

Assignment Report

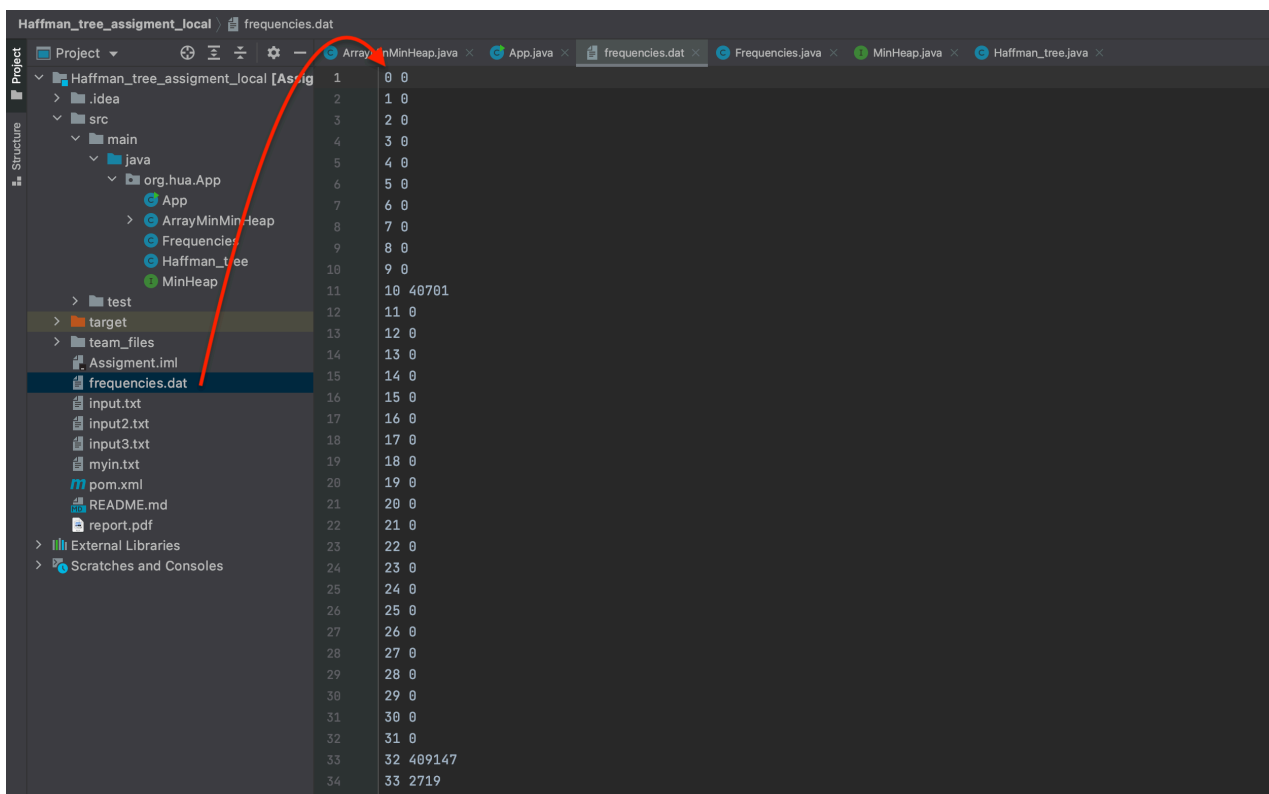
N.Liapis-it21950
A.Chourlias-it219113
C.Zalachoris-it21922

D.Michail
21 December 2020

1ST PART

Αρχικά Δημιουργήσαμε μια κλάση Frequencies, η οποία μέσω της `fileReader` και της `bufferReader` ανοίγει τα αρχεία `.txt`. Έπειτα δημιουργούμε έναν πίνακα συχνοτήτων, από `long's`, για τους ASCII χαρακτήρες και μια `private` μέθοδο `Readcounter` η οποία ξεκινάει να διαβάζει με την σειρά κάθε χαρακτήρα μέσα από κάθε αρχείο, την οποία μέθοδο αναλαμβάνει να καλέσει ο `constructor` για χάρη του χρήστη ώστε στην επόμενη γραμμή του κώδικα ο χρήστης να έχει έναν πίνακα συχνοτήτων(`public final count`) να επεξεργαστεί. Ακόμη, μέσω της μεθόδου `write` τυπώνει το αποτέλεσμα σε ένα άλλο αρχείο `frequencies.dat`, αφού τελειώσει η διαδικασία του `writing` στο αρχείο εξόδου, μέσω του `try-with-resources` statement αυτόματα απελευθερώνουμε πόρους οι οποίοι έγιναν κατάληψη ώστε να εκτελεστεί η παραπάνω διαδικασία. Τέλος, στην κύρια κλάση, πρώτα δημιουργούμε 3 αντικείμενα, κάθε αντικείμενο για κάθε αρχείο, με μία δομή `for` διασχίζουμε τους πίνακες συχνοτήτων και τους προσθέτουμε ώστε να πάρουμε την συνολική συχνότητα από κάθε γράμμα του πίνακα ASCII σύμφωνα με την εμφάνιση του σε κάθε αρχείο(αντικείμενο), και στην συνέχεια καλούμε την `write` για το αντικείμενο στο οποίο προσθέσαμε τις συχνότητες των άλλων δυο.

Στην συνέχεια, για το πρώτο κομμάτι της εργασίας δίνεται ένα στιγμιότυπο οθόνης, και στην συνέχεια μία εξήγηση σχετικά με τα αποτελέσματα τα οποία παρουσιάζονται σε αυτό το στιγμιότυπο.



The screenshot shows an IDE with the project structure on the left and the contents of `frequencies.dat` in the main editor. A red arrow points from the `frequencies.dat` file in the project structure to the editor window.

Project Structure:

- Huffman_tree_assignment_local [Assignment]
- src
 - main
 - java
 - org.hua.App
 - App
 - ArrayMinMinHeap
 - Frequencies
 - Huffman_tree
 - MinHeap
- test
- target
- team_files
- Assignment.iml
- frequencies.dat
- input.txt
- input2.txt
- input3.txt
- myin.txt
- pom.xml
- README.md
- report.pdf
- External Libraries
- Scratches and Consoles

Contents of frequencies.dat:

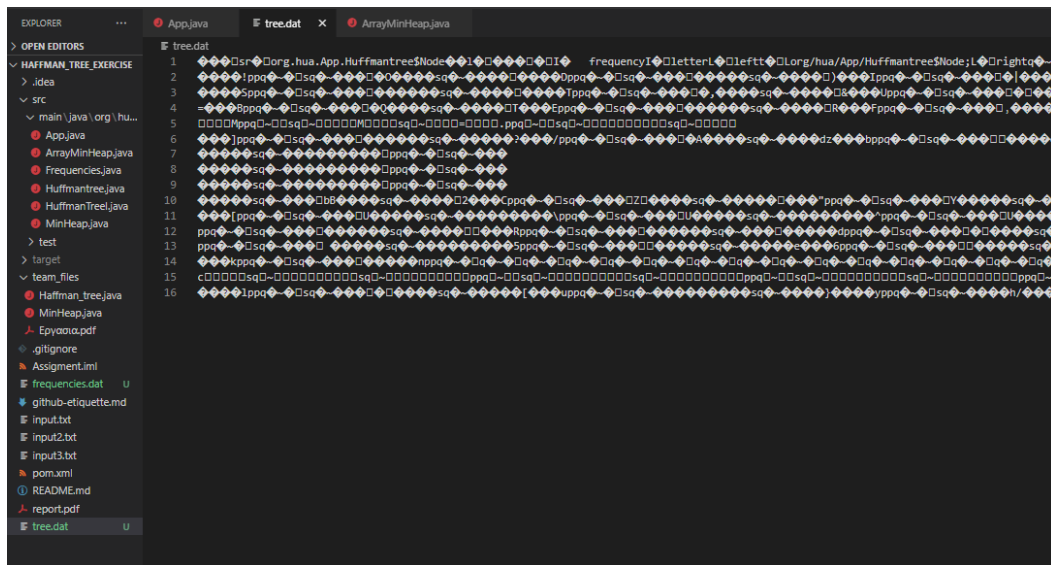
```
1 0 0
2 1 0
3 2 0
4 3 0
5 4 0
6 5 0
7 6 0
8 7 0
9 8 0
10 9 0
11 10 40701
12 11 0
13 12 0
14 13 0
15 14 0
16 15 0
17 16 0
18 17 0
19 18 0
20 19 0
21 20 0
22 21 0
23 22 0
24 23 0
25 24 0
26 25 0
27 26 0
28 27 0
29 28 0
30 29 0
31 30 0
32 31 0
33 32 409147
34 33 2719
35 34 30
```

Στο πιο πάνω στιγμιότυπο βλέπουμε το περιεχόμενο του αρχείου, frequencies.dat, όπου σε κάθε γραμμή υπάρχουν δύο αριθμοί όπου ο πρώτος αναπαριστά την δεκαδική τιμή του χαρακτήρα στον πίνακα ASCII, ενώ ο δεύτερος αριθμός αναπαριστά την συχνότητα του αντίστοιχου χαρακτήρα η οποία έχει υπολογιστεί σύμφωνα με τα παραπάνω.

2ND PART

Ερευνώντας την κωδικοποίηση του Huffman αποκτάμε μια ιδέα για το τι θα ακολουθήσει στα επόμενα κομμάτια της άσκησης. Στο συγκεκριμένο κομμάτι υλοποιούμε την δομή στην οποία αποθηκεύονται τα στοιχεία που συλλέξαμε στο προηγούμενο ερώτημα. Συγκεκριμένα διαβάζουμε τις συχνότητες από το αρχείο `frequencies.dat` και τις εισχωρούμε στην `min heap` που υλοποιήσαμε. Αυτή όπως είδαμε και στο εργαστήριο είναι ένας διάδικος σωρός στον οποίο υπάρχει προτεραιότητα όπου η ριζά είναι πάντα το ελάχιστο κλειδί μεταξύ όλων των άλλων. Αποτελείται από τις βασικές μεθόδους `insert`, `getMin` και `extractMin`, οι οποίες αντίστοιχα προσθέτουν στοιχεία μέσα σε ένα πίνακα, βρίσκουν το στοιχείο με την ελάχιστη τιμή και το εξάγουν αν χρειαστεί. Φυσικά για να λειτουργήσουν αυτές οι μέθοδοι θα πρέπει να υλοποιηθεί η βασική λειτουργία της `min heap` η οποία είναι η `fixup` και η `fixdown`. Η μέθοδος `fixup` με την σειρά της ξεκινά να ελέγχει το τελευταίο στοιχείο που προστέθηκε, αν είναι μικρότερο από τον πατέρα του αλλάζει θέση με αυτόν (σύμφωνα με την `swap` που υλοποιούμε) και συνεχίζεται η ίδια διαδικασία οπότε χρειαστεί. Αντίστοιχα η `fixdown` ξεκινάει με την ριζά ελέγχει εφόσον έχει παιδιά ποιο από τα δυο της παιδιά είναι μικρότερο και έπειτα ελέγχει αν το στοιχείο της ρίζας είναι μικρότερο και από αυτό. Αν δεν είναι τότε αλλάζει τα στοιχεία μέσω της `swap`. Παίρνοντας μια γενική ιδέα για την λειτουργία της `min heap` που υλοποιήσαμε μπορούμε να εξηγήσουμε την δομή του δέντρου Huffman. Αρχικά οφείλουμε να διαβάσουμε το αρχείο των συχνοτήτων από το προηγούμενο κομμάτι της εργασίας. Αυτό το καταφέρνουμε διαβάζοντας σε ένα πίνακα μόνο τις συχνότητες που θα χρειαστούμε και όχι τον χαρακτήρα που αντιπροσωπεύουν. Στην συνέχεια δημιουργούμε την κλάση για την δομή των κόμβων στο δέντρο μας, η οποία αποτελείται από μια μεταβλητή για τις συχνότητες μια για τους χαρακτήρες και τα δυο παιδιά των κόμβων. Δημιουργούμε μεθόδους για να ελέγχουμε αν ο κόμβος έχει παιδιά και έπειτα για την σύγκριση των συχνοτήτων του κάθε κόμβου. Έχοντας λοιπόν δημιουργήσει τους κόμβους είμαστε έτοιμη να υλοποιήσουμε την μορφή ενός δέντρου Huffman. Συγκεκριμένα, δημιουργούμε ένα αντικείμενο για την `min heap` και ξεκινάμε να εισχωρούμε τις συχνότητες που διαβάσαμε μέσα σε αυτή σε μορφή κόμβων όπου τα παιδιά τους δείχνουν σε `null`. Στην συνέχεια, όσο η `min heap` έχει στοιχεία μέσα της εμείς εξάγουμε αρχικά στο αριστερό και έπειτα στο δεξί παιδί του κόμβου τα ελάχιστα στοιχεία και δημιουργούμε με αυτά έναν πατέρα ο οποίος έχει για χαρακτήρα το κενό και για συχνότητα την συνολική από τα δυο παιδιά. Μόλις δημιουργηθεί ο πατέρας τον εισάγουμε πάλι μέσα στην `min heap`. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να γυρίσει η ριζά η οποία έχει την συνολική συχνότητα όλων των κόμβων. Εφόσον τελειώσουμε με την βασική αυτή λειτουργία οφείλουμε να γραφουμε το αντικείμενο σε σειριακή μορφή μέσα στο αρχείο `tree.dat`. Την διαδικασία αυτή επιτυγχάνουμε μέσω της μεθόδου `storeTree` η οποία χρησιμοποιεί τα αντικείμενα της `java`, `FileOutputStream` και `ObjectOutputStream` ώστε να αποθηκευθούν την ριζα του δέντρου μας.

Στην συνέχεια, για το δευτερο κομμάτι της εργασίας δίνεται ένα στιγμιότυπο οθόνης, και στην συνέχεια μία εξήγηση σχετικά με τα αποτελέσματα τα οποία παρουσιάζονται σε αυτό το στιγμιότυπο.



Το αρχείο στο οποίο αποθηκεύουμε το δέντρο μας είναι binary