Viraj Panikh
VHP286
EE360C
Dr. Santacruz

Project 2 report

a) Time complexity of constructHuffmanTree

- inputs: character /frequency array

  - Using Priority Queue from java library add every character sorted by its frequency to the Priority queue. $O(n \cdot \log n)$
  - In the while loop poll the 2 minimom elements from the queue and merge. $O(n)$

  Time Complexity $= O(n\log(n) + n) = O(n\log(n))$

b) Time complexity of encode

- inputs: message to encode

  - generate a hashmap to hold the encoding based on the tree by recorsive tree traversal $- O(n)$

    n = # of unique characters

  - create encoded message $- O(m)$ m = Length of humman message

  Time complexity $= O(m+n)$ n = # of unique characters m = length of input String

c) Time complexity of decode

- inputs: message to decode

  - generate a hash map like previous method $- O(n)$ - # of Characters
  - decode the message by matching each encoded segment to the hashmap $- O(m)$ = length of binary string

  Time complexity $= O(m+n)$ n = # of unique Characters m = length of input S

d) At each step when creating the huffman encoding for each character the algorithm selects either 0 or 1. Characters are stored as leaves of the tree. When reaching a leaf the encoding will be the path to get there from the root + 0 for left child and 1 for right child. So those 2 sybillings have a unique prefix. Since the path to get to each node is unique and the encoding is based on the path the encoding is Prefix free