Δομές Δεδομένων

Πέμπτο Παραδοτέο

Γεώργιος-Νεκτάριος Εγγλέζος it21824
Αντώνης Δρόσος it218115
Ανδρέας Σαραντόπουλος it218140

Εισαγωγή

Για την υλοποίηση του Πέμπτου Παραδοτέου της εργασίας φτιάξαμε μια καινούργια κλάση την Fifth.

Αλλαγές σε προηγούμενα παραδοτέα:

1. Διορθώσαμε τον έξτρα χαρακτήρα **EOF** που είχαμε φτιάξει για να ξέρουμε που τελειώνει το διάβασμα των bits από το αποκωδικοποιημένο αρχείο huffman. Στην προηγούμενη μορφή του ξεχαστήκαμε και ο κώδικας που του φτιάχναμε περιείχε τον κωδικό άλλου χαρακτήρα ascii. Τώρα το προσθέσαμε σωστά στην κλάση **First** που φτιάχνει το **frequencies.dat** ως έναν έξτρα χαρακτήρα με αριθμό ascii -1 και συχνότητα -1. Έτσι όταν η κλάση **Second** φτιάχνει τα nodes και το δέντρο, να συμπεριλάβει και το EOF. Τέλος στην κλάση Fourth που φτιάχνει το κωδικοποιημένο κείμενο σε 0 και 1 βάλαμε να βρίσκει από την λίστα με τους κωδικούς (**codes.dat)** που φόρτωνε, τον χαρακτήρα EOF και να τον προσθέτει στο τέλος του αρχείου.

2. Για κάποιο λόγο κατά την αποκωδικοποίηση του κώδικα huffman παρατηρήσαμε ότι δεν έβλεπε τα new lines οπότε κάναμε κάποιες πολύ μικρές αλλαγές στις κλάσεις First(Πρώτο Παραδοτέο) και Fourth(Τέταρτο Παραδοτέο).

Οι αλλαγές του κώδικα και για τα δύο αναφέρονται πιο κάτω.

Τι κάνει η Κλάση:

1. Διαβάζει το κωδικοποιημένο αρχείο που φτιάξαμε με τον κώδικα στο Τέταρτο Παραδοτέο.
2. Το μετατρέπει σε ένα char[] array που περιέχει 0 και 1.
3. Διαβάζουμε το tree.dat του που φτιάξαμε στο δεύτερο παραδοτέο.
4. Χρησιμοποιώντας το δέντρο και τον πίνακα των bits αποθηκεύει σε ένα String τους χαρακτήρες που αποκωδικοποιεί.
5. Τα εκτυπώνει στο αρχείο Εξόδου.

Κώδικας Παραδοτέου

Fifth.java

Μεταβλητές:

private static byte[] *byteArr*;

Για την αποθήκευση των bytes από το αρχείο εισόδου.

private static char[] *bits*;

Για την αποθήκευση των 0 και 1.

main()

```
public static void main(String[] args) throws IOException {
   if (args.length == 2) {
       try {
           //Read the bytes from the file
           readEncodedFile(args[0]);
           //Convert all the bytes to a String of 0 and 1
           bytesToBits();
           //Read the tree from tree.dat
           HuffmanNode hNode = readTreeDat();
           //Decode the bits using the Huffman Tree
           System.out.println("Decoding now...");
           String decodedText = decode(hNode);
           //Make the new output file
           makeDecodedFile(decodedText, args[1]);

          System.out.println("File " + args[1] + " is Ready");
       } catch (IOException e) {
           e.printStackTrace();
       }
     } else {
             System.out.println("Expected two words(file for
decoding and name for decoded file)");
   }
}
```

Ελέγχουμε πόσες λέξεις έδωσε ο χρήστης ως είσοδο (αποθηκευμένες στο array args) και αν είναι δύο ακριβώς συνεχίζουμε με τις κλήσεις των μεθόδων που αναφερθούν παρακάτω. Αν δεν έδωσε 2 λέξεις εκτυπώνει ανάλογο μήνυμα και τερματίζει το πρόγραμμα.

readEncodedFile()

```
private static void readEncodedFile(String encodedFile) {

    try {
        InputStream inputStream = new
FileInputStream(encodedFile);
        byteArr = inputStream.readAllBytes();
        inputStream.close();
        System.out.println("Finished reading " + encodedFile +
"!");
    } catch (IOException e) {
        System.out.println("File "+encodedFile +" not found,
exiting now!");
        System.exit(0);
    }
}
```

Η μέθοδος αυτή διαβάζει το αρχείο σε κωδικοποίηση huffman που δίνει ως είσοδο ο Χρήστης. Ανοίγουμε ένα FileInputStream για το αρχείο με όνομα **encodedFile.** Στην συνέχεια διαβάζουμε όλα τα bytes του αρχείου σε ένα array τύπου byte με όνομα byteArr. Τέλος κλείνουμε το stream.

bytesToBits()

```
private static void bytesToBits() {
  bits = new char[8 * byteArr.length];
  for (int i = 0; i < byteArr.length; i++) {
     final byte byteval = byteArr[i];
     int bytei = i << 3;
     int mask = 0x1;
     for (int j = 7; j >= 0; j--) {
        final int bitval = byteval & mask;
        if (bitval == 0) {
           bits[bytei + j] = '0';
        } else {
           bits[bytei + j] = '1';
        }
        mask <<=1;
     }
  }
}
```

Πηγή:
http://www.java2s.com/Tutorials/Java/Data_Type/Array_Convert/Convert_byte_array_to_bit_string_in_Java.htm

Ορίζουμε το μέγεθος του πίνακα bits αφού γνωρίζουμε το μέγεθος του πίνακα byteArr. Στην συνέχεια χρησιμοποιώντας bitwise operations το μετατρέπουμε σε bits.

```
private static HuffmanNode readTreeDat() { //Reads the
tree.dat file from the root folder
   try { // Java Object Serialization
       FileInputStream fis = new FileInputStream("tree.dat");
       ObjectInputStream ois = new ObjectInputStream(fis);

       HuffmanNode tree = (HuffmanNode) ois.readObject();

       ois.close(); //Closing the Streams
       fis.close();
       return tree;
   } catch (FileNotFoundException e) {
       System.out.println("File tree.dat not found, exiting
now!");
       System.exit(0);
   } catch (IOException e) {
       System.out.println("Error initializing stream");
   } catch (ClassNotFoundException e) {
       e.printStackTrace();
   }
   return null;
}
```

Χρησιμοποιούμε τον κώδικα που μας δόθηκε στην εκφώνηση του Δεύτερου παραδοτέου για το διάβασμα ενός HuffmanNode από το αρχείο tree.dat που φτιάχνει ο κώδικας που υλοποιήσαμε στο Δεύτερο Παραδοτέο κάνοντας και τους κατάλληλους ελέγχους με try-catch και το επιστρέφουμε.

decode()

```
private static String decode(HuffmanNode root) throws
IOException {
    HuffmanNode currentNode = root;
    String decoded = "";
    for (int i = 0; i < bits.length; i++) {
        if (!currentNode.isLeaf()) {
            if (bits[i] == '0') {
                currentNode = currentNode.getLeft();
            } else if (bits[i] == '1') {
                currentNode = currentNode.getRight();
            } else {
                System.out.println("error");
            }
        }
        if (currentNode.isLeaf()) {
            if (currentNode.getAscii() == -1) {
                System.out.println("Finished Decoding!");
                break;
            } else {

                decoded += (char) currentNode.getAscii();
                currentNode = root;
            }
        }
    }
    return decoded;
}
```

Η μέθοδος αυτή μετατρέπει τα bits σε κείμενο. Παίρνει ως παράμετρο το node(δέντρο) που διαβάσαμε από το αρχείο tree.dat. Δημιουργεί ένα HuffmanNode **currentNode** ώστε να κάνουμε ότι χρειάζεται στην επανάληψη χωρίς να χάνουμε την ρίζα. Σε κάθε επανάληψη ελέγχουμε αν το τωρινό node που προαναφέραμε είναι φύλο. Αν δεν είναι ελέγχουμε το index του bit αν είναι 0 ή 1 και αναλόγως αλλάζουμε την τιμή του currentNode είτε με το αριστερό είτε με το δεξί παιδί node. Αν το node είναι φύλο ελέγχουμε πρώτα αν είναι ο χαρακτήρας EOF που φτιάξαμε στο προηγούμενο παραδοτέο ωστέ να ξέρουμε ότι τελειώσαμε με την αποκωδικοποίηση του δέντρου. Αν δεν είναι προσθέτουμε τον χαρακτήρα του node σε ένα String (θα μπορούσαμε και σε κάποιο StringBuilder ή StringBuffer). Όταν φτάσουμε το EOF σταματάει η επανάληψη με ένα break και επιστρέφει το String με το κείμενο που φτιάξαμε.

makeDecodedFile()

```
private static void makeDecodedFile(String file,String fileName) throws IOException {
  PrintWriter writer = new PrintWriter(new FileWriter(fileName));
  writer.print(file);
  writer.close();
}
```

Η μέθοδος αυτή φτιάχνει το αρχείο εξόδου και εκτυπώνει μέσα σε αυτό το String που έφτιαξε η προηγούμενη μέθοδος.

Αλλαγές παλιού κώδικα:

Οι αλλαγές είναι για το **EOF** που διορθώσαμε και το **new line** που δεν δούλευε σωστά.

First.java

Όταν φτιάχναμε το αρχείο frequencies.dat βάλαμε στο τέλος να γράφει και τον έξτρα χαρακτήρα EOF.
Η μέθοδος:

```
private static void makeDatFile(int[] freq) throws IOException {
  PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("frequencies.dat")));
  for (int i=0;i<freq.length;i++) //to print all the values from 0-127
  {
    out.write(i+":"+freq[i]+",");
  }
  out.write(-1+":"+-1);
  out.close();
  System.out.println("Frequencies.dat is ready in the project folder!");
}
```

Εδώ για το new line όποτε μετατρέπαμε μια γραμμή κειμένου σε έναν πίνακα χαρακτήρων βάζουμε πλέον και το '**\n**' στην τελευταία θέση του ώστε να το μετρήσει στο "**frequencies.dat**".
Η μέθοδος:

```
private static char[] extractTextFromLine(String line){
    char[] arr = new char[line.length()+1];
    for (int i=0;i<line.length();i++)
    {
        arr[i] = line.charAt(i);
```

```
        }
        if (line.length()>0){
            arr[line.length()]= '\n';
        }

        return arr;
    }
```

Fourth.java

1. Αφαιρέσαμε την μέθοδο που είχαμε φτιάξει στο προηγούμενο παραδοτέο για την δημιουργία του EOF.
2. Στην μέθοδο EncodeInputFile βάλαμε στο τέλος ένα iterator για να βρούμε τον κωδικό του EOF και να το προσθέσει στο StringBuilder με το κωδικοποιημένο κείμενο σε 0 και 1.
   Η μέθοδος:

```
private static StringBuilder encodeInputFile(String inputFile) throws
IOException {
  BufferedReader inputFileStream = readFile(inputFile);
  if (inputFileStream == null) { //If
    System.out.println("inputFile doesn't exist");
    System.exit(0);
  }
  while ((currentLine = inputFileStream.readLine()) != null) {
    char[] arr;
    arr = currentLine.toCharArray();//To get the characters from the Line
    for (int i = 0; i < arr.length; i++) {
      String code = getCode(codeList, arr[i]); //get the code of the
character
      encodedText.append(code);
    }
  }
  for (code c:codeList) {
    if (c.getC()==-1){
      encodedText.append(c.getCode()); //EOF
      break;
    }
  }
  return encodedText;
}
```

Για το new line προσθέσαμε στο σημείο που έφτιαχνε την κωδικοποίηση σε 0 και 1 να βάζει στο τέλος κάθε γραμμής που μετέτρεπε και τον κωδικό του '**\n**'.
Η μέθοδος:

```
private static StringBuilder encodeInputFile(String inputFile) throws
IOException {
  BufferedReader inputFileStream = readFile(inputFile);
  if (inputFileStream == null) { //If
     System.out.println("inputFile doesn't exist");
     System.exit(0);
  }
  while ((currentLine = inputFileStream.readLine()) != null) {
     char[] arr =new char[currentLine.length()+1];
     for (int i = 0; i < currentLine.length(); i++) {
        String code = getCode(codeList, currentLine.charAt(i)); //get the code of
the character
        encodedText.append(code);
     }
     arr[currentLine.length()]='\n';
     String code = getCode(codeList, arr[currentLine.length()]); //get the code
of the character
     encodedText.append(code);
  }
  for (code c:codeList) {
     if (c.getC()==-1){
        encodedText.append(c.getCode()); //EOF
        break;
     }
  }
  return encodedText;
}
```

Ενδεικτικό Αποτέλεσμα

<u>Σωστή Εκτέλεση</u>

Για να τρέξει το πρόγραμμα έχουμε αφήσει μέσα στον φάκελο Project το “**tree.dat”** και ένα κωδικοποιημένο αρχείο “**output**” που φτιάξαμε στο προηγούμενο παραδοτέο. Σε περίπτωση που τα σβήσατε ή δεν υπάρχουν τρέξτε την εντολή:
“**java -cp target/Project-1.0-SNAPSHOT.jar org.hua.project.Main Files_To_Read/test_Input.txt output**” για να τρέξουμε τον κώδικα μέχρι και το Τέταρτο παραδοτέο.

Τρέχουμε την εντολή:

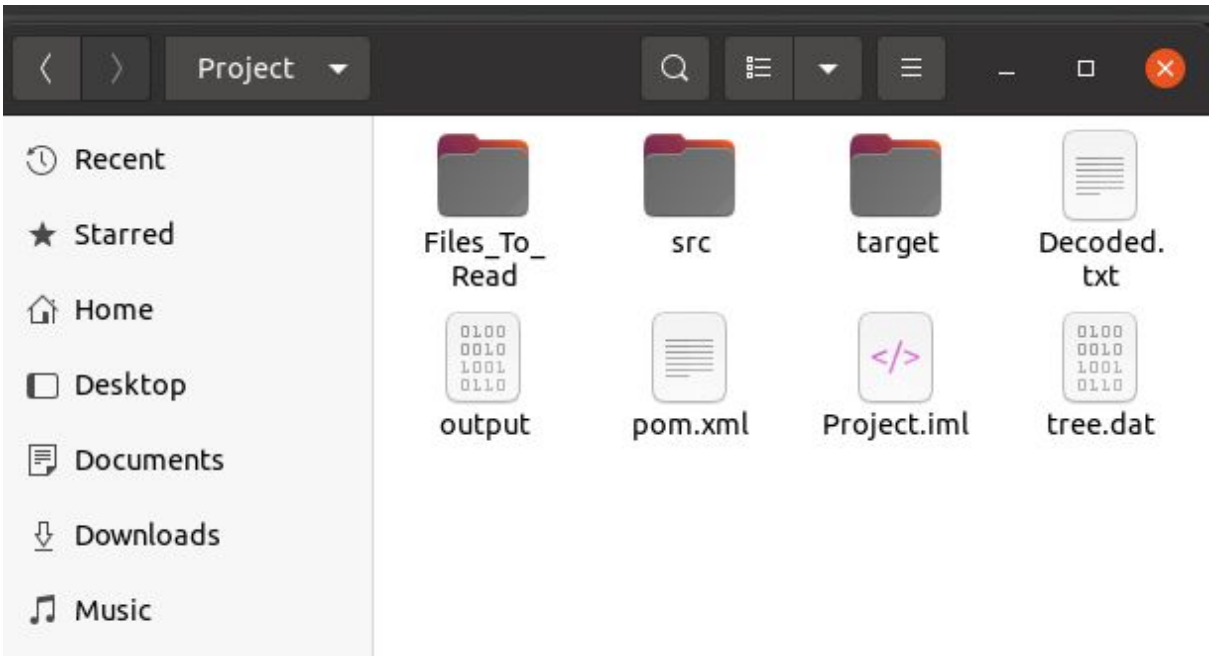
“**java -cp target/Project-1.0-SNAPSHOT.jar org.hua.project.Fifth output Decoded.txt**” για να τρέξουμε τον κώδικα του Πέμπτου παραδοτέου. Στην από πάνω εντολή δίνουμε ως είσοδο τα δύο ονόματα **output** που είναι το αρχείο που φτιάξαμε στο προηγούμενο παραδοτέο και το όνομα που θέλουμε να έχει το αρχείο εξόδου δηλαδή “**Decoded.txt**”.

```
/Project$ java -cp target/Project-1.0-SNAPSHOT.jar org.hua.project.Fifth output Decoded.txt
Finished reading output!
Decoding now...
Finished Decoding!
File Decoded.txt is Ready
```

Τρέχουμε την εντολή “**cat Decoded.txt**” για να δούμε το περιεχόμενο του αρχείου.

```
Chapter 1

It is a truth universally acknowledged, that a single man in
possession of a good fortune, must be in want of a wife.

However little known the feelings or views of such a man may be
on his first entering a neighbourhood, this truth is so well
fixed in the minds of the surrounding families, that he is
considered the rightful property of some one or other of their
daughters.

My dear Mr. Bennet, said his lady to him one day, have you
heard that Netherfield Park is let at last?

Mr. Bennet replied that he had not.

But it is, returned she; for Mrs. Long has just been here, and
she told me all about it.

Mr. Bennet made no answer.

Do you not want to know who has taken it? cried his wife
impatiently.

_You_ want to tell me, and I have no objection to hearing it.

This was invitation enough.

Why, my dear, you must know, Mrs. Long says that Netherfield is
taken by a young man of large fortune from the north of England;
that he came down on Monday in a chaise and four to see the
place, and was so much delighted with it, that he agreed with Mr.
Morris immediately; that he is to take possession before
Michaelmas, and some of his servants are to be in the house by
the end of next week.
```

Τα αρχεία που μένουν τελικά στον φάκελο μας είναι αυτά:



Λάθος Εκτέλεση

Στην περίπτωση που στον φάκελο project δεν υπάρχει το αρχείο tree.dat ή το αρχείο που δίνει ο χρήστης ως είσοδο (στο παραπάνω παράδειγμα είχε όνομα “output”) ή δεν δώσει δύο ακριβώς ονόματα στην είσοδο εμφανίζει στον χρήστη το error και σταματάει το πρόγραμμα.

Εδώ λείπει το αρχείο εισόδου από τον φάκελο “**Project**”.

```
$ java -cp target/Project-1.0-SNAPSHOT.jar org.hua.project.Fifth output Decoded.txt
File output not found, exiting now!
```

Εδώ λείπει το “**tree.dat**” από τον φάκελο “**Project**”.

```
 java -cp target/Project-1.0-SNAPSHOT.jar org.hua.project.Fifth output Decoded.txt
Finished reading output!
File tree.dat not found, exiting now!
```

Εδώ δίνουμε μόνο μία τιμή σαν είσοδο στον κώδικα.

```
/Project$ java -cp target/Project-1.0-SNAPSHOT.jar org.hua.project.Fifth output
Expected two words(file for decoding and name for decoded file)
```