CS3110: Assignment 4

Problem 1:

Funciton findMaxNoneOverlapActivities:

Q = []

For i = n to 1:

Add i to Q

While j > 1

If j.endTime > i.startTime

i = j

j --

End While

End for

Return |Q|

Problem 2:

Function findMinExamRooms:

Sort exams by start time

For i=1 to n  
 j←1  
 While (exam i not scheduled)  
 lastj← finish time of the last exam currently scheduled on room j  
 if si≥lastj  
 schedule request i on resource j  
 j←j+1  
 End While  
 End For

Return j

Complexity: O(NlogN)

**Proof:**

Since we need to schedule all exams, the problem is to find the largest number of overlapped intervals at certain time. To prove greedy algorithm works:

1. Let d = number of classrooms that the greedy algorithm allocates.
2. Classroom d is opened because we needed to schedule a job, say j, that is incompatible with all d-1 other classrooms.
3. Since we sorted by start time, all these incompatibilities are caused by lectures that start no later than sj .
4. Thus, we have d lectures overlapping at time sj + ε

It has been proved that greedy algorithm never schedules two incompatible lectures in the same classroom. So there are at least d classroom scheduled.

**End Proof.**

Problem 3:

Function findMinEmpHour

Ret←0

Sort appointment time

For i=1 to n

Ret++

For j=i+1 to n

If diff(j.time - i.time) < 3600 seconds

i = j

Else

Break

End For

End for

Return Ret

Complexity: O(NlogN)

**Proof:**

This greedy algorithm can be proved by:

1. Let the optimal result be T.
2. At certain time, T-1 intervals have been selected.
3. By selecting the earliest time of remaining appointments and add one interval, there is no additional cost to add other appointments within that interval.

Hence the greedy algorithm is correct.

**End Proof**

Problem 4:

1. True.

**Proof (by contradiction)**

1. If the minimum spanning tree changes then at least one edge from the old graph G in the old minimum spanning tree T must be replaced by a new edge in tree T' from the graph G' with squared edge weights.
2. The new edge from G' must have a lower weight than the edge from G. This implies that there exists some weights C1 and C2 such that C1 < C2 and C12 >= C22.
3. This is a contradiction.

**End Proof**

1. True.

**Proof:**

1. Let G=(V,E)G=(V,E) be the original graph.
2. Suppose there are two distinct MSTs T1 and T2, ∃e∈(E1−E2).
3. Let e∉E2, adding it to T2 creates a cycle. By cycle property the most expensive edge of this cycle (e’) does not belong to any MST.
4. So there are two cases here:
   1. if e′=e then e′∈E1 (because e∈E1−E2).
   2. If e′≠e then e′∈E2.
5. Both cases are contradicting with the fact that e′ is not in any MST.

**End Proof**

Problem 5:

a)

Function checkStillValid(newEdge):

Find the unique path P from newEdge.from to newEdge.to in T

w\*←0

For w in P

If w > w\*

w\* ← w

If newEdge.weight > w\*

Return true

Else

Return false

**Proof:**

1. Adding a new edge to MST must create a cycle. There are two cases:
   1. For all edges in that cycle, if the new edge is maximum, updating the MST by containing new edge will not decrease the total cost.
   2. If the new edge is not maximum, that edge should be included to MST.

**End Proof**

Complexity: BFS to search for path from newEdge.from to newEdge.to is O(|V|)

b)

Based on the algorithm above, remove w\* and add new edge to construct new MST.

Function updateMST(newEdge):

Find the unique path P from newEdge.from to newEdge.to in T

w\*←0

For w in P

If w > w\*

w\* ← w

If newEdge.weight > w\*

Return

Else

Remove w\* in T

Add newEdge to T