Homework 8

Due Wednesday, April 6th at 11:59 pm ET on Gradescope

Solution guidelines. For problems that require you to provide an algorithm, you must give the following:

1. a precise description of the algorithm in English and pseudocode (*),
2. a proof of correctness,
3. an analysis of the asymptotic running time and space.

You may use algorithms from class as subroutines. You may also use any facts that we proved in class, e.g. correctness of subroutines, running time of subroutines. You should be as clear and concise as possible in your write-up of solutions.

(*) It is fine if the English description concentrates on the high level ideas and doesn't include all the details. But the reader should not have to figure out your solution solely based on the pseudocode. You can also add comments to your pseudocode, in fact that is best practice. FYI we will share a document on good pseudocode style with you and it will also be discussed in labs and lecture.

Format for writing a solution for a DP problem. Dynamic programming solutions follow a pretty strict pattern. Here we give you a guide for what should be included in a full credit solution.

1. Clearly state what the subproblems are and what corresponds to the final solution. *For WIS write OPT(i) is the value of the max compatible schedule if we consider the first i jobs* $1, 2, \ldots i$ *in increasing order of finish time. OPT(n) is the solution on the entire input.* Note, that for some problems OPT will have 2 variables, e.g knapsack problem.
2. Write and explain the recursive formula to compute OPT(i). A proper proof would include induction on i (or on both variables). However, for these kind of problems it is pretty clear what the inductive steps and assumption are. For this reason we accept if you say that assuming the formula is correct for every lower index, then my formula is correct because XYZ. Don't forget about the border cases in your formula.
3. clearly state the dimensions of the memoization table and what its values are, e.g. M[i][j] = OPT(i,j).
4. Brief pseudocode, short and clean is better. It should be clear from your code what the dimensions of the table are, how you initialize the border case, in what order you iterate over the table to fill the values (both recursive and bottom-up are valid) and how the recursive formula is used. Any preprocessing steps, e.g. order the jobs, should be written too.
5. running time analysis.

Problem 1. *(10 points)*

You are currently half-way through your Spring semester and you already started to make plans for the upcoming Summer. You decided and bought the tickets for large music festival that will take place in Boston. There will be n musicians and your ticket grants you access to all of the concerts, unfortunately you can only attend one concert at the time and there can be no overlaps between concerts you attend. You will therefore have to make a choice of which concerts to attend. As an input you will be given an array **start**[ ] of length n containing the start time for each concert. You will also receive array **duration**[ ] with the duration of each concert. You can assume concerts are indexed 1 through n in the increasing order of their start time.
Your goal is to pick concerts in such manner that you **maximize** the time of your concert attendance. (If you start attending a concert you have to finish it).

1. What algorithm from class could be used to solve this problem? How would you transform the inputs to this problem into the ones needed for the algorithm from class? Make sure you understand how that algorithm works before proceeding.

2. Due to the large number of attendees, the festival organizers have decided to limit the number of concerts each attendee can enter to k (so the input now consists of the arrays `start` and `duration`, and the number k). Your goal is to find a set of at most k concerts that do not overlap, and whose total duration is as long as possible. Design a polynomial-time dynamic programming algorithm for the problem. Please organize your answer as follows:

   (a) Define the set of subproblems that your algorithm will solve. [*Hint:* Consider having one subproblem for each $i \in \{1, ..., n\}$ and $j \in \{1, ..., k\}$. Think carefully about the order in which to consider the tournaments.]

   (b) How many such subproblems are there?

   (c) Suppose we create a table M to store the value of the subproblems we solve. Write a recursive formula to compute `M[i,j]` in terms of other entries of M (that is, the solutions to smaller subproblems). Don't forget the base cases.

   (d) Explain briefly why your formula is correct. (This explanation is really the proof of correctness for your algorithm.)

   (e) Write pseudocode for your algorithm (it should fill the memoization table and find the optimal subset of tournaments).

   (f) show code for how to retrieve the actual games in your optimal schedule. You can do this either as part of the original pseudocode or write a backtracking algorithm to be run after.

   (g) What is the asymptotic running time of your algorithm? (Only write the Θ formula, no proof needed.)

Problem 2. *(10 points)*

You're a pilot for Terrier Airlines and your airline allowed you to plan your monthly schedule for the upcoming flights. You have an option of choosing either long-haul or short-haul flights. Long-haul flights are paid better but regulations require that pilots have a day of rest before a long-haul flight. Short-haul flights on the other hand do not have such requirement. You are able

to do only one flight per day, and you can assume that on the first day of month you're ready to take either long-haul or short-haul flight. Terrier Airlines also provided you with expected rates for flights this month (rates may differ on different dates). So if you're flying long-haul on day i that means you can't have any flights scheduled on day $i-1$. If you're doing short-haul flight there are no such restrictions. Your goal is to earn as much money this month as you can.

1. In the table below you can see the amounts of money you can earn each day. Find an optimal schedule for yourself based on the table. Write both the schedule and how much you will earn.

days	1	2	3	4	5	6	7	8	9	10
Short-haul	\$200	\$300	\$100	\$150	\$400	\$200	\$150	\$300	\$200	\$400
Long-haul	\$350	\$400	\$500	\$300	\$200	\$300	\$500	\$600	\$400	\$600

2. Let $earn(i)$ denote the total amount of money you can earn in the first i days. Write a recursive formula to compute $earn(i)$. Your formula may use the maximum function as well as calls to $earn(j)$ where $0 \leq j < i$.

3. In order to make your computation more efficient you make use of a memoization table E. What value should be stored in $E[i]$ and what is the size of E?

4. Write the pseudocode of an efficient dynamic programming algorithm to compute the values of $earn(i)$ for all i and fill the table E. (As always, analyze your algorithm for running time and correctness.)

5. Write pseudocode that takes the filled table E as input and prints the schedule along with the total amount that the pilot should use to earn as much as possible. (As always, analyze your algorithm for running time and correctness.)