Problem Set 1

yourid

data structures and algorithms 2 - fall 2023

due september 8, 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 *Order class proof using limits*

Prove that $n^k \in o(a^n)$ for $a > 1$. Use the limit definition of the order class. If you find you need to use L'Hôpital's rule, note that $(a^n)' = (\log a)a^n$.

Solution:

PROBLEM 2 *FlyMe Airlines*

An airline, FlyMe Airlines, is analyzing their network of airport connections. They have a graph $G = (A, E)$ that represents the set of airports A and their flight connections E between them. They define $hops(a_i, a_k)$ to be the smallest number of flight connections between two airports. They define $maxHops(a_i)$ to be the number of hops to the airport that is farthest from a_i, i.e. $maxHops(a_i) = max(hops(a_i, a_j)) \, \forall a_j \in A$.

The airline wishes to define one or more of their airports to be "Core 1 airports." Each Core 1 airport a_i will have a value of $maxHops(a_i)$ that is no larger than any other airport. You can think of the Core 1 airports as being "in the middle" of FlyMe Airlines' airport network. The worst flight from a Core 1 airport (where "worst" means having a large number of connections) is the same or better than any other airport's worst flight connection (i.e. its $maxHops()$ value).

They also define "Core 2 airports" to be the set of airports that have a $maxHops()$ value that is just 1 more than that of the Core 1 airports. (Why do they care about all this? Delays at Core 1 or Core 2 airports may have big effects on the overall network performance.)

Your problem: Describe an algorithm that finds the set of Core 1 airports and the set of Core 2 airports. Give its time-complexity. The input is $G = (A, E)$, an undirected and unweighted graph, where $e = (a_i, a_j) \in E$ means that there is a flight between a_i and a_j. Base your algorithm design on algorithms we have studied in this unit of the course.

Solution:

PROBLEM 3 *Counting Shortest Paths*

Given a graph $G = (V, E)$, and a starting node s, let $\ell(s, t)$ be the length of the shortest path in terms of number of edges between s and t. Give a clear description of an algorithm that computes the number of distinct paths from s to t that have length exactly $\ell(s, t)$.

Solution:

PROBLEM 4 *Coffee*

Charlottesville is known for some of its locally-roasted coffees, each with its own unique flavor combination. Suppose a group of coffee enthusiasts are given n samples $c_1, c_2, ..., c_n$ of freshly-brewed Charlottesville coffee by a coffee-skeptic. Each sample is either a *Milli* roast or a *Shenandoah Joe* roast. The enthusiasts are given each pair (c_i, c_j) of coffees to taste, and they must collectively decide whether: (a) both are the same brand of coffee, (b) they are from different brands, or (c) they cannot agree (i.e., they are unsure whether the coffees are the same or different). Note: all n^2 pairings are tested, including $(c_i, c_i), (c_i, c_j)$, and (c_j, c_i), but not all pairs have "same" or "different" decisions.

At the end of the tasting, suppose the coffee enthusiasts have made m judgments of "same" or "different." Give an algorithm that takes these m judgments and determines whether they are *consistent*. The m judgments are *consistent* if there is a way to label each coffee sample c_i as either *Shenandoah Joe* or *Milli* such that for every taste-comparison (c_i, c_j) labeled "same," both c_i and c_j have the same label, and for every taste-comparison labeled "different," both c_i and c_j are labeled differently. Your algorithm should run in $O(m + n)$ time. Prove the correctness and running time of your algorithm. Note: you do *not* need to determine the brand of each sample c_i, only whether the coffee enthusiasts are consistent in their labelings.

Solution:

PROBLEM 5 *Ancient Population Study*

Historians are studying the population of the ancient civilization of *Algorithmica*. Unfortunately, they have only uncovered incomplete information about the people who lived there during Algorithmica's most important century. While they do not have the exact year of birth or year of death for these people, they have a large number of possible facts from ancient records that say when a person lived relative to when another person lived.

These possible facts fall into two forms:

- The first states that one person died before the another person was born.
- The second states that their life spans overlapped, at least partially.

The Algorithmica historians need your help to answer the following questions. First, is the large collection of uncovered possible facts internally consistent? This means that a set of people could have lived with birth and death years that are consistent with all the possible facts they've uncovered. (The ancient records *may not be accurate*, meaning all the facts taken together cannot possibly be true.) Second, if the facts are consistent, find a sequence of birth and death years for all the people in the set such that all the facts simultaneously hold. (Examples are given below.)

We'll denote the n people as $P_1, P_2, \ldots, P_n$. For each person P_i, their birth-year will be b_i and their death-year will be d_i. (Again, for this problem we do not know and cannot find the exact numeric year value for these.)

The possible facts (input) for this problem will be a list of relationships between two people, in one of two forms:

- P_i *prec* P_j (indicates P_i died before P_j was born)
- P_i *overlaps* P_j (indicates their life spans overlapped)

If this list of possible facts is not consistent, your algorithm will return "not consistent". Otherwise, it will return a possible sequence of birth and death years that is consistent with these facts.

Here are some examples:

- The following facts about $n = 3$ people are **not** consistent: P_1 *prec* P_2, P_2 *prec* P_3, and P_3 *prec* P_1.
- The following facts about $n = 3$ people **are** consistent: P_1 *prec* P_2 and P_2 *overlaps* P_3. Here are two possible sequences of birth and death years:
  $b_1, d_1, b_2, b_3, d_2, d_3$
  $b_1, d_1, b_3, b_2, d_2, d_3$
  (Your solution only needs to find one of any of the possible sequences.)

Your answer should include the following. Clearly and precisely explain the graph you'll create to solve this problem, including what the nodes and edges will be in the graph. Explain how you'll use one or more of the algorithms we've studied to solve this graph problem, and explain why this leads to a correct answer. Finally, give the time-complexity of your solution.

Solution: