CMPT 307 : Quiz 3 (Total Marks: 55)
November 15, 2019

1. (15 points) Answer the following questions providing a brief justification for your answers.

   (a) A graph where all edges are distinct can have more than one minimum spanning trees. True or false?
   Answer: False. Look at question 4(b) of homework 5.

   (b) A graph where all edge weights are distinct can have more than one shortest paths between two vertices u and v. True or false?
   Answer: True. Let $w(u,w1) = 1, w(u,w2) = 2, w(w1,v) = 4, w(w2,v) = 3$. This is an example.

   (c) Adding a number w on the weight of every edge of a graph might change the shortest path between two vertices u and v. True or false?
   Answer: True

   (d) Multiplying all edge weights by a positive number might change the shortest path between two vertices u and v. True or false?
   Answer: False. The weight of a path P becomes $c \times w(P)$, which does not change the ordering.

   (e) Let T be a minimum spanning tree of a graph G. Then for any two vertices u,v the path from u to v in T is a shortest path from u to v in G. True or false?
   Answer: False. Let w(u, v) = 3, w(u, w) = 2, w(w, v) = 2. Then the minimum spanning tree contains the edges (u, w) and (w, v), and not (u, v) which is the shortest path from u to v.

   (f) Suppose you are given a connected weighted graph $G = (V,E)$ with a distinguished vertex s and where all edge weights are positive and distinct. It is possible for a tree of shortest paths from s and minimum spanning tree in G to not share any edges. True or false?
   Answer: False. Let $e = (s,v)$ be the minimum cost edge incident to s. Then dist[s,v] is the cost of edge e. Consider a cut $S = s$ and $V - S$. The minimum cost edge between S and $V - S$ is e. Therefore, by the cut property, there exists a minimum spanning tree that contains e. Thus, e is present in both the shortest path tree from s and the minimum spanning tree.

2. (10 points) For each of the methods of divide-and-conquer, greedy algorithm and dynamic programming, determine the properties from the following list that are appropriate for the method.

(a) Make a choice at each step

(b) Each choice depends on solutions to subproblems

(c) Bottom up solution, from smaller to larger subproblems

(d) Solve the subproblem arising after the choice is made

(e) The choice we make may depend on previous choices, but not on solutions to subproblems

(f) Top down solution, problems decrease in size

(g) Divide the initial problem into subproblems.

(h) Subproblems are disjoint.

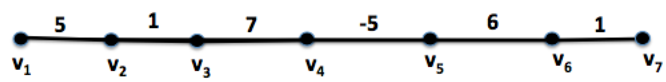
Answer:

Divide and conquer b, c, g, h

Greedy: a, d, e, f

Dynamic Programming b, c, g

3. (15 points) We would like to apply the *Bellman-Ford (BF)* algorithm to the following path graph with v_1 as the source vertex.



(a) How many iterations are needed if the edges are considered in the following order.

$$(v_6, v_7), (v_5, v_6), (v_4, v_5), (v_3, v_4), (v_2, v_3), (v_1, v_2)$$

Ans: The following table illustrates the progress of the algorithm.

Iteration	0	1	2	3	4	5	6	7
$dist(v_1)$	0	0	0	0	0	0	0	0
$dist(v_2)$	∞	5	5	5	5	5	5	5
$dist(v_3)$	∞	∞	6	6	6	6	6	6
$dist(v_4)$	∞	∞	∞	13	13	13	13	13
$dist(v_5)$	∞	∞	∞	∞	8	8	8	8
$dist(v_6)$	∞	∞	∞	∞	∞	14	14	14
$dist(v_7)$	∞	∞	∞	∞	∞	∞	15	15

(b) Determine the order of the edges such that the shortest path distances from v_1 can be determined in two iterations.

Ans: If the edges are considered in the order $(v_1, v_2), (v_2, v_3)$, (v_3, v_4), $(v_4, v_5), (v_5, v_6)$ and (v_6, v_7), the BF algorithm will find all the distances correctly during the first iteration. The second iteration is needed to make sure that all the distances are correctly computed.

(c) What is the worst case running time of the BF algorithm on an arbitrary directed graph with n vertices and m edges?

Answer: The worst case running time results when there exists a shortest path of length $n-1$ and the edges of the shortest paths are considered in the reverse order. In this case the running time is $O(n(n+m))$.

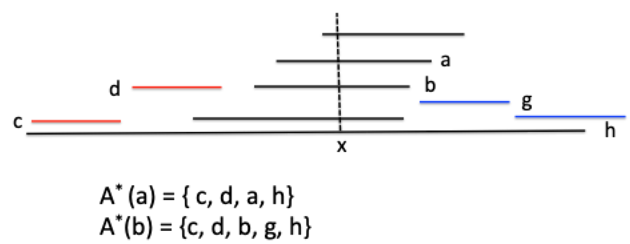
4. (15 points) Consider a set S of day activities (see the figure) where the ith activity a_i has starting time s_i and finish time f_i. An activity a_j contains an activity a_i if the interval $[s_j, f_j)$ contains the interval $[s_i, f_i)$. An activity a_i is called a proper activity if it does not contain another activity. Two activities a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap. The problem we are solving is to select a maximum size subset (A^*) of mutually compatible activities.

(a) Show that there exists an optimal solution A^* where all the activities are proper activities.
Ans: See homework 5 solution 1(a).

(b) For an arbitrary proper interval x, compute $A^*(x)$. What is the running time?
Ans: Let s_x and f_x be the starting time and finishing time of activity x, respectively. We first eliminate all the activities which are not compatible with x. The cost to implement this is $O(n)$. Let S_L (S_R) be the activities compatible with x which lie on the left (right) side of x. Starting from the right (left), we find optimal greedy solution of the activities $S_L \cup \{x\}$ ($S_R \cup \{x\}$). These solutions require $O(n \log n)$ time to compute since we need to sort the endpoints of the intervals of S_L and S_R first. The union of these two optimal solutions is $A^*(x)$. The total cost is $O(n \log n)$.

(c) It is possible for two proper intervals a and b realizing $A^*(a)$ and $A^*(b)$ such that $|A^*(a)| \neq |A^*(b)|$. (see the figure). Show that the size difference of $A^*(a)$ and $A^*(b)$ is at most one for any arbitrary proper activities a and b.
Ans: The proof is very similar to the one given in the solution of question 1(d) of homework 5.

Let x be a point on the line. Let S_x be the set of intervals that overlap x. Clearly, at most one interval of S_x can be picked in the optimal solution A^*. Let S_L and S_R be the intervals of $S - S_x$ that lie to the left and the right of x. The optimal mutually compatible activities can be determined by taking the mutually compatible activities of S_L and S_R, and probably one of the intervals of S_x. In the figure we notice that $A^*(a) = \{c,d,a,h\}$ and $A^*(b) = \{c,d,b,g,h\}$. Since all the involved intervals are proper intervals, if any one interval of S_x is selected, this selection might make at most two of the optimal compatible activity intervals of $S_L \cup S_R$ not compatible. Therefore, the size of the optimal compatible intervals of S is at most the size of the optimal compatible activity intervals of $S_L \cup S_R$ plus one.



(d) Design and analyze an algorithm to compute A^*, given S.
Ans: This is described in the lecture slides.