Greedy 1:

a. Assume T and T′ are distinct minimum spanning trees. Then some edge (u; v) is in T but not in T′ . Since all edge weights are distinct the edges on the unique path from u to v in T′ must then be strictly lighter than (u; v) contradicting the fact that T is a minimum spanning tree. It can easily be seen by example that the second-best minimum spanning trees is not unique.

b. Let T be a minimun spanning tree. We wish to _nd a tree that has the smallest possible weight that is larger than the weight of T. We can insert an edge (u; v) 62 T by removing some other edge on the unique path between u and v. By the above property such a replacement must increase the weight of the tree. By carefully considering the cases it is seen that replacing two or more edges will not produce a tree better than the second best minimum spanning tree.

c. To compute this in O(V2) time for all nodes in the tree do the following: For each node v perform a traversal of the tree. Inorder to compute max(v; k) for all k 2 V simply maintain the largest weight encounted so far for the path being investigated. Doing this yields a linear time algorithm for each node and we therefore obtain a total of O(V2) time.

d. Using the idea of the second subexercise and the provided algorithm in the third subexercise, we can now compute T2 from T in the following way:
Compute max(u; v) for all vertices in T. Compute for any edge (u; v) not in T the difference w(u; v) - max(u; v). The two edges yielding the smallest positive difference should be replaced.

Greedy 2:

a. Suppose a permutation of S is ⟨r1,r2,...,rn⟩, the total completion time is ∑i=1n(n−i+1)⋅pri. The optimal solution is to sort pi into increasing order.

b. Preemption will not yield a better solution if there is no new task. Each time there is a new task, assume that the current running task is preempted, let the current condition be a new scheduling task without preemption.