Problem A. Adjacent Product Sum

Let $a_1 \geq a_2 \geq \ldots \geq a_n$. Then, we can show that one of the optimal ways to arrange them on the circle is: $a_1, a_3, \ldots, a_{2\lfloor \frac{n-1}{2} \rfloor + 1}, a_{2\lfloor \frac{n}{2} \rfloor}, \ldots, a_4, a_2$. In other words, first put all elements on odd positions, then all on even positions, in a different order.

For example, for $n = 7$, this would look as: $(a_1, a_3, a_5, a_7, a_6, a_4, a_2)$.

Let's show that it's optimal. We will use a simple fact: if $a \geq b$, $c \geq d$, then $ac + bd \geq ad + bc$ (as this is equivalent to $(a - b)(c - d) \geq 0$).

Let's show by induction that for each k there exists an optimal arrangement (optimal = with the largest possible sum of adjacent products) in which elements $a_1, a_2, \ldots, a_k$ form a consecutive segment, and numbers a_{k-1} and a_k are at the ends of this segment. To show this for $k = 2$, suppose that a_1 and a_2 aren't adjacent. Let the element directly clockwise to a_1 be a_x, and directly clockwise to a_2 be a_y. Then, let's reverse entire subsegment from a_x to a_2 (considered clockwise). The sum of products will change by $a_1a_2 + a_xa_y - a_1a_x - a_2a_y \geq 0$.

Now that we proved our statement for $k = 2$, let's make the induction step: consider an optimal arrangement in which this holds for $k - 1$. Wlog, in this segment a_{k-2} is the first element, and a_{k-1} the last clockwise (that is, if we traverse this segment clockwise, a_{k-2} is the first element we see, and a_{k-1} the last one.

Suppose that a_k isn't adjacent to a_{k-2}. Then, let a_x be the element next counterclockwise to a_{k-2} (by choice $x > k$), and a_y be the next element counterclockwise to a_k (by choice $x \geq k-1$). Then, let's reverse the (clockwise) segment from a_k to a_x. The sum will change by $a_ka_{k-2} + a_xa_y - a_{k-2}a_x - a_ka_y$, which is nonnegative as $a_{k-2} \geq a_y$ and $a_k \geq a_x$.

So, just sort the elements, arrange them in this order, and find the sum of products.

Problem B. Binary Arrays and Sliding Sums

Let's try to see how arrays a with the same $f(a)$ look. Consider all arrays a for which $f(a) = b$. What do we know about them?

First, we know $a_{i+k} - a_i = b_{i+1} - b_i$. Consider elements $a_i, a_{i+k}, a_{i+2k}, \ldots, a_{i-k}, a_i$ (add k until we meet a_i again for the first time). What's the length of this cycle? It's $\frac{n}{\gcd(n,k)}$, where $\gcd(n, k)$ denotes the greatest common divisor of n and k.

Let's denote the set of indexes for which a_i is in this cycle by S. If at least for one $i \in S$, $b_{i+1} - b_i \neq 0$, then we can uniquely determine the values of all a_i for $i \in S$ (as $a_j - a_i$ can only be 1 when $a_j = 1, a_i = 0$, and can only be -1 when $a_j = 0, a_i = 1$). If all these b_i are equal, then all a_i are also equal, but here we have two choices: to make all of them equal to 0 or to 1.

Now, suppose that there are t cycles, in which all a_i have to be equal. If we choose x of these cycles to have all ones, and the rest to have all zeros, then the sum on each subsegment of length k increases by $x\frac{k}{\gcd(n,k)}$ (each cycle intersects with each segment of length k by exactly $\frac{k}{\gcd(n,k)}$ cells). So, for a given a, all arrays a_1 with $f(a_1) = f(a)$ look as follows:

- Consider all $\gcd(n, k)$ cycles $(a_i, a_{i+k}, a_{i+2k}, \ldots)$ for each i from 1 to $\gcd(n, k)$.
- For each of these cycles: if not all a_i in it are equal, then in all arrays a_1 with $f(a_1) = f(a)$, these a_{1i} will be the same as in a.
- If there are t cycles in which all a_i are equal, and x of them consist of all ones, then in a_1 in these cycles, all elements are also equal, and exactly x of them consist of all ones.

Now, let's count the number of different equivalent classes. Let $l = \gcd(n, k)$. Suppose that t of these cycles have all equal elements. Then there are $\binom{l}{t}$ ways to choose these cycles, and $2^{\frac{n}{l}} - 2$ ways to fill each

other cycle. As for t cycles with equal elements, we only care how many of those cycles are all ones, and there are $t+1$ possibilities. So, we have to find

$$\sum_{t=0}^{t=l}(t+1)\binom{l}{t}(2^{\frac{n}{l}}-2)^{l-t}$$

Denote $2^{\frac{n}{l}}-2=a$. Note that $t\binom{l}{t}=\frac{l!}{(l-t)!(t-1)!}=l\binom{l-1}{t-1}$. Now, rewrite:

$$\sum_{t=0}^{t=l}(t+1)\binom{l}{t}a^{l-t}=\sum_{t=0}^{t=l}\binom{l}{t}a^{l-t}+l\sum_{t=1}^{t=l}\binom{l-1}{t-1}a^{l-t-1}=(a+1)^l+l(a+1)^{l-1}$$

(Here we just used that $(x+1)^k=\sum_{t=0}^{k}x^t\binom{k}{t}$)

So, we can solve each case in $O(\log)$ with simple binary exponentiation.

Problem C. Count Hamiltonian Cycles

First, let's get some trivial bounds on the length of the shortest cycle. Take any position i ($1\le i\le 2n-1$). Let the numbers of `W`, `B` in $s[1:i]$ be $left_W, left_B$ correspondingly.

Consider any Hamiltonian cycle. Suppose that it contains x edges connecting node from $\{1,2,\dots,i\}$ to $\{i+1,i+2,\dots,2n\}$. The degree of each node in Hamiltonian cycle is 2, so the number of edges connecting nodes from $\{1,2,\dots,i\}$ is $\frac{2i-x}{2}$. So, x has to be even. Also, note that the number of edges connecting nodes from $\{1,2,\dots,i\}$ can't exceed $2min(left_W,left_B)$. So,

$$\frac{2i-x}{2}\le 2min(left_W,left_B)\implies x\ge 2(left_W+left_B)-4min(left_W,left_B)=2|left_W-left_B|$$

Also, of course, $x\neq 0$, as otherwise graph wouldn't be connected. So, $x\ge 2max(1,|left_W-left_B|)$.

It turns out that these bounds are all achievable, so if we just wanted to learn the length of the shortest, we could just find all these bounds over each position i and sum them up. (Proof that these bounds are achievable will follow from what's written below).

But we are interested in the number of such cycles. For that, let's analyze the structure of cycles in which all of these bounds hold. Again, choose some i and define $left_W, left_B$ as before.

It's easy to see that:

- If $left_W > left_B$, then the graph on $\{1,2,\dots,i\}$ consists of $left_W-left_B$ paths of form $WBWBW\dots BW$.
- If $left_B > left_W$, then the graph on $\{1,2,\dots,i\}$ consists of $left_B-left_W$ paths of form $BWBWB\dots WB$.
- If $left_B=left_W$, then the graph on $\{1,2,\dots,i\}$ consists of a single path $WBWB\dots WB$.

It's easy to see that if this holds for every i, then all bounds on the numbers of edges which connect a node from $\{1,2,\dots,i\}$ to $\{i+1,i+2,\dots,2n\}$ will match exactly.

Now, let dp_i denote the number of ways to connect edges on the first i nodes so that all first i conditions hold. How do we update this dp?

Assume now $left_W > left_B+1$. If the next character is `W`, then we have to add a path from a single node. If the next character is `B`, then we have to reduce the number of paths by 1, by connecting `B` to the ends of some two paths. But here we run into a problem: it matters if the path has a length larger than 1 or not, as in that case, it has two "distinct"ends or not. To solve this issue, let's modify our problem a bit.

Let's find the number of oriented Hamiltonian cycles. Then we could divide this number by 2 and obtain the answer to the initial problem. All the conditions remain the same, just paths become oriented.

Now, making a transition is easy! As now, when we are processing `B` in the case above, we don't have to choose 2 ends; we have to choose a right end and a left end. There are precisely $(left_W - left_B) \cdot (left_W - left_B - 1)$ to do so.

Now updating the dp is easy; you are just dealing with some cases. Briefly:

- $left_W > left_B$. If next character is `W`, $dp_{i+1} = dp_i$. If next character is `B`, then, if $left_W > left_B + 1$, $dp_{i+1} = dp_i(left_W - left_B) \cdot (left_W - left_B - 1)$. If $left_W = left_B + 1$, $dp_{i+1} = 2dp_i$ (there are two ways to prolong the chain).
- $left_B > left_W$ is symmetric.
- $left_W = left_B$. Wlog next character is `W`. Then we have to connect this `W` to the only black end of the previous chain, so $dp_{i+1} = dp_i$.

We can calculate this in $O(n)$.

Problem D. Distance Parities

Let's add an edge between each i, j, the distance between which is odd. Let's show that if there is any graph for which the distances match, this graph also has to work.

For any i, j between which the distance is odd, the parity will match. For any i, j between which the distance is even, say, $2t$, there is a node k on the shortest path between them, for which $d(k, i) = 1$, $d(k, j) = 2t - 1$. Therefore, we drew edges (k, i) and (k, j), and the distance between i, j in our graph is 2, so the parities also match.

So, we should just add all these edges, and check if the graph is connected and all parities match. $O(n^3)$ with Floyd-Warshall.

Problem E. Excellent XOR Problem

Let X be the XOR of all the elements. If $X \neq 0$, we can split into two parts arbitrarily, and their XORs will be different.

If $X = 0$, then let's try to split the array into 3 parts and see when it's impossible. Let a_x be the first nonzero element of the array (if all elements are 0, clearly there is no solution). Let the first subarray be $a[1 : x]$. We want to split the rest into two parts so that XOR in both isn't 0 or a_x (the XORs of those parts can't be equal, as then the XOR of the entire array would be a_x, not 0).

So, if there is no such split, then for any y with $x + 1 \leq y \leq n$, XOR on subarray $a[x + 1 : y]$ is 0 or a_x. However, this means that each element there is 0 or a_x. So, all elements of the array are 0 or a_x. But in this case, XOR of any subarray is always 0 or a_x, so there can't be any such split.

So, the solution is: if $X \neq 0$, split arbitrarily. If $X = 0$, find the first nonzero element a_x, take subarray $a[1 : x]$, and try all ways to split the rest into two parts. If you haven't succeeded, there is no solution.

Problem F. F*** 3-Colorable Graphs

We will show that the answer is always 2 or 3. Let's initially color nodes of the first part (from 1 to n) in color 1 and the remaining in color 2.

It's clear that if we don't add any edges, the graph remains 2-colorable (and therefore 3-colorable).

If we add one edge (u, v), then just color node v in color 3, coloring will remain proper.

Now, suppose that we added two edges, and the graph is no longer 3-colorable. Let's consider some cases. Suppose that one of these two edges connects two nodes from different parts. Then, if the other edge is (u, v), we can just color node v in color 3, and the coloring will remain proper. So, we have to add edges, which connect nodes from the same part.

Let these edges be (u_1, v_1) and (u_2, v_2). If these edges share a node, say, $u_1 = u_2$, then we can just color node u_1 in color 3, and the coloring will be proper. Otherwise, let's try coloring some endpoint of each of these two edges in color 3. If we can't get proper coloring this way, it means that all of the edges $(u_1, u_2), (u_1, v_2), (v_1, u_2), (v_1, v_2)$ are present in this graph. And indeed, if such a cycle of length 4 is present in the graph, we can add edges (u_1, v_1) and (u_2, v_2), obtaining a subgraph on 4 nodes u_1, v_1, u_2, v_2, in which each pair of nodes is connected, and which, therefore, isn't 3-colorable.

Even if there is no 4-cycle, we can always add at most 3 edges to make the graph not 3-colorable. It's enough to show that we can choose 2 nodes u_1, v_1 from the first part, and nodes u_2, v_2 from the second part, such that at least 3 edges among edges $(u_1, u_2), (u_1, v_2), (v_1, u_2), (v_1, v_2)$ are present in this graph (then we can add all remaining edges to obtain a complete graph on 4 nodes). Suppose that it doesn't exist. Then there is no edge (u, v) with u in the first part, v in the second, such that there is at least one more edge from u and at least one more edge from v. Then for any edge (u, v), at least one of u, v is a leaf. Then consider any non-leaf node u, it can be connected only to leaves, so u together with these leaves forms a separate connected component, and the graph is not connected. This contradicts the problem statement (graph is told to be connected).

So, answer is 2 if there is a 4-cycle u_1, v_2, v_1, u_2, and 3 otherwise. How to check if there is a 4-cycle?

Here is an algorithm that, for a given node, checks if it's in some 4-cycle in $O(n)$, giving total runtime of $O(n^2)$. Consider node v, and it's neighbors $u_1, u_2, \ldots, u_k$. v is contained in a 4-cycle iff there is a node $v_1 \neq v$, which is connected to at least two nodes among $u_1, u_2, \ldots, u_k$.

Then, just start going through all neighbors of nodes $u_1, u_2, \ldots, u_k$, marking nodes which we meet (except v). If we have to mark some node twice, we found a 4-cycle. This takes $O(n)$ per node.

Problem G. Graph Problem With Small n

For convenience, we will number the vertices from 0 to $n - 1$.

First, we denote the subset of vertices $i_1, i_2, \ldots, i_k$ as $2^{i_1} + 2^{i_2} + \ldots + 2^{i_k}$. Thus, each subset of vertices of the graph corresponds to some number $mask$ from 0 to $2^n - 1$.

Solution in $O(2^n n^3)$.

Let $dp[mask][u][v]$ denote whether there exists a Hamiltonian path starting at vertex u and ending at vertex v on the vertices in the subset $mask$ of the graph (thus $mask$ must contain the bits u and v). We can start by making $dp[2^v][v][v] = true$ for each vertex v.

We will go through the subsets in increasing order of their size. Note that $dp[mask][u][v] = true$ if and only if $mask$ contains some vertex $w \neq v$ such that:

- w is connected to v
- $dp[mask - 2^v][u][w] = true$

(This vertex w — is the vertex adjacent to v in this potential Hamiltonian path).

So, for a given $mask$ and a given pair (u, v), we can search all the candidates w in $O(n)$, giving $O(2^n n^3)$ in total.

Solution in $O(2^n n^2)$.

Instead of the Boolean variable $dp[mask][u][v]$, let's keep the number $dp_1[mask][u]$. It will contain the bit

v if and only if $dp[mask][u][v] = true$. Also, for each vertex u, let's keep a number $neigh_i$ that contains bit v if and only if there is an edge between vertices u and v.

Again, let's calculate the value of $dp_1[mask][u]$ by iterating over $mask$ in increasing order of subset size. For each vertex v that is in $mask$, let's try to determine whether the bit v is in $dp_1[mask][u]$. Recall that $dp[mask][u][v] = true$ if and only if there is some vertex $w \neq v$ in $mask$ such that: w is connected to v and $dp[mask - 2^v][u][w] = true$. This is equivalent to saying that $dp_1[mask - 2^v][u]$ and $neigh_v$ have at least one bit in common! We can find out if they share a bit in a single `AND` operation. So, for data $mask, u$ we can compute $dp_1[mask][u]$ in $O(n)$, and therefore the total asymptotics is $O(2^n n^2)$.

Solution for $O(2^n n)$.

This is still not enough, we need to do some more optimization.

Let's calculate the value of $dp_1[mask][n-1]$ for all $mask$. We do this in $O(2^n n)$. Note that each Hamiltonian path of the original graph must pass through the vertex $n-1$. From the calculated values of dp_1 it is easy to understand which vertices are reached by the Hamiltonian path from vertex 1. Consider some other two vertices $u, v < n-1$. It is easy to see that there is a Hamiltonian path between vertices u and v if and only if there exist two subsets $mask_u, mask_v$ such that:

- $u \in mask_u, v \in mask_v$
- Every vertex except $n-1$ is in exactly one of these subsets. Vertex $n-1$ is in both subsets.
- $u \in dp_1[mask_u][n-1], v \in dp_1[mask_v][n-1]$

In other words, only if there exists a partition of vertices into 2 groups (with $n-1$ in each group), that there exists a Hamiltonian path from u to $n-1$ in the first group and from $n-1$ to v in the second group.

But how to quickly determine this for each pair of vertices? For each vertex u, let's keep a number ans_u, whose bit v is present if and only if there is a Hamiltonian path from u to v in the graph. Now, let's iterate over all $mask_u$ ($mask_v$ is uniquely defined by $mask_u$, as $2^n - 1 - mask_u + 2^{n-1}$), and for each $mask_u$, let's iterate over all u. For each of these u we make $ans_u = ans_u$ `OR` $dp_1[mask_v][n-1]$.

The overall asymptotics is $O(2^n n)$.

Problem H. Help Me to Get This Published

Preface. Help me to get this published, please. Sorry for the long tutorial. It requires only one small piece of knowledge:

Well-known fact. If Gallai coloring contains at least three different colors, then there is a color that spans a disconnected graph.

Now let's dive into some interesting observations I haven't seen before and couldn't find in the literature. If you find them, sorry :(. If not, help me to get this published.

We will also denote the color of the edge between nodes i, j by $c(i, j)$.

Lemma 1. Suppose that color 1 spans a disconnected graph, let A be one of the connected components of this graph. Then for any nodes $u, v \in A$ and $w \notin A$, $c(u, w) = c(v, w)$.

Proof of Lemma 1. First, consider any edge (u_1, v_1) of color 1 in A. For any node $w \notin A$, consider triangle $u_1 v_1 w$. $c(u_1, v_1) = 1$, but $c(u_1, w) \neq 1, c(v_1, w) \neq 1$, so $c(u_1, w) = c(v_1, w)$.

Now, consider any nodes $u, v \in A$. There is a simple path of color 1 between then, let it be $x_0, x_1, \ldots, x_k$ with $x_0 = u, x_k = v$. Then $c(w, u) = c(w, x_0) = c(w, x_1) = \ldots = c(w, x_k) = c(w, v)$, as desired. □

Theorem 2. For any Gallai coloring, the following inequality holds:

$$\sum_{i=1}^{n} \frac{1}{2^{d(i)}} \geq 1$$

Proof of Theorem 2. We will prove this statement by induction by n. It's trivial for $n \le 3$.

Consider $n \ge 4$. If there are at most 2 different colors, then $\sum_{i=1}^{n} \frac{1}{2^{d(i)}} \ge \frac{n}{4} \ge 1$, as desired. Otherwise, there exists a color that spans a disconnected graph. Wlog, this is color 1.

Consider any connected component of color 1, A. By **Lemma 2.1**, for any node $v \notin A$ there exists some color $c(v)$, such that all edges between v and A have color $c(v)$.

Let k be the number of different colors among $c(v)$ (for $v \notin A$). Consider the graph obtained by compressing A into a single node a with $d(a) = k$. Here, $c(a, v) = c(v)$ for $v \notin A$. By the induction hypothesis, we have:

$$\sum_{v \notin A} \frac{1}{2^{d(v)}} + \frac{1}{2^k} \ge 1$$

Now, consider separately the graph spanned by nodes in A. Let $d_1(v)$ denote the color degree of node v with respect to this graph (where $v \in A$). By the induction hypothesis, we have:

$$\sum_{v \in A} \frac{1}{2^{d_1(v)}} \ge 1$$

Now note that for each $v \in A$, $d(v) \le d_1(v) + k$ (as outside of A v has precisely k different colors). Then we can write:

$$\sum_{i=1}^{n} \frac{1}{2^{d(i)}} = \sum_{v \notin A} \frac{1}{2^{d(v)}} + \sum_{v \in A} \frac{1}{2^{d(v)}} \ge \sum_{v \notin A} \frac{1}{2^{d(v)}} + \sum_{v \in A} \frac{1}{2^{d_1(v)+k}} = \sum_{v \notin A} \frac{1}{2^{d(v)}} + \frac{1}{2^k} \sum_{v \in A} \frac{1}{2^{d_1(v)}} \ge \sum_{v \notin A} \frac{1}{2^{d(v)}} + \frac{1}{2^k} \ge 1$$

This proves **Theorem 2**. □

This is already a necessary condition for our degree sequence. But it's not sufficient.

Lemma 3. Consider Gallai coloring of K_n, in which one of color degrees is $n-1$. Then the color degrees are $n-1, n-1, n-2, n-3, \ldots, 1$ in some order.

Proof of Lemma 3. Without loss of generality, let node n have $d(n) = n-1$. All colors $c(i, n)$ for $1 \le i \le n-1$ are different, wlog $c(i, n) = i$ for $1 \le i \le n-1$.

For any $1 \le i < j \le n-1$, from triangle ijn we get that $c(i, j)$ is i or j. For each i, j, let's draw an oriented edge $i \to j$ if $c(i, j) = i$, and $j \to i$ if $c(i, j) = j$. As there are no rainbow triangles, there are no directed cycles of length 3 in this graph.

Remember that if the tournament graph contains no directed cycle of length 3, it is acyclic. This is a well-known fact, and can be proved by considering the shorted directed cycle $(x_1, x_2, \ldots, x_k, x_1)$. Clearly $k \ge 4$. If edge between x_1, x_3 is oriented $x_3 \to x_1$, then there is a shorter directed cycle (x_1, x_2, x_3, x_1), else there is a shorter directed cycle $(x_3, x_4, \ldots, x_1, x_3)$. So, nodes $1, 2, \ldots, n-1$ can be arranged in a row, so that all edges go from left to right. The color degree of the i-th node in this row will be exactly i. □

Theorem 4. Let $d(1) \le d(2) \le \ldots \le d(n)$ be the color degree sequence of some Gallai coloring of graph K_n. Let's also define $d(0) = 0$. Then, for each $1 \le k \le n$:

$$\sum_{i=k}^{n} \frac{1}{2^{d(i)-d(k-1)}} \ge 1$$

Proof of Theorem 4. For $k = 1$ this result is the same as the result of **Theorem 2**. Let's consider $k \ge 2$.

Consider any $i \ge k$. How many different colors go from i to nodes $1, 2, \ldots, k-1$? Suppose that there are edges from at least x different colors from i to nodes from 1 to $k-1$. Let's choose one edge of each

color, and denote the endpoints of these edges different from i by $v_1, v_2, \ldots, v_x$. Consider graph on nodes $v_1, v_2, \ldots, v_x$ and node i. It contains $x+1$ nodes, and the color degree of node i in it is x, so in this subgraph, at least one more node has to have color degree x, by the lemma above. But then $x \le d(k-1)$.

Now, consider subgraph on nodes $k, k+1, \ldots, n$. For any $i \ge k$, there are at most $d(k-1)$ colors going from i to nodes $1, 2, \ldots, k-1$, so the color degree of node i in this new graph is at least $d(i) - d(k-1)$. Then we can just apply **Theorem 2** to this graph. □

It turns out that this condition is actually sufficient. This part is not very interesting, I will skip it, but the idea is that you can get construct Gallai colorings with such degree sequences as follows: take Gallai coloring of K_{n-1}, take any node i, clone it, and add an edge between i and its clone. This way, we can make $d(i) \to d(i), d(i)$, or $d(i) \to d(i)+1, d(i)+1$. These operations allow to achieve all valid degree sequences.

Now, we can use these criteria to actually solve the problem. We will make a dp, going from degrees $n-1$ to 1. When at number k, we will maintain the number of ways to use cnt numbers $\ge k$, with the current value of $\lfloor \sum 2^{k-x} \rfloor$ being equal to sum (where x ranges over all cnt numbers $\ge k$). This gives a simple $O(n^4)$ dp solution.

Problem I. Increasing Grid

Let's subtract from $a_{i,j}$ the number $i+j-1$. It is easy to see that now all rows and columns must be non-decreasing. Also, $0 \le a_{1,1}$ and $a_{n,m} \le 1$. So, all numbers must be between 0 and 1. This is also a sufficient condition: if $0 \le a_{i,j} \le 1$, then in the initial table this number was $i+j-1$ or $i+j$, that is, it was in the range from 1 to $n+m$.

So, we are interested in tables of ones and zeros that do not descend through rows and columns, some elements of which are given to us (if $a_{i,j} - (i+j-1) \notin \{0, 1\}$ before subtraction, then such tables do not exist, and we immediately print 0).

For each one, all the numbers in the rectangle to the right and down from it must also be ones. For each zero, all numbers in the rectangle to the left and up from it must also be zeros. So, if some zero is to the right and down from some one, the answer is 0.

We can do the filling above in the following way: for each row from 1 to n, for each column from 1 to m, if $a_{i,j} = 1$, then make $a_{i+1,j}$ and $a_{i.j+1}$ also equal to 1 (if these cells exist). If at some point we have to make zero equal to one, then there are no solutions. Then we will do the same thing with zeros.

Now we need to figure out how to count the number of ways to assign other values.

Note that the tables of ones and zeros that do not descend along the rows and columns have the following form: there is a path from the lower left corner to the upper right corner that goes only up and to the right along the sides of the grid, such that on one side of this path there are only zeros, and on the other side only ones. Let's calculate the number of such paths if we already know some values.

Consider the nodes of our table. They are formed by the intersection of $n+1$ horizontal and $m+1$ vertical lines. Let's denote by $dp_{i,j}$ the number of ways to draw a path that goes only up and to the right, from the lower left corner (i.e., the intersection of the nth horizontal and 0th vertical) to the node at the intersection of the ith horizontal and jth vertical, so that all the numbers to the left of this path are zeros, and all the numbers to the right are ones. We start with $dp_{n,0} = 1$, and we need to find $dp_{0,m}$.

This dp is quite easy to calculate: we need to add to $dp_{i,j}$ some of the numbers $dp_{i+1,j}$ and $dp_{i,j-1}$, depending on whether the path satisfies all conditions.

The asymptotics is $O(n^2)$.

Problem J. Jewel of Data Structure Problems

First, let's see how to determine the beauty of a given permutation.

If the whole permutation is odd, then its beauty is n. Suppose that the permutation is even.

For i, we denote by c_i the number of inversions that include p_i, that is, the number of indices j for which $j < i$ and $p_j > p_i$, or $j > i$ and $p_j < p_i$. If any of the numbers c_i is odd, then we can consider a subsequence of all elements except the i-th, and it will be odd. In this case the beauty will be $n - 1$.

Otherwise, all c_i are also even. Suppose that there is at least one inversion in the permutation, say, $i < j$ and $p_i > p_j$. Then consider the subsequence containing all elements except i and j. It is odd (because we have removed all inversions containing p_i or p_j, but the inversion formed by p_i and p_j we have to remove only once). So, in this case the beauty is $n - 2$. If there are no inversions, then the permutation is $(1, 2, \ldots, n)$. All its subsequences are sorted, so its beauty is -1.

It remains to understand how to find this beauty quickly. Note the following: c_i is even if and only if $p_i \bmod 2 = i \bmod 2$. Let us show this. Let x — the number of indices j from 1 to $i - 1$ for which $p_j < p_i$. Then to the left of p_i $i - 1 - x$ elements form an inversion of a_i. On the right $p_i - 1 - x$ elements form an inversion of a_i. Thus, in total a_i lies in $i - 1 - x + p_i - 1 - x = (i + p_i) - 2(x + 1)$ inversions. So c_i is even if and only if $i + p_i$ is even, i.e. only if i and p_i have the same parity.

Let us calculate the parity of p from the beginning. It is easy to see that the parity of p is equal to the parity of $n - cycles$, where $cycles$ — the number of cycles in the permutation p. Indeed, swapping two elements changes the parity of the permutation (a very well-known fact), and we can sort our permutation by exactly $n - cycles$ of swapping two elements.

Also, we keep cnt_eq, cnt_same_par — the number of indices for which $p_i = i$, and the number of indices for which $p_i \bmod 2 = i \bmod 2$. It is clear how to update these values after each exchange.

Now, we answer each query as follows: if the permutation is odd, the beauty is n. Otherwise, if $cnt\_same\_par \neq n$, the beauty is $n - 1$. Otherwise, if $cnt\_eq \neq n$, then the beauty is $n - 2$, otherwise, the permutation is sorted and the beauty is -1.

The asymptotics is $O(n + q)$.

Problem K. King of Swapping

Let's look at the position where the number n is located. We must be able to move n to any position. At the same time, we can move n from position u to position v (in one operation) if and only if $p_u > p_v$.

Thus, if we can move n from any vertex to any vertex, then in an oriented graph whose edges are (u_i, v_i), we can reach any vertex from any vertex (such graphs are also called **strongly connected**).

It turns out that this is a sufficient condition. Before proving this, let us tell you how to determine that a graph is strongly connected. It is enough to check that it is possible to get from vertex 1 to any other vertex and from any other vertex to vertex 1. Each of these checks is done by one dfs, and therefore the asymptotics is $O(n + m)$.

Now we prove that this is a sufficient condition.

Consider any oriented cycle in our graph, i.e. the sequence $v_1, v_2, \ldots, v_k$ such that we have operations $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k), (v_k, v_1)$. We will show that in this loop, we can rearrange the elements as we like. Without loss of generality, the numbers in positions $v_1, v_2, \ldots, v_k$ are $1, 2, \ldots, k$. Let us show that we can get any permutation of the numbers from 1 to k.

First, we show that from any situation, we can get a permutation where $p_{v_1} = 1, p_{v_2} = 2, \ldots, p_{v_k} = k$. Indeed, first we move k around until it is in v_k, then $k - 1, \ldots$, then 2 and 1.

Now we show that from the permutation $(1, 2, \ldots, k)$ we can get some cyclic shift of the permutation $(k, k - 1, \ldots, 1)$. We show this by induction, for $k = 1, 2$ the statement is obvious. Transition: first, we exchange k with the next number in the cycle $k - 2$ times, obtaining the permutation $(2, 3, \ldots, k - 2, k, k - 1, 1)$. Now "combine into one element" k and $k - 1$ (we can exchange them together: $(k, k - 1, x) \to (k, x, k - 1) \to (x, k, k - 1)$). We have an assertion for $k - 1$.

Now we show that from the shift $(k, k - 1, \ldots, 1)$ we can get any other permutation. Let us reverse all

operations: if the original operation was (u_i, v_i), we will consider the operation (v_i, u_i). Suffice it to show that by applying these operations we can get from any permutation to a cyclic shift $(k, k-1, \ldots, 1)$, but we do it in exactly the same way: first we put k in place, then $k-1, \ldots, 2, 1$.

This completes the proof of the statement for the loop. Then it is easy to show that we can get any permutation if the graph is strongly connected. For example, consider any two positions u, v $(u \neq v)$. There is a simple cycle with positions u and v. In this cycle we can rearrange the elements in any way, so we can simply swap p_u, p_v. So, we can swap any two elements, so we can get any permutation from any permutation.

Problem L. Least Annoying Constructive Problem

Odd n. Let $n = 2k + 1$.

We will imagine these nodes on a circle, numbered from 1 to n, and use the cyclic notation of nodes; that is, node $n + i$ is the same as node i for any i.

Consider the following construction. For each i from 1 to n, output the following k edges: $(i, i+1), (i-1, i+2), (i-2, i+3), \ldots, (i-k+1, i+k)$. On the circle, view these as k edges parallel to edge $(i, i+1)$. Let's call such k edges for a given i block i.

Let's show that this works. Consider any $n - 1 = 2k$ consecutive edges. Such $2k$ edges completely contain some block, wlog block 2, and contain some suffix of block 1, let's say of length x, and some prefix of block 3 of length $k - x$.

After we drew block 2, we have $k + 1$ components remaining: k edges and a single node $3 + k$. The idea is that the i-th edge of block 1 connects i-th and $i + 1$-st of these components, the same for i-th edge of block 3. Then it's easy to see that the suffix of block 1 of length x connects the last $x + 1$ components into a single component, and the prefix of block 3 of length $k - x$ connects this resulting component and the first $k - x$ components.

Even n. Let $n = 2k + 2$. Let's once again draw $2k + 1$ nodes on a circle, and let the center of this circle be point $2k + 2$. We will once again number the points on the circle cyclically mod $2k + 1$ (note that this doesn't apply to the center node $2k + 2$).

Then, let the "block i"consist of the following k edges for each i from 1 to $2k + 1$: $(i, i+1), (i-1, i+2), (i-2, i+3), \ldots, (i-k+1, i+k)$, **and** the edge $(i + k + 1, 2k + 2)$. You can visualize this as: draw k edges parallel to $(i, i+1)$, and the edge from the center to the last remaining point. Then, write down edges of these blocks for each i from 1 to $2k + 1$, in this order.

The proof that this works is the same as in the odd case.

Problem M. Most Annoying Constructive Problem

First of all, forget about subarrays of length 1 whatsoever. By word subarray I only refer to subarrays of length ≥ 2.

Let $f(n) = \frac{n(n-1)}{2} - \lfloor \frac{n-1}{2} \rfloor$. I claim that for all $n \geq 4$, such a permutation exists for all $0 \leq k \leq f(n)$, and doesn't exist for larger k. $n \leq 3$ I am sure you can do yourself.

Lemma. For $n \geq 4$, we must have at least $\lfloor \frac{n-1}{2} \rfloor$ even subarrays.

Proof. We will show this by induction. For $n = 4, 5$ we can verify this with bruteforce.

Now, consider $n \geq 6$. Suppose that some permutation $[p_1, p_2, \ldots, p_n]$ of length n contains at most $\lfloor \frac{n-1}{2} \rfloor - 1$ even subarrays. By induction, however, we know that $p[3 : n]$ has to contain at least this many even subarrays. So, p contains exactly this many even subarrays, and all of them are in $p[3 : n]$.

So, first, $p_1 > p_2 > p_3$ (as subarrays $p[1 : 2], p[2 : 3]$ have to be odd), and also for any $i \geq 3$, both subarrays $p[1 : i], p[2 : i]$ have to be odd. So, for any $i \geq 3$, the number of elements in $p[2 : i]$ that are smaller than p_1 has to be even. As $p_2 < p_1$, $p_3 < p_1$, we get $p_4 > p_1, p_5 > p_1, \ldots, p_n > p_1$. So, $p_1 = 3, p_2 = 2, p_3 = 1$. But then consider an array $p[2 : i]$ for any $i \geq 4$. It's odd, but 2 forms only one inversion: with 1, so the

array $p[3:i]$ turns out to be even. So, arrays $p[3:i]$ are even for all $i \geq 4$, and $n-3 \leq \lfloor \frac{n-1}{2} \rfloor - 1$, which leads to contradiction for $n \geq 6$. $\square$.

Now, let's look at how we might construct such permutations. Once again, for $n \leq 5$ we construct everything simply by bruteforce. Now, consider $n \geq 6$, and suppose that we already know how to construct the permutations for all admissible pairs (n_1, k) with $n_1 < n$. Let's consider some cases.

- $k = 0$. Then permutation $[1, 2, \ldots, n]$ works.
- $1 \leq k \leq n-2$. Then permutation $[1, 2, \ldots, k-2, k-1, k+2, k, k+1, k+3, k+4, \ldots, n]$ works.
- $n-1 \leq k \leq f(n-2)+n-1$. Then consider a permutation of length $n-2$ with $k-(n-1)$ odd subarrays, and append to it $n, n-1$. Note that in such permutation p subarray $p[n-1:n]$ is odd, and for any $1 \leq i \leq n-2$, subarrays $p[i:n-1], p[i:n]$ have different parities, so we added precisely $n-1$ odd subarrays.
- $k = f(n)$. Consider the following infinite sequence a:

  $$4, 1, 6, 3, 8, 5, 10, 7, 12, 9, 14, 11, 16, \ldots$$

  Here at odd positions, we have even integers, starting from 4, and on even positions, we have odd integers, starting from 1. Yes, 2 is missing, but it's not a problem.

  The key is that the only even subarrays in this sequence are of the form $a[2i:2i+1]$. The proof is left to the reader as an exercise.

  So, we can take first n elements of this sequence, and "compress"them to a permutation, getting a permutation with only $\lfloor \frac{n-1}{2} \rfloor$ even subarrays.
- $f(n-2)+n \leq k < f(n)$. It turns out that the following construction works. Choose p_1 and p_n somehow, and arrange the remaining elements in $p[2:n-1]$ according to the permutation for $solve(n-2, f(k-2))$. Proof of this fact is left to the reader.

  If you believe this, then you can try all n^2 pairs, until you find the one that works. You can check one pair in $O(1)$, after you precomputed the number of odd subarrays that particular first/last elements add, even in $O(n^2)$.

Total complexity: $O(n^2)$. If you know how to write checker better than in $O(n^2)$, tell me.

Problem N. No Zero-Sum Subsegment

First, calculate the sum $S = 2D + C - B - 2A$ of all numbers in the array. If $S = 0$, clearly, there is no solution. If $S < 0$, let's multiply everything by -1 (just swap A with D and B with C), the answer won't change. Now we suppose that $S > 0$.

We will model the situation as a walk on the integer line. We start at point 0, we have to make A jumps to the left by 2, B jumps to the left by 1, C jumps to the right by 1, D jumps to the right by 2, and we should never visit the same point twice. Our route, clearly, will end at point S.

Suppose we jump over segment $[L, L+1]$ at least 3 times. We can jump over this segment only in 3 ways: $L \longleftrightarrow L+2$, $L \longleftrightarrow L+1$, $L-1 \longleftrightarrow L+1$. So, we must use all 3, and our journey has to be $L-1 \to L+1 \to L \to L+2$, or $L+2 \to L \to L+1 \to L-1$.

Consider the first time in our route when we get to the point ≥ 1. It's not hard to show that we can't get to any point < 0 after this: this would mean that we jumped over segment $[0, 1]$ three times, but by the observation above, it's impossible, as we start from 0. So, if we are ever going to the left of 0, we are doing this at the very beginning, then getting back to point 1, and staying in the > 0 area. Similarly, if we ever go to the $> S$ area, it's in the very end.

Now let's consider the ways in which we can go to this < 0 area and get back to 1. It turns out that there are only 3 ways to do so:

- **Way 1.** Jumping to the left by 1, and then to the right by 2.
- **Way 2.** Jumping to the left k times by 2, then jumping to the left 1 time by 1, then jumping to the right $k+1$ times by 2.
- **Way 3.** Jumping to the left k times by 2, then jumping to the right 1 time by 1, then jumping to the right k times by 2.

The ways to go to the $> S$ area by a jump from $S-1$ and get back to S are symmetric.

Now, we have 4 ways to "start"our journey: 3 ways to go into the < 0 area, and the remaining way is to avoid going into the < 0 area. Similarly, we have 4 ways to "end"the journey. Now, let's fix this "start"and the "end"of the journey and look at the ways to get from point point X to point Y with $X \le Y$, with A -2s, B -1s, C 1s, D 2s, if we aren't allowed to go to $< X$ and $> Y$ areas.

First, we can't jump to the left by 2. Indeed, suppose we jumped $L+2 \to L$. Then we have to jump over $[L, L+1]$ at least 3 times, so our route has to contain $L+2 \to L \to L+1 \to L-1$. If it contains $L+1 \to L-1$, therefore, it has to contain $[L, L-2], [L-1, L-3]$, and so on, impossible.

Second, if we ever jump to the left by 1, like $L+1 \to L$, it's within a subroute $L-1 \to L+1 \to L \to L+2$.

So, we must have $A = 0$, $D \ge 2B$, and in the end, we are just arranging C jumps of length 1, B jumps of length 3, and $D-2B$ jumps of length 2 however we want, which is easy to count.

Now, let's iterate over all possible ways to start and end the journey (4^2 in total), and sum up the numbers of journeys over all these ways. For a fixed way, we know that we have to use all the -2s in the "start"+ "end". Note that with this information, we can learn the exact number of each jump we will use in "start"+ "end"together. There are, however, some options.

Let's look at an example. Let's say that we have 4 -2s initially, and we are choosing **Way 2** for the "start"and **Way 3** for the "end". Then in the "start"we use k_1 -2s, 1 -1, k_1+1 2s, and in the "end"we use k_2 2s, 1 1, k_2 2s. So, we use 4 -2s, 1 -1, 1 1, 5 2s in total, and the number of different ways is equal to the number of solutions of $k_1 + k_2 = 4$ in positive integers (that is, 3). As we know exactly what jumps we will use for "start"+"end we can find the number of ways to arrange the remaining jumps, as discussed in the paragraph above. After that, we just multiply these numbers of ways.

After we precalculated all factorials and inverse factorials up to $3 \cdot 10^6$, the complexity is $O(1)$ per test case.