Kick Start 2019 - Round C

Analysis: Catch Some

Test set 1 (Visible)

Firstly, observe that if Bundle switches the color of her shirt then she should never wear a previously worn color again as that would not give us a better answer. Also, for dogs of the same color, if she observes one dog then she should have observed all dogs who are at a position less than that dog. The condition that she doesn't need to return to her home after observing the last dog is a little tricky. Let's first solve the problem where she must return home after observing the last dog.

For this simplified version, we notice that the order of the color of shirts that Bundle has to put on doesn't matter. This gives rise to the following dynamic programming approach.
Let dp[i][j] denote the minimum time Bundle needs to observe j dogs such that she has either observed or has decided she'll not observe dogs of colors 1 to i.
Hence, dp[C][**K**] becomes our answer, where C is the total number of different colors.

To calculate dp[i][j] we loop upon the number of dogs Bundle could have observed for the (i-1)th color, and take the minimum cost among them. That is,
dp[i][j] = min(dp[i-1][j] + 0, dp[i-1][j-1] + 2 * $X_{i,1}$, dp[i-1][j-2] + 2 * $X_{i,2}$, dp[i-1][j-3] + 2 * $X_{i,3}$ ...)
Where, $X_{c,d}$ represents the distance of the d-th dog of c-th color from Bundle's house.

Since the size of the loop for a particular color is the number of dogs of previous color + 1, the sum of sizes of all loops is O(**N**). Hence, we have O($\mathbf{N}^2$) states and do a total of O(**N**) transitions. This gives us a run time of O($\mathbf{N}^2$) for a given color as the last color.

To complete the solution, we simply do this considering every color as the last color and add in the cost of observing dogs of the last color separately. This will incur an additional O(**N**) loop to iterate over all colors resulting in a final complexity of O($\mathbf{N}^3$).

Test set 2 (Hidden)

We can optimize the previous solution to not do the same calculations with every color as the last color. This can be done by adding a boolean parameter in our state representing if we have already decided on the last color. We simply don't add the cost of returning back home whenever we set this parameter to true.

Essentially, we have dp[i][j][k] where i and j still represent the same things and k is a boolean denoting if we have chosen the last color in the first i colors.

Now, dp[i][j][0] = min(dp[i-1][j][0] + 0, dp[i-1][j-1][0] + 2 * $X_{i,1}$, dp[i-1][j-2][0] + 2 * $X_{i,2}$ ...)
dp[i][j][1] = min(min(dp[i-1][j][1] + 0, dp[i-1][j-1][1] + 2 * $X_{i,1}$, dp[i-1][j-2][1] + 2 * $X_{i,2}$ ...),
min(dp[i-1][j][0] + 0, dp[i-1][j-1][0] + $X_{i,1}$, dp[i-1][j-2][0] + $X_{i,2}$ ...))

Again we have O($\mathbf{N}^2$) states and do a total of O(**N**) transitions. But we just do this once and not once for every color, yeilding a final complexity of O($\mathbf{N}^2$).