Name: Harshit Jain
Access ID: hmj5262
Recitation: 8

Problem 0 **Points:**

Acknowledgements

(a) I worked with Yug Jarodiya.

(b) I did not consult with anyone in my group members.

(c) I did not consult any non-class materials.

Problem 1

Points:

Given: A Set $C = \{0, 1, ..., n-1\}$ of characters

To show: Represent any optimal prefix-free code on C using only $(2n - 1 + n\lceil \log n \rceil)$ bits

Proof:

An optimal prefix-free code on C has an associated full binary tree with n leaves and $(n-1)$ internal vertices. These tree can be coded by a sequence of $n + (n-1) = (2n-1)$ bits.

Height of full binary tree = $\lceil \log n \rceil$

To associate the n members of $C\{0, 1, ..., n-1\}$ with the n leaves of the tree, which can be done by listing them in the order in which preorder traversal encounters them, $\lceil \log n \rceil$ is enough bits to represent each of the n members of C, and no delimiters are needed if each is allocated $\lceil \log n \rceil$ bits. For n leaves, it require $n\lceil \log n \rceil$ bits to represent.

Total bits needed to represent optimal prefix-free code on $C =$

$$n \textit{ (for leaves)} + (n-1) \textit{ (for internal nodes)} + n\lceil \log n \rceil \Rightarrow \underline{2n - 1 + n\lceil \log n \rceil \text{ bits}}$$

Problem 2

Points:

The Idea: Put more frequent symbols at smaller depth

Greedy approach: Continually merge least frequent symbols/nodes until you have a full ternary tree encoding all symbols. In this case, take the three lowest frequency symbols, and merge them into one root R. Then from the set of the remaining nodes and R, take the three lowest nodes and merge them continuously until a full ternary tree is made.

Optimal Proof: Suppose we have a tree T with three lowest frequent symbols not as deep as possible. Then at least one has a smaller depth. Switch it with one of the deepest nodes that is more frequent.This improves the encoding length. Thus T is not optimal. Since T is not optimal, then we know that the three lowest frequencies must be at the largest depth.

Algorithm 1: Ternary Huffman Algorithm

Input : $f = f[1], \cdots, f[n]$; Γ has n symbols
Output: T

$T =$ empty tree;
$H =$ priority queue ordered by f;
for $i = 1$ *in* n **do**
  insert(H, i);
end for
for $k = (n+1)$ *in* $(2n - \lceil \frac{n}{2} \rceil)$ **do**
  $i =$ extract min(H);
  $j =$ extract min(H);
  $k =$ extract min(H);
  Create a node z in T with children $i, j,$ and k ;
  $f[z]$= $f[i]$ + $f[j]$ + $f[k]$;
  insert(H, z);
end for
return T

Problem 3

Points: