**Advanced Data Structures (COP 5536)**

**Spring 2017**

**Programming Project Report**

Lakshmi Saranya Kalidindi

80946311

skalidindi@ufl.edu

This project is comprised of 3 files that has multiple classes in each file.

Compiling Instructions:-

1. Adsproject

Compile = Javac Adsproject.java

Run = java Adsproject

1. Encoder

Compile = javac encoder.java

Run = java encoder <inputFile>

1. Decoder

Compile = javac decoder.java

Run = java decoder <encoded.bin> <codetable.txt>

**Function prototypes**

1. Adsproject

This file was created to test and analyze which priority queue structure is best in performance. The three structures, binary heap, 4 way cache optimized heap and pairing heap were implemented and is run 10 times to build a Huffman tree. This file is again run 5 times to get the average best time. I found out that four way heap with cache optimization came out to be the better. In one instance, four way heap took 6235 milliseconds, pairing heap took 8740 milliseconds and binary heap took 6826 milliseconds.

1. Encoder

Since I got four way heap to be the better performing data structure, Huffman tree is generated using this priority queue structure. This file needs an input file that can have up to 100,000,000 lines and each line to have decimal values in the range of 0 to 999,999. The output of this file would be 2 files, encoded.bin and code\_table.txt and print statement that gives the time required to run the encoder on that input file.

1. Decoder

This file takes 2 files, encoded.bin and code\_table.txt, as input and gives an output in a file named decoded.txt. The aim is to ensure that the file decoded.txt should be exactly same as the input given for the encoder. The first step is to create a decoder tree from the code table given as input. Then print the output file, converting the bytes from encoded.bin with the help of the decoder tree.

**Performance analysis:**

It was found out that the least running time to build a Huffman tree was taken by the four way cache optimized heap. Below is a detailed analysis of 3 of the runs:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Four way heap** | **Binary heap** | **Pairing heap** |
| **Run 1** | 4738 | 5444 | 8728 |
| **Run 2** | 4235 | 4826 | 8140 |
| **Run 3** | 4840 | 4961 | 9002 |

Pairing heap takes the largest time since the tree can grow to any length. Also while extracting the minimum, all the children should be combined by a complex procedure called 2 pass scheme which supposedly gives better performance. Even though it is easier to implement, this queue structure did not show optimum performance. The four way heap takes lesser time than the other binary heap, though by a small difference. Even though it would seem like there would be more number of comparisons in four way heap, it would have to traverse a tree with smaller height in four way, (since log4n is smaller than log2n).

**Decoding algorithm:**