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 lose $a_i + b_i \times k$ grams of topping. You would like to eat the pizza while losing 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.

Solution

$P(l, r)$ is the problems of determining the minimal amount of topping lost if only slices from l (included) to r (not included) are left. When $l < r$, the slices left are labelled $l \dots r-1$. When $r < l$, the slices left are labelled $l \dots n-1$ and $0 \dots r-1$. When $l = r$ there are no slices left.

Question 1.2

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

Solution

Base cases: For any $0 \le l < n$, we have $P(l, l) = 0$.

For any $l \ne r$, there are $k = r - l \mod n$ slices left and we have $P(l, r) = \min\big(a_l + b_l k + P(l+1, r), a_r + b_r k + P(l, r-1)\big)$.

In the recurrence equation above, we consider the arguments of P to be elements of $\mathbb{Z}_n$, that is we take the arguments modulo n. For instance $P(n+3, -6) = P(3, n-6)$.

Question 1.3

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

Solution

We assume that the arrays A and B are of length n, contain the parameters a_i and b_i and we assume for $0 \leq i, k < n$ that $0 \leq a_i + b_i \times k$.

```
savetoppings($A$, $B$, $n$, $M$, $l$, $r$)
    **if** $l = r$ **then** **return** $0$
    **if** $M[l, r] < 0$ **then**
        $k \leftarrow r - l \mod n$
        $c \leftarrow A[l] + B[l] \times k +$ savetoppings($A$, $B$, $n$, $M$, $l$+1, $r$)
        $d \leftarrow A[r] + B[r] \times k +$ savetoppings($A$, $B$, $n$, $M$, $l$, $r$-1)
        $M[l, r] \leftarrow \min(c, d)$
    **return** $M[l, r]$

starteating($A$, $B$, $n$)
    $M \leftarrow$ 2D array of size $n \times n$ initialized with $-1$
    $t \leftarrow A[n-1] + B[n-1] \times n +$ savetoppings($A$, $B$, $n$, $M$, 0, $n$-1)
    **for** $i = 0$ **to** $n - 2$ **do**
        $t' \leftarrow A[i] + B[i] \times n +$ savetoppings($A$, $B$, $n$, $M$, $i$+1, $i$)
        $t \leftarrow \min(t, t')$
        **return** $0$
    **return** $M[l, r]$
```

Question 1.4

What is the worst-case complexity of your algorithm?

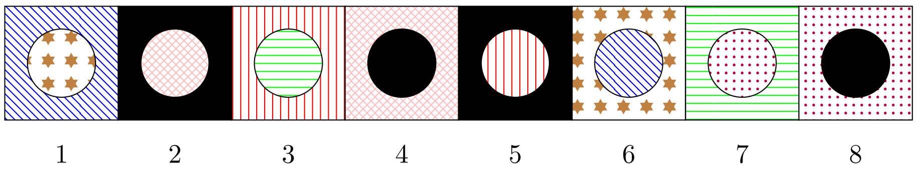
Solution

$O(n^2)$

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?

$5 + 2 + 4 + 2 + 5 + 1$

Solution

19

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., $\{(4, 5), (6, 8)\}$.

The swaps that reduce the cost are the following.

- $(1, 4)$ reduces the cost by 3.
- $(2, 4)$ reduces the cost by 2.
- $(3, 4)$ reduces the cost by 1.
- $(5, 4)$ reduces the cost by 1.
- $(6, 4)$ reduces the cost by 2.
- $(1, 8)$ reduces the cost by 3.
- $(3, 8)$ reduces the cost by 3.

- $(7, 8)$ reduces the cost by 1.

However, the swaps $(1, 8)$ and $(3, 8)$ are incompatible with moving the black marble in 4, so it is better to do $\{(1, 4), (7, 8)\}$.

Solution

$\{(1, 4), (7, 8)\}$

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.

Solution

For $1 \leq i \leq n$, let $P(i)$ be the problem of determining the minimal possible cost reachable from the starting configuration without moving any marble located in $j > i$.

Question 2.4

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

Let b_i denote the position of marble i's corresponding box.

To evaluate $P(i)$ when i is black, let us consider the minimal cost reachable by swapping i with a coloured marble $1 \leq j < i$. After that swap, no marble to the right of j inclusive can be moved, so we would obtain a minimal reachable cost of $P(j-1) + |b_j - j| - |b_j - i|$. Indeed, in the case of swapping marble i with marble j, we have to subtract the coloured marble's original distance from its corresponding box, and then add its final distance from its box.

On the other hand, when i is coloured, let us consider the minimal cost reachable by swapping i with a black marble $1 \leq j < i$. After that swap, no marble to the right of j inclusive can be moved, so we would obtain a minimal reachable cost of $P(j-1) + |b_i - j| - |b_i - i|$.

There is also the possibility of not moving marble i, in which case the minimal reachable cost would be equal to $P(i-1)$.

Solution

$P(0)$ and $P(1)$ are the base cases and their solution is the cost of the starting configuration.

Let b_i denote the position of marble i's corresponding box.
For $1 < i \leq n$, if i is black then

$$P(i) = \min\left(P(i-1), \min_{1 \leq j < i}\left(P(j-1) + |b_j - j| - |b_j - i|\right)\right)$$

if i is coloured then

$$P(i) = \min\left(P(i-1), \min_{1 \leq j < i}\left(P(j-1) + |b_i - j| - |b_i - i|\right)\right)$$

Question 2.5

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

Solution

Each inner minimization can be done in $O(n)$. There are $O(n)$ values $P(i)$ to compute, so the overall complexity is $O(n \times n) = O(n^2)$.

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?

Solution

Each vertex $v \in V$ represents a single polygon. E is the set of pairs (v_1, v_2) such that polygons v_1 and v_2 share a border.

Question 3.2

List the variables for the ILP problem.

Solution

b_v for $v \in V$, where $b_v = 1$ indicates that polygon v is colored in blue
r_v for $v \in V$, where $r_v = 1$ indicates that polygon v is colored in red

These are integer variables.

Question 3.3

List the constraints.

Solution

$$\begin{aligned}
b_{v_1} + b_{v_2} &\le 1 \text{ for } (v_1, v_2) \in E, \text{ indicating that no neighbours can be both colored blue} \\
r_{v_1} + r_{v_2} &\le 1 \text{ for } (v_1, v_2) \in E, \text{ indicating that no neighbours can be both colored red} \\
b_v + r_v &\le 1 \text{ for } v \in V, \text{ indicating that no polygon can have both colors} \\
&\left.\begin{aligned} 0 &\le b_v \\ 0 &\le r_v \\ b_v, r_v &\in \mathbb{Z} \end{aligned}\right\} \text{ for } v \in E, \text{ indicating that the variables } b_v \text{ and } r_v \text{ are } \{0,1\} \text{ variables}
\end{aligned}$$

Question 3.4

State the objective function.

Solution

$$\max \sum_{v \in V} (b_v + r_v)$$

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.)

Solution

This problem is better model as ILP because each bag is to be entirely stacked in a single column. LP would allow bags to be half-stacked in one column and half-stacked in another column.

Question 4.2

List the variables for the problem.

Solution

$$x_{ij} \text{ for } 1 \leq i \leq n \text{ and } 1 \leq j \leq p\text{: whether or not to place bag } i \text{ in column } j$$

The x_{ij} are integer variables.

Question 4.3

List the constraints.

Solution

$$\sum_{j=1}^{p} x_{ij} = 1 \text{ for } 1 \leq i \leq n\text{: each bag is placed in a single column}$$

$$\sum_{i=1}^{n} x_{ij} \leq h \text{ for } 1 \leq j \leq p\text{: each column contains no more than } h \text{ bags}$$

$$\left.\begin{array}{l} 0 \leq x_{ij} \leq 1 \\ x_{ij} \in \mathbb{Z} \end{array}\right\} \text{ for } 1 \leq i \leq n \text{ and } 1 \leq j \leq p\text{, indicating that the variables } x_{ij} \text{ are } \{0,1\} \text{ variables}$$

Question 4.4

State the objective function.

Solution

$$\min \sum_{i=1}^{n} \sum_{j=1}^{p} x_{ij} f_{ij}$$