Prob1: ~~5~~ 4 Prob2: 3 Total: 7

CS345A: Design and Analysis of Algorithms

Quiz 2

Marks = 12 Date: 09 September 2023

NAME: DIVYANSH

ROLL No: 210355

Important Instruction:
You might <u>lose 1 mark</u> if you write answer to any question <u>outside</u> the box assigned for that question.

Problem 1 Attempt any one of the following problems.

1. Easy Problem (*3 marks*)
You are given a directed graph $G = (V, E)$ on $n = |V|$ vertices, where each edge $e \in E$ has a capacity $c(e) > 0$. There is a source vertex $s \in V$ and every vertex is reachable from s. Let P be any path from s to a vertex $v \in V$. Capacity of P is defined as the capacity of the minimum capacity edge lying on P. The following is a pseudocode of the algorithm (similar in spirit to Dijkstra's algorithm) for computing the capacity of the maximum capacity path from s to v for each $v \in V\backslash\{s\}$. It makes use of a variable $L(v)$ which is initialized appropriately in the beginning and updated suitably subsequently during the algorithm. For each $v \in V$, it computes $Cap(v)$ that stores the capacity of the maximum capacity path from s to v. Fill in each blank with an appropriate expression. You may use **min** and **max** operators if you wish.

Procedure MaximumCapacity(G, s)

```
1  $U \leftarrow V$;
2  $S \leftarrow \emptyset$;
3  for each $v \in V$ do $L(v) \leftarrow 0$ ✓;
4  $L(s) \leftarrow \infty$;
5  for $i = 0$ to $n - 1$ do
6      $y \leftarrow$ vertex from $U$ such that y has max value of L ✓;
7      $Cap(y) \leftarrow$ $L(y)$ ✓;
8      Move $y$ from $U$ to $S$;
9      for each $(y, v) \in E$ do
10         if ($\min(L(y), c(y,v)) > L(v)$ ✓ and $v \in U$ ✓)
           then $L(v) \leftarrow$ $\min(L(y), c(y,v))$ ✓;
11     end
12 end
```

3

2. Hard Problem (*6 marks*)
There are n jobs, labeled $J_1, J_2, \ldots, J_n$ to be executed on a single server. Each job takes exactly 1 minute to get executed on the server. The server can execute only 1 job in a minute, and is available for precisely n minutes from $\{1, \ldots, n\}$. For any $1 \leq i \leq n$, Job J_i has 2 parameters, namely, P_i and D_i that denote respectively the profit and deadline associated with J_i. Note that D_i is a positive integer from set $\{1, \ldots, n\}$. Job J_i will give you profit P_i if it is executed within its deadline D_i. For example, suppose Job J_i has deadline $D_i = 5$ and profit $P_i = 2023$. If job J_i is executed during 5th minute or earlier, it will give you profit of rupees 2023. But if it is executed during 6th minute or later, you will not get any profit from this job. You have to schedule all the jobs (assigning each job to a unique minute from $\{1, \ldots, n\}$) such that you get the maximum total profit. Design a polynomial time algorithm to compute such a schedule.

Note: You need to just describe the algorithm in plain English (preferable) or pseudocode. There is no need to prove its correctness or analyse its time complexity.

Answer: Let J be the set of all n jobs.
Let d be maximum of all the D_i's.
Let X_d be the set of jobs which can be executed in the d^{th} interval without exceeding its deadline.

→ Now if $|X_d| > 0$, then execute the job with maximum profit in the d^{th} interval and remove that job from set J. After this update d to d-1 and repeat again untill reached the 1st interval.

→ If $|X_d| == 0$, then just update d to d-1 and repeat again untill reached the 1st interval.

$\boxed{4}$

All the jobs executed in this manner will lead to maximum profit.

You are not scheduling all the jobs.

Problem 2

Attempt any one of the following problems.

1. Easy Problem (*3 marks*)
   Suppose $G = (V, E)$ is a directed acyclic graph (DAG) on n vertices and m edges. We carry out DFS traversal of G. Let $D(v)$ and $F(v)$ denote respectively the start time and the finish time of the DFS traversal of vertex v. The following lemma states a necessary condition on the start time or finish time of vertices u and v if (u, v) is an edge in G. Fill in the blank properly and provide the proof of the lemma in the box that follows.

Lemma 0.1 *If $(u, v) \in E$, then* $F(u) > F(v)$

Proof: Theorem-1: There will not be any path from v to u.
$\Rightarrow$ this will be because the path and edge (u, v) will constitute a cycle and that is not possible in a DAG.
How,

① If DFS of u called before v } As v is reachable from u (edge) the DFS of u will be called ~~[illegible]~~ and finish inside DFS of u so, $D(u) < D(v) < F(v) < F(u)$

② If DFS of v is called before u } u is not reachable from u (theorem-1) so, DFS of u will be called after DFS of v has finished, So, $D(v) < F(v) < D(u) < F(u)$.

In both the cases we can observe that $F(u) > F(v)$.

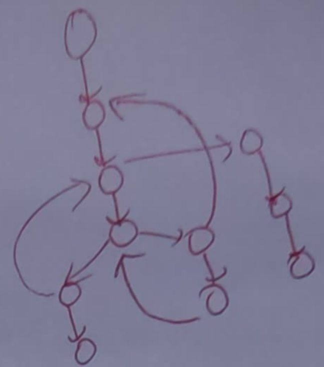
2

In the following box, describe the algorithm in plain English (preferable) or pseudocode for computing topological ordering of G using only the start time or the finish time of vertices.

Answer: We start doing the DFS of graph and keep maintaining the $D(v)$ and $F(v)$ for all $v \in V$. Now the vertex in the decreasing order of finish time will give ~~the~~ a valid topological ordering.
So the first finished vertex goes to the right end of order and ~~sub~~ ~~[illegible]~~ subsequent finished ones will added to the ~~[illegible]~~ left of last one. So this order (from left to Right) will result in a topological order.

1

Ex:



1
1

2. Hard Problem (*6 marks*)
Let $G = (V, E)$ be a directed graph on $n = |V|$ vertices and $m = |E|$ edges. There is a vertex $s \in V$ which has a path to every other vertex in V. We carry out a DFS traversal of G starting from s. You are given the resulting DFS tree T rooted at s. You are also given the start time $D(v)$ and the finish time $F(v)$ of the DFS traversal for each vertex $v \in V$. A path P in G is said to be a proper path if it satisfies the following properties.

- P has at most 1 backward edge.
- P has 0 or more tree edges.

Note that $\langle v \rangle$ is also a proper path from v to v. We now define LowPt(v) of a vertex $v \in V$ as follows.

$$\text{LowPt}(v) = \max(\text{DFN}(w) \mid \text{there is a proper path from } v \text{ to } w.)$$

Design an $O(m + n)$ time algorithm to compute LowPt(v) for all $v \in V$.

Note: Please read the properties of the proper path carefully to avoid any wrong interpretation. You need to just describe the algorithm in plain English (preferable) or pseudocode. There is no need to prove its correctness or analyse its time complexity.

Answer:
Let G' be the strongly connected component graph of G (as discussed in class) which will be a DAG.
Let $P_{SCC}(v)$ be the max DFN of any ~~vest~~ vertex in the SCC of v.
T be the topological ordering of G'.
We start processing T from right to left.
For the rightmost SCC, the LowPt(v) of all the vertex v will be $P_{SCC}(v)$. Now temp be $P_{SCC}(v)$. (temp ← $P_{SCC}(v)$ when v is of the last SCC).
For any vertex v in second last SCC, $P_{SCC}(v)$ is max($P_{SCC}(v)$, temp). We again update temp to max(temp, $P_{SCC}(v)$).
We repeat till the beginning (covering all vertex) of G while traversing T.

0

This doesn't work out.

Trace gives wrong answer.

4