CMPT 307 - Data Structures and Algorithms: Problem Set 2

1. **Unique MST.** Prove that if an undirected connected graph $G = (V, E)$ has *distinct* weights $c(e) \geq 0$ on its edges $e \in E$, then it has exactly one Minimum Spanning Tree. For your proof, use the correctness analysis of Kruskal's or Prim's algorithm.

2. **Dijkstra vs. Prim.** As you may have noticed, Dijkstra's and Prim's algorithms are very similar. In this question, you will explore their similarities and differences.

   (a) Show that running Dijkstra's algorithm on an undirected, connected graph G (with non-negative edge costs) will produce a spanning tree of G.

   (b) Prove or give a counter-example to the following statement: A tree produced by Dijkstra's algorithm on a given (undirected, connected, weighted) graph G is a Minimum Spanning Tree.

3. **Greedy hiker.** Consider the Knapsack Problem: Given positive integer weights $w_1, w_2, \ldots, w_n$ and a positive integer W (all in binary), find a subset $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i$ is maximized, subject to the constraint that $\sum_{i \in S} w_i \leq W$.

   A greedy algorithm for this problem is: Sort all weights w_i's from heaviest to lightest. Go through them one by one, adding w_i to the knapsack, if the resulting total weight of the knapsack is at most W.

   (a) Show that this greedy algorithm is not correct in general. That is, given an example of an instance $(w_1, \ldots, w_n, W)$ where the greedy algorithm fails to finds an optimal solution.

   (b) Consider the special case of the problem, wher the weights w_i are distinct powers of 2, i.e., each $w_i = 2^{t_i}$, for some nonnegative integer t_i. Prove that the greedy algorithm is correct in this special case. (Your proof of correctness should argue by induction that every partial solution produced by the algorithm can be extended to a global optimal solution.)

4. **Largest Independent Set in a Forest** Consider the following problem.
   Given: A forest $F = (V, E)$ (i.e., a collection of unconnected trees).

Find: A largest independent set for F (i.e., a largest subset $S \subseteq V$ of vertices of F such that no two vertices in S are connected by an edge in F).

Consider the following greedy algorithm that attempts to solve this problem.

Algorithm GreedyForest($F = (V, E)$)
$S \leftarrow \emptyset$; % initially our independent set S is empty
while $V \neq \emptyset$
    `find any leaf` $v \in V$ `in the forest` F; % i.e., find $v \in V$ of degree ≤ 1
    $S \leftarrow S \cup \{v\}$;
    `remove` v `and any neighbor` w `of` v `from` V;
endwhile
`return` S

(a) Assuming that a forest is given by the adjacency lists, analyze the running time of the algorithm GreedyForest using the order notation. (Try to find the fastest implementation for the algorithm, perhaps using an appropriate data structure to speed up finding a leaf in the body of the while-loop.)

(b) Decide if the algorithm GreedyForest is correct. That is, *either* construct an example of the forest where this algorithm fails to find a largest independent set, *or* give a proof that this algorithm always succeeds in finding a largest independent set in any given forest.

5. **Circular interval scheduling.** Consider the clock face, labeled by hours (from 0 to 24). You are given n circular segments (intervals on the circle), with ith segment having a start time s_i and finish time f_i, and occupying the segment of the circle from time s_i to f_i going clockwise. For example, you may have a segment from 1 to 4 (on the clock face), or a segement from 22 to 2 (which you can think of as going from 10pm to 2am).

   Given n circular intervals $(s_1, f_1), \ldots, (s_n, f_n)$, find a maximum size collection S of intervals (schedule) such that no two intervals in S overlap. (As for the problem of interval scheduling considered in class, we assume that the intervals are *open*, so two intervals sharing endpoints only are not overlapping; thus, e.g., it's OK to put $(1, 17)$ and $(17, 1)$ into the same schedule.) Give a $poly(n)$-time algorithm for this problem.

6. **Recurrences** Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers:

   (a) $T(n) = 2T(n/2) + n^3$.
   (b) $T(n) = 16T(n/4) + n^2$.
   (c) $T(n) = T(n-1) + n$.
   (d) $T(n) = T(\sqrt{n}) + 1$.
   (e) $T(n) = \sqrt{n}T(\sqrt{n}) + n$.