CS211FZ Data Structures & Algorithms (II)

Lab 1 – Binary Trees and their Applications

Objectives

- Learn how to construct and use binary trees
- Exercise how to apply binary trees to real-world applications
- Reflect the knowledge learned in the class

NOTE:

- **Do NOT use "package" in your source code**
- **You must submit the source code files, i.e., the ".java" files.**
- **You are allowed to use course reference books or class notes during the lab.**
- **Sharing your work with others is NOT allowed.**

Task 1: Huffman Decoder

In the attached file "**Lab1_Task1_Cipher.txt**", the first line contains the Huffman codewords and the second line contains the cipher text that is encoded using the Huffman codewords. The organisation of the codewords is shown in Figure 1. Your task is to use the codewords provided in the file to decode the Huffman coded message and print the message on screen.

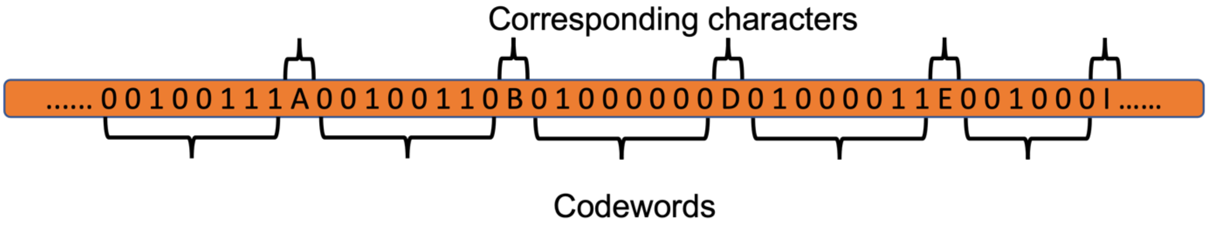
Corresponding characters
...... 0 0 1 0 0 1 1 1 A 0 0 1 0 0 1 1 0 B 0 1 0 0 0 0 0 0 D 0 1 0 0 0 0 1 1 E 0 0 1 0 0 0 0 I ......
Codewords

Figure 1. Codewords Organisation. There are no character "1" or "0" in the encoded message.

1. You must name your class file "**HuffmanDecoder.java**".

2. Construct a binary tree for the Huffman codewords. The method must be named: "**public void constructHuffmanTree(String codewords)**". Provide comment for the method to explain how the algorithm works.

3. Use the constructed binary tree to decode the cipher text and print the decoded text on screen. NOTE: you must use the binary tree to decode the cipher text. The decoding method must be named: “**public String decode(String cipher)**”. Provide comment for the method to explain how your decoding algorithm works.

4. Check your results with your teaching assistants.

Task 2: Sorting Values using Binary Search Trees

The attached file “**Lab1_Task2_RandomNumbers.txt**” contains 1000 integer numbers, separated using white space characters. There are also duplicated numbers in the file. Your task is to insert all the numbers from the file into a binary search tree, then perform in-order traversals to print the values (including the duplicated numbers) in both ascending and descending order from the same binary search tree (only one tree is allowed).

1. You must name your file “**BSTSorting.java**”.

2. You must provide a method called “**public void insert(int key)**” method for inserting values into the tree.

3. You must provide a method called “**public void sortAscending( )**” that will print values in an ascending order using in-order traversal algorithm that you have learned in the class. The implementation of the method must be a recursive method.

4. You must provide another method called “**public void sortDescending( )**” that will print the values in a descending order using in-order traversal from the same tree. The implementation must be a recursive method.