**HOMEWORK 3 SOLUTIONS**

**Ans1**. S ← [0] // The starting point

t ← g[m] // the total miles to Santa Monica

miles ← 0; i ← 1

while miles < t – n do

while g[i] ≤ miles + n do // where g[i] is the distance from usc to ith gas station

i ← i + 1

endWhile

miles ← g[i − 1]

Add g[i − 1] to the end of the S list

endWhile

return S

Proof: Suppose S is an optimal route which involves the sequence of stops s0,s1, ...,sk where si = the miles from USC of the i-th stop, s0 = USC, and sk = Santa Monica. Let s’1 be the distance from Newark of the last station before going more than n miles. If s1 = s’1 we are done. So suppose s1 != s’1 . Then because of the car’s p mile limit we have: (i) s1 < s’1 and (ii) s2 − s1 ≤ n. Hence, s2 – s’1 ≤ n. So S’= s’,s’1 ,s2, . . . ,sk is feasible and has the same number of stops as S. Hence, S’ is optimal.

Complexity: It runs in O(n) as we traverse the complete distance just once.

**Ans2**. We implement BFS. And check if in every iteration, there is a node with no incoming edges, then it is the topological order of the system.

But if every node has at least one incoming node, it means that G has a cycle.

It runs in O(n). To prove we will follow the edge we are currently at. Since all edges have incoming nodes, we will visit the nodes until we revisit any new node. The nodes encountered between these successive visits constitutes a cycle.

**Ans3**. Let sequence S be s1, s2, … sn.

And sequence S’ be s’1, s’2 ,….. s’m.

We check the first event in S that is same as s’1, and then check for event that is same as s’2 and so on. We k1,k2, …. Are the numbers that have matched. And i and j point towards the current positions in S and S’.

Initially i=1, j=1;

While i<n and j<m

If si is same as s’j then,

Let kj=i

Let i=i+1, j=j+1

Else

Let i=i+1

Endwhile

If j=m+1 return the subsequence is k1,…..km

Else

Return “ S’ is not a subsequence of S. ”

The algorithm runs in O(n).

Proof: Assume that S’ is the same as subsequence sl1,…slm. We prove that algorithm will be successful in finding a match and will have kj<lj or all j=1,…..m .

Let j=1, k1 event is same as s’1, so k1<l1.

Now when j>1, assume j-1<m . now by induction when the algorithm find the match kj-1 and kj-1<lj-1 . The algo lets kj be first event after kj-1 that is same as s’j if the event exists.

We know that lj is such an event and lj>lj-1>=kj-1. So slj=s’j and lj>kj-1. The algo finds the first index, so kj<lj.

**Ans4.**  i) Construct min heap of k sublists.

ii) Take 2 sublists and compare their first element i.e. the minimum element in the list. This comparison takes O(k) time.

iii) Extract the minimum element from the root and place it in the new sublist. And using heapify update the root in the sublist. This extraction takes O(log k) time.

As we have n elements for comparison and so total time for extraction takes O(n\* log k )

And the comparison take O(k).

So overall time – O(n\*log k + k) =O(n log k )time.

**Ans5.** We want to cover a set of points on a line using intervals of length 8 and the goal is to use the minimum number of intervals.

Efficient greedy algorithm: Starting from leftmost point, at each step, place the beginning of the interval at the first uncovered point. And this will cover the houses in the 4 mile radius. Repeat this process till the rightmost house gets covered in the tower signals.

We claim that this algorithm gives us the optimal solution.

Proof by contradiction: Assume greedy is not optimal.

Let g1 < g2 < . . . < gp denote the set of start points of the intervals chosen by the greedy algorithm. Similarly, let f1 < f2 < . . . < fq denote the set of start points of the intervals in an optimal solution with f1 = g1, f2 = g2, . . . , fr = gr for largest possible value of r. Note that gr+1 > fr+1 by the greedy choice of algorithm. We change the optimal solution a bit by making fr+1 = gr+1 i.e. we push the (r + 1)th interval further.

The new configuration is a feasible solution since it still cover all the points. This is true since there cannot be any points in (fr+1, gr+1) according to greedy solution.

This new solution has one more start point in common with greedy solution which is a contradiction.