TEAM PROJECT 3

Project 3B – Morse Code

Luke Janis

Dawson Ploudre

Reece Gillam

Garrett Morgan

MARCH 8TH, 2023

TABLE OF CONTENTS

1. SYSTEM DESIGN 3

2. UML DIAGRAM 4

3. TEST CASES 5

4. TEAM MEMBER CONTRIBUTION 6

5. POSSIBLE IMPROVEMENTS 6

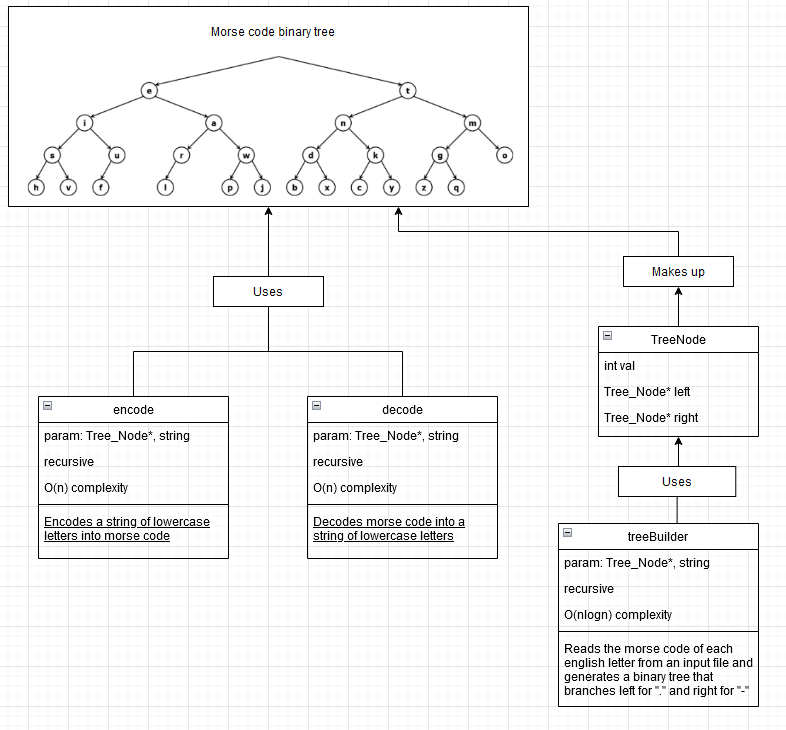
1. SYSTEM DESIGN

Treebuilder uses a version of the common binary tree node to build a binary tree where each node holds a value for each letter of the alphabet as an integer. The tree is organized through morse code where a left branch means a dot while the right branch would be equivalent to a dash. Depending on the morse code of a character, a series of left and right branches were taken to represent the code and the value for the character would be stored at the unique locations.

Encode is a wrapper function that appends together all the morse code sequences returned to it from the findchar function by iterating through the user input string until the string has been fully encoded. The findchar functions find each character one at a time through pre-order traversal and building the equivalent morse code depending on the branches taken to get to the character upon finding the letter within the binary tree. The morse code data is a string because it holds a sequence of dots and dashes without knowing the size of the final encoded data and the function append made it possible to continually add encoded characters to the rest of the encoded string making it all one object in the end.

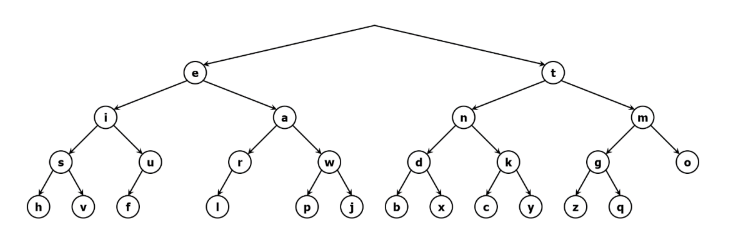
Decode breaks down the user’s input morse code to each sequence of dots and dashes that corresponds to a letter and processes the string down taking left and right branches as indicated by the next character in the sequence until there is nothing but the but the character at the node that was indicated by the sequence of dots and dashes. Strings are used to pass the whole length of the morse code through a recursive function to make use of the erase function and follow the morse code until reaches the end where the function can output the character at the corresponding location.

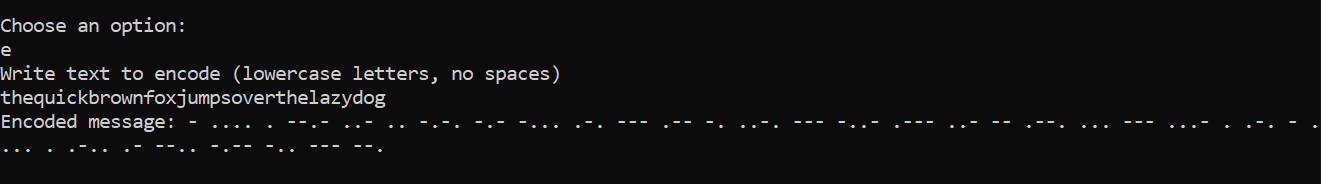
2. UML DIAGRAM



3. TEST CASES

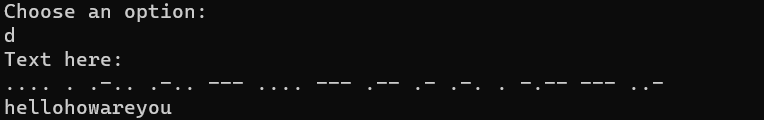
1. First, let’s take a look at a sample input for our encode function. Our sample input will be the string “thequickbrownfoxjumpsoverthelazydog”, a string which contains every letter in the alphabet. We should expect the function to return a string with each letter in morse code, in the same order as the original input string, separated by spaces. We should expect the morse code to line up with our binary tree used to represent morse code which is also seen below.





As we can see, the function correctly encoded the string. Each letter is encoded correctly, using our binary tree, where each right is represented with a ‘-’ and each left is represented using a ‘.’.

1. Next, let’s take a look at our decode function. We will use the sample input “.... . .-.. .-.. --- .... --- .-- .- .-. . -.-- --- ..-”, which in decoded form should be “hellohowareyou”. We should expect the function to return the string “hellohowareyou”.



As we can see, the function correctly processed our encoded message and returned the decoded string “hellohowareyou”. From this, we can see that our function correctly traversed our binary tree, using a ‘.’ to represent a left and a ‘-’ to represent a right, which allowed it to correctly decode our encoded string. We can also see that it correctly recognized that spaces were used to separate each character in the encoded string, which allowed it to process each character effectively.

4. TEAM MEMBER CONTRIBUTION

Luke Janis – Hosted repository and created the basic shell of the program. Completed the treeBuilder function and the UML diagram in the project report.

Dawson Ploudre – Worked on the decoding function with Garrett. Completed the Test Cases for the project report.

Reece Gillam – Made the encode function and wrote the system design section.

Garrett Morgan – Worked on the decoding function with Dawson. Added possible improvements to the project report.

5. POSSIBLE IMPROVEMENTS

* We could’ve made an extra class and header file to separate the code. It looks a little messy all in one file, if we made the class and header file we could find what we are searching for more efficiently.
* Some areas of the code could have been more simplistic to make the code look better and more efficient.
* We could’ve made an input file and a sample output file so we wouldn’t have to type to test repeatedly.
* We could include error checking. Like if someone chooses to decode and inputs letters.
* Made sure that there was not much extra data movement caused using string functions.