Alexis Miranda

COMP 157

12/5/16

Project 3

Input size:

Frequency table: n

Huffman tree: m (# of unique elements in n)

Compression table: m (# of unique elements in n or size of Huffman tree)

Compression: n

Decompression: k (length of compressed data file)

Basic operations:

Frequency table: O(n)

Huffman tree: O(m) (Dependent on # of unique elements in n. problem size decreases by 1 each time)

Compression table: O(m) (Again dependent on # of unique elements in n. This will be the size of the tree. Efficiency of traversing the whole tree is just the size of the tree)

Compression: O(n)

Decompression: O(k) (length of compressed data file)

Pseudo code (see attached pseudo.py):

createFreqTable(data):

table = {}

**for** i **in** range(len(data)):

**if**(data[i] **in** table.keys()):

val = table[data[i]]

table[data[i]] = val + 1

**else**:

table[data[i]] = 1

total = len(data)

**for** i **in** (table.keys()):

val = table[i]

table[i] = val/total

tableList = list(table.items())

tableList = sorted(tableList, key=**lambda** x: x[1])

**return** tableList

createHTree(table):

**while**(len(table) > 1):

A = table.pop()

B = table.pop()

sumVal = A.data + B.data

ac = Node(A)

bc = Node(B)

n = Node(sumVal)

n.add\_lChild(ac)

n.add\_rChild(bc)

**if**(len(table) > 0):

insert(n, table)

**else**:

table.insert(0, n)

**return** table

createCTable(tree):

cTable = {}

postOrder(tree,"",cTable)

**return** cTable

postOrder(tree, string, dic):

**if**(tree.lChild == NULL **and** tree.rChild == NULL):

dic[tree.data] = string

**return**

**if**(tree.lChild != NULL):

postOrder(tree.lChild, string + "0", dic)

**if**(ltree.rChild != NULL):

postOrder(tree.rChild, string + "1", dic)

**return**

decompress(compressStr, tree):

curr = tree.head

decStr = ""

**for** i **in** range(len(compressStr)):

**if**(compressStr[i] == "0"):

curr = curr.lChild

**if**(curr.lChild == NULL **and** curr.rChild == NULL):

decStr = decStr + curr.data

curr = tree.head

**if**(compressStr[i] == "1"):

curr = curr.rChild

**if**(curr.lChild == NULL **and** curr.rChild == NULL):

decStr = decStr + curr.data

curr = tree

**return** decStr

compress(string):

**for** i **in** range(len(string)):

compressStr = compressStr + cTable[data[i]]