Meeta Marathe
November 10, 2017
PA3

Report

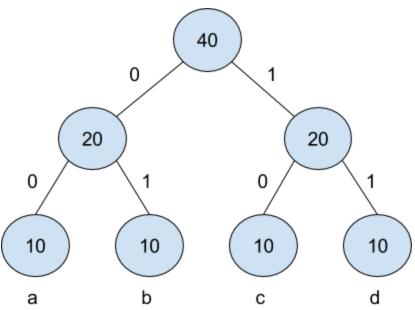**check1.txt**

Running compress.cpp:

- Header
    - line 98 10
    - line 99 10
    - line 100 10
    - line 101 10
- Encoded string
    - 00011011000110110001101100011011000110110001101100011011000110110001101100
    01101100011011

Manually building Huffman coding tree:

- Since there are four non-zero frequencies, there are four leaf nodes: a, b, c, and d
    - All have a frequency of 10
    - In the min heap, they are organized with the lowest letter at the top and the
    highest letter at the bottom because they have the same frequency
- a and b are popped first, and they get a parent node with frequency 20 that is put into the
heap
    - a is child 0, and b is child 1
- c and d are popped next, and they get a parent node with frequency 20 that is put into the
heap
    - c is child 0, and d is child 1
- The parents of the pairs of children are popped last, and their parent is created with
frequency 40
    - Parent of a and b is child 0, and parent of c and d is child 1
    - The new parent becomes the root

- Tree:

- Codes of letters
  - The codes are formed through starting at the root and going down the tree until we reach a leaf node, keeping track of the 0's and 1's we pass by.
  - a: 00
  - b: 01
  - c: 10
  - d: 11
- Encoded string of check1.txt (abcdabcdabcdabcdabcdabcdabcdabcdabcdabcd)
  - 0001101100011011000110110001101100011011000110110001101100011011000110110001101100011011
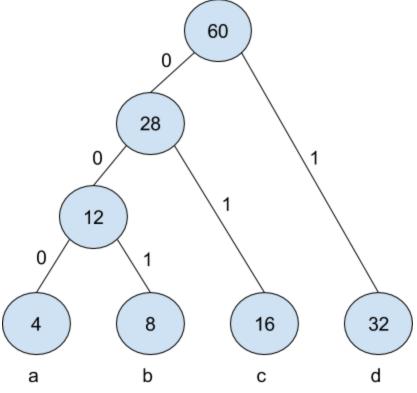 - Same as output of compress.cpp, success!

check2.txt

Running compress.cpp:
- Header
  - line 98 4
  - line 99 8
  - line 100 16
  - line 101 32
- Encoded string

- ○ 0000000010010010010101010101010101011111111111111111111111111111111111
  01010101010101001001001001000000

Manually building Huffman coding tree:
- Since there are four non-zero frequencies, there are four leaf nodes: a, b, c, and d
  - ○ a has frequency 4, b has frequency 8, c has frequency 16, and d has frequency 32
  - ○ In the min heap, they are organized with the lowest frequency at the top and the highest frequency at the top
- a and b are popped first, and they get a parent node with frequency 12 that is put into the heap
  - ○ a is child 0, and b is child 1
- The parent of a and b and the node c are popped next, and they get a parent node with frequency 28 that is put into the heap
  - ○ The parent of a and b is child 0, and c is child 1
- The parent of c and the node d are popped next, and their parent is created with frequency 60
  - ○ Parent of c is child 0, and d is child 1
  - ○ The new parent becomes the root
- Tree:

- Codes of letters

- ○ The codes are formed through starting at the root and going down the tree until we reach a leaf node, keeping track of the 0's and 1's we pass by.
  - ○ a: 000
  - ○ b: 001
  - ○ c: 01
  - ○ d: 1
- ● Encoded string of check2.txt (aabbbbccccccccddddddddddddddddddddddddddddddddcccccccccbbbbaa)
  - ○ 000000001001001001010101010101010111111111111111111111111111111111 0101010101010101001001001001000000
  - ○ Same as output of compress.cpp, success!

File Header

In my file header, I only printed out the non-zero frequencies of the symbols. I print the ASCII value and a space and then the frequency to keep track of which symbol has which frequency. For files that don't use as many characters, this significantly reduces the space for the header. For example, in check1.txt and check2.txt, instead of using 4 * 256 = 1024 bytes for the header, it only uses 4 * 2 * 4 = 32 bytes for the header because there are only four symbols in the files. However, if many different characters are used, this method will have around the same space efficiency of the original header.