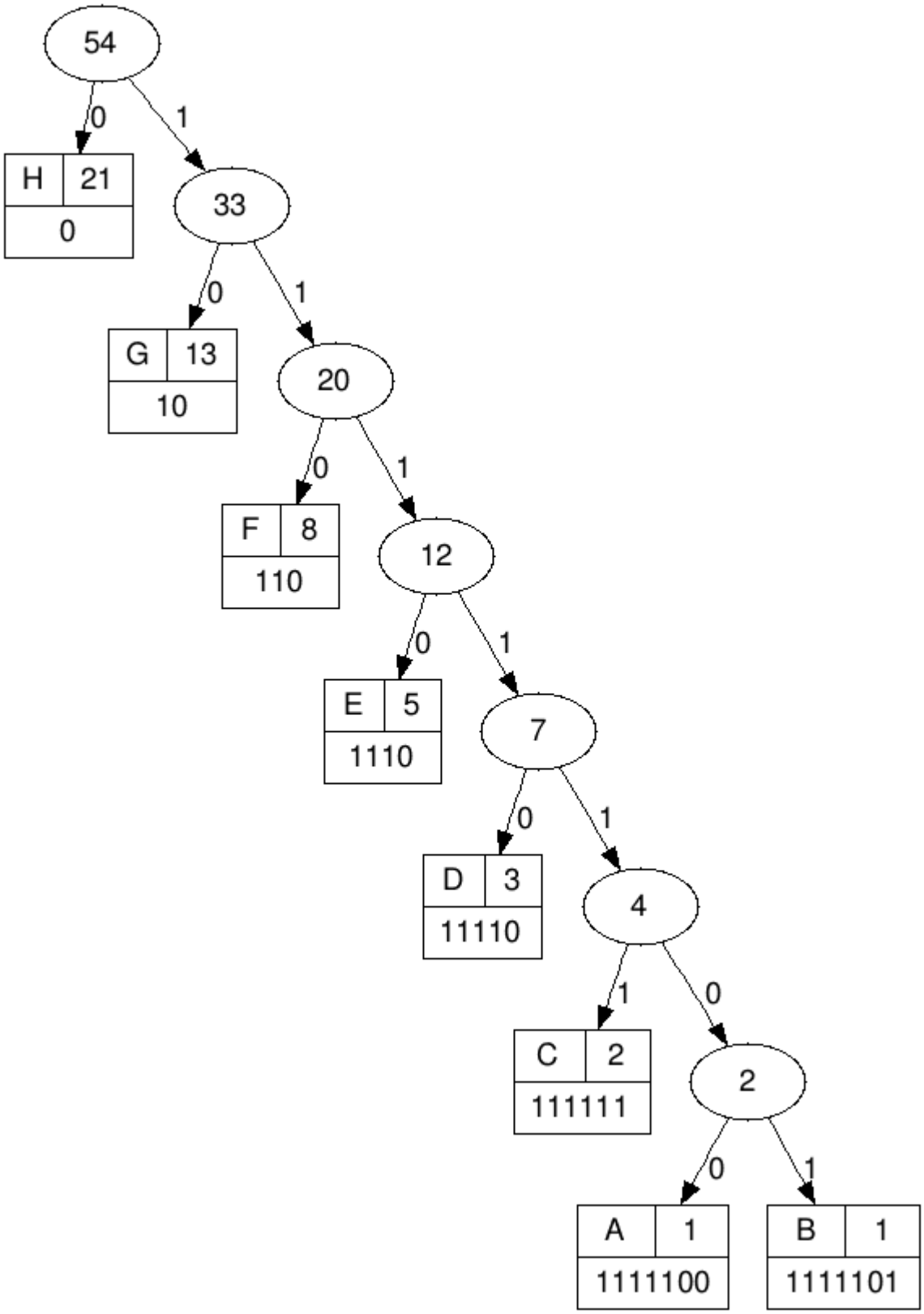
Assignment 2:

Installation

1. To run the file, on a command prompt type ‘python huffman.py’

Explanation

1. Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code
2. A Fibonacci series is a series of numbers in which each number (*Fibonacci number*) is the sum of the two preceding numbers. The simplest is the series 1, 1, 2, 3, 5, 8, etc.
3. For our given problem, we consider each number in the series to be a node, and the number from the series denotes the frequency of that particular node. For example, if our series is [1,1,2,3], this means that {a:1, b:1, c:2, d:3}. We then use this Symbol-to-Frequency table and construct our Huffman encoded tree.
4. First, we prompt the user to enter a Fibonacci series number. We cap this number to 26 because we create a map of ascii alphabets to frequencies in order to construct our Hoffman tree.
5. We then use python’s ‘heap’ library extensively to generate the Hoffman tree from a dictionary that was made by calling a Fibonacci function recursively, therefore, generalizing our program
6. The Hoffman tree for 8 Fibonacci numbers in the series is as follows (cont. on next page):


54
0
1
H
21
0
33
0
1
G
13
10
20
0
1
F
8
110
12
0
1
E
5
1110
7
0
1
D
3
11110
4
1
0
C
2
111111
2
0
1
A
1
1111100
B
1
1111101

9. As one can notice, the Fibonacci series gives a very interesting result. The shape of the Hoffman tree is skewed, that is, a Fibonacci series Hoffman code gives a skewed left or right tree.
10. The worst case for Huffman coding (or, equivalently, the longest Huffman coding for a set of characters) is when the distribution of frequencies follows the Fibonacci numbers (http://mathforum.org/kb/thread.jspa?messageID=3400647)
11. The Huffman algorithm considers the two least frequent elements recursively as the sibling leaves of maximum depth in code tree. The Fibonacci sequence (as frequencies list) is defined to satisfy F(n) + F(n+1) = F(n+2). As a consequence, the resulting tree will be the most unbalanced one, being a full binary tree.
12. This is why run length coding or other forms of compression are usually applied prior to Huffman coding. It eliminates many of the worst case scenarios for Huffman coding.
13. The output of our program is as follows:

```
Enter the end number for Fibonacci Series:
8
The fibonacci series is [1, 1, 2, 3, 5, 8, 13, 21]

The Symbol<->Frequency dictionary for encoding is as follows: defaultdict(<type 'int'>, {'a': 1, 'c': 2, 'b': 1, 'e': 5, 'd': 3, 'g': 13, 'f': 8, 'h': 21})

The Huffman Encoding is as follows:

Symbol    Weight    Huffman Code
h         21        0
g         13        10
f         8         110
e         5         1110
d         3         11110
c         2         111111
a         1         1111100
b         1         1111101

Process finished with exit code 0
```