Kick Start 2019 - Round D

Analysis: Food Stalls

Test set 1 (Visible)

We will start by fixing the position of the warehouse. Once that is done, we know the cost of putting up a stall at any spot. This allows for a dynamic programming approach where we let dp[i][j] represent the cost of putting j stalls in the first i spots. dp[i][j] is minimum of dp[i - 1][j] and (dp[i-1][j-1] + distance of ith spot from warehouse + $\mathbf{C}_i$) (we skip i if we have chosen to place the warehouse at i). dp[**N**][**K**] + cost of building the warehouse represents our solution for the chosen location of the warehouse.

By doing this for all possible positions for the warehouse, we will get the minimum cost for placing all **K** stalls. So this approach has a complexity of $O(\mathbf{N}^2\mathbf{K})$.

Test set 2 (Hidden)

Now suppose we decided on the **K** + 1 spots where we will place our stalls and warehouse but are yet to decide where to place our warehouse, let's say the co-ordinates of these spots are Y_1, Y_2, ..., $Y_{K + 1}$ in increasing order. Obviously, it'll be the spot Y_j such that $|Y_j - Y_1| + |Y_j - Y_2| + ... |Y_j - Y_{K + 1}|$ is minimum. This is the classic post office location problem and is solved by putting the warehouse in the median point of our chosen points (in case of even number of points with two medians, any one will yield an optimal answer).

Once we have this observation, we know that we must put floor(**K**/2) stalls on the left of our warehouse and **K**-floor(**K**/2) stalls on the right. If we can calculate, for every position of the warehouse, the minimum cost to place these points, we will be done. Here we will show how to calculate the minimum cost to place floor(**K**/2) stalls to the left of every position of the warehouse.

For simplicity, we assume the given points $\mathbf{X}_1$, $\mathbf{X}_2$, ..., $\mathbf{X}_\mathbf{N}$ to be sorted. Now, we will maintain a max-heap of size floor(**K**/2). For position i, we define $V_i = \mathbf{X}_\mathbf{N} - \mathbf{X}_i + \mathbf{C}_i$. Initially, we will store V_1, V_2, ..., $V_{\text{floor}(\mathbf{K}/2)}$ in our heap. We'll also maintain the sum of all the elements in our heap.

If we consider that we are placing the warehouse at $\mathbf{X}_{\text{floor}(\mathbf{K}/2)+1}$, then the minimum cost of placing all floor(**K**/2) stalls to the left of it is given by sum of all the elements in our heap - floor(**K**/2) * ($\mathbf{X}_\mathbf{N} - \mathbf{X}_{\text{floor}(\mathbf{K}/2)+1}$). We try to get this cost for all other valid placements of the warehouse.

So, for i = floor(**K**/2) + 1 to **N** - (**K**-floor(**K**/2)) - 1, we check if V_i is less than the maximum element of our heap. If it is then we remove the maximum and insert V_i and update the sum of all our heap elements by subtracting the difference of the removed element and the inserted element. We subtract floor(**K**/2) * ($\mathbf{X}_\mathbf{N} - \mathbf{X}_{i+1}$) from the sum and compare it with the minimum cost obtained thus far, updating it with the current cost if current cost is lower.

We can use a similar approach to get the cost of all elements to the right, there V_i will just be $\mathbf{X}_i$ and we should iterate from right to left and in each iteration, subtract (**K**-floor(**K**/2)) * $\mathbf{X}_i$ from the sum of the elements in the heap and update the minimum cost with that.

Finally we see for which point i, the sum of cost of placing floor(**K**/2) stalls on the left of it, **K**-floor(**K**/2) stalls on the right and $\mathbf{C}_i$ is minimised and output this value.