Problem Set 3

data structures and algorithms 2 - fall 2023

yourid

due october 20, 2023 at 11:59 pm

Collaboration Policy: You are encouraged to collaborate with up to 4 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite them. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators: list your collaborators

Sources: list your sources

problem 1 *Scenic Highways*

In this problem we'll describe something similar to solving a problem with Dijkstra's algorithm, and ask you some questions about this new problem and a possible algorithm.

You are taking a driving vacation from town to town until you reach your destination, and you don't care what route you take as long as you maximize the number of scenic highways between towns that are part of your route. You model this as a weighted graph G, where towns are nodes and there is an edge for each route between two towns. An edge has a weight of 1 if the route is a scenic highway and a 0 if it is not.

You need to find the path between two nodes s and t that maximizes the number of scenic highways included in the path. You design a greedy algorithm that builds a tree like Dijkstra's algorithm does, where one node is added to a tree at each step. From the nodes adjacent to the tree-nodes that have already been selected, your algorithm uses this greedy choice:

Choose the node that includes the most scenic highways on the path back to the start node s.

When node t is added to the tree, stop and report that path found from s to t. Reminder (in case it helps you): the greedy choice for Dijkstra's algorithm was:

Choose the node that has the shortest path back to the start node s.

1. What is the key-value stored for each item in the priority-queue? Also, what priority-queue operation must be used when we need to update a key for an item already stored in the priority-queue?

2. Draw one or more graphs to convince yourself that the greedy approach describes above does not work. Then explain in words as best you can the reason this approach fails or a situation that will cause it to fail (i.e., a counterexample).

Solution:

problem 2 *As You Wish*

Buttercup has given Westley a set of n tasks $T = t_1, \ldots, t_n$ to complete on the farm. Each task $t_i = (d_i, w_i)$ is associated with a deadline d_i and an estimated amount of time w_i needed to complete the task. To express his undying love to Buttercup, Westley strives to complete all the assigned tasks as early as possible. However, some deadlines might be a bit too demanding, so it may not be possible for him to finish a task by its deadline; some tasks may need extra time and therefore will be completed late. Your goal (inconceivable!) is to help Westley minimize the

deadline overruns of any task; if he starts task t_i at time s_i, he will finish at time $f_i = s_i + w_i$. The deadline overrun (or lateness) of tasks—denoted L_i—for t_i is the value

$$L_i = \begin{cases} f_i - d_i & \text{if } f_i > d_i \\ 0 & \text{otherwise} \end{cases}$$

Design a polynomial-time algorithm that computes the optimal order W for Westley to complete Buttercup's tasks so as to minimize the maximum L_i across all tasks. That is, your algorithm should compute W that minimizes

$$\min_{W} \max_{i=1,\ldots,n} L_i$$

In other words, you do not want Westley to complete *any* task *too* late, so you minimize the deadline overrun of the task completed that is most past its deadline.

1. What is your algorithm's greedy choice property?

   Solution:

2. What is the run-time of your algorithm?

   Solution:

3. Consider the following example list of tasks:

   $$T = \{(2,2), (11,1), (8,2), (1,5), (20,4), (4,3), (8,3)\}$$

   List the tasks in the order Westley should complete them. Then argue why there is no better result for the given tasks than the one your algorithm (and greedy choice property) found.

   Solution:

PROBLEM 3 *Course Scheduling*

The university registrar needs your help in assigning classrooms to courses for the spring semester. You are given a list of n courses, and for each course $1 \leq i \leq n$, you have its start time s_i and end time e_i. Give an $O(n \log n)$ algorithm that finds an assignment of courses to classrooms which minimizes the *total number* of classrooms required. Each classroom can be used for at most one course at any given time. Prove both the correctness and running time of your algorithm.

Solution:

PROBLEM 4 *Ubering in Arendelle*

After all of their adventures and in order to pick up some extra cash, Kristoff has decided to moonlight as the sole Uber driver in Arendelle with the help of his trusty reindeer Sven. They usually work after the large kingdom-wide festivities at the palace and take everyone home after the final dance. Unfortunately, since a reindeer can only carry one person at a time, they must take each guest home and then return to the palace to pick up the next guest.

There are n guests at the party, guests $1, 2, \ldots, n$. Since it's a small kingdom, Kristoff knows the destinations of each party guest, $d_1, d_2, \ldots, d_n$ respectively and he knows the distance to each guest's destination. He knows that it will take t_i time to take guest i home and return for the next guest. Some guests, however, are very generous and will leave bigger tips than others; let T_i be the tip Kristoff will receive from guest i when they are safely at home. Assume that guests are willing to wait after the party for Kristoff, and that Kristoff and Sven can take guests home in any

order they want. Based on the order they choose to fulfill the Uber requests, let D_i be the time they return from dropping off guest i. Devise a greedy algorithm that helps Kristoff and Sven pick an Uber schedule that minimizes the quantity:

$$\sum_{i=1}^{n} T_i \cdot D_i.$$

In other words, they want to take the large tippers the fastest, but also want to take into consideration the travel time for each guest. Prove the correctness of your algorithm. (Hint: think about a property that is true about an optimal solution.)

Solution: