Greedy Algorithms

Q1. Given an undirected weighted graph **G(V,E)** with positive edge weights. G has n vertices and m edges. What is the time complexity of Dijkstra's single source shortest path algorithm if a priority queue is used to store the distances of the vertices from source.

a. O(nm)
b. O(n^2m)
c. O(nlog(n))
d. O(mlog(n))
e. O(nlog(m))
f. **None of above**

Q2. Given a weighted, directed graph **G(V,E)** with n vertices and m edges with positive weights. Suppose graph G contains **cycles**. Is it possible that Dijkstra's shortest path algorithm will change the distance of a vertex it has already included inside the fringe?

YES / NO ____No_____
If Yes, illustrate by giving an example of a graph. If No, explain why.

A cycle will only increase the distance of a vertex already inside the fringe.

Q3. Given a weighted, directed graph **G(V,E)** with n vertices and m edges with positive weights. Suppose you find the shortest path 'P' in the graph G from source vertex 'S' to vertex 'A'. Now you modify the graph G by **adding** a constant positive number to all the edge weights. Let's call the modified graph G^+. Is the path 'P' still the shortest path from source vertex 'S' to vertex 'A' in graph G^+?

YES / NO ____No_____
Briefly justify your answer.

It's easy to construct a counterexample. Suppose there are two paths P_1 and P_2 from S to A. Suppose P_2 has **more hops** than P_1. If P_2 was originally the shortest path, then after adding a positive number to all edge weights, P_1 may become the new shortest path from S to A in G^+.

Q4. You are given a weighted, directed, connected graph G(V,E) with positive edge weights. Suppose you have computed shortest paths from source vertex 'S' to every other vertex in the graph. Now, you replace the weights of all edges in G by taking their square-roots. Will your shortest paths change?

If yes, then give an example, if not, then prove it.

Yes, the shortest path will change. See example below.

S: A;3, D;2
A: B;3
B: C;3
D: C;5

Q5. Given a weighted, directed graph **`G(V,E)`** with n vertices and m edges with positive weights. Suppose you find the shortest path 'P' in the graph G from source vertex 'S' to vertex 'A'. Now you modify the graph G by **multiplying** a constant positive number to all the edge weights. Lets call the modified graph G^x. Is the path 'P' still the shortest path from source vertex 'S' to vertex 'A' in graph G^x?

YES / NO ___**Yes**_____
Briefly justify your answer.
Scaling by a positive number 's'.
$s P_x < s P_y$, if $P_x < P_y$

Q6. Suppose G is a connected, undirected graph whose edges all have positive weight. Let M be a minimum spanning tree of this graph. Now, we modify the graph by adding 7 to the weight of each edge. Is M guaranteed to be a minimum spanning tree of the modified graph?

YES / NO ___**Yes**_____

Q7. Your friend runs a company that delivers organs for transplantation to hospitals. Since lives are at stake, these organs have to be delivered as quickly as possible. Your friend transports these organs in special medical vehicles (MV) through fast highways. There are a number of petrol stations along the highway and your friend has to **minimize the time** spent in stopping for petrol. A full tank on the MV can hold ***T*** litres of petrol. The vehicle consumes petrol at the rate of ***R*** litres per km. If your friend decides to stop at a station then adding fuel to the tank takes ***F*** litres per minute (assume that adding fuel to the tank is the only delay incurred at the station). The starting point of the trip is your friend's office 'O' and ending point is the hospital 'H'. You are given the locations (L_1, L_2, L_3, … L_n) of petrol stations along the O-H route (assume that the O-H route is a straight line and L_1, L_2 are distances in kilometers from the starting point O along that line. The distance between any two stations will not be greater than T/R km). Assume that MV's fuel tank is empty at starting point O.

You have to design a **greedy** algorithm that decides when and where to stop for petrol and how much to fill, while minimizing the overall time required in stopping for petrol.

The MV stops at every petrol station and fills the tank with fuel that is just sufficient to get to the next station.

Q8a. You have studied that we can find the single source shortest paths in a weighted graph G(V,E) with nonnegative edge weights using Dijkstra's algorithm. The time complexity for Dijkstra's algorithm using heap-based priority queue is O(m log(n)). If you are told that the input graph G will always be a DAG then can you design an algorithm for single source shortest paths that has better time complexity than Dijkstra's algorithm?

If YES, then give a clear description of your algorithm and its running time. If NO, then explain why not.

YES.

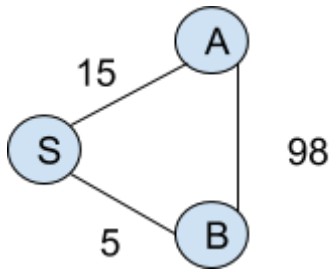
1. Construct the topological ordering of G (let's call that ordering 'P')
2. Initialize the distance of source vertex S to 0 and all other vertices to infinity
3. For each vertex u in P
   Do the following for all neighbors v of u;
   If dist(v) > dist(u) + weight of edge (u, v)
   dist(v) = dist(u) + weight of edge (u, v)

Running time: O(m+n)

Q8b. Given an undirected weighted graph **`G(V,E)`** with positive edge weights. The graph has n vertices and m edges. You already know Dijkstra's algorithm for finding the single source shortest path to all vertices in **G**. Your friend says that he can slightly modify Dijkstra's algorithm for finding the single source longest path to all vertices in **G**. The change he has made to Dijkstra's algorithm is that instead of checking and updating the minimum distance at each step from source, he now checks and updates the maximum distance from source.

Prove that your friend's algorithm is correct or give a counterexample.

The friend's claim is false.
Counter example:



The friend's algorithm will report 15 as the longest path from S to A, which is not correct.

Q8c. You are given a directed graph G (V, E), where the vertices are numbered v_1, v_2, .. v_n. Each edge is directed from a vertex with lower index to a vertex with higher index (i.e., every directed edge has the form (v_i, v_j) if i<j). **Every vertex, except for v_n has at least one directed edge coming out of it.** The length of the path is the number of edges in it. We want to find the length of the longest path starting from v_1 and ending at v_n. You have to determine if the following pseudocode will find the correct solution.

```
currentNode = v1
L = 0
while there is an edge coming out of currentNode
    Choose the edge (currentnode, v_j) such that j is as small as possible
    currentNode = v_j
    L = L+1
return L as length of longest path
```

Will the above greedy approach work? YES / NO ___NO_____
If Yes, prove it. Otherwise provide a counterexample.
V1: V2, V3
V2: Vn
V3: V4
V4: Vn

Q8d. You are given a weighted, directed connected graph G(V,E) with no cycles and positive edge weights. Design a linear time algorithm for finding single source *longest* paths in the graph.

Describe your algorithm and its running time clearly.
G(V,E) is a DAG. Let 'S' be a source vertex. Assuming there are no incoming edges into node 'S'. w(u,v) is weight of edge (u,v).
ALGORITHM:-

1. Find topological ordering of vertices in G
2. Initialize for all $u \in V$ dist(u) = $-\infty$
3. dist[S] = 0;

4. for each u∈V in the topological ordering
5. for each edge (u,v) in E
6. dist(v) = max (dist(v), dist(u) + w(u,v))

Time Complexity: O(m+n)

Q8e. Given a **directed acyclic graph** G(V, E) with positive edge weights w_e and a source vertex S, we carry out the following transformation.

1. Create a new graph G_x(V, E) which has the same set of vertices and edges as the original graph G, but the weights are negated, i.e. $-w_e$.
2. We are given an implementation of a shortest path (SP) algorithm that works in presence of negative weights. We run SP algorithm on G_x and it returns the shortest paths from S to all other vertices in G_x.

What does the output of SP algorithm represent in the original graph G?

Clearly explain your answer.

The output of SP algorithm represents the longest path in the DAG G(V,E).

Q9. Recall the interval scheduling problem discussed in class, where you were given 'n' intervals, each with a start time (s_i) and a finish time (f_i) and you had to find the maximum subset of mutually compatible jobs. You know you can use the Earliest-Finish-Time-First (EFTF) greedy algorithm to solve this problem. Which of the following statements hold true for the EFTF algorithm.

Suppose EFTF says that the following intervals should be scheduled in the order given below.

I_a, I_c, I_d, I_b, I_f, I_e

$f(I_a) \leq s(I_c)$ **TRUE / FALSE** _____T_____

$f(I_a) < s(I_d)$ **TRUE / FALSE** _____T_____

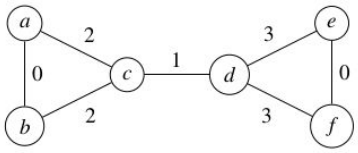
$s(I_d) < s(I_b)$ **TRUE / FALSE** _____T_____

$s(I_d) < f(I_b)$ **TRUE / FALSE** _____T_____

Q10.

Prim's algorithm for minimum spanning tree works for undirected connected graphs with negative edge weights.	TRUE / FALSE

Given the following weighted undirected graph G(V, E), what is the maximum number of minimum spanning trees in this graph?



Maximum number of minimum spanning trees in G are: ___4_____
Brief justification of your answer:

a-b, e-f and c-d are in every MST. There is a choice between a-c and b-c and also between d-e and d-f

Q11. We have an ordered list of n words. We cannot reorder the words. The length of the i^{th} word takes up w_i space, (for simplicity, we will assume that there are no spaces between words.) We want to lay out this ordered list of words into lines.
The length of a line is the sum of the lengths of the words on that line. The page width is L and no line can be longer than L. Suppose the length of line is K, then the penalty for that line is L − K. The total penalty is the **maximum** of the line penalties. The problem is to find a layout that **minimizes the total penalty**. You have to determine if the following pseudocode for a greedy algorithm will correctly solve this problem.

```
for i= 1 to n
      If the ith word fits in the current line, then place it there
      else place the ith word on a new line
```

Will the above greedy approach work? YES / NO ____NO_____
If Yes, prove it. Otherwise provide a counterexample.
Suppose L = 5
$w_1 = 3$
$w_2 = 2$
$w_3 = 2$
$w_4 = 4$
Greedy algorithm gives a maximum of line penalties as 3, whereas the optimal is 2

Q12)
Suppose you have coded 'n' programs in the last year and you want to take backup of your hard work on a USB drive. Let's call the programs P_1, P_2, … P_n. Program P_i requires k_i bytes of space. The capacity of the USB drive is C bytes, where

$C < \sum_{i=1}^{n} k_i$

Notice C is less than the total space required, so you can't save all the 'n' programs.

A) Can you design an efficient and optimal greedy algorithm to **maximize the number of programs** you can save on the drive?

If YES, then **describe** your algorithm. If NO, then explain why not.
YES / NO _____YES_____

Sort according to k_i in ascending order and save on USB drive in that order.

B) For the same USB drive problem, can you design an efficient and optimal greedy algorithm to **maximize the space utilization** of the drive?

If YES, then **describe** your algorithm. If NO, then explain why not.
YES / NO ____NO_____

At this point you just need to have an intuition that a greedy solution with local optimization doesn't seem to work. We'll study this later in the course.

Q13. Recall the "Interval Partitioning" problem, where requests are in the form of time intervals and many identical resources are available. We want to schedule all the requests using as few resources as possible. E.g. classrooms are resources and we want to schedule all lectures in as few classrooms as possible. Following is the greedy "Earliest Start-Time First" algorithm for this problem. If the classrooms are stored in a priority queue where the key is the finish time of the last lecture then how much time does the FOR-loop take? Give your answer in Big-Oh notation and **explain** how you estimated the time complexity.

EARLIEST-START-TIME-FIRST $(n, s_1, s_2, \ldots, s_n, f_1, f_2, \ldots, f_n)$

SORT lectures by start time so that $s_1 \leq s_2 \leq \ldots \leq s_n$.

$d \leftarrow 0$ ⟵ number of allocated classrooms

FOR $j = 1$ TO n

  IF lecture j is compatible with some classroom

    Schedule lecture j in any such classroom k.

  ELSE

    Allocate a new classroom $d + 1$.

    Schedule lecture j in classroom $d + 1$.

    $d \leftarrow d + 1$

RETURN schedule.

O(n log(d))