArrayDeque έχει αυτή την δυνατότητα μέσω της μεθόδου `iterator()`.

4 Κωδικοποιητής

Στο τέταρτο μέρος της εργασίας καλείστε να υλοποιήσετε τον κωδικοποιητή Huffman. Ο κωδικοποιητής θα πρέπει να διαβάζει από την γραμμή εντολών δύο ονόματα αρχείων. Το πρώτο είναι το αρχείο εισόδου που θεωρούμε πως είναι σε κωδικοποίηση ASCII και το δεύτερο το αρχείο εξόδου που θα πρέπει να είναι σε κωδικοποίηση Huffman. Οι παράμετροι της γραμμής εντολών βρίσκονται στην μοναδική παράμετρο της συνάρτησης main (`String args[]`). Δείτε το tutorial της Java στον παρακάτω σύνδεσμο:

- `https://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html`.

Ο κωδικοποιητής θα πρέπει:

- να διαβάσει τη κωδικοποίηση Huffman από το αρχείο `codes.dat`,
- να διαβάσει το κείμενο από το αρχείο εισόδου ανά χαρακτήρα και να γράψει στο αρχείο εξόδου τους ίδιους χαρακτήρες αλλά με την κωδικοποίηση Huffman.

Εδώ είναι το πρώτο σημείο όπου θα πρέπει να αναπαράσταση σας να γίνει σε επίπεδο bits. Επειδή η αναπαράσταση είναι μεταβλητού μήκους σε bits, θα πρέπει να κρατάτε στην μνήμη τα επόμενα bytes. Μόλις οριστικοποιηθούν οι τιμές τους και άρα γνωρίζετε όλα τα bits, μπορείτε να τα γράψετε στην έξοδο.

Η γλώσσα Java περιέχει ένα τύπο `byte` για να αναπαραστήσετε ένα byte. Για να γράψετε μεμονωμένα bits σε ένα byte θα χρειαστεί να χρησιμοποιήσετε τους bitwise operators. Για περισσότερες λεπτομέρειες δείτε το tutorial της Java στον παρακάτω σύνδεσμο:

- `https://docs.oracle.com/javase/tutorial/java/nutsandbolts/op3.html`

Μία πιθανά ευκολότερη λύση είναι να χρησιμοποιήσετε την κλάση BitSet για ενδιάμεση αναπαράσταση.

Τέλος θα χρειαστεί να αντιμετωπίσετε επιτυχώς το πρόβλημα πως ο αριθμός των bits ενός αρχείου δεν είναι πλέον υποχρεωτικά πολλαπλάσιο του 8 και άρα με κάποιον τρόπο πρέπει να ξέρετε πόσα bits έχετε γράψει σε ένα αρχείο. Υπάρχουν διάφοροι μέθοδοι για να το αντιμετωπίσετε, μερικές από τις οποίες είναι:

- να γράφετε στην αρχή του αρχείου τον συνολικό αριθμό από bits,
- μπορείτε να γράφετε στην αρχή τον συνολικό αριθμό από bits modulo 8 και άρα να ξέρετε πόσο bits αντιστοιχούν σε πληροφορία στο τελευταίο byte του αρχείου,
- μπορείτε να προσθέσετε στο αλφάβητο σας ένα επιπλέον χαρακτήρα 'EOF' και να τον γράφετε πάντα στο τέλος του αρχείου.

5 Αποκωδικοποιητής

Στο πέμπτο μέρος της άσκησης καλείστε να υλοποιήσετε τον αποκωδικοποιητή. Ο αποκωδικοποιητής θα πρέπει να διαβάζει από την γραμμή εντολών δύο ονόματα αρχείων. Το πρώτο είναι το αρχείο εισόδου που θεωρούμε πως είναι σε κωδικοποίηση Huffman και το δεύτερο το αρχείο εξόδου που θα πρέπει να είναι σε κωδικοποίηση ASCII. Οι παράμετροι της γραμμής εντολών βρίσκονται στην μοναδική παράμετρο της συνάρτησης main (`String args[]`). Δείτε το tutorial της Java στον παρακάτω σύνδεσμο:

- `https://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html`.

Ο αποκωδικοποιητής θα πρέπει:

- να διαβάσει το δέντρο Huffman από το αρχείο `tree.dat`,
- να διαβάσει το κείμενο από το αρχείο εισόδου ανά bit και χρησιμοποιώντας το δέντρο Huffman να αποκωδικοποιήσει το κείμενο και να το γράψει στην έξοδο σε ASCII.

Μπορεί να είναι πιο εύκολο να διαβάσετε όλο το κείμενο πρώτα στην μνήμη, π.χ σε έναν πίνακα από bytes, να φτιάξετε ένα BitSet και στην συνέχεια να κάνετε την αποκωδικοποιήση. Εδώ έχει σημασία να ξέρετε με κάποιον τρόπο πότε τελειώνει το αρχείο (βλέπε σχετική συζήτηση στο 4ο μέρος της άσκησης).

Καθώς διαβάζετε την είσοδο ανά bit, θα πρέπει να κάνετε μία διάσχιση του δέντρου ξεκινώντας πάντα από την ρίζα. Άμα δείτε 0 θα πρέπει να πάτε αριστερά αλλιώς δεξιά. Αν φτάσετε σε φύλλο, τυπώνετε στην έξοδο τον αντίστοιχο χαρακτήρα και απαναλαμβάνετε την ίδια διαδικασία από την ρίζα.

6 Παραδοτέα

Η άσκηση έχει ένα παραδοτέο ανά μέρος (συνολικά πέντε παραδοτέα με διαφορετικές ημερομηνίες). Σε κάθε μέρος, παραδοτέα είναι ο πηγαίος κώδικας ο οποίος θα πρέπει να μεταγλωττίζεται πολύ εύκολα χρησιμοποιώντας το maven. Δεν θα γίνουν δεκτές λύσεις που χρειάζονται εξωτερικά προγράμματα ή που δεν μεταγλωττίζονται επιτυχώς σε περιβάλλον Linux. Μαζί με τον πηγαίο κώδικα θα πρέπει να υπάρχει και ένα αρχείο README το οποίο να περιγράφει την διαδικασία μεταγλώττισης και εκτέλεσης. Τέλος θα πρέπει να υπάρχει και ένα αρχείο report.pdf το οποίο να περιγράφει αναλυτικά

την δουλειά σας, να εξηγεί τον κώδικα σας, και να περιέχει παραδείγματα εκτέλεσης του κώδικας σας. Αναφορά θα πρέπει να υπάρχει σε όλα τα παραδοτέα.

Προσοχή η βαθμολόγηση δεν γίνεται μόνο με βάση την λειτουργικότητα αλλά και με βάση την ποιότητα του κώδικα. Επιπρόσθετα σημαντικό ρόλο παίζει και η αναφορά.