Grid – Analysis

Idea: AmirReza PourAkhavan, Mohammad Hossein Paydar, Implementation and Analysis: Ivan Lupov

The analysis will focus over the subtasks with $N = 1$. For their two dimensional counterparts it is enough to store and process the relevant information for all columns the same way it is stored for a singular row. This way we will take care of jumps that move in the same column. Also note that for clarity we skip the row index in any mentions of a – it is implied to be equal to 0.

Subtask 1: $N = 1, M \leq 200$

The dynamic programming state dp$[x]$ = max cost to reach position x should be obvious to contestants. Here it is enough to iterate over the position $y < x$, such that $y \to x$ is the last jump before steping on position x.

Subtask 2: $N = 1, A_j \leq A_j + 1$

The additional restriction here allows for a greedy idea: if possible, it is always better to jump as far as possible – the penalty stays the same but the difference in $|a_j - a'_j|$ can only grow. Thus it is optimal to jump from $(0, 0)$ to $(0, m - 1)$ immeadiately. It is worth noting that in the 2d case there appear to be two paths $(0, 0) \to (0, m - 1) \to (n - 1, m - 1)$ and $(0, 0) \to (n - 1, 0) \to (n - 1, m - 1)$ however both of them give the same number of coins.

Subtask 3: $N = 1, C = 0$

Here there is no penalty for making jumps and since $|x - y| \geq 0$, we can feel free to spam them. Making short jumps between adjacent cells in the grid is optimal and since $N = 1$ we can calculate the profit of this strategy with a simple for-loop. In the 2d case we will have to use dynamic programming.

Subtask 4: $N = 1, M \leq 50\,000$

A common idea in problems involving some cost function with absolute values (e.g: $|x - y|$) is to split its computation in two cases, depending on the sign of the inner value (here: $x - y$). Returning to our dynamic programming idea from subtask 1, here this reduces to:

$$\text{dp}[x] = \max\{\max_{y<x}\{\text{dp}[y] + a_y \mid a_x \leq a_y\} - a_x, \max_{y<x}\{\text{dp}[y] - a_y \mid a_x > a_y\} + a_x\}\}$$

In short, for lower (than a_x) values we care about the maximum dp $- a$ value and for higher we care about dp $+ a$ value. Here the time complexity is $O(m \log A)$ where A is the maximum value in the grid – still not optimal.

Subtask 5: $N = 1$

Here we will develop the key idea of the solution. Coding the 100-points solution is simply a matter of taking care of the columns the same way we will take care of the singular row now.

Key observation about the behaviour of the "absolute value" function is that $|x - y| = \max\{x - y, y - x\}$. This is really convienient because writing our dynamic programming transition was some equation of $|\cdot|$and maxs, but not we can make it simply an equation of maxs.

$$\begin{aligned} \text{dp}[x] &= \max_{y<x}\{\text{dp}[y] + |a_x - a_y|\} \\ &= \max_{y<x}\{\text{dp}[y] + a_x - a_y, \text{dp}[y] + a_y - a_x\} \\ &= \max\{\max_{y<x}\{\text{dp}[y] - a_y\} + a_x, \max_{y<x}\{\text{dp}[y] + a_y\} - a_x\} \end{aligned} \tag{1}$$

Thus it is enough to keep track of the maximum value only for the dp $- a$ and dp $+ a$ expressions. Note that we group terms such that all mentions of y can be taken care of together – this is common when rearranging equations to get a neater result.