

ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ:

Αρχικά δημιουργήσαμε τις κλάσεις που θα χρησιμοποιήσουμε στον κώδικα μας την Main, HuffmanNode , TreeMaker, MyComparator, Files.

Στην HuffmanNode δηλώσαμε δύο μεταβλητές την data για τις συχνότητες και την c για τους χαρακτήρες από 0 μέχρι 127 επίσης δηλώσαμε το left και right ως δυο pointers.

Στην MyComparator συγκρίνουμε με βάση τις τιμές δεδομένων των κόμβων.

Στην TreeMaker δημιουργούμε μια function με όνομα create που παίρνει ως παραμέτρους τον πίνακα charfreq για τις συχνότητες των χαρακτήρων ένα charArray για τους χαρακτήρες και ένα n για το μέγεθος των πινάκων. Μετά δημιουργούμε μια ουρά προτεραιότητας και με την βοήθεια της for γεμίζουμε τα Nodes με τους χαρακτήρες και τις συχνότητες τους και τους βάζουμε σε ουρά(queue)

Στη συνέχεια δημιουργούμε ένα root node με τιμή null.

Μετά κάνουμε ένα while loop που παίρνει τα δύο μικρότερα Nodes από την ουρά μέχρι το size της να γίνει 1. Μέσα στο while παίρνουμε την πρώτη τιμή της ουράς με το peek και την καταχωρούμε στην μεταβλητή x και με το poll διαγράφουμε την τιμή από την ουρά μετά παίρνουμε την δεύτερη τιμή της ουράς και κάνουμε το ίδιο για την μεταβλητή y. Στη συνέχεια δηλώνουμε ένα Node f στο οποίο αναθέτουμε την

πρόσθεση της συχνότητας των 2 Node x και y . Μετά ορίζουμε το πρώτο Node ως το αριστερό παιδί και το δεύτερο ως το δεξί παιδί. Ύστερα αναθέτουμε την τιμή του root ως το f και χρησιμοποιώντας την add προσθέτουμε την Node f στην ουρά προτεραιότητας . Τέλος επιστρέφουμε την τιμή root.

Στην κλάση Files δημιουργήσαμε μια μέθοδο Read File που παίρνει παράμετρο ReadName για το path του αρχείου, ένα πίνακα ακεραίων για τις συχνότητες και ένα πίνακα χαρακτήρων για τους χαρακτήρες ascii. Μετά χρησιμοποιήσαμε την εντολή BufferedReader σε συνδυασμό με το FileReader για να μπορέσουμε να διαβάσουμε τις συχνότητες από το αρχείο.

Συνεχίζοντας χρησιμοποιήσαμε ένα while loop .Μέσα στο loop διαβάζουμε τις συχνότητες από το αρχείο με την βοήθεια του indexOf και substring(indexOf: βρίσκει την θέση μετά από ένα συγκεκριμένο string που έχουμε δήλωση, substring: παίρνει ότι υπάρχει μετά από την θέση του indexOf) και τις αποθηκεύουμε στον πίνακα asciitable χρησιμοποιώντας το Integer.valueOf(μετατρέπει το string σε integer).Χρησιμοποιήσαμε μια for για την σωστή τοποθέτηση των frequencies.

Μετά δημιουργήσαμε 2 μεθόδους writeToFile η μια παίρνει ως παραμέτρους ένα χαρακτήρα ένα ακέραιο και μια συμβολοσειρά και η άλλη παίρνει ως παραμέτρους ένα ακέραιο και μια συμβολοσειρά για να για να γράψουμε το δέντρο Huffman.

Στην κλάση Main δημιουργήσαμε μια function `printcode` που παίρνει τους παραμέτρους ένα node και μια συμβολοσειρά δηλώσαμε ένα constructor για να μπορούμε να χρησιμοποιήσουμε την κλάση files και κάναμε ένα if για να ελέγχουμε εάν στο αριστερό και δεξιό παιδί η τιμή είναι null, εάν είναι και οι δυο null τυπώνουμε στο αρχείο `tree.dat` το μονοπάτι του κάθε φύλλου και τερματίζει η `printcode`, εάν δεν είναι null τότε αυτοκαλείται με άλλες παραμετρους, και συνεχίζει μέχρι να φτάσει στην ρίζα του δέντρου.

Στην Main δηλώσαμε ένα πίνακα `charArray` του οποίου αρχικοποιήσαμε τις θέσεις του με χαρακτήρες `ascii` από 0 έως και 127. Μετα δηλώσαμε ένα πίνακα `charfreq` με 128 θέσεις στον οποίο αποθηκεύσαμε τις συχνότητες των χαρακτήρων με την βοήθεια της κλάσης files. Δηλώσαμε το `root` στο οποίο αποθηκεύσουμε την τιμή που επιστρέφει η `create` και τέλος καλούμε την `printcode` για να τυπώσουμε το δέντρο huffman.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΚΤΕΛΕΣΗΣ:

The screenshot shows the Apache NetBeans IDE interface. The main window displays the source code of the `Main.java` file, which implements a Huffman tree construction algorithm. The code uses the `Scanner` class to read input, the `HuffmanNode` class to represent nodes, and the `TreeMaker` class to build the tree. The output window shows the execution results, including the build success message and the finished time. A Notepad window on the right displays the generated Huffman tree structure, which is a binary tree with nodes containing character frequencies and Huffman codes.

```
public static void main(String[] args) throws IOException {  
  
    Scanner s = new Scanner(System.in);  
    HuffmanNode root;  
  
    // number of characters.  
    int n = 128;  
    char[] charArray;  
    int[] charfreq;  
    charfreq = new int[128];  
    charArray = new char[128];  
  
    for (int i = 0; i < 128; i++) {  
        charArray[i] = (char) i;  
    }  
  
    Files f1 = new Files();  
    TreeMaker tree = new TreeMaker();  
    f1.ReadFile("C:\\Users\\35797\\Desktop\\ΧΑΡΟΚΟΠΕΙΟ\\3ο ΕΞΑΜΗΝΟ\\ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ\\tree.dat");  
    root = tree.create(charfreq, charArray, n);  
}
```

Output - Run (HuffmanProject_2) x

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ HuffmanProject_2 ---  
-----  
BUILD SUCCESS  
-----  
Total time: 2.274 s  
Finished at: 2020-12-20T22:16:20+02:00  
-----
```

tree.dat - Notepad

```
File Edit Format View Help  
2203297-880176-411921->e:204231  
2203297-880176-411921-207690->s:103575  
2203297-880176-411921-207690->h:104115  
2203297-880176-468255-219969->i:107638  
2203297-880176-468255-219969->n:112331  
2203297-880176-468255-248286->o:120723  
2203297-880176-468255-248286-127563-61819-30576-15216-7354->A:3609  
2203297-880176-468255-248286-127563-61819-30576-15216-7354-3745-1869-869-432-  
2203297-880176-468255-248286-127563-61819-30576-15216-7354-3745-1869-869-432-  
2203297-880176-468255-248286-127563-61819-30576-15216-7354-3745-1869-869-432-  
2203297-880176-468255-248286-127563-61819-30576-15216-7354-3745-1869-869-432-  
2203297-880176-468255-248286-127563-61819-30576-15216-7354-3745-1869-869-432-  
2203297-880176-468255-248286-127563-61819-30576-15216-7354-3745-1869-869->1:4  
2203297-880176-468255-248286-127563-61819-30576-15216-7354-3745-1869->0:1000  
2203297-880176-468255-248286-127563-61819-30576-15216-7354-3745->E:1876  
2203297-880176-468255-248286-127563-61819-30576-15216-7862->T:3749  
2203297-880176-468255-248286-127563-61819-30576-15216-7862-4113->q:2033  
2203297-880176-468255-248286-127563-61819-30576-15216-7862-4113-2080->R:1029  
2203297-880176-468255-248286-127563-61819-30576-15216-7862-4113-2080-1051-481-  
2203297-880176-468255-248286-127563-61819-30576-15216-7862-4113-2080-1051-481-  
2203297-880176-468255-248286-127563-61819-30576-15216-7862-4113-2080-1051-481-  
2203297-880176-468255-248286-127563-61819-30576-15216-7862-4113-2080-1051-570-  
2203297-880176-468255-248286-127563-61819-30576-15216-7862-4113-2080-1051-570-  
2203297-880176-468255-248286-127563-61819-30576-15216-7862-4113-2080-1051-570-  
2203297-880176-468255-248286-127563-61819-30576->v:15360  
2203297-880176-468255-248286-127563-61819->, :31243  
2203297-880176-468255-248286-127563-65744->y:32138  
2203297-880176-468255-248286-127563-65744->g:33606  
2203297-1323121-552828-262728->a:128252  
2203297-1323121-552828-262728-134476->d:65761  
2203297-1323121-552828-262728-134476-68715-33786->, :15811  
2203297-1323121-552828-262728-134476-68715-33786-17975-8476-4183->x:2085  
2203297-1323121-552828-262728-134476-68715-33786-17975-8476-4183->g:2085  
<----->  
Ln 1, Col 1 100%
```

Created from: Μυριάνθη Αποστολίδη , Χάρη Παπαμιχαήλ, Φώτη Γαλανό