Lab 13 - Morse code

Weichen Chai

Spring 2023

Introduction

This paper seeks to answer the questions presented in"The last assignment
" for ID1019. In this assignment, we will explore how the functional language Elixir can be utilized to solve the morse code encoding and decoding problem, working with processing of tuples, lists and trees.

1. Morse code

Morse code differs from Huffman coding in that it employs shorter sequences of symbols for frequently used characters and longer sequences for less commonly used characters. However, what sets Morse code apart is that it uses pauses as a signal to indicate when one character ends and another begins, making it easier to decode the message compared to Huffman coding.

Encoding in the given Morse tree is quite expensive time wise so another encoding method should be utilized.

0.1 Encoding

Starting with the first module for Huffman, we begin by using the code given under this section and filling in the blanks.

```
def encode_table() do
    tree = morse()
    Enum.sort(codes(tree, []))
    //sorts output of tree place them in alphabeticall order
 end


def codes({:node, ascii, a, b}, list) do
 //lookup table of ASCII characters and morse code sequences
  left = codes(a, list ++ [ ?-])
  right = codes(b, list ++ [ ?.])
  [{ascii, list}] ++ left ++ right
  //node character and morse code.
end
def codes( ascii, code) do
  case ascii do
     nil ->     [ ]
     //in case of empty
     _   ->  [{ascii, code}]
  end
 end
```

The characters for ".", "-" are represented in Elixir with ?., ?- for ASCII

Then we simply use the encoder for this table, shown in class, it should have a complexity of O(n m) where n is the length of the message and m is the length of the Morse codes

```
def encode(text) do
    table = encode_table()
    encode(text, table)
  end
  ...
```

After running the code, I ended up with

```
.-- . .. -.-. .... . -. -.-. .... .- ..
```

When entering my name:"weichenchai"

0.2 Decoding

To meet the requirement of having a decode data structure with a search complexity of O(m), where m is the length of Morse codes, the Morse code tree provided is a suitable structure. This is because each dot or dash in the Morse code only requires one step down the tree, and the space indicates that the corresponding character has been reached.

The decoding process involves traversing down the Morse code tree while reading the Morse codes. When a dot is encountered, the decoder goes down the left side of the tree, and when a dash is encountered, it goes down the right side. When a space is encountered, the decoder stops and retrieves the corresponding character.

```
def decode(code) do
  decode(code, morse())
end

def decode([], _, _) do  []  end
//base case

def decode([char|rest], {:node, t, right, left}) do
  case char do
    ?. -> decode(rest, left)

    ?- -> decode(rest, right)

    ?\s -> [t| decode(rest, morse())]
  end
end
```

The answer to the question given to us at the start for decryption the Morse code then gives the answers:

all your base are belong to us,

The second answer is a YouTube link to ”Never gonna give you up”

```
https://www.youtube.com/watch?v=dQw4w9WgXcQ
```