CSE Ph.D. Qualifying Exam, Spring 2025
Algorithms

Instructions:

Please answer three of the following four questions. All questions are graded on a scale of 10. If you answer all four, all answers will be graded and the three lowest scores will be used in computing your total. **This exam is closed-book, closed-notes.**

Questions:

1. **Greedy: 24-Hour Scheduling**

   Consider a processor that can operate 24 hours a day, every day. People submit requests to run *daily jobs* on the processor. Each such job comes with a *start time* and an *end time*; if the job is accepted to run on the processor, it must run continuously, every day, for the period between its start and end times. (Note that certain jobs can begin before midnight and end after midnight.)

   Given a list of n such jobs, your goal is to accept as many jobs as possible (regardless of their length), subject to the constraint that the processor can run at most one job at any given point in time. Provide an algorithm to do this with a running time that is polynomial in n. Prove that your algorithm returns the optimal solution. You may assume for simplicity that no two jobs have the same start or end times.

   Example. Consider the following four jobs, specified by (*start-time, end- time*) pairs.

   (6pm, 6am), (9pm, 4am), (3am, 2pm), (1pm, 7pm).

   The optimal solution would be to pick the two jobs (9pm, 4am) and (1pm, 7pm), which can be scheduled without overlapping.

2. **Dynamic Programming: Longest Palindromic Sequence** A palindrome is a nonempty string over some alphabet that reads the same forward and backward. Examples of palindromes are all strings of length 1, `civic`, `racecar`, and `aibohphobia` (fear of palindromes).

   In this problem, we consider the problem of finding the longest palindrome that is a *subsequence* of a given input string. For example, given the input `character`, your algorithm should return `carac`.

   Let n be the length of the string.

   (a) Give a brute force algorithm that takes $O(2^n)$ time to solve the longest palindrome subsequence problem. Provide pseudocode to illustrate your algorithm. Analyze its time complexity.

   (b) Give a dynamic-programming algorithm that takes $O(n^2)$ time and $O(n)$ space solve the longest palindrome subsequence problem. Provide pseudocode to illustrate your algorithm. Analyze its time complexity.

3. **Dynamic Programming: Buy More**

   George is the new manager of the local Buy More, a consumer-electronics store whose main revenue source comes from selling computers. George has accurate predictions of the quantity of computer

sales in the next n months, where d_i denotes the number of sales in month i. Assume that sales occur at the beginning of each month, with unsold computers stocked in inventory until the beginning of the next month. It costs C to keep a single computer in stock for a month. The Buy More's inventory can keep up to I computers in stock. Each month George can choose to order any number of computers (which conveniently arrive before that month's sales), but can only keep up to I computers in stock at the end of the month. Each time that George submits an order, there is a fixed shipment cost of R. George currently has no computers in inventory. Help George design an algorithm that is polynomial in n and I to meet the n monthly demands (d_i), whilst minimizing the cost.

Note that the cost consists of both the fixed cost R for submitting an order and the marginal cost C for each computer kept in inventory per month. For every month i, the demand d_i must be met. No more than I computers may be left in inventory at the end of the month.

Let n be the total number of months.

a) Give the recurrence relation for minimizing the total cost while meeting the demand, including base cases. Your answer should include R, I, d_i, and C.

b) Give the pseudocode of a bottom-up approach to implement your dynamic programming algorithm. Also include the pseudocode to find the optimal solution which tells George how many computers he should order each month. Show that your algorithm achieves time and space complexity $O(nI)$.

Examples

$$n = 4, \quad \{d_0 = 2, d_1 = 3, d_2 = 1, d_3 = 2\}, \quad R = 15, \quad C = 2, \quad I = 5$$

- An order for 8 computers cannot be placed at the beginning of month 0 because this would result in $8 - d_0 = 6$ computers leftover at the end of month 0, which exceeds the inventory limit I.
- If an order size of 2 is placed for month 0, 3 for month 1, 1 for month 2 and 2 for month 3, the total cost will be: $R + R + R + R = 15 + 15 + 15 + 15 = 60$
- If orders are placed of size 3 for month 0, 4 for month 1, 0 for month 2 and 1 for month 3, the total cost will be: $R + 1C + R + 2C + 1C + R = 15 + 2 + 15 + 4 + 2 + 15 = 53$

4. **NP-complete: Campus Tours**

The office of Administration at Georgia Tech is planning for the Campus Visit and Tour Experience Program in which various Campus visit tours are offered to groups of prospective students and their families to become more familiar with the Georgia Tech buildings and amenities. Each tour starts from Georgia Tech Student Center and after visiting some (not necessarily all) buildings and landmarks, in a particular order, returns to the first point. Therefore, if we represent the map of the campus with a directed graph G, each node represents a building or landmark, and each edge shows a directed path from one point to another. Accordingly, the set of the various tours can be represented as a set $S = \{R_1, R_2, \ldots, R_n\}$, where n is the number of the various tours and R_i is the route (circuit) that the visitors pass along tour i, starting from and ending at the Student Center. Now the CampusTours problem is defined as the following:

Given the Directed Graph $G = (V, E)$, the set of the campus tours $S = \{R_1, R_2, \ldots, R_n\}$, and an integer number k, is there a subset of k tour(s) in S such that a visitor group can be sure that they have seen all the buildings and landmarks of Georgia Tech?
(*Note: It is possible to visit a node more than once during these k tours.)*

Prove that the `Campus Tours` problem is NP Complete. You can use the fact that `Vertex Cover` problem is NP-complete.

VertexCover(G, k) is the problem of deciding whether in graph G there exists a subset of vertices S of size at most k such that all edges in G have at least one endpoint in the selected vertex set S. A *simple cycle* in a graph is a cycle with no repeated vertices (except for the start and end vertex).

Hint: When constructing the graph G' for the CampusTours problem, you may consider using edges of the graph G in the VertexCover problem as vertices of G'.