# Tree: Huffman Decoding


by vatsalchanana

Problem | Submissions | Leaderboard | Discussions | Editorial

Huffman coding assigns variable length codewords to fixed length input characters based on their frequencies. More frequent characters are assigned shorter codewords and less frequent characters are assigned longer codewords. A Huffman tree is made for the input string and characters are decoded based on their position in the tree. We add a '0' to the codeword when we move left in the binary tree and a '1' when we move right in the binary tree. We assign codes to the leaf nodes which represent the input characters.

For example :

```
         {φ,5}
      0 /     \ 1
     {φ,2}   {A,3}
    0/   \1
{B,1}  {C,1}
```

Input characters are only present on the leaves. Internal nodes have a character value of ϕ. Codewords:

```
A - 1
B - 00
C - 01
```

No codeword appears as a prefix of any other codeword. Huffman encoding is a prefix free encoding technique.

Encoded String "1001011" represents the string "ABACA"

You have to decode an encoded string using the Huffman tree.

You are given pointer to the root of the Huffman tree and a binary coded string. You need to print the actual string.

**Input Format**

You are given a function,

```
void decode_huff(node * root, string s)
{

}
```

The structure for node is defined as :

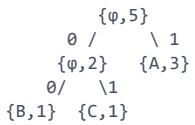
```
struct node
{
    int freq;
    char data;
    node * left;
    node * right;

}node;
```

**Note:**
Internal nodes have data='\0'(ϕ )

**Output Format**

Output the decoded string on a single line.

**Sample Input**




```
S="1001011"
```

**Sample Output**

```
ABACA
```

**Explanation**

```
S="1001011"
Processing the string from left to right.
S[0]='1' : we move to the right child of the root. We encounter a leaf node with value 'A'. We add 'A' to the decoded string.
We move back to the root.

S[1]='0' : we move to the left child.
S[2]='0' : we move to the left child. We encounter a leaf node with value 'B'. We add 'B' to the decoded string.
We move back to the root.

S[3] = '1' : we move to the right child of the root. We encounter a leaf node with value 'A'. We add 'A' to the decoded string.
We move back to the root.

S[4]='0' : we move to the left child.
S[5]='1' : we move to the right child. We encounter a leaf node with value C'. We add 'C' to the decoded string.
We move back to the root.

 S[6] = '1' : we move to the right child of the root. We encounter a leaf node with value 'A'. We add 'A' to the decoded string.
We move back to the root.

Decoded String = "ABACA"
```

Submissions:28004

Max Score:20
Difficulty: Medium

Rate This Challenge:

More

Current Buffer (saved locally, editable)

Java 8

```
/*
   class Node
      public  int frequency; // the frequency of this tree
       public  char data;
       public  Node left, right;

*/

void decode(String S ,Node root)
    {

```

```
12
13
14      }
15
```

Line: 1 Col: 1

Upload Code as File  Test against custom input  Run Code  Submit Code