Algorithms Project 2 -- Marshall Thompson

1. Implementation Specifics

Inputs

It is assumed that there are two lists given as input. First, a set of symbols. Second, is the corresponding frequencies of occurence for those symbols.

Implementation

Approach

My approach relies on a min heap to achieve optimality. A min heap allows for easy retrieval of the minium element of the set. This is necessary as a greedy approach to constructing Huffman trees relies on pairing the two symbols with minimum probability. Furthermore, when inserting nodes back into the heap the order is maintained, so further removals from the heap remain in the proper order.

The minimum elements are disireable for creating an optimal Huffman Coding as we want the symbols with the greatest probability to appear near the top of the tree and thus have a shorter encoded length. This is true of pairs of symbols as well which are created in the Huffman Encoding Algorithm. Symbols with greater probability will be paired later in the algorithm and thus be higher in the tree.

A heap is optimal in finding the minimum (O(logn)) over brute force and unordered binary tree (O(n)).

For my implementation I used the Python heapq library for a min heap, and I created my own Node class for ease of use and comparison. The Node class can be seen below

PseudoCode

```
class Node:
    symbol,
    freq,
    left,
    right,
    huffman_code

    # overrides < operator
    def overridePythonLessThan(otherNode)
        return this.freq < otherNode.freq

def huffman(syms, freqs):
    huffman_heap = Heap

    n = len(syms)
    for i=0-->n:
        add (syms[i], freq[i]) to huffman_heap

    while len(huffman_heap) > 1:
```

```
        left_node = huffman_heap.pop()
        right_node = huffman_heap.pop()

        left_node.huffman_code = 0
        right_node.huffman_code = 1

        new_freq = left_node.freq + right_node.freq

        new_node = Node(new_freq, left=left_node, right=right_node)
```

2. Numerical Results

Table	Graph

	Experimental	Theoretical
0	170000	0
1	8000	2356
2	12000	9426
3	32000	28280
4	79000	75414
5	182000	188536
6	406000	452486
7	907000	1055801
8	1961000	2413261
9	4177000	5429837
10	9098000	12066306
11	64434000	26545873
12	42489000	57918269
13	91141000	125489583
14	239631000	270285257
15	463356000	579182693
16	1017514000	1235589746
17	2298547000	2625628211
18	4935764000	5560153859
19	10405186000	11738102592