Name: Jaime Arana-Rochel

Homework #2

Q1

a) Proof

Suppose there is an optimal solution $O = \{o_1, o_2, \ldots, o_m\}$ not given by the greedy algorithm with m total coins. Let $D = \{d_1, d_2, \ldots, d_n\}$ be the solution set returned by the greedy algorithm. Thus, we assume $m < n$. We now move case by case, checking the denominations of the coins.

Case: 10 cents
Let i be the number of 10 cent coins in O and j be the number of 10 cent coins in D. If $i > j$, that means the greedy algorithm produced a solution using less coins of this denomination than in O and we arrive at a contradiction that O was optimal to begin with. If $i < j$, we also arrive at a contradiction. Since the greedy algorithm always chooses as many 10 cent coins as possible, the remainder of the desired amount is less than 10 cents. But if $i < j$, then the remainder of the desired amount is ≥ 10 cents for O, meaning that the greedy algorithm stays ahead of the other solution and we are done. Finally, lets say that $i = j$. We then need to check how many of the other coin denominations are used to complete the remainder of the desired amount.

Case: 5 cents
Let i' be the number of 5 cent coins in O and j' be the number of 5 cent coins in D. Using the same arguments for the case of 10 cent denominations, we see that if $i' \neq j'$ we arrive at contradictions to our assumption that O was optimal. If $i' = j'$, then we proceed to the case of 1 cent denominations.

Case: 1 cent
Let i'' be the number of 1 cent coins in O and j'' be the number of 5 cent coins in D. If $i'' > j''$, then we contradict our initial assumption that $m < n$ since O would then have more coins than D. Because the remainder of the desired amount at this point is < 5, the greedy algorithm will choose as many 1 cent coins (j'') as needed, and thus it can't be that $i'' < j''$. By elimination, it must be the case that $i'' = j''$.

Because the number of coins of each denomination was shown to be the same, we contradict our initial assumption that $m < n$. So $m = n$, meaning that the greedy algorithm does indeed produce an optimal solution.

b) Let our denomination set $= \{1cent, 4cents, 5cents\}$. If we are given the amount of 8 cents, the greedy algorithm would choose one 5 cent coin and three 1 cent coins. However,

the optimal solution would be to choose two 4 cent coins. Thus the greedy algorithm fails for this denomination set.

Q2: Exercise 2 Chapter 4

a) True. For the original graph G, Kruskal's algorithm would have created the tree T in order of increasing edge costs. This edge order is maintained when Kruskal's algorithm is applied to the new graph G' with edge costs c_e^2. That means that T is still a MST for the new graph.

b) False. Consider a graph G with $V = \{s, u, v, t\}$, $E = \{(s, u), (s, v), (u, t), (v, t)\}$ with edge costs $\{4, 1, 5, 7\}$ respectively. The shortest path P in this graph would be $s -> v -> t$ which has cost 8 as opposed to the path through u which has cost 9. If we square the edge costs, then the path through v becomes 50. However the cost through u is 41, meaning that our original path P is no longer valid.

Q3: Exercise 5 Chapter 4

The greedy algorithm is as follows: Starting from the western endpoint, we move along the path eastward. Upon encountering a house at position p_i, we move exactly four miles eastward and place a base station at position b_i. We continue this process for the remaining stretch of the road, ignoring the houses that are already covered by the base station placed in the last iteration.
We see that this algorithm essentially places a base station at the easternmost position b_i, 4 miles away from a house, such that the house and any houses in between it and the base station are covered.

Claim: This greedy algorithm halts.
Proof: This algorithm obviously terminates since at some point we will reach the easter endpoint of the road and no longer be able to place base stations.

Claim: This algorithm is optimal in that it places as few base stations as possible.
Let $B = \{b_1, b_2, \ldots, b_n\}$ be the set of base station positions, in increasing order, given by the greedy algorithm. Let $O = \{o_1, o_2, \ldots, o_m\}$ be an optimal solution to the problem. We will show that the greedy algorithm stays ahead of the optimal solution O by claiming that $b_k \geq o_k$ for each k.

Proof by Induction on base positions:
Base case $k = 1$: This is obviously true since the greedy algorithm always chooses the easternmost position after a house to place the base station.
Induction Step: Let our induction hypothesis be the assumption that $b_k \geq o_k$ is true for arbitrary $k = i \geq 1$.
Now lets suppose $k = i + 1$. By the induction hypothesis, the greedy algorithm's first i

stations covers all the houses covered by the first i stations in O. In the final step of the greedy algorithm, we choose b_{i+1} to be as easternmost as possible (meaning it's as large as possible) such that all houses between b_i and b_{i+1} are covered. Because this position is as large as it can be, we have that b_{i+1} must be at least as large or larger than o_{i+1}. Thus, by induction we have that $b_k \geq o_k$ for each k.

Therefore, since our greedy algorithm stays ahead of the optimal solution O in that it places base stations in the largest position possible such that all houses are covered, the solution B must be optimal.

Q4: Exercise 10 Chapter 4

For this problem, we will assume that all of the edge costs are distinct. We can achieve this by the pertubation of edge costs argument given in the text.

a)
We will represent the MST T using an adjacency list. Our new edge is $e = (v, w)$. Using the adjacency list, we find the $v - w$ path P in T. For all edges in P, if the cost is less than the cost of the new edge c, then e cannot be in the MST T. This is due to the Cycle Property which implies that e cannot be in the MST since it would be the most expensive edge in the cycle formed by adding this edge to the graph G. Therefore T would remain unchanged. If however some edge in the path P which is part of that cycle has greater cost than e, then by the Cycle Property that edge in the path would no longer be in the MST. We would simply switch that edge and the newly added edge e, meaning T changes.

Since this algorithm only needs to search the entire tree T given by the adjacency list, we can find the $v - w$ path in linear time in the number of edges and vertices. We only do cost comparisons for $n-1$ edges, meaning that that whole operation is also linear.That gives us an asymptotic running time of $O(|V|)$ for the whole algorithm.