CO3002/7002 Assignment 2

Released Mar 9, 2021 Deadline Mar 29, 2021 5:00 pm

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

Submit your work on Blackboard, as a pdf file. It can either be produced electronically (e.g. MS Word) then converted to pdf, or handwritten then scanned/photographed.

Question 1 (30 marks)

This question is to be attempted individually.

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 give an example input, state the solution returned by the greedy algorithm on that input, and another solution that is better than the greedy solution for that input.

(a) **Input**: An undirected *complete* graph G (i.e. there is an edge (with weight) between any two vertices in the graph); a starting vertex s and a destination vertex t in G.

Problem: Find a path that begins at s, ends at t, and visits every other vertex in G exactly once, and such that the total weight of this path is as small as possible.

Algorithm: Let u be the current vertex (which initially is s). Find the minimum-weight edge among all edges (u, v) where v is neither t nor any vertex already visited. Add this edge to the path, and update the current vertex to v. Repeat this until t is the only vertex not yet visited. At this point add the edge from the current vertex to t as the final edge of the path. [10 marks]

(b) **Input**: A set Y of youtube channels, a set P of people and a and a set of pairs (y_i, p_j) indicating person p_j subscribed channel y_i.

Problem: Choose a subset Y' of youtube channels from Y so that everyone in P subscribes to at least one channel in Y', and that the number of channels in Y' is as small as possible.

Algorithm: Repeatedly add to Y' the channel that would give the largest number of 'new' subscribers (i.e. those who did not subscribe to any channel already in Y'), until everyone in P has subscribed something in Y'. [10 marks]

(c) **Input**: A set of objects, each with a weight between 0 and 1.

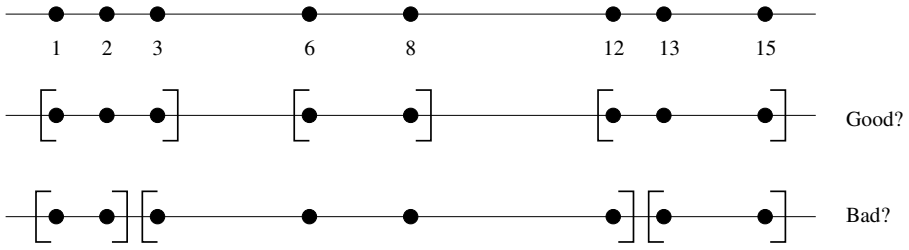
Problem: Put all objects into a number of containers, where each container can hold objects of total weight at most 1, and such that the number of containers used is minimised.

Algorithm: Sort all objects in decreasing order of weights. For each object in this sorted order, among all existing containers that would fit this object, put it into the one with the largest total weight of objects already in there. If no existing container can fit this object in, put it into a new container. [10 marks]

Question 2 (70 marks)

This question can be attempted in groups.

A fundamental problem in machine learning and data analytics is to partition or classify objects in a 'natural' way based on their 'distances' to each other. In this question, you are given an integer N, and n points $x_1, x_2, \ldots, x_n$ in a horizontal line. We want to partition the n points into N groups so that intuitively within each group the points are 'close' together. The following is an example of what may be a good way of partitioning the following $n = 8$ points into $N = 3$ groups, and what may be a bad way:



More precisely, each point x_i is a number on the x-axis. To measure how 'good' a partitioning is, we define the error of a group of points $x_p, x_{p+1}, \ldots, x_q$ to be

$$(x_p - \mu)^2 + (x_{p+1} - \mu)^2 + \cdots + (x_q - \mu)^2$$

where $\mu = (x_p + x_{p+1} + \cdots + x_q)/(q - p + 1)$ is the average of the points. Our objective is to find the optimal way of partitioning the points into N groups so that the sum of errors of all groups is minimised.

For example, the 'bad' way of partitioning for the above example gives an error of:
Group 1: average = (1+2)/2 = 1.5, error = $(1 - 1.5)^2 + (2 - 1.5)^2 = 0.5$.
Group 2: average = (3+6+8+12)/4 = 7.25, error = $(3 - 7.25)^2 + (6 - 7.25)^2 + (8 - 7.25)^2 + (12 - 7.25)^2 = 42.75$.
Group 3: average = (13+15)/2 = 14, error = $(13 - 14)^2 + (15 - 14)^2 = 2$.
Total error = 0.5 + 42.75 + 2 = 45.25.

We divide the points in such a way that only consecutive points (along the line) may fall into the same group.

(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. (Hint: one possible approach is to define $F(i, k)$ to be the error of the optimal partition of the first i points $x_1, \ldots, x_i$ into k groups, where $i \leq n$ and $k \leq N$. The value of $F(i, k)$ can be found recursively: consider all possible locations where the last (k-th) group can begin.)
- 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 8-point example above as input. You only need to show the contents of the completed dynamic programming tables.

[55 marks]

(b) The above assumes the number of groups N is known in advance. But often it is useful to let the algorithm work out what is the 'natural' number of groups. The above objective (minimising the total error of groups) cannot be used directly in this case, since the 'optimal' solution would always be to divide the n points into n groups with each point in its own group, and the error of each group (and the total error) is 0. To counter this, we introduce a 'penalty' P for each additional group used, and the objective becomes finding a partition that minimise the following 'cost':

$$E_1 + E_2 + \cdots + E_N + N \cdot P$$

where E_i is the error of group i, N is *not* a given parameter (the algorithm has to find out what it is) and P is a user-supplied penalty value.

Design an efficient algorithm for this problem. It is sufficient to give the pseudocode of the algorithm; or give a mathematical recursive formulation and a clear indication of how it could be turned into an efficient algorithm. Your algorithm only need to find out the optimal cost; no traceback is required.

(Hint: one recursive formulation is to consider the minimum cost of partitioning the first i points, then consider where the last group can begin.) [15 marks]

Marking criteria

Question 1 will be marked based on correctness of counterexample given. Correct counterexample but with missing/incorrect explanations may cost you up to half of the marks. Partial marks may be given to "almost correct" counterexamples.

Question 2(a) will be marked roughly according to this table:

0-15	Not any ideas towards a correct algorithm (e.g. gave some greedy-type heuristics that do not always give the correct answer).
15-25	There are some ideas towards a correct algorithm but the algorithm would have been inefficient (e.g. have a broadly correct recursive formulation but is exponential time due to not using dynamic programming tables).
25-35	Some ideas of a DP-based algorithm but with many problems, or is a formulation that gives a poor (although polynomial) time complexity. Analysis/example execution missing or incorrect.
35-45	Broadly correct ideas of a DP-based algorithm, but with some errors in pseudocode. Analysis/example execution has some mistakes.
45-55	Everything is mostly correct, maybe with some small mistakes. Traceback is included.
55-60 (bonus)	As above, and algorithm/analysis includes how the time complexity can be further optimised (compared with a standard application of DP).

Question 2(b) will be marked roughly according to this table:

0-5	Not any ideas towards a correct algorithm (e.g. gave some greedy-type heuristics that do not give the correct answer).
5-10	Some ideas of a recursive formulation, but with major errors or was not turned into DP.
10-15	Mostly correct ideas of a DP-based algorithm, maybe with some minor errors.