Assignment 4

CS-GY 6033 INET Spring 2025

Due date: 11:55pm on April 12th 2025 on Gradescope

Instructions:

Dynamic Programming

For each of the dynamic programming problems below, you must:

- Define the table you are using, and define each entry
- Describe how to initialise your table
- Describe the relationship between the entries in your table
- Describe which entry in the table stores your final result
- Provide the pseudocode that shows how to fill up your table
- Justify the runtime of your algorithm

Question 1

13 points

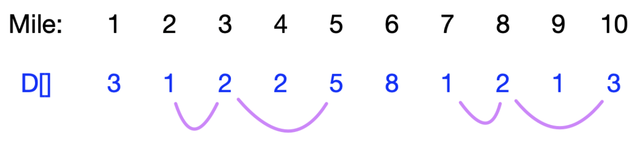
(a) In this question, you will provide a new solution to the toll-both problem from our practice set. In this version of the solution, you must use the table $T[1,\ldots,n,1,\ldots,n]$ where entry $T[i,j]$ represents the minimum cost of a route from position $(1,1)$ *to* position (i,j). Be sure to include all the steps of your DP solution. Note that the original input to this problem is the same as that for the practice problem.

(b) Provide the pseudo-code that outputs the coordinates of the minimum-cost route, using the table $T[]$ from above.

Question 2

15 points

(a) Consider a hiking trail that goes from mile marker 1 to mile marker n. At each marker, there is exactly ONE rock. Each rock must be carried a certain number of miles. The input to the problem is $D[1,2,\ldots,n]$ where $D[i]$ is the **distance** that we much carry rock i. For example, if $D[1]=4$, the rock at marker 1 must be carried at least 4 miles. At each mile marker, we have the choice of picking up the rock, or continue walking with no rock. We cannot carry more than one rock at a time, but we are allowed to drop a rock, and pick a new one up at the same mile. Any rock that we pick up MUST be carried the minimum number of miles. In the example below, the rock pick ups and drop offs show that we are able to transport four rocks.



Your job:

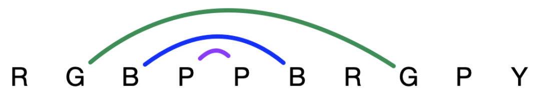
Write a DP solution that solves the problem of finding the **maximum** number of rocks that can be successfully transported during the hike. Be sure to include a properly defined DP table, describe the initialization, how the table is filled up, the pseudo-code, and the runtime.

(b) Update the above solution so that you also print out the optimal solution: this consists of the the mile markers where you pick up rocks.

Question 3

12 points

Suppose we have a sequence of characters in the array $C[1, 2, \ldots, n]$ where each character represents a **color**. The characters are selected from the set $\{R, B, G, Y, P\}$. An example of such a sequence is $RBBGYPRGBYPYPY$. Using these characters, we would like to draw a **rainbow**, where an arc of a certain color can be made be connecting two of the same color. For a proper rainbow to be drawn, colored arcs cannot cross each other! An example of a rainbow is shown below:



Arcs of different colors have different values. Yellow arcs are worth \$100. Blue arcs are worth \$200. Green arcs are worth \$300. Pink arcs are worth \$400. Red arcs are worth \$500.

Your Job:

Update the relevant DP problem from class so that it returns the **maximum value** of a rainbow that is drawn using from the characters in $C[1, 2, \ldots, n]$.

You must properly define the DP table, explain the initialisation, justify how you fill in the entries, provide the pseudo-code, clearly show which value is returned, and justify the runtime.

Question 4

15 points

(a) Suppose we are given a set of n boxes, where each box has an associated *weight* and a *height*. The goal is to pile up the boxes in such a way that we make a tower of total height at least M, by selecting the boxes with the **minimum total weight**. The constraint is that the boxes must be piled in *decreasing* order of height (for example, a box of height 5 can be placed on a box of height 6, however the reverse is not possible). Note that the boxes cannot be rotated, because base is distinct from the height.

The input is given in the arrays: $H[1, 2, \ldots, n]$, $W[1, 2, \ldots, n]$, where $H[i]$ is the height of box i, and $W[1, 2, \ldots, n]$ is the weight of box i.

Design a dynamic programming solution that solves the problem of determining the minimum total weight of the boxes that create a tower of height T.

(b) Write the pseudo-code for the procedure that outputs the indices of the boxes that are used in the highest tower. The output must be in the order from largest to smallest.

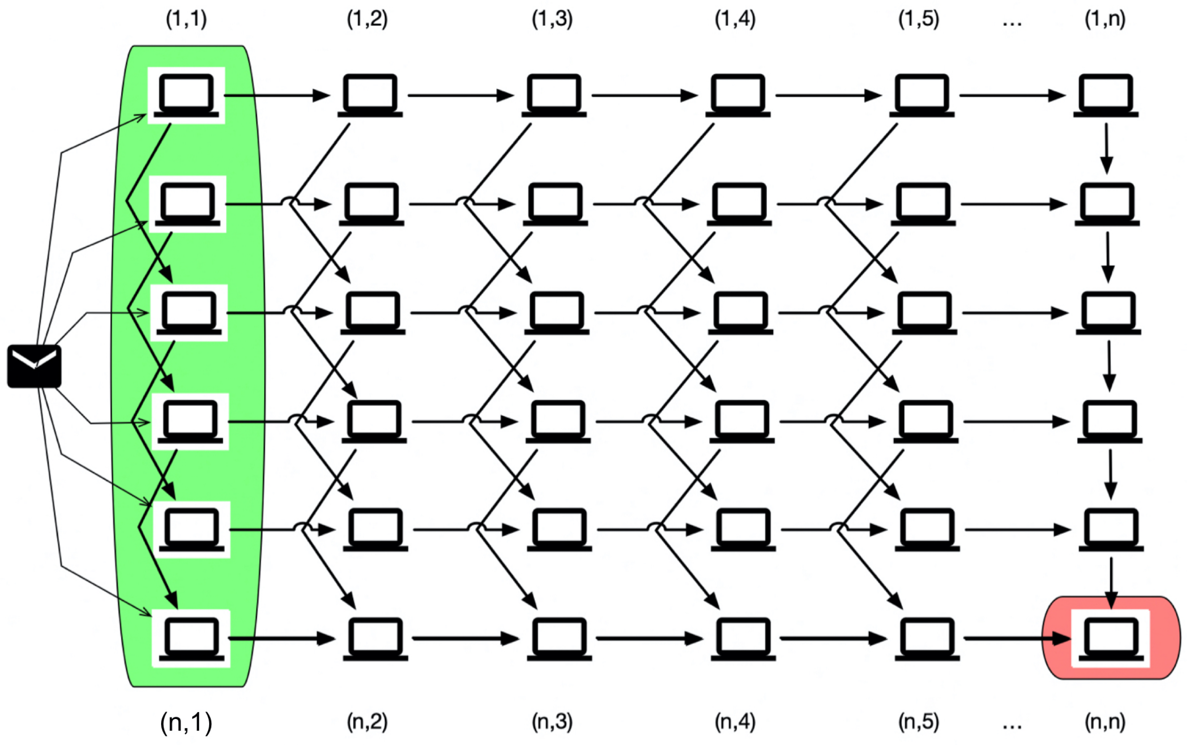
Question 5

15 points

A computer network is configured as shown in the figure below, where each computer is indexed as (i, j) for $i = 1 \ldots n$ and $j = 1 \ldots n$. The input message can be sent to one of the computers in position $(j, 1)$. From the first column, the message can be sent to another computer in the system, as long as it follows directly along one of the paths shown in the figure. The message must arrive safely at the computer with index (n, n). However, not all routes are safe. Each computer in the network has a certain chance of a security breach. If the message is passed along a route that contains a computer with a security leak, the message is *not* successfully transmitted. The chance of a breach at computer (i, j) is contained in the variable $B(i, j)$. The goal is to find the route with the *highest* chance of a successful transmission (in other words, the smallest chance of a breach).

The input to this problem is the array $B[1\ldots n, 1\ldots n]$: where $B[i,j]$ is the probability of a breach at computer (i,j). Your job is to design a dynamic programming solution to this problem. Your solution must output the maximum chance of a successful transmission (or 0 if not such route is possible). Your solution must use the following DP table: $T[1\ldots n, 1\ldots n]$ where $T[i,j]$ represents the highest chance that the message leaves one of the input computers and reaches computer (i,j). You must justify the runtime of $O(n^2)$.

Notes on probability : a probability is a value from 0 *to* 1*. If the chance of a breach at computer* A *is* x*, then the chance of a successful transmission is* $1-x$*. If the chance of success at computer* A *is* x *and the chance of a success at computer* B *is* y*, then the chance of a success at* ***both*** *computers is* $x \cdot y$*.*


(1,1)
(1,2)
(1,3)
(1,4)
(1,5)
...
(1,n)
(n,1)
(n,2)
(n,3)
(n,4)
(n,5)
...
(n,n)