CO3002/7002 Assignment 2

Released Mar 9, 2022 Deadline Mar 30, 2022 5:00 pm

This assignment consists of two parts. The first part is to be completed individually. The second part can be completed in groups of size up to three. Further instructions about group work will be provided separately.

You should prepare your answers electronically, in a Word document or pdf file. Where mathematical formulas, diagrams etc are needed you can use handwriting and insert them into the document as pictures, but any substantial amount of text should be typeset and not handwritten. You should have the individual and group parts in different files; there are separate submission points for the two parts on Blackboard.

In line with university policy, marking will be done anonymously. Please do not include your name or other personally identifiable information in your submission.

Part A (Individual, 30 marks)

In each of the following, a greedy algorithm is proposed for the given problem. Give a simple counterexample for each of them to show that the proposed greedy algorithm does not always return the optimal solution. You should (1) give an example input, (2) state the solution returned by the greedy algorithm on that input, and (3) state the optimal solution (or indeed any better solution than the greedy one) for that input.

(a) **Input**: An undirected graph $G(V, E)$.

Problem: Find a minimum spanning tree T of G.

Algorithm:

Choose an arbitrary vertex s in G
Initialise $S := \{s\}$, $T := \{\}$ (empty set), and $u := s$
while $S \neq V$ **do**
  Find the minimum-weight edge e among all edges (u, v) where v is not in S
  Add v to S, add e to T, and set $u := v$
end while

(Note: this is *not* Prim's algorithm.) [10 marks]

(b) **Input**: A set A of animals, and a set of pairs (a_i, a_j) indicating animal a_i cannot be put together with animal a_j (because for example a_i attacks or eats a_j).

Problem: Choose a subset A' of animals from A so that the animals in A' can be safely put together, and that the number of animals in A' is as large as possible.

Algorithm: Initialise A' to be empty. For each animal in A, count the number of other animals it can be safely put together. In decreasing order of this count, consider each animal in turn and add it to A' if it can be put safely together with all other animals already in A'. [10 marks]

(c) **Input**: A set of n tasks, each with a required "amount" of "work"; a set of m workers all of whom can work on any of the tasks.

Problem: Assign the tasks to the workers so that the work is "distributed as evenly as possible among the workers", or more precisely, the amount of work assigned to the busiest worker (the one with the largest amount of work) is as small as possible.

Algorithm: Sort the tasks in decreasing order of amount of work. For each task in this sorted order, assign it to the worker who currently (i.e., based on the assignment made so far) has the least amount of work assigned. [10 marks]

Part B (Group, 70 marks)

Your country is being invaded. As part of the invasion, a long line of enemy armoured vehicles are moving along a road towards you. You can launch some drone attacks to damage or destroy some of those vehicles. Unfortunately you only have one drone, so you have to plan your attacks carefully to maximise the damage you cause to the enemy.

The line of n vehicles is moving at a constant speed past the point where you are stationed (and where you launch your attack). For simplicity, we assume they move one unit of distance per unit of time, and that the vehicles are spaced one unit of distance apart from each other. Each enemy vehicle has a certain "value" (how much it is worth to destroy it). So, we can assume there is a sequence of vehicles $x_1, x_2, \ldots, x_n$ with values $v_1, v_2, \ldots, v_n$. At time $i = 1, 2, \ldots$, vehicle x_i is directly under your drone and is the only one you can attack. Based on satellite imaging, you have all the information about these vehicles in advance.

Your drone can only perform one attack at a time. After it fired, it needs to return to base, refill its ammunition ("ammo" for short), and go up in the air again. The more ammo it carries, the more powerful the strike is, but it also takes more time to be loaded up. To be precise, it needs one unit of time for returning to base, w units of time to refill it with w units of ammo, and another unit of time to go up in air. Thus in total it needs $w+2$ units of time between two strikes, if it is to be loaded with w units of ammo. You can assume you have unlimited ammo, but the drone can carry at most W units of ammo each time. Assume w and W are all integers.

A strike at an enemy vehicle may not completely destroy it; we assume that with w units of ammo, one strike causes w^2 units of damage to the vehicle (i.e. reduces its value by w^2), and if the value goes down to 0 it is completely destroyed. Note that each strike uses up all the ammo the drone carries, and you only get at most the value of that vehicle. For example if the drone carries 5 units of ammo and fires it at an vehicle with value only 7, you only caused 7 units of damage (reduced its value from 7 to 0), not 25.

You can have the drone loaded and up in the air already before the enemy arrives (so the first strike does not require loading time and it can carry maximum ammo). Your task is to determine the timing of the strike(s) and how much ammo to carry in each strike, so as to maximise the total damage to the vehicles.

The following is an example. Suppose $n = 13$, $v_1..v_{13} = [2, 3, 0, 1, 4, 4, 1, 0, 4, 0, 1, 6, 4]$, $W = 3$. Suppose the drone initially carries 3 units of ammo and first strikes at time $t = 2$, causing $\min(v_2, 3^2) = 3$ units of damage to x_2. Then it returns to base and fills up 2 units of ammo, so it is only ready at time $t = 6$. Then it strikes at $t = 6$, causing $\min(v_6, 2^2) = 4$ units of damage to x_6. Then it returns to base and refill with 3 units of ammo, so it will only be ready to fire again at time $t = 11$, and it waits until $t = 12$ to fire and causes $\min(6, 3^2) = 6$ units of damage to x_{12}. In total the damage done is $3 + 4 + 6 = 13$ units. Alternatively, if the drone strikes at times $t = 1, 5, 9, 13$, with 2 units of ammo each time, it causes a damage of $2 + 4 + 4 + 4 = 14$ (why?) There are many other possible strategies. Your task is to find the

best one.

(a) Design an efficient algorithm for this problem. You should: (the following is described in terms of the design of a dynamic programming algorithm, since you almost certainly should use it)

- In plain English or using some mathematical notation, give a recursive formulation of the problem.
- Give the pseudocode of the algorithm that is based on this formulation.
- Analyse the time and space complexity of your algorithm.
- Illustrate how your algorithm works when given the 13-vehicle example above as input. You only need to show the contents of the completed dynamic programming tables.

[50 marks]

Hint: One possible approach is to define $V(i)$ to be the optimal total damage where the input consists of only the vehicles $x_1, \ldots, x_i$. At time i, either the drone doesn't strike, or it does with w units of ammo on board for some w. In the latter case, the previous strike (if there is one) can only happen at time $i - w - 2$ or earlier, and we have to find the optimal solution of the problem where x_{i-w-2} is the last vehicle. Consider all these possible scenarios and use recursion for the subproblems. Note that if $i - w - 2 \leq 0$ then there cannot be a previous strike (i.e. this is the first strike) so it can carry maximum ammo with no loading time; you need to handle this correctly.

There are other correct approaches; the above is just one possible formulation.

(b) Unfortunately, as it turns out, you don't have unlimited ammo; you only have M units of ammo, so all the strikes together can use at most M units of ammo. Again assume M is an integer.

Design an efficient algorithm for the same problem in this limited-ammo case. You only need to give a recursive formulation and/or pseudocode and explain the time complexity.

Hint: one possible approach is to define $V(i, k)$ to be the optimal total damage where the input consists of only the vehicles $x_1, \ldots, x_i$ and with k units of ammo available. Consider all possibilities on what happens at time i as in (a).)

[20 marks]

Marking criteria

Marks are given roughly according to these tables:

Part A:

0	Not giving any counterexample at all. E.g. giving some algorithm, pseudocode, actual program code, copying pages and pages of explanation of the given problem (or even some unrelated problem) from some websites, textbooks or "research papers". (Yes, some of you did that.)
2-3	Wrong understanding of the problem, and wrong counterexample for it. But at least it is some kind of an example (some input instance and some alleged greedy/optimal solution).
5	A correct counterexample for a misunderstood version of the problem; or correct understanding of the problem but two of input/greedy solution/optimal solution wrong.
7-8	Correct understanding of the problem but one of input/greedy solution/optimal solution wrong.
10	Correct counterexample (input) and correct explanation (greedy solution and optimal solution).

Part B: (The mark you get within each range depends on how many of the required elements (pseudocode, analysis, example execution) is present/correct.)

(a)	(b)	
0	0	Completely irrelevant answers; copying from some textbook general explanations about dynamic programming; copying from some websites / textbooks / "research papers" the solution of some other problem. (Again, some of you actually did that.)
≥ 10	≥ 4	Any genuine attempt at the question, no matter how wrong it is.
10-20	4-8	Not any ideas towards a correct algorithm (e.g. gave some greedy-type algorithms); or gave an inefficient, brute-force type algorithm.
15-25	6-10	There are some ideas towards a recursive formulation, but not a DP algorithm (did not use dynamic programming tables).
20-40	8-16	Some correct ideas DP-based algorithm but with major problems in formulation/pseudocode/analysis/example execution (where required).
30-45	12-20	Broadly correct ideas of a DP-based algorithm, but with some minor problems in formulation/pseudocode/analysis/example execution (where required).
45-50		Everything is mostly correct, maybe with some small mistakes. Traceback is included.