Q1 [20 Points] Fruit Frenzy, Revisited

In the dilemma that you helped Dinosaur Bobby solve in Assignment 1, there were \( n \) gardens. Each garden \( i \) was described by two integers: \( e_i \), the easiness of reaching garden \( i \), and \( q_i \), the number of fruits in garden \( i \). Garden \( i \) was called a smarter choice than garden \( j \) if \( e_i > e_j \) and \( q_i > q_j \).

Bobby is now employed at a circus and wants your help again in selecting a smart garden. But a couple of things are different this time around.

1. First, Bobby wants to find a sequence of gardens (\( i_1, i_2, \ldots, i_k \)) such that each garden \( i_{r+1} \) is a smarter choice than the previous garden \( i_r \); such a sequence is called a chain. Essentially, Bobby wants to impress his manager by saying “Look, I could’ve selected garden \( i_1 \). But I found a smarter choice \( i_2 \). And then an even smarter choice \( i_3 \)…And then an even smarter choice \( i_k \). That’s why I picked garden \( i_k \).”

2. Each garden \( i \) additionally has a non-negative weight \( w_i \). And Bobby wants you to find a chain with the highest total weight.

Design a dynamic programming algorithm to help Bobby.

(a) [5 Points] Define an array or arrays storing the necessary information from subproblems. Clearly define what each entry means, and how you would compute the final answer given the array entries.

(b) [5 Points] Write a Bellman equation and briefly justify its correctness.

(c) [5 Points] Describe a bottom-up implementation to compute the array entries.

(d) [5 Points] Analyze the worst-case running time and space complexity of your algorithm.
Q2 [25 Points] The Manager Wants the Tallest Giraffes

You are a new hire at the circus and your manager has assigned you your first task. There are \( n \) giraffes in a line, and giraffe \( i \) has height \( H[i] \). The manager wants you to study their heights carefully. Later, the manager will ask you a series of queries of the form “What is the maximum height across giraffes \( i \) through \( j \)?”, where \( i \) and \( j \) (with \( i \leq j \)) will vary from one query to the next. You ought to be able to answer these queries rapidly.

Of course, there are only \( \Theta(n^2) \) possible queries, so if you pre-compute all the \( \Theta(n^2) \) answers and store them in a table, then you can answer any query in \( O(1) \) time simply by looking at your table.

However, the manager wants you to be quicker. You need to study the giraffes’ heights (and pre-compute whatever table you want) in \( O(n \log n) \) time. Afterwards, you still need to be able to answer any query in \( O(1) \) time simply by looking at your table entries.

(a) [15 Points] Define the table that you would pre-compute. Clearly define what each entry means and how you would answer any query in \( O(1) \) time if all the table entries were pre-computed.

(b) [10 Points] Write a Bellman equation for the table entries, justify its correctness, and provide a bottom-up implementation.

Q3 [20 Points] Scheduling Performances

Having moved up the hierarchy, you are now the circus manager. Your first task is to schedule performances for the next \( n \) days. Here are some inputs and constraints:

- There are \( p \) prepared performances. Each performance requires one ringmaster.
- There are \( m \) ringmasters. Ringmaster \( j \) can only lead performances in the set \( P_j \).
- Exactly \( k \) of the prepared performances should be scheduled on each day.
- No performance should be scheduled twice in the same day (or the audience will get bored!).
- No ringmaster should be scheduled to lead more than \( L \) performances across all \( n \) days (as you’re not paying them all that much!).

(a) [10 Points] Use the network flow framework to design an efficient algorithm that either finds a feasible schedule or declares that there isn’t one. Clearly define each component of the network.

(b) [10 Points] Justify the correctness of your algorithm. Analyze the worst-case running time of the naïve Ford-Fulkerson algorithm on your network from part (a).

Q4 [25 Points] Scheduling Staff Duties

Performances are just the “front end”. You also need to schedule the staff to perform various duties during the \( n \)-day event in the “back end” for the circus to run smoothly.

Exactly \( a_i \) staff members should be scheduled on day \( i \). Each staff member \( j \) is available on a subset of days \( A_j \subseteq \{1, \ldots, n\} \), and can be scheduled on at most \( L \) days from \( A_j \).
(a) [10 Points] Use the network flow framework to design an efficient algorithm that either finds a feasible schedule or declares that there isn’t one. Clearly define each component of the network. Justify the correctness of your algorithm and analyze the worst-case running time of the naïve Ford-Fulkerson algorithm on your network.

(b) [15 Points] You realize that these constraints are too tight and ask the staff members for help. They agree to the following: each staff member $j$, in addition to coming in on at most $L$ days from $A_j$, can also come in on at most $L'$ days outside of $A_j$ (i.e. from $\{1, \ldots, n\} \setminus A_j$). Use the network flow framework to design an efficient algorithm under these relaxed constraints. Justify its correctness and the worst-case running time when using the naïve Ford-Fulkerson algorithm.