*ASSUMPTIONS PAGE*

**BINARY TREE LOGIC**

Binary Trees are a type of graph that separates each node into two branches that each contain subtrees. In these types of graphs, each node has at most two children (left and right). Binary search trees are a type of binary tree in which the value of the left child is less than the parents and the value of the right child is greater than the parents. These trees can be complete, full, both, or neither. This program will make use of binary search trees to encode/decode Morse code.

**PROGRAM LOGIC**

The program will use two different trees to create the binary search tree: a typical binary tree, and the binary search tree node which will actually create the binary search tree to help decoded the Morse code. The tree will place a single dot on the left side and a single dash on the right side. The program will print the tree as given on its side. It will display when the user has entered the string to be encoded/decoded. The user will manually change the string in the main program. They will then uncomment the section the code that they are looking to use to have the encoding/decoding displayed to the screen. Everything is created in the classes and a class object is used in the main.

**PROGRAM ASSUMPTIONS**

This program will assume that the user will input a string. that are delimited by spaces . Due to this, the program will only accept strings as a parameter, not characters. Only one word is decoded/encoded at a time.

The user does not need to worry about case or space since the program will account for both options.

Though the program will simply display an error message and end if the user accidentally inputs

Anything but a character for encoding and anything but . or \_ for the morse code. We can assume that the user is aware there is no Morse code value saved for numbers(though the program will account for that if tried).

The program will assume that the user will input the string needing to be decoded with spaces as the delimiter (i.e. The user will insert “.\_(\*space\*)..\_” for decoding). This will ensure there is no confusion between characters that are similar.

The program will assume that the user will be using a period (.) to represent the dots in the Morse code instead of the bullet point (·). An error will be thrown if an unrecognized character is given and the program will terminate.

The program will assume the user will only be attempting to decode only one single word

*ALGORITHM EFFICIENCY*

**BST**

* insert(BTNode\*& node, string morse, string alph)🡪 in general has a complexity of O(logn), worst case has complexity of O(n).
  + To make this complexity better, you could use a linked list which has better insertion time O(1), however, using a BST has a better search (which will be used more frequently.) This would be the better option

**Create\_Dictionary**

* Create\_dictionary::add\_To\_Dict(string alph, string morse\_char)🡪 typically O(1) complexity.

**Conversion**

* void retrieve\_File\_Info()🡪 worst case O(n2), but I believe this one is O(nlogn) since it is iterating through the file and inserting to binary tree.
* void print\_BST()🡪 has O(n) complexity since it is iterating through to add the spaces. If I wanted to improve the complexity, I could just opt to only have the characters appear on a single line and remove my for loop. This would reduce it to O(1)
* string find(BTNode \*&node, string input)🡪 has complexity O(logn). If I wanted to improve this, I could use a hash tree instead of creating a binary tree which has a complexity of O(1), this would not be incredibly helpful for creating a binary search tree, however, so this is ideal for what we are doing.
* string encoder(string input)🡪 worst case is O(n3) since it has to iterate three different times (including using .erase() ). It is this high due to error handling. There isn’t too much I could do to reduce it since this is mostly error handling. Disregarding the error handling it has complexity O(n)
* string decoder(string input)🡪complexity of O(nlogn), worst case O(n2) due to iterating through a string and having to search through binary tree. This could possibly be shortend slightly by only using a dictionary and decoding this way.

*REFERENCES*

Succinct Encoding of Binary Trees. (n.d.). Retrieved July, 2020, from <https://www.geeksforgeeks.org/succinct-encoding-of-binary-tree/>

Virtual Functions (n.d.) Retrieved July, 2020

<https://www.geeksforgeeks.org/virtual-function-cpp/>

Print fill Nodes Binary Tree. (n.d.) Retrieved July

<https://www.geeksforgeeks.org/print-full-nodes-binary-tree/>

Data Structures: Binary Search Trees (n.d.) Retrieved July,

<https://www.tutorialspoint.com/data_structures_algorithms/binary_search_tree.htm>

Stack Exchange(n.d) (used a lot for various topics) Retrieved July, 2020 from

<https://stackexchange.com/>

Know thy complexities!(n.d.) Retrieved July, 2020 from

<https://www.bigocheatsheet.com/>

UML Class Diagrams Explained with Examples, June 13, 2019. Retrieved July, 2020 from

<https://creately.com/blog/diagrams/class-diagram-relationships/>

**GITHUB URL**

<https://github.com/n-edmond/Morse_Code>