Ali At-Tayeb

27732929

Assignment 3 - Comp 352

**1)**

**c)** My tree structure is quite simple. It has an internal class node that can connect to two nodes and it can hold an integer value. This internal class implements the interface base node. I made an internal class leaf that also implements the interface so we can easily store the internal nodes and leafs within the same array of base nodes. There is also a default constructor, get and set method for the root of the tree. My base class is quite simple as I did not find many similarities between the Huffman tree and the splay tree. One is a general tree, while the other is a binary search tree.

**e)** My Huffman class utilizes the root implemented in the basic tree class. It also uses the node class from the basic tree to initialize the internal nodes. A leaf class is created internally to store leafs in the Huffman tree. I also use a quicksort class to sort the character by frequency and a hash class to distribute the frequencies to their respective characters.

**g)** Similarly, I use the root implemented in the basic tree to build my splay tree. Since the class had all the node components, I used them and implemented the splay methods using the node class from the basic tree.

**h)** I decided to implement a splay tree since we are continuously doing the same three actions. That being said, the splay tree structure guarantees an average log n time per access operation. Since keeping track of the height and having a balanced tree are not a priority, the splay tree seems like a better option. Also, if it happens that a certain value is accessed often, it will be quicker to access after the first time.

**4)**

**a)**

Given string: will the accessible romance coast?

Encoded: 01001110110111001110001011100111100010001010011010011110111101111011011011110111011100010101011001010100011010101001111000101001011010001111010110101100011

The encoded version requires 155 bits. Traditional ASCII coding requires 8 bits per character. Since the given function contains 34 characters, it would require 272 bits. That being said, the Huffman coding requires only 56% of the bits that ASCII coding would demand. I believe that the encoded string did match the frequency, as most of the characters included are high in the frequency table. Other than, c and the interrogation point, all the other characters appear quite frequently.

**b)**

Comparisons: 28118

Parent change: 140775

Find: 1000

Add: 1286

Remove: 391

I believe I made the right decision of implementing a splay tree. There is a total of 2677 operations. Dividing the number of comparisons by the number of operations, we get approximately 10.5. If I implemented an AVL tree, with 2677 operations, the height would roughly be 11 (log 2677). That being said, the proportion of comparison to the number of elements is roughly the same as the height. The splay had approximately the same performance and operations were saved by not calculating the height difference every time.

For future designs, I would spend more time on the design portion before starting to program. I spent an immense amount of time implementing a binary search tree delete function, to finally realize that the splay trees deletion process is much simpler.