DS2 Theoretical HW #4

1. A = 10

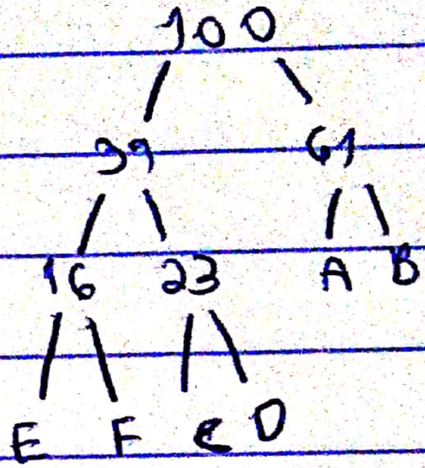
B = 11

C = 010

D = 011

E = 000

F = 001



$$\text{compression rate} = \frac{(2\cdot31 + 2\cdot30 + 3\cdot13 + 3\cdot10 + 3\cdot9 + 3\cdot7)}{(3\cdot31 + 3\cdot30 + 3\cdot13 + 3\cdot10 + 3\cdot9 + 3\cdot7)} = 0.7967$$

Question 2

A full binary tree (sometimes proper binary tree or 2-tree) is a tree in which every node other than the leaves has two children. Suppose we have an optimal prefix code which is not represented by a full tree. That implies that there is at least one node in the tree which has only one child. If were to remove all those nodes from the tree and replace them with each of their only children, the tree would still represent the same characters, but the characters which were under the deleted nodes are now all represented by at least one bit less. This shows that our 'optimal' prefix code could be shortened, and therefore was not 'optimal' beforehand. Thus we have proven by contradiction that if a prefix code is optimal, it must be full.

Question 3a

When n is a power of 2, a Huffman tree encoding n characters will be perfect.

Question 3b

When n is not a power of 2, a Huffman tree encoding n characters will be a full binary tree with the absolute value of the difference between the height of any two leaves will be at most 1.

Question 4

If the frequencies of the letters L1 through Ln satisfy the condition $f1 > f2 > ... > f n > f1 /2$, then in the process of building the Huffman tree, the priority queue will initially be [L1 ,L2 , ... , Ln], with the highest priority being Ln and the lowest being L1 . Therefore, the first time we take the two letters with the highest priority, we will take Ln and Ln-1 . They will both be put beneath a new node with frequency f n+f n-1 . Since $f n > f1 /2$, and $f n-1 > f n$ are both true, therefore $f n-1 > f1 /2$ is also true. Therefore $f n+f n-1 > f1$. This means that the new node will be put at the back of the priority queue. For all the other nodes in the priority queue, since the frequencies of the next two nodes to be taken from the queue are both larger than the frequencies of the previous two nodes that were taken from the priority queue, their sum is also larger than the sum of the previous two. Therefore all subsequent two nodes will be put under a new node and inserted into the back of the priority queue. In order to end up with a Huffman tree containing leaves which are at least 2 levels apart, it must occur that two subtrees which have a height difference of at least 2 are joined under a new node. However since we have shown that every new node will always be put at the back of the priority queue, and every new subtree has a height of one plus the height of the highest subtree it was made of, it follows that any two adjacent nodes have a height difference no greater than 1. Therefore it will never occur that two subtrees with a height difference of 2 or greater are joined, so all the leaves must be on the same level or on two consecutive levels.

Question 5a

If the code is not a prefix code, then the code for at least one of the characters is the prefix of another. Since in encoding trees, codes are represented by paths from the root, such a tree will have a character node with at least one child.

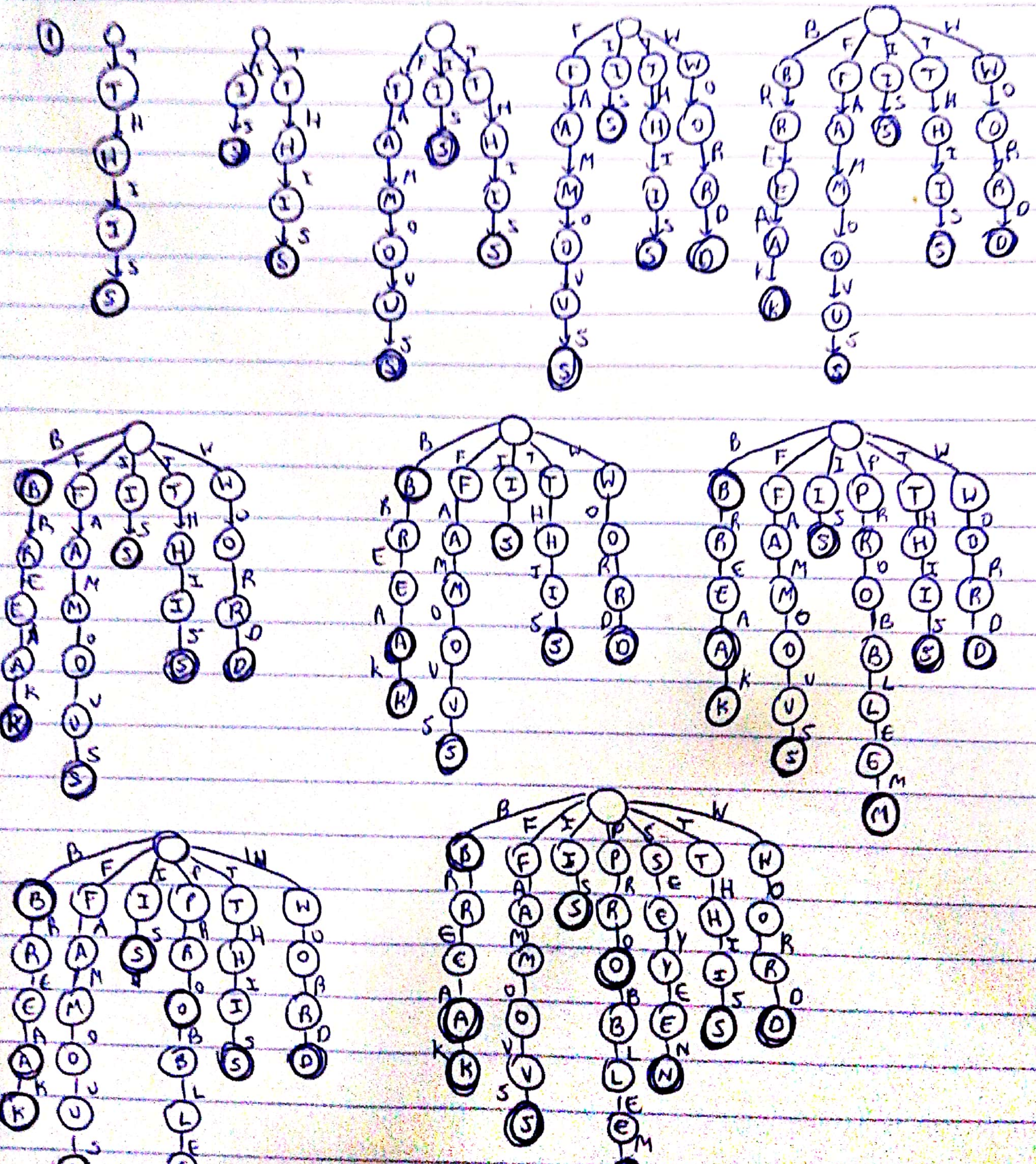
Question 6b

Since the frequency of L1 is equal to the sum of all other frequencies, then every time that two nodes are joined from the priority queue, their summed frequencies will still be less than that of L1 , unless it is the only other node remaining in the queue. That means that all other nodes besides L1 will be joined into a subtree, and then L1 will be joined to the subtree containing all the other nodes. Therefore a tree with such a property must have a leaf with a height of 1.

Question 5c

The frequencies in this tree are the Fibonacci sequence with each term multiplied by f1 . Since each two consecutive frequencies have a sum equal to the next frequency in the queue, it will at every stage join a subtree to a single leaf. Therefore such a tree will have a structure such that each non-leaf node will have one child which is a leaf and one child which is not a leaf. The last non-leaf node is an exception and will instead have two children which are leaves.

1



Question 2

A trie can represent a set of strings where each string may appear multiple times, but only if the structure is modified slightly. Instead of each node having a boolean property which indicates if the node is the end of a word, it can instead contain an integer property which will be the number of words that end in that spot. For example, if the node is not an end of word, said property will contain the value zero. If it is the end of a word, and that word only appears once, the value it contains will be one. If there are n copies of that word, the terminal node will have the value n in the aforementioned property

```
③ list< string> insert_spaces (string s, const trie & words)
{     list <string> sentences;
      string word= "";
      for(int i=0; i<s.size(); ++i)
       { word += s[i];
         if(!words.has_path(word))
          { break; }
         if(words.has_word(word))
         { list<string> rest = insert_spaces (s.substr(i+1, s.size()), words);
            for(auto i= rest.begin(); i!= rest.end(); ++i)
             { sentences.insert (word + ' ' + *i);
         }
       }}
      return sentences; }
```