Competitive Programming and Mathematics Society

COMP4128 Revision Workshop

25T2

CPMSoc Programming

Table of contents

1 Ticket Hoarding

2 Lights

3 Bars

4 Routing

5 Game of Tiles

Ticket Hoarding

There are N ($1 \leq N \leq 3 \cdot 10^5$) days on which tickets are sold. The price per ticket on day i is a_i ($1 \leq a_i \leq 10^9$). Also:

- At most M ($1 \leq M \leq 10^9$) tickets can be bought on a single day.
- If x tickets are bought on day i, all subsequent days will have ticket prices per ticket increased by x.

Find minimum cost to buy K ($1 \leq K \leq \min(NM, 10^9)$) tickets.

Example

$N = 4$, $M = 2$, $K = 3$, $a = [8, 6, 4, 2]$

The best way to buy the ticket is as follows:

- Buy 0 tickets on the first day. The prices per ticket for the remaining days are $[6, 4, 2]$.
- Buy 0 tickets on the second day. The prices per ticket for the remaining days are $[4, 2]$.
- Buy 1 ticket on the third day with cost 4. The price per ticket for the last day is $[3]$.
- Buy 2 tickets on the last day with cost 6.

Cost is $4 + 6 = 10$.

Subtasks

Subtask 1

Purchasing tickets *do not* affect the prices on subsequent days.

Subtask 2

Purchasing tickets *do* affect the prices of subsequent days, but $a_i = 1$ for all i.

Subtask 3

No additional constraints.

Subtasks

Subtask 1

Purchasing tickets *do not* affect the prices on subsequent days.

Subtasks

Subtask 1

Purchasing tickets *do not* affect the prices on subsequent days.

Greedily purchase tickets from days with cheaper cost!

Subtasks

Subtask 2

Purchasing tickets *do* affect the prices of subsequent days, but $a_i = 1$ for all i.

Subtasks

Subtask 2

Purchasing tickets *do* affect the prices of subsequent days, but $a_i = 1$ for all i.

The amount that price per ticket increases by is equal to the number of tickets bought before it.

Subtasks

Subtask 2

Purchasing tickets *do* affect the prices of subsequent days, but $a_i = 1$ for all i.

The amount that price per ticket increases by is equal to the number of tickets bought before it.

This is the number of pairs of tickets bought on different days.

Subtasks

Subtask 2

Purchasing tickets *do* affect the prices of subsequent days, but $a_i = 1$ for all i.

The amount that price per ticket increases by is equal to the number of tickets bought before it.

This is the number of pairs of tickets bought on different days.

To minimise this additional cost, purchase tickets from the least number of days possible!

Solution

Subtask 1 solution works!

Solution

Subtask 1 solution works!

Why?

Solution

Subtask 1 solution works!

Why?

Doing so in this way minimises the total cost of tickets excluding price increases. It also minimises the days on which the tickets are bought, which minimises cost due to price increases.

Complexity is $O(N \log N)$.

Lights

There are N ($1 \leq N \leq 1000$) light bulbs in a row and N switches. The i-th switch toggles the i-th light bulb. Initially, all lights are off.

In each move, Anubhav is allowed to select a set of these switches and toggle the lights associated with them in a *move*. He is not allowed to press the same set of switches in two different moves.

Anubhav is very picky – after exactly M ($1 \leq M \leq 1000$) moves, he wants the first V ($0 \leq V \leq N$) lights on and the rest off. How many ways can he win, modulo 10657201? Order that he chooses the sets doesn't matter.

Example

$N = 3$, $M = 3$, $V = 0$

There are 7 ways listed below.

$$\langle 0, 0, 1 \rangle, \langle 0, 1, 0 \rangle, \langle 0, 1, 1 \rangle$$
$$\langle 0, 0, 1 \rangle, \langle 1, 0, 0 \rangle, \langle 1, 0, 1 \rangle$$
$$\langle 0, 0, 1 \rangle, \langle 1, 1, 0 \rangle, \langle 1, 1, 1 \rangle$$
$$\langle 0, 1, 0 \rangle, \langle 1, 0, 0 \rangle, \langle 1, 1, 0 \rangle$$
$$\langle 0, 1, 0 \rangle, \langle 1, 0, 1 \rangle, \langle 1, 1, 1 \rangle$$
$$\langle 0, 1, 1 \rangle, \langle 1, 0, 0 \rangle, \langle 1, 1, 1 \rangle$$
$$\langle 0, 1, 1 \rangle, \langle 1, 0, 1 \rangle, \langle 1, 1, 0 \rangle$$

Solution

We call the resultant pattern V. Let $\oplus$ be the bitwise XOR operator. The problem turns into choosing M vectors $V_1, V_2, \ldots, V_M \in \mathbb{F}_2^N$ such that $\oplus_{i=1}^{M} V_i = V$.

Solution

We call the resultant pattern V. Let $\oplus$ be the bitwise XOR operator. The problem turns into choosing M vectors $V_1, V_2, \ldots, V_M \in \mathbb{F}_2^N$ such that $\oplus_{i=1}^{M} V_i = V$.

Consider DP?

Solution

We call the resultant pattern V. Let $\oplus$ be the bitwise XOR operator. The problem turns into choosing M vectors $V_1, V_2, \ldots, V_M \in \mathbb{F}_2^N$ such that $\oplus_{i=1}^{M} V_i = V$.

Consider DP?

Let f_x be the number of ways if order matters to choose x distinct vectors who's XOR gives V. Then $dp_x = \frac{1}{x!} f_x$ is the answer.

Solution

We call the resultant pattern V. Let $\oplus$ be the bitwise XOR operator. The problem turns into choosing M vectors $V_1, V_2, \ldots, V_M \in \mathbb{F}_2^N$ such that $\oplus_{i=1}^{M} V_i = V$.

Consider DP?

Let f_x be the number of ways if order matters to choose x distinct vectors who's XOR gives V. Then $dp_x = \frac{1}{x!} f_x$ is the answer.

Choose $x-1$ vectors, $V_x = V \oplus V_1 \oplus V_2 \oplus \cdots \oplus V_{x-1}$. There are $(x-1)!\binom{2^N}{x-1}$ ways to do this. What's the issue?

Solution

We call the resultant pattern V. Let $\oplus$ be the bitwise XOR operator. The problem turns into choosing M vectors $V_1, V_2, \ldots, V_M \in \mathbb{F}_2^N$ such that $\oplus_{i=1}^{M} V_i = V$.

Consider DP?

Let f_x be the number of ways if order matters to choose x distinct vectors who's XOR gives V. Then $dp_x = \frac{1}{x!} f_x$ is the answer.

Choose $x-1$ vectors, $V_x = V \oplus V_1 \oplus V_2 \oplus \cdots \oplus V_{x-1}$. There are $(x-1)!\binom{2^N}{x-1}$ ways to do this. What's the issue?

The issue is if V_x is the same as one of the chosen $x-1$. Suppose this is V_{x-1}. Then $V_x \oplus V_{x-1} = 0$, so $V_1 \oplus V_2 \oplus \cdots \oplus V_{x-2} = V$ – this is f_{x-2}.

Solution

We call the resultant pattern V. Let $\oplus$ be the bitwise XOR operator. The problem turns into choosing M vectors $V_1, V_2, \ldots, V_M \in \mathbb{F}_2^N$ such that $\oplus_{i=1}^{M} V_i = V$.

Consider DP?

Let f_x be the number of ways if order matters to choose x distinct vectors who's XOR gives V. Then $dp_x = \frac{1}{x!} f_x$ is the answer.

Choose $x-1$ vectors, $V_x = V \oplus V_1 \oplus V_2 \oplus \cdots \oplus V_{x-1}$. There are $(x-1)!\binom{2^N}{x-1}$ ways to do this. What's the issue?

The issue is if V_x is the same as one of the chosen $x-1$. Suppose this is V_{x-1}. Then $V_x \oplus V_{x-1} = 0$, so $V_1 \oplus V_2 \oplus \cdots \oplus V_{x-2} = V$ – this is f_{x-2}.So

$$f_x = (x-1)!\binom{2^N}{x-1} - (x-1)(2^N - x + 2) f_{x-2}.$$

Solution

Now, the answer is just $dp_M = \frac{1}{M!} f_M$. What is the base case?

Solution

Now, the answer is just $dp_M = \frac{1}{M!} f_M$. What is the base case?

Base case is $dp_0 = \begin{cases} 1, & V = 0 \\ 0, & V \neq 0 \end{cases}$, and $dp_1 = 1$.

Bars

A town has N ($1 \leq N \leq 5 \cdot 10^5$) houses in a row, each with one inhabitant. Houses can become bars. If house i becomes a bar, it generates p_i ($1 \leq p_i \leq 10^9$) profit per customer.

Each inhabitant is a customer of the bar immediately to the left and right of it's house (regardless of whether they themselves open a bar or not) as long as they exist.

Determine the maximum profit of the town if bars are optimally placed.

Example

$N = 4$, $p = [5, 2, 2, 6]$

It is optimal to place bars at the first and last house. Each will have 3 clients, providing a total profit of $3 \cdot (5 + 6) = 33$.

Subtasks

Subtask 1

$N \leq 5000$.

Subtask 2

No additional constraints.

Subtasks

Subtask 1

$N \leq 5000$.

Subtasks

Subtask 1

$N \leq 5000$.

Consider a DP solution.

Subtasks

Subtask 1

$N \leq 5000$.

Consider a DP solution.

Let dp_i be the maximum profit achieved by the first i houses, placing a bar at house i. The answer is dp_n.

Subtasks

Subtask 1

$N \leq 5000$.

Consider a DP solution.

Let dp_i be the maximum profit achieved by the first i houses, placing a bar at house i. The answer is dp_n.

Base case is $dp_1 = 0$.

Subtasks

Subtask 1

$N \leq 5000$.

Consider a DP solution.

Let dp_i be the maximum profit achieved by the first i houses, placing a bar at house i. The answer is dp_n.

Base case is $dp_1 = 0$.

Recurrence is

$$dp_i = \max_{j<i}\{dp_j + (i-j)\cdot(p_i + p_j)\},$$

with time complexity $O(N^2)$.

Solution

The recurrence from Subtask 1 is

$$dp_i = \max_{j<i}\{dp_j + (i-j)\cdot(p_i + p_j)\}.$$

What does $(i-j)\cdot(p_i+p_j)$ remind you of?

Solution

The recurrence from Subtask 1 is

$$dp_i = \max_{j<i}\{dp_j + (i-j)\cdot(p_i + p_j)\}.$$

What does $(i-j)\cdot(p_i+p_j)$ remind you of?

Double the area of trapezium with parallel side lengths p_i, p_j and height $i-j$.

Solution

The recurrence from Subtask 1 is

$$dp_i = \max_{j<i}\{dp_j + (i-j)\cdot(p_i + p_j)\}.$$

What does $(i-j)\cdot(p_i+p_j)$ remind you of?

Double the area of trapezium with parallel side lengths p_i, p_j and height $i-j$.

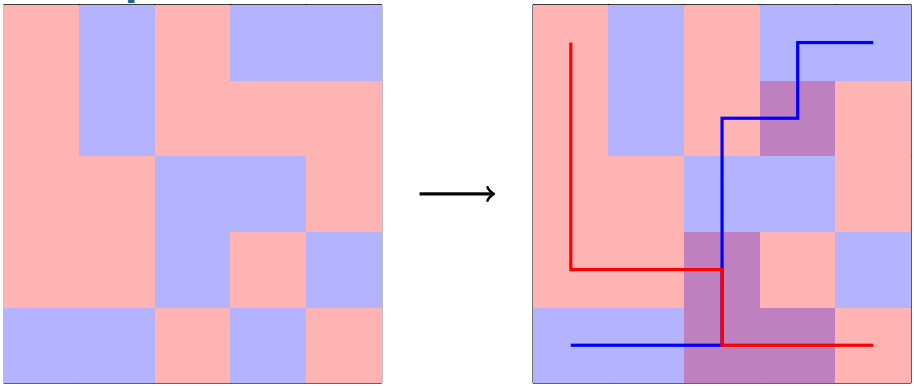
Can we now solve the problem? Hint: Plot (i, p_i) for all i.

Routing

Given an $N \times N$ ($3 \leq N \leq 500$) grid where each cell is coloured either red or blue, determine the minimum number of cells to colour purple such that:

- You can move from cell $(1, 1)$ to cell (N, N) by only passing through red or purple cells.
- You can move from cell $(1, N)$ to cell $(N, 1)$ by only passing through blue or purple cells.

Example



Need to colour 4 cells as shown above.

Subtasks

Subtask 1

You are only required to ensure Condition 1 holds.

Subtask 2

No additional constraints.

Subtasks

Subtask 1

You are only required to ensure Condition 1 holds.

Subtasks

Subtask 1

You are only required to ensure Condition 1 holds.

Dijkstra from top-left to bottom-right, with the cost of walking to red cell 0 and the cost of walking to a blue cell 1.

Solution

Do the same solution as Subtask 1 for bottom-left to top-right on blue cells.

Solution

Do the same solution as Subtask 1 for bottom-left to top-right on blue cells.

The answer is the sum of this and the result from Subtask 1.

Complexity is $O(N^2)$ with 0-1 BFS or $O(N^2 \log N)$ with Dijkstra.

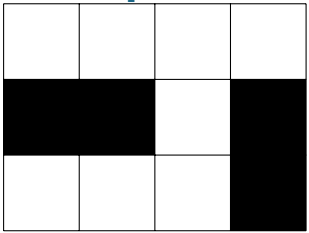
Game of Tiles

Two players play a game over a rectangular board with R rows and C columns ($1 \leq R, C \leq 50$). Initially some cells are painted black and the rest remain white.

Player 1 and 2 alternate turns making a move. First one that cannot make a valid move loses. Player 1 starts by choosing a white cell and writing the number 1. Each subsequent move i consists of writing i on an unused white cell adjacent (horizontally or vertically) to cell numbered $i - 1$.

Given the board, determine who wins.

Example



Player 2 will always win on this board.

Subtasks

Subtask 1

$R = 1$.

Subtask 2

No additional constraints.

Subtasks

Subtask 1

$R = 1$.

Subtasks

Subtask 1

$R = 1$.

Player 1 wins if there exists a component with an odd number of white cells.

Solution

Chess-board colour each component and create a bipartite graph. When does Player 1 win?

Solution

Chess-board colour each component and create a bipartite graph. When does Player 1 win?

Player 1 wins if not all component matchings are perfect. Otherwise, Player 2 wins. Why?