Due: on August 12, before 2:00 PM.

Part 1

You have a big pizza with n slices. Unfortunately, there's so much cheese on it that whenever you remove a piece from the pizza, some of the toppings fall off. Before you start eating you have analysed each slice i and determined that if you take the slice when there are k slices remaining (including that slice), then you will loose $a_i + b_i \times k$ grams of topping. You would like to eat the pizza while loosing a little topping as possible. At any point, you may only eat a piece with an empty space next to it (except for the first piece, for which you may eat any one).

The objective of this part is to obtain a dynamic programming algorithm to solve the following task (T): determine the minimal amount of topping lost as you eat the entire pizza.

Question 1.1

Define the subproblems used in a dynamic programming approach to solve task T.

Question 1.2

Express the recurrence relationship between subproblems and give the solution to the base case(s).

Question 1.3

Give the pseudo-code of an algorithm to solve task T.

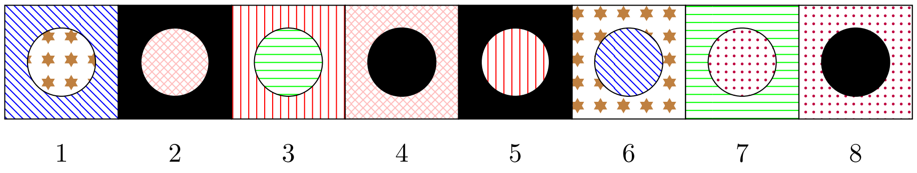
Question 1.4

What is the worst-case complexity of your algorithm?

Part 2

You have a row of n boxes. In each one sits a marble. Some of the marbles are coloured. For each coloured marble, there is a box of the same colour. All other marbles and boxes are black. Marbles are not necessarily sitting in a box of their own colour. All coloured marbles have a different color from one another.

The *cost* of a configuration is defined as the total sum of the distances between each coloured marble and its corresponding box. For example, consider the following configuration with $n = 8$ boxes and marbles.



Question 2.1

What is the cost of the configuration displayed above?

Question 2.2

You wish to swap some marbles such that the cost is minimised. However, the following rules apply:

- Each swap involves one coloured marble and one a black marble.
- If two marbles are swapped, then all marbles sitting between them must remain in their original location.
- No marble may be swapped twice.

In the configuration above, what set of legal swaps leads to a configuration with the smallest cost? Answer using the following format: e.g., $\{(4, 5), (6, 8)\}$.

Question 2.3

We would like to develop an $O(n^2)$ Dynamic Programming algorithm to determine the minimal possible cost reachable from an input starting configuration. Subquadratic algorithms (faster than $O(n^2)$) may exist but are not required.

Define the subproblems.

Question 2.4

Give the recurrence equation(s) linking the subproblems and give the solution to the base case(s).

Question 2.5

Prove that your approach leads to an algorithm with $O(n^2)$ complexity.

Part 3

You are given a set of points in the plane. Some of these points have line segments between them. These line segments do not overlap, except at endpoints. These line segments form a set of (potentially irregular) polygons. Polygons are not self-intersecting, and do not overlap, although they may share edges and/or vertices. You wish to colour as many of the polygons with one of two colours (say blue and red) such that no two polygons that share an edge have the same colour. Polygons may remain uncoloured. Two uncoloured polygons may share an edge.

We would like to solve this problem using integer linear programming (ILP).

Question 3.1

The first step in the modelisation of this problem is to represent it as a problem on a graph (V, E). What would each vertex $v \in V$ represent? What would each edge $e \in E$ represent?

Question 3.2

List the variables for the ILP problem.

Question 3.3

List the constraints.

Question 3.4

State the objective function.

Part 4

You are stacking n bags onto a plane. Bags must be stacked in columns, up to h high. The plane's cargo hold is p columns wide. The amount of force that bag i placed in pile $q \in [1, p]$ applies to the plane is given by f_{iq}. You wish for the sum of these forces from all bags to be as small as possible.

We aim at modeling this problem as a mathematical optimization problem.

Question 4.1

Is it better to model this problem using Linear Programming or Integer Linear Programming? Why? (One or two sentences in English should be sufficient to explain your reasoning.)

Question 4.2

List the variables for the problem.

Question 4.3

List the constraints.

Question 4.4

State the objective function.