CMPSC 465 Data Structures & Algorithms
Spring 2021 Paul Medvedev and Chunhao Wang

Final

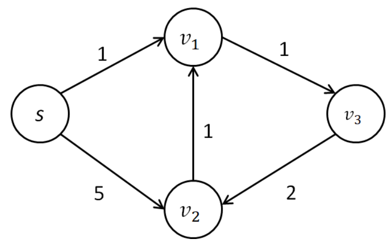
Complete by: May 6th, 9:40 am

Instructions:

- Please log into the regular lecture Zoom meeting.
- If you have a question during the exam, you may ask the Instructor privately via Zoom chat.
- Instructor will announce any major corrections vocally over Zoom.
- All clarifications and corrections will be placed in `https://docs.google.com/document/d/1TJCF7R8bNG8tWQQ6Ogs9YraS3Pr8Pc1aXBq-yirhO0o/edit?usp=sharing`.
- Write your solutions by hand. Typed solutions will not be accepted. You may handwrite on a tablet as well.
- **At 9:40am, you must put your pens down. You have until 9:50 to upload your solutions to Gradescope.**
- You must use a scanning app and not just take pictures.
- You must use the template solution sheet provided on Gradescope.

1. (10 pts.) Shortest paths

Lets define the *pathwidth* of a weighted directed graph G as the integer t such that all the shortest paths from s to all other nodes have at most t edges. For example, for the graph below, the shortest path weight from s to v_1 is 1, from s to v_2 is 4, and from s to v_3 is 2. The lengths of the respective shortest paths are 1, 3, and 2, respectively. The pathwidth t is then 3.



(a) (8pts) Suppose that you are given the pathwidth as a parameter t to your algorithm. Give pseudo-code for an $O(|E| \cdot t)$ algorithm for solving the single-source shortest path problem. Your algorithm should take as input a weighted directed graph G, a vertex s, and a non-negative integer t. Your algorithm should return a *dist* vector such that $dist(u)$ is the shortest path distance from s to u, for all vertices u. You can assume that there are no negative weight cycles in G.

(Note: 1) you must write pseudo-code to receive full credit, and 2) you do not need to explain the algorithm correctness.)

(b) (2pts) Now suppose that t is not given as a parameter to your algorithm. Briefly explain how your algorithm can be modified so that its asymptotic running time is $O(|E| \cdot pathwidth(G))$. You can describe the change in either plain English or by writing pseudo-code.

Solution

(a) Here is the algorithm

Algorithm 1: SP(G, s, t)

```
for for each vertex u in G do
    dist(u) = ∞
end
dist(s) = 0
for i = 1 to t do
    for each vertex u do
        for each v in the adjacency list of u do
            update(u, v)
        end
    end
end
```

(b) For each iteration of the outer for loop, keep track of whether you have performed at least one updated which resulted in a change of *dist*. As soon as you complete a whole iterations without any changes, you can stop the algorithm.

2. (10 pts.) Dynamic programming

Suppose you are running a software development company. There are N programmers and M projects. Each programmer is assigned to work on one of the projects. Each of the M projects will produce a different profit depending on how many programmers are assigned to this project. Denote the profits of project i with j programmers by $P_{i,j}$. Design a dynamic programming algorithm to find the assignment of programmers that produces the most profit. What is the running time? Justify the correctness of your algorithm.

Solution: Let $B(m, j)$ be the best cost of the best assignment of j programmers to the first m projects. An optimal solution of j programmers to the first m projects assigns some number of programmers, k, to project m, and must assign $j-k$ programmers with maximal profit to the first $m-1$ projects, which, by induction, has cost $B(m-1, j-k)$. Thus, we have

$$B(m, j) = \max_{0 \le k \le j} \{P_{m,k} + B(m-1, j-k)\}$$

Furthermore, the base case is $B(0,0) = 0$, as the profit of assigning no programmer to no project is 0.

The time to compute each value $B(m, j)$ is $O(N)$, and since there are MN subproblems, the total running time is (MN^2).

To recover the assignment, one backtracks from $m = M$ and $j = N$ and assigns k programmers to project m is $B(m, j) = B(m-1, j-k) + P_{m,k}$ and continues to backtrack from $m-1, j-k$.