CS3383, Winter 2019 Assignment # 5 Sample solutions

Rasoul Shahsavarifar
Faculty of Computer Science, UNB

Questions 1) **Basic idea of the solution:** adapt Kruskal's Minimum Spanning Tree algorithm to instead find the Maximum Spanning Tree. (This will minimize the sum of the weights of the edges not in the tree, which is the feedback edge set).

Algorithm FeedbackEdge(weighted graph $G = (V; E)$):
$for\ \ all\ \ u \in V$
  $makeSet(u)$
$S' \leftarrow \emptyset$
sort edges by nonincreasing weight, largest weight first
$for\ \ e \in ordered\ \ edges$
  $if\ \ findSet(u) \neq findSet(v)$
    $merge(findSet(u), FindSet(v))$
  $else$
    $S \leftarrow S \cup \{(u, v)\}$
$return\ \ S'$

Time Analysis: Suppose $|V| = n$ and $|E| = m$.
The initial for loop iterates exactly n times. The sort takes $\Theta(m \log m)$ time, using mergesort. The second (main) for loop iterates m times. Over the entire execution of the algorithm, the findSet calls take $O(m \log m)$ time. The rest of the algorithm contributes some constant factor, so the algorithm as a whole takes $\Theta(n + mlogm)$ time.
Note: We leave the log factor in terms of m rather than n, and keep the n term, because the graph is not necessarily connected. While n could be arbitrarily larger than m, the union-find operations will only use the vertices that have edges, so the amortized cost of findSet is at least $\log(\sqrt{m})$ and at most $\log(\frac{m}{2})$.

Questions 2) **Order:** nondecreasing order by t_i (shortest first).
Summarized Explanation: each customer's service time is added to the waiting time of all those customers served later. The sum of waiting times $\sum_{i=1}^{n} (n - i + 1) \cdot t_i$ is thus at its minimum when the smallest times are the ones which are added the most.

Question 3) The longest a codeword could be $(n - 1)$, the maximum height of a tree with n leaves. This occurs in a degenerate tree that has one leaf at each level except two at the bottom. To have such a tree built for Huffman encoding, let, for each symbol i, $f_i = (1/2)^i$. This doubling sequence makes each newly created node (combining the two least common symbols) have frequency less than all of the remaining symbols, so it will be merged with one symbol at a time. See Figure 1.

Question 4) Consider the base set $U = \{1, 2, dots, 2^k\}$ for some $k \geq 2$. Let $T_1 = \{1, 3, \ldots, 2^k - 1\}$ and $T_2 = \{2, 4, \ldots, 2^k\}$. These two sets comprise an optimal cover. We add sets $S_1, \ldots, S_{k-1}$ by defining $l_i = 2 + \sum_{j=1}^{i} 2^{k-j}$ and letting $S_i = \{l_{i-1} + 1, \ldots, l_i\}$ (take $l_0 = 0$).

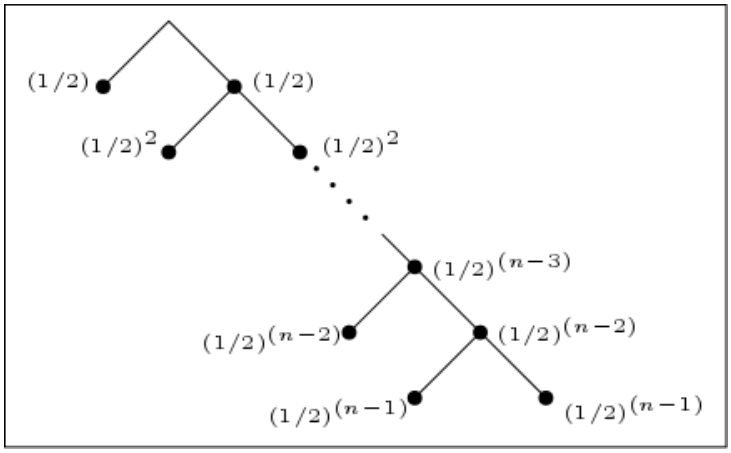

Figure 1: Question 3

Thus, S_1 contains $2^{k-1}+2$ elements and the greedy algorithm picks this first. After the algorithm has picked i sets, each of T_1 and T_2 covers $2^{k-i-1}-1$ new elements while S_{i+1} covers 2^{k-i-1} new elements.
Hence, the algorithm picks the cover $S_1, \ldots, S_{k-1}$ containing $(k-1) = \log(n-1)$ sets. Figure 2 is an illustration for this question.

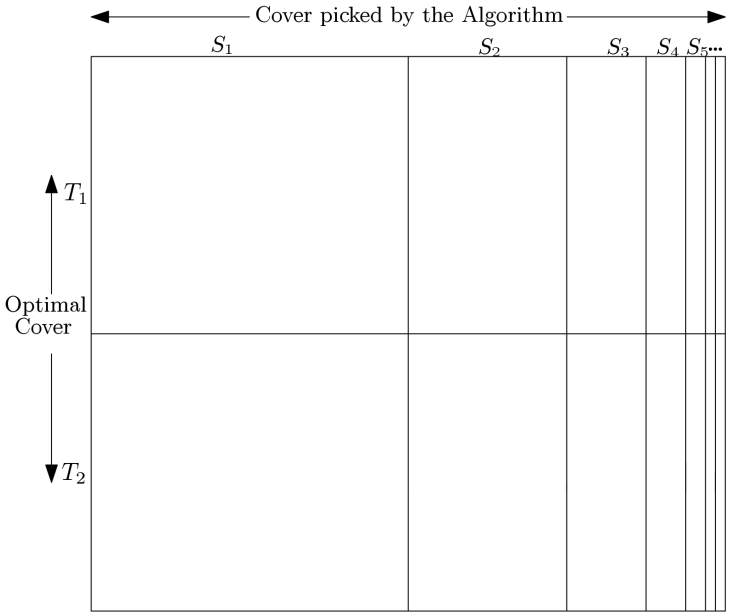

Figure 2: Question 4