Midterm Questions Study Guide (Draft!)

1. What is the asymptotic complexity of the following function? Hint: it is *not* some function of **N**.

```
int mystery(int a, int b, bool c) {
  if (a >= 2) {
    if (c) {
      return mystery(a / 2, b + 1, false);
    } else {
      return mystery(a + 1, b + 1, !c);
    }
  } else {
    return b;
  }
}
```

2. Implement the delete operation on an unbalanced BST

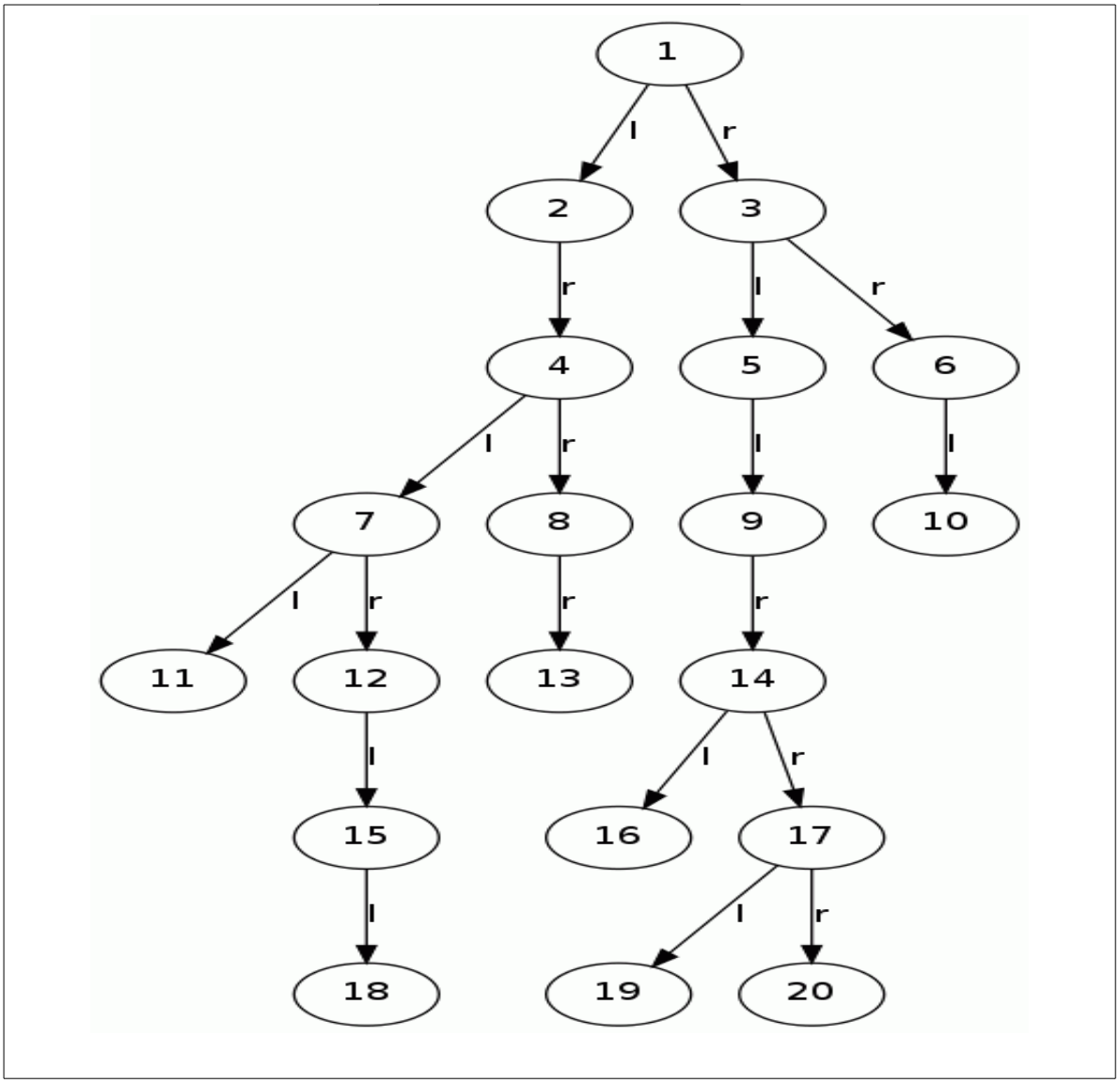
3. Implement AVL or red-black balancing for insert into a binary search tree (yes, write the code and make it work!).

4. Implement the **`verify()`** function for an AVL, red-black, or 2-3 tree. Verify is a useful debugging tool that checks whether the data structure obeys its invariants.

5. Write a function that returns the minimum and maximum lengths of a simple path from the root to each leaf in a binary search tree. You may add add additional helper functions if required. Assume the following declaration:

```
class BST {
  public:
     // your function here
  private:
    struct Node {
      static const int RED = 0;
      static const int BLACK = 1;
      int color;
      SomeType* Data;
      Node* left;
      Node* right;
    };
    Node* root;
};
```

6. Given the following tree, decorate the nodes with pre- and post-order enumerations.



7. 2-3 tree:

a) Given the following trace output, draw a bubbles-and-arrows diagram of the 2-3 tree:

b) Assume p is a pointer to a tree node containing the word “quick”, what is the value of p->middle?

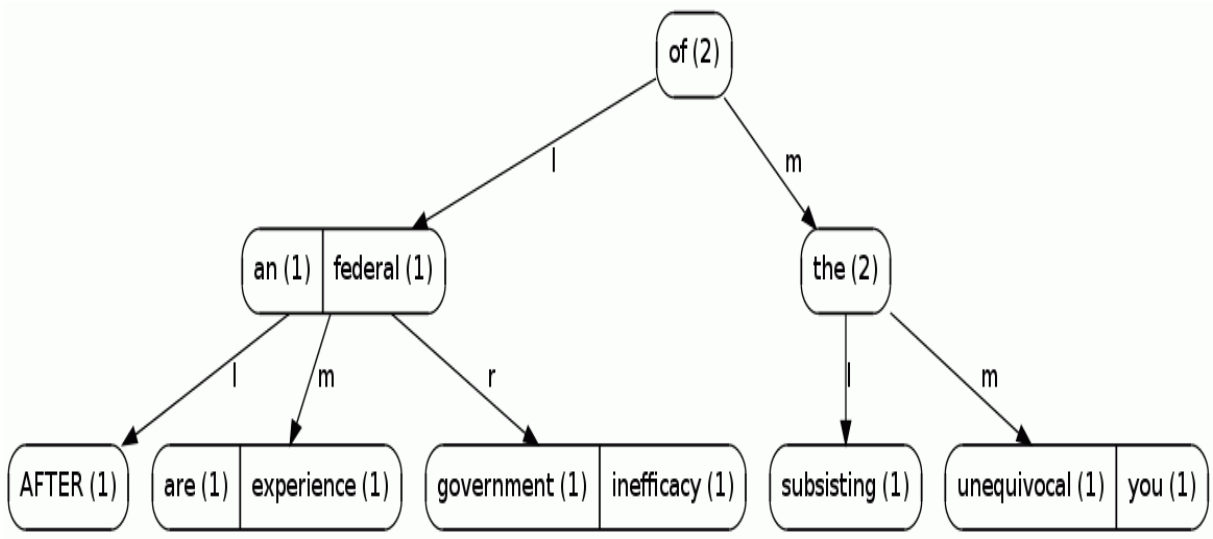
```
0x16fb280 (0x16fb100, 0x16fb250, 0)
   lazy
0x16fb100 (0x16fb040, 0x16fb190, 0)
   fox
```

```
0x16fb040 (0, 0, 0)
   brown
   dog
0x16fb190 (0, 0, 0)
   jumps
0x16fb250 (0x16fb220, 0x16fb0d0, 0)
   quick
0x16fb220 (0, 0, 0)
   over
0x16fb0d0 (0, 0, 0)
   the
```

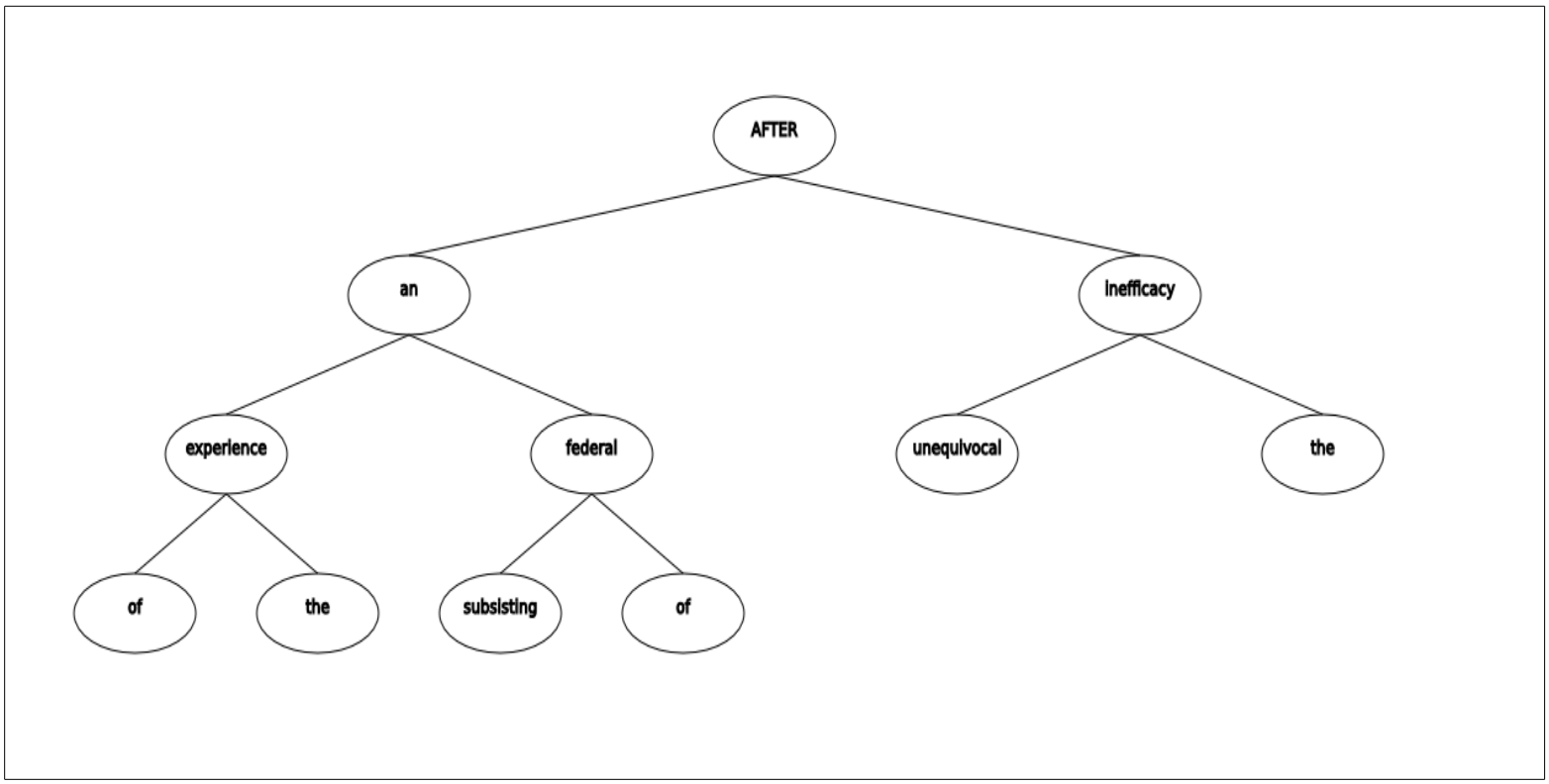
8. Given the following declaration, write a function that calculates the percentage of unused key elements (you may write additional helper functions as needed):

```
class Btree {
  public:
    // your function here
  private:
    class Node {
      Data* key1;
      Data* key2;
      Node* left;
      Node* middle;
      Node* right;
    };
    Node* root;
};
```

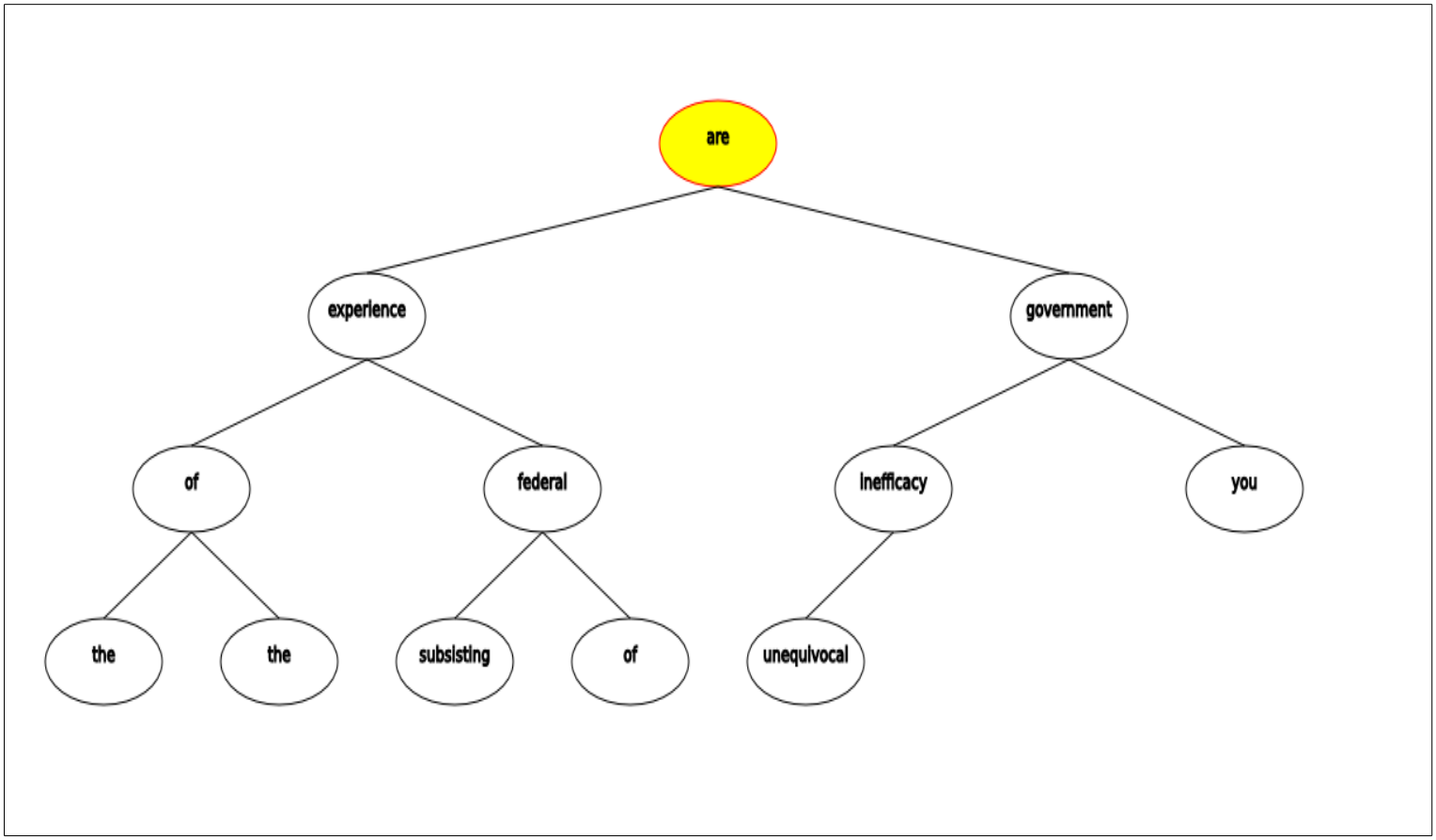
9. Redraw the following 2-3 tree after inserting the word “called”:



10. Redraw the following heap tree structure after inserting the word “government”:



11, Redraw the following heap tree after removing the minimum element:

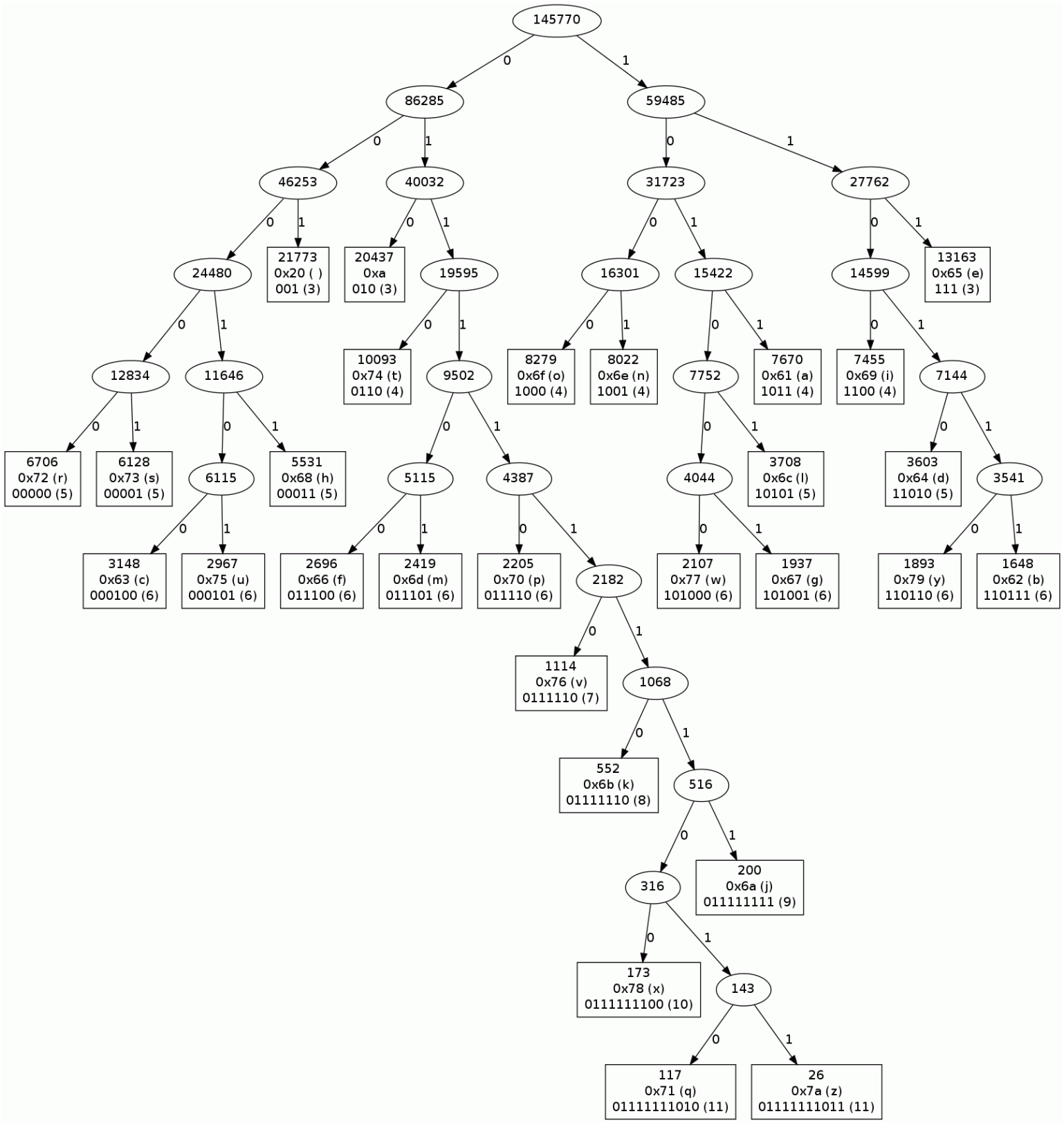


12. Given the following frequency distribution, construct the Huffman tree and encoding:

Symbol	Frequency	Encoding
A	12	
B	3	
C	32	
D	9	
E	14	
F	18	
G	3	
H	18	
I	24	
J	7	

13. Given the following Huffman tree, decode this message:

```
1010 0110 0010 0011
0100 0110 1010 0010
1000 1000 1111 1100
0110 0010 0100 1110
1101 0000 0010 1000
0000 1111 0111 1111
0010 1101 1101 0000
1
```


145770
0
1
86285
59485
0
1
0
1
46253
40032
31723
27762
0
1
0
1
0
1
24480
21773
0x20 ( )
001 (3)
20437
0xa
010 (3)
19595
16301
15422
14599
13163
0x65 (e)
111 (3)
0
1
0
1
0
1
0
1
12834
11646
10093
0x74 (t)
0110 (4)
9502
8279
0x6f (o)
1000 (4)
8022
0x6e (n)
1001 (4)
7752
7670
0x61 (a)
1011 (4)
7455
0x69 (i)
1100 (4)
7144
0
1
0
1
0
1
0
1
6706
0x72 (r)
00000 (5)
6128
0x73 (s)
00001 (5)
6115
5531
0x68 (h)
00011 (5)
5115
4387
4044
3708
0x6c (l)
10101 (5)
3603
0x64 (d)
11010 (5)
3541
0
1
0
1
0
1
0
1
3148
0x63 (c)
000100 (6)
2967
0x75 (u)
000101 (6)
2696
0x66 (f)
011100 (6)
2419
0x6d (m)
011101 (6)
2205
0x70 (p)
011110 (6)
2182
2107
0x77 (w)
101000 (6)
1937
0x67 (g)
101001 (6)
1893
0x79 (y)
110110 (6)
1648
0x62 (b)
110111 (6)
0
1
1114
0x76 (v)
0111110 (7)
1068
0
1
552
0x6b (k)
01111110 (8)
516
0
1
316
200
0x6a (j)
011111111 (9)
0
1
173
0x78 (x)
0111111100 (10)
143
0
1
117
0x71 (q)
01111111010 (11)
26
0x7a (z)
01111111011 (11)