In the arcade game *Dance Dance Revolution* (DDR), players stand on a stage and hit arrows as they scroll across the screen. More specifically, a sequence of n arrows (←, ↑, →, ↓) will scroll across the screen, and as each arrow hits the top of the screen, the player must stand on the corresponding arrow on the stage.

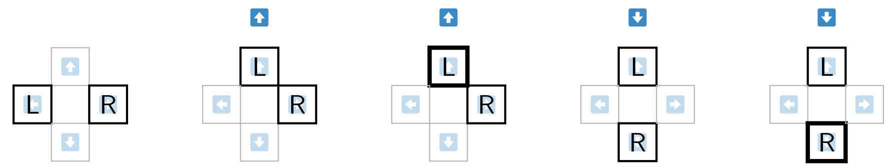
We play a variant of DDR, aptly named *Don't Dance Revolution* (DDR2), where the goal is to play the game like DDR but move as little as possible. The game plays like DDR except when an arrow reaches the top of the screen and the player already has a foot on the correct arrow, then the player is awarded one lazy point. If neither foot is on the correct arrow, then the player must move *exactly* one foot from its current location to the correct arrow on the platform.

Unfortunately, the game is a bit unforgiving: any wrong move will cause the player to lose the game and *all* of their lazy points. Wrong moves include:

- Failing to step on the correct arrow.
- Moving more than one foot at any given time.
- Moving either foot when the player is already stepping on the correct arrow.

You are given a sequence A of n arrows. Assume that your left foot starts on ← and your right foot starts on →, and you have memorised the entire sequence of arrows.

For example, consider the following sequence: ↑↑↓↓. We can earn up to two lazy points as follows:



Start game! *Lazy point!* *Lazy point!*

(a) Show that it is always possible to earn at least $\lfloor n/4 \rfloor$ lazy points during a round of DDR2.

(b) Describe an $O(n)$ algorithm to determine the maximum number of lazy points you can earn in a round of DDR2.

Hint. *We only have two feet...*

Solution.

(a) Consider splitting the sequence into blocks of size 4, leaving the remainder on the end if not divisible. Consider now two cases:

- The block contains all different arrows. One foot can handle three of them, and the other will be able to obtain the rest of them, earning at least one lazy point.
- The block contains at least one repeat. Step on the first instance with one foot, and handle the others with the other. You will earn a lazy point in that way.

In both cases, at least one lazy point is earned. Since there will be $\left\lfloor \frac{n}{4} \right\rfloor$ blocks of 4 in total, we have at least that many lazy points.

(b) We define a Dynamic Programming table DP with dimensions $(n+1) \times 4 \times 4$, with $DP[i][l][r]$ representing the maximum possible score after the ith point in the sequence, l is the position of the left foor and r is the position of the right. Set $DP[n][l][r] = 0$ for all l and r values. Now, for each arrow i, starting from the end, we must find the maximum of the possible moves.

- If l or r is the optimal arrow, then we update $DP[i][l][r] = DP[i+1][l][r] + 1$.
- If l or r is not on the optimal arrow, then we calculate the maximum for each food indi-

vidually. That is, we find

$$\max\{DP[i+1][A[i+1]][r], DP[i+1][l][A[i+1]]\}.$$

The optimal solution will be $DP[0]$ where l and r are the defined starting positions. This algorithm cycles through each element of DP and runs $O(1)$ operations for each. So, our total time complexity will be $O((n+1) \times 4 \times 4) = O(n)$.