CS 170 HW 8

Due **2021-03-16, at 10:00 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, write “none”.

DP solution writing guidelines:

Try to follow the following 3-part template when writing your solutions.

- Define a function $f(\cdot)$ in words, including how many parameters are and what they mean, and tell us what inputs you feed into f to get the answer to your problem.
- Write the “base cases” along with a recurrence relation for f.
- Prove that the recurrence correctly solves the problem.
- Analyze the runtime and space complexity of your final DP algorithm.

2 DP Coding

This semester, we are trying something new: questions which involve coding.

(a) In this question, we consider the maximum subarray sum problem. This is a classic dynamic programming problem and also happens to be asked somewhat frequently during coding interviews.

The setup of the problem is as follows: You are given an array A of numbers. You want to find the subarray sum of the subarray, i.e. slice of the array, with the largest sum. In other words, you want to find

$$\max_{i,j} \sum_{k=i}^{j} A[k].$$

For example, the array $[-1, 2, 6, -3, 5, -6, 3]$ has a maximum subarray sum of 10, which is achieved by the subarray $[2, 6, -3, 5]$.

Describe and analyze a dynamic programming algorithm to solve the Maximum Subarray Sum problem.

(b) This link will take you to a python notebook, hosted on the Berkeley datahub, in which you will implement your dynamic programming algorithm for the problem above, so you can see what it looks like in actual code. Once you have finished, download a PDF of your completed notebook via File → Download as → PDF via HTML, and append the downloaded pdf to the rest of your homework submission. Be careful when selecting pages on gradescope.

Note: Datahub does not guarantee 100% reliability when you save your notebook, and recommends downloading a local copy occasionally to backup progress (via File → Download as → Notebook (.ipynb)).

3 Spaceship

A spaceship uses some *oxidizer* units that produce oxygen for three different compartments. However, these units have some failure probabilities.

Because of differing requirements for the three compartments, the units needed for each have somewhat different characteristics.

A decision must now be made on just *how many* units to provide for each compartment, taking into account design limitations on the *total* amount of *space*, *weight* and *cost* that can be allocated to these units for the entire ship. Specifically, the total space for all units in the spaceship should not exceed 500 cubic inches, the total weight should not exceed 200 lbs and the total cost should not exceed $400,000$ dollars.

The following table summarizes the characteristics of units for each compartment and also the total limitation:

	Space (cu in.)	Weight (lb)	Cost ($)	Probability of failure
Units for compartment 1	40	15	30,000	0.30
Units for compartment 2	50	20	35,000	0.40
Units for compartment 3	30	10	25,000	0.20
Limitation	500	200	400,000	

The objective is to *minimize the probability* of all units failing in all three compartments, subject to the above limitations and the further restriction that each compartment have a probability of no more than 0.05 that all its units fail.

Formulate the *integer programming model* for this problem. An integer programming model is the same as a linear programming model with the added functionality that variables can be forced to be integers.

Side note: Integer programming is often intractable, so we use a linear program as a heuristic. We can take out the integrality constraints and change our model into a linear program. We then round the solution and make sure none of constraints have been violated (you should think about why this won't always give us the optimum)

4 Motel Choosing (optional)

You may submit your solution to this problem if you wish it to be graded, but it will be worth no points.

You are traveling along a long road, and you start at location $r_0 = 0$. Along this road, there are n motels at location $\{r_i\}_{i=1}^n$ with $0 < r_1 < r_2 < \cdots < r_n$. The only places you may stop are these motels, but you can choose which to stop at. You must stop at the final motel (at distance r_n), which is your destination.

Ideally, you want to travel exactly T miles a day and stop at a motel at the end of the day, but this may not be possible (depending on the spacing of the motels). Instead, you receive a *penalty* of $(T - x)^8$ each day, if you travel x miles during the day. The goal is to plan your stops to minimize the total penalty (over all travel days).

Describe and analyze an algorithm that outputs the minimum penalty, given the locations $\{r_i\}$ of the motels and the value of T.