Problem A. Random Order

To solve the problem, write down the index of each number into an array $i_1, \cdots, i_n$. Then you need to find the maximum possible k such that $max(i_1, \cdots, i_k) < min(i_{n-k+1}, \cdots, i_n)$.

This can be done by gradually increasing k and maintaining the maximum on the prefix and the minimum on the suffix as long as the condition above holds. This solution works in $\mathcal{O}(n)$.

Problem B. Rest Point

To solve the problem, let us consider the following subproblem: given the lengths of vectors $a_1, \cdots, a_n$, determine whether it is possible to choose their directions so that their sum is the zero vector.

Then, if the maximum element a is greater than the sum of all the other elements, it is impossible to cancel out this largest vector.

Otherwise, the sum can be the zero vector — this can be proved, for example, as follows. Suppose initially all other vectors are directed along the vector of maximum length and are ordered. Then take the first of them and continuously rotate it by 180, while directing all subsequent vectors during the motion so that they remain codirected and their endpoint lies on the line passing through the extension of the vector with maximum length.

In the end, we obtain all vectors directed toward the vector with maximum length (initially, all vectors except the one with maximum length were directed in the opposite direction). But all motions were continuous, and initially the sum of the lengths of the other vectors was at least the length of the maximum vector. This means that at some moment the sum of all vectors was exactly the zero vector.

Let us return to the original problem. Consider the segment $[l_i; r_i]$ with the maximum l_i. Then consider the sum of all other r_j, denoted $restsum$. If $l_i > restsum$, then no choice of vector lengths can produce the zero vector, because any other value chosen from segment i only increases the maximum, while any other values chosen from the remaining segments only decrease the remaining sum.

Otherwise, we can set the length of segment i equal to l_i, and for all other segments set $len_j = min(r_j, l_i)$ — this number always belongs to the segment $[l_j; r_j]$, since $l_j \le l_i$. Also, the sum of these numbers is at least l_i, because if for some j we have $len_j = l_i$, then the sum condition is immediately satisfied; otherwise, it holds because the sum of the remaining r_j is at least l_i.

The solution works in $\mathcal{O}(n)$.

Problem C. Alternative Worlds I

To solve the problem, let us note several properties of an optimal partition.

1) If the median of the set S_1 is x_1 and the median of the set S_2 is $x_2 \ge x_1$, then $x = med(S_1 \cup S_2) \in [x_1; x_2]$.

x cannot be greater than x_2, because when adding the elements of S_1 to S_2, the number of elements greater than x_2 is smaller than the number of elements not greater than x_2.

By analogous reasoning, we obtain that x cannot be smaller than x_1.

2) There exists an optimal partition in which each set contains at most one non-negative number.

Indeed, if there is a set of length 2 consisting of two non-negative numbers, then it can be split into 2 sets without decreasing the sum of medians.

Otherwise (if the length is at least 3), we can remove the first and the last elements from this set into a separate set. The median of the original set will not change, while the median of the new set will be at least 0. Therefore, the sum of medians does not decrease.

By repeatedly applying the reasoning above (for example, by induction), we obtain that $statement2$ is true.

Now let us describe one of the optimal solutions.

If the number of non-negative numbers is at least the number of negative numbers, then we make a

separate set for each non-negative number and then add the remaining negative numbers one by one to some of these sets.

Then the negative numbers contribute nothing, while all non-negative numbers are summed up — this is the maximum possible sum of medians.

If there are more negative numbers, then *statement*2 implies that if there are *cnt* non-negative numbers, then only *cnt* negative numbers can be added to them so that they contribute nothing. Clearly, it is optimal to add the smallest negative numbers — this way we create *cnt* sets of two numbers, one of which is non-negative and the other negative.

At the same time, *statement*1 implies that it is beneficial to leave the remaining numbers in one set.

Thus, we have obtained an optimal partition for each case, from which it is easy to derive the answer.

Problem D. Alternative Worlds II

To solve the problem, let us note several properties of an optimal partition.

1) If the median of the set S_1 is x_1, and the median of the set S_2 is $x_2 \geq x_1$, then $x = med(S_1 \cup S_2) \in [x_1; x_2]$.

This statement is proved similarly to the corresponding statement in the previous problem.

It also follows from this, in the same way, that only 1 set may have a negative median (otherwise these 2 sets can be merged into one).

2) Let us show that among the sets with non-negative median, only one may contain more than 1 element (that is, an optimal partition with this property exists).

First, if a set with a positive median contains more than 1 element, then this set has odd size; otherwise, one of the elements a, b contributing to the median can be separated into its own set. Then the sum of medians of these elements changes from $(a + b)/2$ to $a + b$.

Also, in a set with a positive median and more than 1 number, all numbers smaller than the median are negative – otherwise, those positive numbers can be moved into separate sets, which does not decrease the sum of medians.

But then, if there are 2 sets with a positive median and more than 1 number each, with medians $a, b(a \leq b)$, then one can separate b into its own set and merge all remaining numbers into 1 set whose median will be equal to a — that is, the sum of medians does not change under this transformation.

This is exactly what had to be proved for statement 2.

3) Let the set with a negative median be called A and have length m, and let the set with the number of positive elements greater than 1 be called B and have size $k = 2x + 1(x \geq 0)$. Let us prove that there exists an optimal partition in which only one of these sets exists.

We will prove this by contradiction. Suppose such sets do exist.

Then the set B contains exactly $x+1$ non-negative numbers (this was proved earlier) and x negative ones. Moreover, since all these numbers do not contribute to the median, it is beneficial to take the x smallest negative ones. Only the smaller non-negative number contributes, so it is beneficial to take only the first $x + 1$ positive ones.

But then in the set A, all negative numbers are no smaller than the negative numbers in the set B, and similarly for the non-negative numbers.

This means that one can, for example, take the smallest and the largest number from B and add them to A, and the sum of medians will not change. Then, by repeating this process many times, the set A will eventually consist of only one element.

4) Now let us show that if only the set A or B exists, then it consists of some prefix of the numbers in the sorted array.

Indeed, if B exists, then it is determined uniquely and consists of a prefix (as proved earlier).

Otherwise, there exists A, consisting of all negative numbers and some non-negative ones. But then A contains the smallest possible non-negative numbers (otherwise, one can replace numbers of A with smaller positive ones placed in separate sets and not decrease the sum of medians).

Thus, it is sufficient to check only the situations in which some prefix of the initially sorted array is a separate set, and all remaining elements are sets of size 1. The maximum sum of medians in such partitions will be the answer.

Problem E. Dark Labyrinth

We will consider all vertices from the current array c as a biconnected component, which we will treat as a single vertex. Initially, $c = [1]$, and this is valid.

1) Suppose the graph contains a cycle of length at most $k+1$ passing through this biconnected component, treated as a single vertex. Then it is claimed that all vertices of this cycle can be added to c.

Indeed, one can place a lamp in the component c, and then, if the cycle length is at most k, place lamps in each of its vertices. Otherwise, if the cycle length is exactly $k+1$, it is enough to place lamps in the next $k-1$ vertices of the cycle after c, since then only one vertex will remain unlit, which means it is still possible to traverse the tunnels incident to it.

Then, by traversing this cycle repeatedly, one can add all its vertices to c, and since it was a cycle, it follows that c will remain a biconnected component.

After that, all lamps can be picked up.

2) Vertex n is still not in c, but every cycle passing through c (considered as one vertex) contains at least $k+2$ vertices.

Then it is impossible to reach any vertices outside c (while still being able to keep all lamps). This is true because in order to start moving along vertices not belonging to c, one has to place a lamp in c and then make a step out of the component c. But after that Oliver will not be able to return to the component c.

We prove this by contradiction. Suppose it is possible to reach a vertex u that has an edge to c. Consider the first time we reach such a vertex. Before that, Oliver had not entered any vertices leading to c, and therefore could neither enlarge c nor collect lamps. That is, vertex u can be reached using k lamps without collecting them.

Let $d(v)$ be the minimum number of edges that must be traversed from c to reach vertex v. Since every cycle through c contains at least $k+2$ edges, and u has an edge to c, we have $d(u) \geq k+1$.

We will prove by induction that, using k lamps, one can reach only vertices v for which $d(v) \leq k$.

Base of induction: $k = 1$. Then, to start moving, there is only one option — place a lamp in the vertex c — and then it is possible to reach only vertices that have a direct edge from c.

Induction step: suppose the statement is true for k; prove it for $k+1$.

Again, we argue by contradiction. Suppose it is possible to reach a vertex v with $d(v) = k+2$. Let this be the first reachable vertex with $d(v) = k+2$. Then the previous vertex u has $d(u) = k+1$, and in order to reach v, it was necessary to place a lamp in u. Consider the first moment when Oliver could enter vertex u — before that, there was no lamp in u, namely the one he would later place there, but he still managed to reach vertex u with $d(u) = k+1$ using only k lamps — contradiction.

This means that it is impossible to reach vertices with $d(v) \geq k+2$, which is exactly what had to be proved.

Thus, it is impossible to reach the considered vertex u with $d(u) \geq k+1$ using k lamps.

Therefore, we obtain the following algorithm: we need to find the shortest cycles from the current component c and, while their length does not exceed $k+1$, contract them; at the end, check whether $n \in c$.

To find the shortest cycle, one can use bfs from the vertex c. Since contraction can happen at most n times and bfs takes $\mathcal{O}(m)$ time, the solution works in $\mathcal{O}(mn)$.

Problem F. Thanos Sort

Let us consider which subsegments of the original array can be obtained by performing Thanos sort. One can notice that for the original array, we either stop sorting immediately (using the whole array), or move to one of the parts obtained by splitting the array in half as evenly as possible. That is, all possible final subsegments of the original array themselves form a tree structure similar to a segment tree (of depth $\mathcal{O}(\log n)$).

Suppose some segment at depth h (for the root, $h = 0$) is sorted, and all its ancestors are not sorted. Then it can be reached with probability 2^{-h}, and if it has length len, then it contributes $len \cdot 2^{-h}$ to the expectation. Therefore, to maintain the answer, after each update, we will recompute the contribution of all relevant vertices.

Then, when changing the value at some index, we traverse all $\mathcal{O}(\log n)$ vertices of the segment tree and, for each of them, check whether the corresponding segment is now sorted.

Next, we need to carefully recompute the answer — if the new deepest sorted segment is higher than the previous one on this path, then we need to remove the contribution of all segments below it (each corresponding vertex can be reached from one of the considered $\mathcal{O}(\log n)$ vertices by one edge, since only one element changed). If the new deepest sorted segment is lower than the previous one, then we need to add the segments lower on the path, similarly to the case above.

Thus, only $\mathcal{O}(\log n)$ vertices are considered for each update.

To determine for each of the $\mathcal{O}(\log n)$ vertices on the path whether the corresponding segment is sorted, one can build an array of differences of adjacent elements and a segment tree over it, and then use binary lifting on this tree to find the largest sorted segment, from which the deepest vertex corresponding to a sorted segment can be computed. This array of differences can also be maintained in the segment tree itself.

The solution described above works in $\mathcal{O}(n + q \log n)$.

Problem G. Reconstruction

We are given a graph showing, for each number, its nearest number (its index). One can observe that this graph will always consist of connected components, each of which is a cycle of length 2, and each vertex of this cycle may have one incoming edge from some vertex, which in turn may also have at most one incoming edge, and so on.

Therefore, to check existence, it is sufficient to verify that the given graph has the corresponding structure.

It is also obvious that, in order to minimize the maximum element of a, the distances between elements should be as small as possible. Therefore, the distance between the elements of each cycle must be 1, then to the next vertex in each direction 2, and so on, increasing the distance by 1 each time.

This gives us a set of components, for each of which there are two distances from the ends, a_i and b_i. Then, between components that are adjacent to each other on the number line, the distance must be $max(x, y) + 1$, if the two components have distances x, y between the bordering vertices and the previous points in their components.

So, to reconstruct the answer, we need to restore the optimal order of the components and their orientation (which side has a at the end and which side has b).

At the same time, the lower part of the first component x and the upper part of the last component y contribute nothing.

However, suppose that the problem is cyclic and they contribute an additional value $max(x, y) + 1$. Then it can be solved greedily: take the component with the maximum value among all a_i, b_i. Since it definitely contributes, it is optimal to pair this maximum value with the next maximum value among all remaining ones. Thus, we can merge the two found blocks into one and add their contribution and the resulting sum.

By repeating this process many times, we obtain an optimal solution for the cyclic problem. To return to the current problem, it is enough to remove the split at the maximum value and reconstruct the answer.

The solution complexity is $\mathcal{O}(n \log n)$.