CSE 101, Spring 2018
Practice questions for Quiz 2

1. Answer True or False with a brief explanation, counterexample or proof.

   (a) Let M be an MST of G. If M contains an edge of weight w, than all MSTs of G must contain an edge of weight w.

   True: Let T_1 and T_2 be two MSTs. Suppose that an edge of weight w_1 is in T_1 and not in T_2.
   Add the edge of weight w_1 to T_2. Now you have created a cycle.
   Case 1: w_1 is the heaviest edge in the cycle. Add one of these lighter edges to T_1 that will create a cycle and remove w_1 and you have decreased the total weight of T_1 which was assumed to be an MST.
   Case 2: There exists and edge e' in the cycle that has weight greater than w_1. Then we can add the edge of weight w_1 to T_2 and remove the edge e' and we have decreased the total weight of T_2 which we assumed to be an MST.

   (b) Let G be an undirected connected graph with positive edge weights. If G has two edges with the same weight then it must have more than one MST.

   False: consider the graph:
   A:(B,1)
   B:(A:1),(C:1)
   C:(B:1)

   (c) When using the union-find data structure, any rank k vertex has fewer than 2^k vertices in its subtree.

   False: from the book and the slides we proved that each rank k vertex must have equal to or more than 2^k vertices.

2. Let G be a weighted graph, with edge weights that are positive and distinct. Let T be a minimum spanning tree for this graph. Now suppose we replace each edge weight we by its square w_e^2 . Must T be a minimum spanning tree for this new graph?

   Yes. If you square every edge, then they will be in the same relative order as before so when you run Kruskal's algorithm, you will have the same result.

3. Does negating all the edge weights in a weighted undirected graph G and then finding the minimum spanning tree give us the maximum-weight spanning tree of the original graph G?

   Yes: Since $w(MST) \leq w(T)$ for any other tree T, when you negate the edge weights, you have that $-w(MST) \geq -w(T)$ for any other tree T.

4. Given a connected undirected graph G with distinct positive edge weights and given an edge e. Design a linear time algorithm ($O(|V| + |E|)$) that determines if e is part of a minimum spanning tree of G.

   (*Hint:* Use the following fact: Edge $e = (u, v)$ does not belong to an MST of G if and only if v and w can be joined by a path consisting entirely of edges that are cheaper than e.)

   Remove $e = (u, v)$ from the graph along with all other edges of weight $w(e)$ and higher to make a new graph G'. Then run explore on u in G' and if v is discovered then e is not part of an MST and if v is not discovered then e is part of an MST.

5. Recall event scheduling problem from class (the event scheduling problem is to find a largest compatible set - a set of non-overlapping events of maximum size). Suppose that instead of always selecting the first activity to finish, we select the last activity to start that is compatible with all previously selected activities. Prove that this strategy yields an optimal solution.

The proof for this is identical to the proofs of correctness for the first event to finish. It just reverses time. In fact, we can reduce one to the other by reversing the start and finish times of all events.

6. Suppose you are running a trucking company that ships packages from San Diego to LA. The amount of packages requires you to send a number of trucks each day from SD to LA. Each truck has a limit of W on the maximum amount of weight they are allowed to carry. Boxes arrive in LA one by one, and each package i has weight w_i. The trucking station is quite small so at most one truck can be at the station at any time. Company policy requires that boxes are shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after his make it to LA faster.

   The company is using the following greedy algorithm: they pack boxes in the order they arrive and whenever the next box does not fit, they send the truck on its way and repeat on the next truck.

   (a) Determine the
   - Instance,
   - Solution format,
   - constraints and
   - objective function

   for this problem.

   Instance: $(w_1, w_2, \ldots, w_n)$ the weights of the packages and W, the maximum amount of weight the truck can carry.

   Solution format: intervals of indices $[1, \ldots, i_0], [i_0 + 1, \ldots, i_1], \ldots, [i_k + 1, \ldots, n]$ such that packages $i_j + 1$ through i_{j+1} go on the jth truck.

   Constraints: $w_{i_j+1} + \cdots + w_{i_{j+1}} \leq W$ for each j.

   Objective: Minimize k the number of trucks.

   (b) Prove this strategy is optimal.

   Proof:(modify the solution)
   MTS Lemma: *For some input $I = (w_1, \ldots w_n; W)$, let $[1, \ldots, g]$ be the packages loaded into the first truck by the greedy choice. suppose that OS is a solution that does not load packages $[1, \ldots g]$ onto the first truck. Then there exists a solution OS' that loads packages $[1, \ldots g]$ onto the first truck. and the number of trucks used in OS' is at most the number of trucks used in OS.*
   Proof of Lemma: By the greedy choice, you cannot fit any more packages on to the first truck than $[1, \ldots, g]$ so the first truck of OS must take $[1, \ldots, i']$ with $i' < g$. Create OS' from OS by moving packages $i' + 1, \ldots, g$ from the next few trucks onto the first truck. The first truck does not exceed the capacity of W and since we are taking packages away from the next few trucks. If they were not beyond the capacity of W to start with, they will not be beyond the capacity of W. It is clear that $|OS| = |OS'|$. ■
   (Note: with this construction, it may be the case that the second, third,... trucks are empty.)
   Induction Part:
   Claim: *For any input of any size $n \geq 1$, the greedy solution is optimal.*
   Base Case: For $n = 1$, put the package on the first truck and it is the optimal solution.
   Induction Hypothesis: Suppose that for some $n > 1$, the greedy strategy is optimal for all inputs of size k such that $1 \leq k < n$.
   Inductive Step: Suppose $I = (w_1, \ldots, w_n; W)$ is an input of size n. Then let OS be any solution. Then by the MTS lemma, there exists a solution OS' that fills the first truck with packages $[1, \ldots, g]$ and uses at most as many trucks as OS. Therefore, we have that:

$$T(OS) \overset{MTS}{\geq} T(OS') = T([1, \ldots g] + S(I')) \overset{IH}{\geq} T([1, \ldots g] + GS(I')) = T(GS)$$

where $T(S)$ is the number of trucks used in the solution S, and $I' = (w_{g+1}, \ldots, w_n; W)$. ■

Implementation:
Keep a running sum of package weights until that sum exceeds W. Include all packages just before the sum exceeds W into the first truck. Then recurse on the remaining packages.

```
procedure Trucks(w_1, ..., w_n;W)
    if n == 0:
        return [ ]
    i = 0
    sum = 0
    while sum < W and i < n:
        i = i + 1
        sum = sum + w_i
    return [1, ..., i]∘ Trucks(w_{i+1}, ..., w_n;W)
```

(c) Implement and determine runtime.

Runtime: The algorithm adds the weight of each package only once. Since there are n packages, the runtime is $O(n)$.

7. Suppose that you and your friends are going to hike the John Muir trail this summer. You want to hike as much as possible per day but, you do not want to hike after dark. On a map there are a large set of good stopping points for camping and you design your trip based on the following system: Every time you come to a potential stopping point, determine whether you can make it to the next one before nightfall. If you can make it then keep hiking, otherwise stop and camp. You claim that with this strategy you will minimize the number of camping stops you must make.

(a) Suppose the stopping points are located at distances $x_1, \ldots, x_n$ from the trailhead. Also assume that your group can hike d distance per day (independent of terrain, weather conditions and so forth.)

Determine the

- Instance,
- Solution format,
- constraints and
- objective function

for this problem.

Instance: $(x_1, x_2, \ldots, x_n)$ the stopping points from the trailhead and d, the maximum distance you can travel in a day.

Solution format: $[i_1, i_2, \ldots, i_k]$, the list of stopping points.

Constraints: $x_{j+1} - x_j \leq d$ for each j

Objective: Minimize k, the number of stops.

(b) Prove this strategy is optimal.

Proof:(modify the solution)
MTS Lemma: *For some input $I = (x_1, \ldots x_n; d)$ suppose that OS is a solution that does not stop at the first greedy location x_g. Then there exists a solution OS' that does stop at x_g and the number of stops in OS' is at most the number of stops in OS.*
Proof of Lemma: By the greedy choice, x_g is the farthest you can possibly go in one day. Let OS be a solution that does not stop at x_g, then the first stop of OS is at stopping point $x_{i'} < x_g$. Create OS' from OS by stopping at x_g as your first stop instead of $x_{i'}$ and then continuing along the same sequence of stopping points after x_g in OS' and leaving all further

stops the same. Suppose that x_j is the stop in OS that is right before x_g and x_{j+1} is the stop in OS right after x_g. Then since $x_{j+1} - x_j \leq d$ and $x_j < x_g$, we have that $x_{j+1} - x_g < d$ and so in OS' we can stop at x_g and then get to x_{j+1} the next day and continue in the sequence of OS until the end. At the very least, there is one stop of OS before x_g and so it follows from the construction of OS', $OS' = (OS - \{\text{all stops before } x_g \text{ in } OS\}) \cup \{x_g\}$, that $|OS| \geq |OS'|$ ■

Induction Part:

Claim: *For any input of any size $n \geq 1$, the greedy solution is optimal.*

Base Case: For $n = 1$, stop at the only stopping point which is the end of the trail. This is the only solution and it is optimal.

Induction Hypothesis: Suppose that for some $n > 1$, the greedy strategy is optimal for all inputs of size k for $1 \leq k < n$.

Inductive Step: Suppose $I = (x_1, \ldots, x_n; d)$ is an input of size n. Then let OS be any solution. Then by the MTS lemma, there exists a solution OS' that stops at x_g with the number of stops of OS' is at most as many stops as OS. Therefore, we have that:

$$|OS| \overset{MTS}{\geq} |OS'| = |\{x_g\} \cup S(I')| \overset{IH}{\geq} |\{x_g\} \cup GS(I')| = |GS|$$

where $I' = (x_{g+1}, \ldots, x_n; d)$. ■

Implementation:

Scan through the locations until one of them exceeds d. Camp at the first stopping point just before the distance d. Then recurse on the remaining packages.

```
procedure Camps(w_1, ..., w_n;W)
    if n == 0:
        return [ ]
    i = 1
    while x_i < d and i < n:
        i = i + 1
    return x_{i-1} ∘ Camps(x_i, ..., x_n;d)
```

Implement and determine runtime.

Runtime: The algorithm compares each camp only once. Since there are n camps, the runtime is $O(n)$.