Due 28th July 2022 at 4pm Sydney time

In this assignment we apply dynamic programming. There are *four problems* each worth 20 marks, for a total of 80 marks.

Your solutions must be typed, machine readable PDF files. *All submissions will be checked for plagiarism!*

For each question requiring you to design an algorithm, you *must* justify the correctness of your algorithm. If a time bound is specified in the question, you also *must* argue that your algorithm meets this time bound.

To describe a dynamic programming algorithm, you must include:

- the subproblem specification,
- the recurrence,
- any base cases,
- how the overall answer is calculated, including (if necessary) the order in which the subproblems are solved and
- the time complexity analysis.

The recurrence, base cases and final answer calculation must each be accompanied with (often brief) worded reasoning to justify the correctness of the algorithm.

Partial credit will be awarded for progress towards a solution.

The clarifications may be updated further in the Assignment 4 FAQ thread on the Ed forum: https://edstem.org/au/courses/8646/discussion/937894.

General clarifications: Nil.

Question 1

You are playing a video game, where you control a character in a grid with m rows and n columns. The character starts at the square in the top left corner $(1, 1)$, and must walk to the square in the bottom right corner (m, n). The character can only move one square at a time *downwards or rightwards*. Every square (i, j), other than the starting square and the ending square, contains a known number of coins $a_{i,j}$.

1.1 **[10 marks]** Design an algorithm which runs in $O(mn)$ time and determines the maximum number of coins that your character can accumulate by walking from $(1, 1)$ to (m, n) using a combination of downwards and rightwards moves.

You can arrive at a square from above or from the left. Which is better?

- **Do the starting and ending squares contain any coins?**

  No.

- **What is the output format?**

  Only the maximum number of coins, *not* the optimal path.

1.2 **[10 marks]** After playing this game many times, you have broken the controller, and you can no longer control your character. They now walk randomly as follows:

- if there is only one possible square to move to, they move to it;
- otherwise, they move right with probability p and down with probability $1 - p$.

Note that this guarantees that the character arrives at (m, n).

Design an algorithm which runs in $O(mn)$ time and determines the *expected* number of coins that your character will accumulate by walking from $(1, 1)$ to (m, n) according to the random process above.

Recall that for a discrete random variable X which attains values $x_1, \ldots, x_n$ with probabilities $p_1, \ldots, p_n$, the *expected value* of X is defined as

$$\mathbb{E}(X) = \sum_{i=1}^{n} p_i x_i.$$

As in 1.1, you can arrive at a square from above or from the left.

Recall the *linearity of expectation*: for random variables X and Y and constants a and b, we have $\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y)$.

- **What is expected value? What does the given formula mean?**

  Expected value is a calculation of the weighted average over all possible outcomes. The x_i in the formula are outcomes (number of coins accumulated on a path), and the p_i are the probabilities of those outcomes (probability of following that path). Note that you do *not* necessarily have to use this formula directly in your algorithm.

Question 2

You are managing a garage with two mechanics, named Alice and Bob. Each mechanic can serve at most one customer at a time.

There will be n customers who come in during the day. The ith customer wants to be served for the entire period of time starting at time s_i and ending at time e_i. You may assume that the customers are indexed by their order of arrival, i.e. $s_1 < s_2 < \ldots < s_n$.

For each customer i, the business will:

- make a_i dollars if customer i is served by Alice;
- make b_i dollars if customer i is served by Bob;
- lose c_i dollars if customer i is not served.

Your task is to maximise the net earnings of the garage, which is calculated as the total amount made minus the total amount lost.

2.1 **[8 marks]** Consider the following greedy algorithm.

Process each customer i in order of arrival as follows.

- If both Alice and Bob are available at time s_i:
  - if $a_i \geq b_i$, assign customer i to Alice;
  - otherwise, assign the customer to Bob.
- If only one mechanic is available at time s_i, assign customer i to that mechanic.
- If neither mechanic is available at time s_i, do not serve customer i.

Design an instance of the problem which is not correctly solved by this algorithm. You must:

- specify a number of customers n,
- for each customer, provide values for s_i, e_i, a_i, b_i and c_i,
- apply the greedy algorithm to this instance and calculate the net earnings achieved, and
- show that a higher net earnings figure can be achieved.

> - **Can a job be paused partway?**
>
>   No.
>
> - **Can a customer wait, in order to start later than their scheduled start time?**
>
>   No.

2.2 **[12 marks]** Design an algorithm which runs in $O(n^3)$ time and determines the maximum net earnings of the garage.

> - **What is the target time complexity?**
>
>   We've increased it from $O(n^2)$ to $O(n^3)$. There is a (very difficult) quadratic time solution, but cubic time solutions will now be eligible for full marks.
>
> - **Is the information about the customers known from the start?**
>
>   Yes. You know all s_i, e_i, a_i, b_i, c_i before any customers arrive.

Question 3

You are given a simple directed weighted graph with n vertices and m edges. The edge weights *may* be negative, but there are no cycles whose sum of edge weights is negative.

3.1 [**10 marks**] An edge e is said to be *useful* if there is some pair of vertices u and v such that e belongs to **at least one** shortest path from u to v.

Design an algorithm which runs in $O(n^3)$ and determines the set of useful edges.

> First find a sufficient condition for an edge to *not* be useful, and then prove that it is also a necessary condition.

3.2 [**10 marks**] An edge is said to be *very useful* if there is some pair of vertices u and v such that e belongs to **every** shortest path from u to v.

Design an algorithm which runs in $O(n^3)$ and determines the set of very useful edges.

> - **Does the input specify particular vertices u and v?**
>
>   No.

Question 4

There are $2n$ players who have signed up to a chess tournament. For all $1 \leq i \leq 2n$, the ith player has a known skill level of s_i, which is a non-negative integer. Let $S = \sum_{i=1}^{2n} s_i$, the total skill level of all players.

In the tournament, there will be n matches. Each match is between two players, and each player will play in exactly one match. The *imbalance* of a match is the absolute difference between the skill levels of the two players. That is, if a match is played between the ith player and the jth player, its imbalance is $|s_i - s_j|$. The *total imbalance* of the tournament is the sum of imbalances of each match.

The organisers have provided you with a value m which they consider to be the ideal total imbalance of the tournament.

Design an algorithm which runs in $O(n^2 S)$ time and determines whether or not it is possible to arrange the matches in order to achieve a total imbalance of m, assuming:

4.1 [**4 marks**] all s_i are either 0 or 1;

4.2 [**16 marks**] the s_i are distinct non-negative integers.

> From each match, consider the player with higher skill level only. What should the total of their skill levels be?

> - **What is the output format?**
>
>   Only YES or NO, *not* the full list of matches.