Josh Barber Assignment 9 March 19, 2023

Problem 1:

Pseudocode:

```
Greedy Algorithm (A):
denominations = [0.25, 0.1, 0.05, 0.01]
while i < length of denominations
        while A >= denominations[i]
                A -= denominations
        i += 1
```

We can be greedy in assuming we can use the highest denomination and continuously subtract it from the value 'A'. If the largest denomination doesn't fit in the value, switch to the next largest, until all denominations have been used. Time Complexity is O(A).

Problem 2:

Pseudo code:

```
Huffman (Symbol Frequency Object as SFO for short):
    heap = Heapify (SFO)
    while heap length > 1
        left = pop heap (smallest frequency)
        right = pop heap (smallest frequency)
        left.code = 0; right.code = 1
        create node Z = (left.frequency + right.frequency, left=left, right=right
                                                    symbol = None)
        heap.push(Z)
    return heap
```

```
PrefixEncode(message, heap):
    codes = StoreCodes(heap)
    encodedMessage = None
    for letter in message:
        encodedMessage += codes[letter]
    return encodedMessage

StoreCodes(node, value)
    newValue = value + node.code
    if (node.left)
        StoreCodes(node.left, newValue)
    if (node.right)
        StoreCodes(node.right, newValue)

    if not node.left and not node.right
        codes[node.symbol] = newValue

PrefixDecode(message, heap)
    decodedMessage = None
    for number in message
        if number = 0
            heap = heap.left
        else
            heap = heap.right
        if heap.left and heap.right is None
            decodedMessage += heap.symbol
    return decodedMessage
```

Huffman Algorithm runs in $O(n \log n)$ time. The storing codes algorithm runs in $O(n)$ time, in which PrefixEncode will run in $O(|n| + |v|)$ time, where v is every letter in the message. PrefixDecode will run in $O(|e|)$ time, where it will back track each node to decode, where e is the length of encoded message.

Each letter in the message will get its own unique code depending on its frequency. This helps compress the message, as 8 bits per letter isn't necessary.