Problem A. PE class

Subtasks 1 and 2

Let's iterate through all possible strings of length 2^{2n} and check each one. For $n = 2$, all cases can be examined manually.

Subtask 3

There are only two cases: $a_i = 0$ or $a_i = -1$. When $a_i = 0$, the answer will be $LRLRLR\ldots$, and when $a_i = -1$, the answer will be $RLRLRL\ldots$.

Subtask 4

If $a_1 = 0$, then $s_1 = L$, otherwise $s_1 = R$. Then we can keep track of how many L's and R's have been placed in the prefix, and from this we can uniquely calculate the next character.

Subtask 6

Claim. The multiset of numbers obtained for L is the same as the multiset obtained for R.

Proof. It can be proven using mathematical induction. In any case, there will be adjacent L and R. It is easy to notice that the obtained numbers for them are the same. And since they do not affect all the others, we can remove them. Then we can proceed to the problem with $n - 2$.

It turns out that each number in the array a occurs an even number of times. If we leave half of the occurrences for each number, these will be the obtained numbers for all L. Let's denote this array as La. Now, using the minimum number of R symbols, we will obtain all the numbers from the array La. To do this, we will sort the values in the array in ascending order and insert the necessary elements in this order.

Problem B. Batyr I and Tima the Great

- Author — Batyr Sardarbekov;
- Developers — Dinmukhamed Tursynbay, Aibar Kuanyshbay, Batyr Sardarbekov.

Subtask 1 ($m = 0$) — 5 points

In this subtasks, it is simple to see that you either move in circle clockwise or anticlockwise so answer is minimum between $|x_i - y_i|$ or $L - |x_i - y_i|$.

Subtask 2 $L, m, q \leq 10^2$ — 8 points

Let's build graph for each query, then Batyr's roads has weight 1 and Tima's roads has weight 0. And run shortest path algorithms to find answer.

Subtask 3 $L, m, q \leq 10^3$ — 11 points

Let's build graph before processing queries like in second subtask, and then run BFS or Dijkstra for each pair in query to find answer.

Subtask 4 $m, q \leq 10^3$ — **10 points**

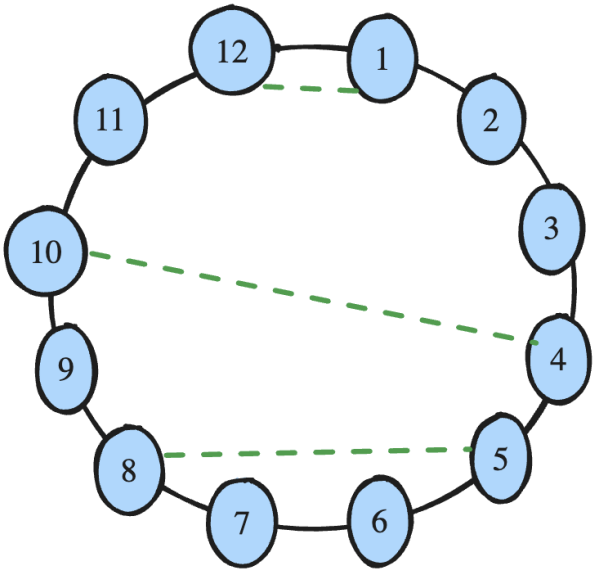
Notice that in previous subtask, you can build graph with nodes which appear only in one of queries or in Tima's edges. so we have $2(m+q)$ nodes on which we can build graph, and solve similar to third subtask.

Subtask 5 $b_i < a_{i+1}$ — **12 points**

Consider Tima's roads as segments. Observe that no two segments intersect, and none of them completely cover each other. When moving from x to y, the distance covered is equal to $y - x -$ sum, where sum is the total length $\sum(b_i - a_i)$ of segments that lie entirely within the range $[x, y]$.

Subtask 6 $a_i < a_{i+1}, b_{i+1} < b_i$ — **14 points**

Consider Tima's roads as segments. Note that while no segment intersects, each of them covers the other, implying a hierarchy among them. Add segment $[0, L]$ and then build a tree graph on this set of segments. For each segment, identify the shortest segment covering it, which becomes its parent in the tree. Additionally, in this tree, each point is part of the shortest segment covering it. Achieve this using a scanline: sort segments by their left side and maintain a stack of segments to easily find the parent. The weight of each edge is the distance between the closest points of the parent and son segments. To better understand how to handle queries, consider the given graph below, where $L = 12$, and Tima's roads are represented by the green lines.



As example to understand, let's use 3 and 9. so 3 will be linked to segment $[1, 12]$. while 9 to $[4, 10]$ in our tree. So in this case it is optimal to move from 3 to segment $[4, 10]$ and then from 10 to 9, so answer is 2. And let's consider another query 4, 9 in this case it is trivial we move from 4 to 10 and then 9, so answer is 1. in first example we moved to segment $[4, 10]$ which is son lca ($[1, 12]$) of segments $[4, 10]$ and $[1, 12]$, and in second it is going to lca ($[4, 10]$). It can be shown, that it's optimal to either go from both points to lca, or going to son of lca and then between them to each other. Time complexity is $O(nlogn)$.

Subtask 7 $|x_i - y_i| = 1$ $(1 \leq i \leq q)$ — **18 points**

In this version of problem, if x_i and y_i are connected by Tima's roads, the answer is 0; otherwise, it is 1. To determine if they are connected in that way, let's merge two of Tima's roads if they intersect each other but do not entirely cover one another. First, connect roads by their endpoints. When considering

roads a and b, we need to identify all the roads that intersect them. For instance, if $a < b$, we deem some endpoints of road u as valid if u lies between a and b, and the rightmost endpoint from u is greater than or equal to b. Conversely, if $a > b$, for endpoints u in between, we need to check if the leftmost endpoint from u is less than or equal to b. We can find such endpoints using a segment tree and use a disjoint-set union (dsu) structure to connect them as components. This process can be accomplished in $O(n \log n)$ time.

Subtask 8 No additional constraints — 22 points

After building components same to subtask 7, we are left we almost same problem as subtask 6 but you also need to consider that compare to subtask 6 components are set of points rather than segment.

Problem C. Lazy, but honest

- $D = 1$, 5 points.

  We have to complete each task we receive on the same day. First, check that every $b_i \leq a_i$. Then print the number of days with $b_i \neq 0$. $O(n)$

- $N \leq 18$, 7 points

  If we decide to work some day it's always optimal to do as many tasks as possible and we start from the earliest tasks.

  For each day we have two options: not to work or to do as many tasks as possible. Use recursion. Store for each day how many tasks are left uncompleted in that day. $O(2^n * n)$

- All a_i equal, 18 points.

  Greedy solution: if we decide not to work on that day can we complete all tasks(current and future) in any number of working days? If not we must work or else skip that day. To check that we can use a segment tree to find an interval with maximal value $b_l - a_l + b_{l+1} - a_{l+1} + \ldots + b_r - a_r$. $O(nlog(n))$

- $D = N$, 18 points.

  Let's find a solution for the suffix i with the maximal sum of b_j in working days. To move from i to $i-1$: If we can complete a_{i-1} tasks in current working days use them. Else find the largest b_j in not working days and mark it as a working day until we complete all a_{i-1} tasks. $O(nlog(n))$

- $N \leq 2000$, 16 points.

  Let x_i be the number of completed tasks in days $1, 2, \ldots i$. Then x_i must be not greater than $b_1 + b_2 \ldots + b_i$ and not less than $b_1 + b_2 \ldots + b_{i-D+1}$.

  Use $dp_{i,j} = x$ where i number of days, j number of working days, x maximal number of completed tasks. Transitions are easy and don't forget to check $b_1 + b_2 \ldots + b_{i-D+1} \leq dp_{i,j} \leq b_1 + b_2 \ldots + b_i$. $O(n^2)$

- No additional constraints, 36 points

  Use the same dp from the previous subtask. The key observation is that dp_i is a convex function. Then we can use *set* to store $dp_{i,j+1} - dp_{i,j}$. And transitions are just inserting b_i. To check bounds for value x store min value and max value in dp. $O(nlog(n))$