Kick Start 2020 - Round H

Analysis: Rugby

We can divide this problem into two parts: choosing where to place the line, and choosing the permutation of players on the line (in other words, which players should be on which points on the line).

Regarding the permutation of players, the first observation that we can make is that preserving the initial X-ordering of the players is an optimal strategy. That means we should place the player with the lowest **X** coordinate value as the first player on the line, the player with the second lowest **X** coordinate value as the second player on the line, and so on. If there are ties, we can break them arbitrarily.

We can prove that this greedy strategy is optimal. First let us assume that there are two players with X-coordinates $\mathbf{X}_1$ and $\mathbf{X}_2$, and let us denote the X-coordinate of the start of line as startX, whereas $\mathbf{X}_1 \leq \text{startX} < \mathbf{X}_2$. If we do not preserve the players' relative position on the final solution, we have that the first player will move for (startX + 1) - $\mathbf{X}_1$ steps, and the second player will move for $\mathbf{X}_2$ - startX steps, so the final solution equals -$\mathbf{X}_1$ + $\mathbf{X}_2$ + 1 steps. However, if we preserved the players' relative positions, the first player would move for startX - $\mathbf{X}_1$ steps, and the second player would move for $\mathbf{X}_2$ - (startX + 1) steps, so the final solution would equal -$\mathbf{X}_1$ + $\mathbf{X}_2$ - 1 steps, which is lower than the above strategy. For the other possible values of startX, which are startX < $\mathbf{X}_1$ or $\mathbf{X}_2 \leq$ startX, both permutations for the players result on the same amount of steps. Therefore, we can reach the conclusion that, for any solution, by swapping two players' order in order to preserve the players' initial relative positions, the total number of steps remains the same or decreases.

Now imagine that we have an arbitrary optimal solution S (not necessarily sorted on the players' X-coordinates). If we perform a swap, as described above, the solution does not get any worse, so it is still optimal. If we perform such a swap enough times we will reach a point in which the players are in non-descending order of their X coordinate, which is what our greedy solution looks like, and the solution is still optimal. Therefore, our greedy solution is optimal.

Regarding where to place the line, let *solutionFirstX* be the **X** position of the first point on the line, and let *solutionRow* be the **Y** position of the line.

Test Set 1

Given the constraints of the Test Set 1, we can iterate through all possible values for *solutionFirstX* and *solutionRow* and check which combination results on the minimum total number of steps. The grid is unbounded, however we can bound the possible values for *solutionFirstX* based on the initial **X** positions of the players, because placing the line entirely to the left of the player with the lowest **X** coordinate would make them all take an extra step to get there, which intuitively leads to a worse solution. The same applies on the right side. Let *minPlayerX* represent the **X** value of the player the most to the left, and *maxPlayerX* represent the **X** value of the player the most to the right. The number of possible values for *solutionFirstX* range between *minPlayerX*-(**N**-1) and *maxPlayerX*. Given similar definitions for *minPlayerY* and *maxPlayerY*, the same logic applies for *solutionRow*, which ranges between *minPlayerY* and *maxPlayerY*.

To calculate what is the total number of steps for the players to reach the line at *candidateFirstX* and *candidateRow*, iterate through all players in non-descending X-coordinate order and add their Manhattan distances to their relative locations on the line. In other words, the total number of steps equals the sum of $|\mathbf{X}_i$ - (candidateFirstX + (i - 1))| + $|\mathbf{Y}_i$ - candidateRow|, for every player i in non-descending order of $\mathbf{X}_i$.

The complexity of given solution is O(**N** × log **N** + (*maxPlayerX* - *minPlayerX* + **N**) × (*maxPlayerY* - *minPlayerY* + 1) × **N**).

Test Set 2

Given that the limits for **N** and the players initial positions are much greater in Test Set 2, the above mentioned solution would not be fast enough.

To divide our solution into smaller steps, we should notice that horizontal and vertical movement can be solved independently. In other words, the value *solutionRow* does not affect how many horizontal steps the players will take, as well as *solutionFirstX* does not affect how many vertical steps the players will take.

Let us start by finding the value for *solutionRow*. We can observe that most players will have to move vertically to reach *solutionRow*, either by moving up or moving down (and some will not have to move because they are initially on *solutionRow*). We want the total number of steps to be as small as possible, so a good candidate is the Y-position of the player that is positioned as the median among all players, regarding their initial Y-position.

We can prove that this is optimal as follows: first, let us start with the assumption that **N** is odd. If we set *solutionRow* to be equal to $\mathbf{Y}_{\lceil \mathbf{N}/2 \rceil}$ (the Y-position of the player in the middle), then we can see that there will be at least ⌊**N**/2⌋ players who are initially below or at the same line as the one in the middle, and at least ⌊**N**/2⌋ players who are initially above or at the same line as the one in the middle. Let us call *totalStepsBelow* the total number of steps taken by the ⌊**N**/2⌋ players who are initially below or at the same line as the one in the middle, and let us call *totalStepsAbove* the total number of steps taken by the ⌊**N**/2⌋ players who are initially above or at the same line as the one in the middle. As is, the total number of steps equals *totalStepsBelow* + *totalStepsAbove*. If we move *solutionRow* one position up, then the player in the middle and all the players that were initially below or at the same line as the one in the middle will have to take one extra step to reach *solutionRow*, and, in the best scenario, all the players that were initially above as the one in the middle will have to take one less step to reach *solutionRow*. Therefore, the total number of steps would equal (*totalStepsBelow* + ⌊**N**/2⌋ + 1) + (*totalStepsAbove* - ⌊**N**/2⌋), which equals *totalStepsBelow* + *totalStepsAbove* + 1, which is worse than our first guess. The same applies for moving *solutionRow* one position down. A similar logic can be used for when **N** is even.

Now we can use a similar approach for finding the value for *solutionFirstX*. Taking into account the fact that we want to preserve the players initial X-ordering, we can observe that for each player i, who has k_i players to his left, the optimal value for *solutionFirstX* would be $\mathbf{X}_i - k_i$, because this way he would not have to move horizontally at all. Using the same logic described for finding *solutionRow*, we can find the optimal value for *solutionFirstX* by picking the median of all players optimal values for *solutionFirstX*.

Using these strategies we can find the value for *solutionRow* in O(**N** × log **N**) time by sorting the players by their initial Y-coordinate and picking the median value. Similarly, we can find the value for *solutionFirstX* in O(**N** × log **N**) time by sorting the players by their initial X-coordinate and picking the median optimal value. Finally we can calculate the total number of steps in O(**N**) time, by using the strategy mentioned on Test Set 1 analysis. Therefore, the time complexity of the solution is O(**N** × log **N**).