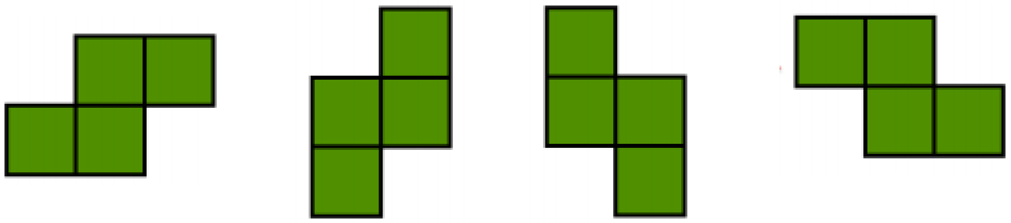
Editorial of Kent Nikaido Contest 1 (Moscow Pre-Finals ACM ICPC Workshop 2016, Round 3)

Filipp Rukhovich

29.03.2016

1 Problem A. Tetris Puzzle

Let's reformulate the statement. In this problem, given an $N \times N$ grid, some cells of them are covered by S-mino tiles of four types:

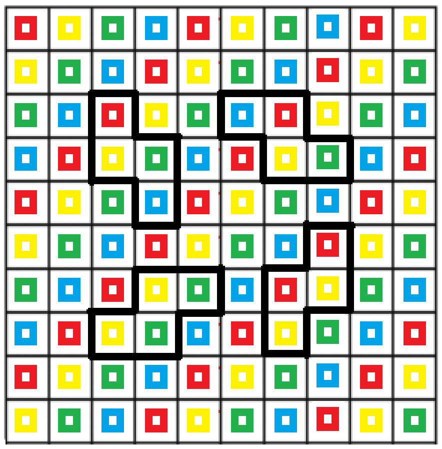

(1) (2) (3) (4)

It is only known which cells are covered by one of the tiles; the task is to compute the parity of number of tiles which are of 3rd or 4th type.

Let's colorize the grid in 4 color numbered 0, 1, 2 and 3 so that cell (i, j) has color $(i+j) \bmod 4$:

It can be seen from the following picture that for any fixed color C:

- every tile of type 1 or 2 covers 0 or 2 cells of C;
- every tile of type 3 or 4 covers exactly 1 cell of C:



It brings us to simple $O(N^2)$-solution: let's just calculate the parity of number of covered cells with some fixed color (for example, 0), and it will be the answer!

Time complexity of described solution is $O(N^2)$.

2 Problem B. Shift and Paint

There are two big cases in this problem:

2.1 Case 1: N > L

1. It's obvious that right cycle shift (described in statement) and left cycle shift (the inverse of right) are possible to perform;

2. Let's assume that $L \geq 4$, and let $a?bcd$ be subsegment of length $L+1$ of the row, where a, b, c, d are balls, and ? is subsegment of length $L-3$ of the row. Then, the cycle shift of subsegment bcd can be easily made by following sequence of operations:

   $a?bcd \to ca?bd \to cda?b \to ca?bd \to bca?d \to ba?dc \to a?dbc$ (1);

   It means that **any subsegment of length 3 can be cyclically-shifted** using the scheme (b, c, d are balls, E, F, G, H - some subsegments of the row so that lengths of $FbcdE$ and $GHbcd$ are equal to N):

   $FbcdE \to$ ... the sequence of shifts ...$GHbcd \to GHdbc \to$ ... the same, but inversed sequence, all shifts are also inversed ... $\to FdbcE$ (2);

3. Let's assume again that $L \leq 4$. Then $a?bcd$ (a, b, c, d-balls, ? - subsegment of length $L-3$) can be performed into $a?cdb$ and then into $ab?cd$. Including the scheme similar to (2), it can be easily proved that **cycle shift of subsegment of length $L-2$ is possible operation** ;

4. If L is even, then we can swap any two neighbouring elements; it means that two colorings are equal iff numbers of balls colored by color 1, 2, ..., K are equal for both coloring; so, the answer is number of ways to put N unique balls to K different boxes, which is C_{N+K-1}^{K-1};

5. If L is odd and there two balls of same color, we are also able to swap any two neighbouring balls (the schemes are $?a?a?bc? \to aabc????? \to acab???? \to aacb \to ?a?a?cb?$, $?a?ba? \to aba??? \to aab??? \to ?a?ab?$ or similar); so, two coloring with equal numbers of balls of each color are equal if there are two balls of equal colors;

6. If L is odd and all colors in the row are unique, so some permutation can be transformed only in permutation of equal parity; the schemes similar to previous two gives the fact that transformation to any such permutation is possible ($?a?b? \to ab???, ab?cd? \to abcd?? \to badc \to ba?dc?$); it means that set of colorings of balls in N fixed different colors is divided by 2 classes of equivalence.

7. From previous two items implies that if L is odd then answer is $C_{N+K-1}^{K-1} + C_K^N$.

Time complexity of the solution is $O(N \ log \ MOD)$, in this task $MOD = 10^9 + 7$.

2.2 Case 2: N = L

In this case, the task is classical problem about k-ary necklace which can be solved using Burnside's lemma; the answer is $\frac{1}{N}\sum_{q|N} K^{gcd(q,N)}$.

Time complexity of the solution $O(N)$ can be achieved precalculating all degress of K.

3 Problem C. Pianist

Assume that some solution exist; so there are non-negative integers $x_1, y_1, x_2, y_2, ..., x_7, y_7$, so that in the solution there are x_1 movings from A to B, y_1 movings from B to A, x_2 movings from B to C, y_2 movings from C to B, ..., x_7 movings from G to A and y_7 movings from A to G.

Let's build the directed graph $G(x_1, y_1, ..., x_7, y_7)$ in the following way:

- vertices of the graph are keys A, B, C, D, E, F, G;
- there are exactly x_1 edges from A to B, y_1 edges from B to A, ..., y_7 edges from A to G.

It can be proved using the theory of eulerian graphs that the solution exists iff there are such $x_1, y_1, ..., x_7, y_7)$ so that:

1. all edges can be achieved from A and vice versa; in other words, all edges and vertex A are lying in some component of strong connection;
2. there are exactly x_1 edges from A to B, y_1 edges from B to A, ..., y_7 edges from A to G;
3. $x_1 + y_2 = C_2, x_2 + y_3 = C_3, ..., x_7 + y_1 = C_1 - 1$;
4. $x_2 + y_1 = C_2, x_3 + y_2 = C_3, ..., x_7 + y_1 = C_1 - 1$;

Last three constraints brings us to a linear system:

$$\begin{pmatrix}
1&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&1&1&0\\
0&1&0&0&0&0&0&0&0&0&0&0&1&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{pmatrix} \times \begin{pmatrix} x_1\\ y_1\\ x_2\\ y_2\\ x_3\\ y_3\\ x_4\\ y_4\\ x_5\\ y_5\\ x_6\\ y_6\\ x_7\\ y_7 \end{pmatrix} = \begin{pmatrix} C_2\\ C_2\\ C_3\\ C_3\\ C_4\\ C_4\\ C_5\\ C_5\\ C_6\\ C_6\\ C_7\\ C_7\\ C_1 - 1\\ C_1 - 1 \end{pmatrix}$$

Let's enumerate our variables and C-values as $z_1, ..., z_{14}$ and $D_1, ..., D_{14}$ in such way that the system would seem like follows:

$$\begin{pmatrix}
1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&0&0&0&0&0\\
&&&&&&\ddots&&&&&&&\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
1&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{pmatrix} \times \begin{pmatrix} z_1\\ z_2\\ z_3\\ \vdots\\ z_{13}\\ z_{14} \end{pmatrix} = \begin{pmatrix} D_1\\ D_2\\ D_3\\ \vdots\\ D_{13}\\ D_{14} \end{pmatrix}$$

If $D_1 - D_2 + D_3 + ... - D_{14} \neq 0$, then the solution does not exist. Otherwise, it can be found that $z_{13} = A_{13} - z_{14}, z_{12} = A12 + z_{14}, ..., z_1 = A_1 - z_{14}$, where $A_{13} = D_{13}, A_{12} = D13 - A13, A_{11} = D12 - A_{12}$ etc. We can find some segment $[L..R]$, so that z_{14} satisfies constraints 3 and 4 iff $z_{14} \in [L..R]$. If this segment is degenerate, then the solution does not exist; if $L + 1 < R$ then the solution exist because $z_{14} = L + 1$ makes all $x - s$ and $y - s$ to be positive, and the graph is strongly-connected. The remaining case is $L \leq R \leq L + 1$; in this case, we can simply build the graph for $z_{14} = L$ or $z_{14} = R$ and check the constraint 1 to be satisfied.

The time complexity of the solution is $O(1)$:)

4 Problem D. Driving

It's obvious that optimal path contains each edge only one or two times. So, our task is to duplicate some edges so that degree of each vertex would become even; theory of eulerian graphs gives us the fact that this constraint is necessary and sufficient.

Suppose that no edges were duplicated; then our first step is to calculate parity of degree of vertex:

$par(v) := degree(v)\ mod\ 2$ for every vertex v.

Then, build minimal spanning tree of the graph, and let B be set of edges of that tree.

Proposition 4.1 . *In optimal path, all duplicated edges lie in B.*

Proof. Consider the edge e which doesn't lie in B; then, there is a cycle contains only e and edges $e_1, e_2, ..., e_k$ from B. Let weight of e to be 2^a, and weights of other edges in cycle to be $2^{a_1}, 2^{a_2}, ..., 2^{a_k}$. It is known from the theory of minimal spanning trees that $a \geq max(a_1, a_2, ..., a_k)$; all weights in given graph are pairwise-different, so 1) $a > max(a_1, a_2, ..., a_k)$ 2) $2^a > 2^0 + 2^1 + ... + 2^{a-1} \geq 2^{a_1} + 2^{a_2} + ... + 2^{a_k}$. Last fact means that it is strictly more optimal to duplicate edges $e_1, e_2, ..., e_k$ from B then edge e, with the same influence on parities of vertexes.

But what edges from B should be duplicate? Let v_1 to be some leaf vertex of B; there is a unique edge e_v in B incident to v; current parity of v uniquely informs us whether should we duplicate e_v or not. After duplicating (or non-duplicating), erase v and e_v from B and repeat the operation. This algorithm: 1) shows that which edges from B should be duplicating can be determined uniquely; 2) can be performed in $O(n + m)$ using queue or depth-first search (don't forget precalculate degrees of 2 modulo $10^9 + 7$!).

5 Problem E. Coins

Let Niwango be the first player, or the player number 1, and Nikomoba - player number 2. Also we say that Niwango(Nikomoba) gets the i-th coin if after the game, i-th coin is heads-up(tails-up).

If players both play optimally, then they get coins according to the following rules:

1. Let $S \subset 1, 2, ..., N be the set of positions in the row, so that \forall pos \in S \forall i \in (pos, N] : p_{pos} < p_i$; let S contain numbers $pos_1 < pos_2 < ... < pos_k = N$;

2. If $pos \notin S$ then the player number $2 - (pos\ mod\ 2)$ gets this coin;

3. First player gets coin number pos_1;

4. For $i = 2, 3, ..., k$, player number $2 - (pos_{i-1}\ mod\ 2)$ gets coin number pos_i.

For example, if coins have values $5, 2, 6, 3, 7, 8, 7, 9$, then $S = 2, 4, 7, 8$, first player gets 1st, 2nd, 3rd, 5th and 8th coins.

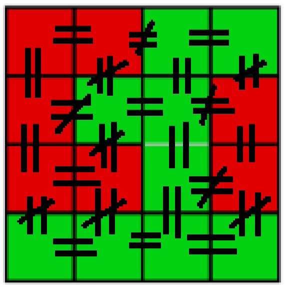
A proof of this non-trivial fact is left to the reader. Having this fact, all actions we should do are:

1. Storing S in, for example, std::map<Position, value> (only positions from S are available in the map) ;
2. Storing and maintaining an array $whoGets[1..N]$, where $whoGets[i]$ is number of player who gets this coin;
3. Storing and maintaining a variable $answer$, where $answer = \sum_{i=1}^{N} P_i * (whoGets[i] == 1)$.

All these structures, variables and invariant can be easily maintained after each operation of decreasing of p_i in total working time $O(QlogN)$ (just delete all the elements which are not in S after decreasing, then insert the element in map if necessary; invariants are maintained trivially. The estimation can be proved using the method of potentials (with $\Phi = |S| * log(N)$).

6 Problem F. Forbidden Puzzle

In this problem, the key observation follows. Consider a correct solution of the puzzle, and connect all pairs of neighbouring cells by signs of equality or inequality (of colors of these cells):



The idea is that in each 2×2-square, number of inequalites is exactly 2!

It brings us to the solution. We will reduce the problem to finding a maximum flow. The network will be building as follows:

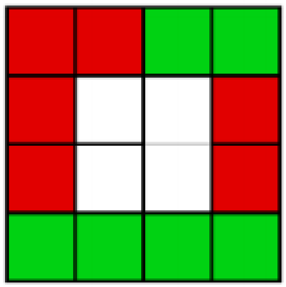
1. There are $(n-1)*(n-1)+2$ vertexes numbered $0, 1, 2, ..., (n-1)*(n-1)+1$ in the network;
2. Vertex $s = (n-1)*(n-1)$ is the source, $t = (n-1)*(n-1)+1$ is be the sink;
3. Vertex number $v = i*(n-1)+j$ is associated with 2×2 square of cells $(i, j), (i+1, j), (i+1, j+1), (i, j+1)$;
4. Vertex $v = i*(n-1)+j$ is called even if $(i+j)\ mod\ 2 = 0$; otherwise, this vertex is odd. Thereby, vertexes in the network are odd, even, source or sink;
5. Source is connected with all even vertices by edges; capacity of each edge is 2;
6. All odd vertices are connected with sink by edges; capacity of each edge is 2;

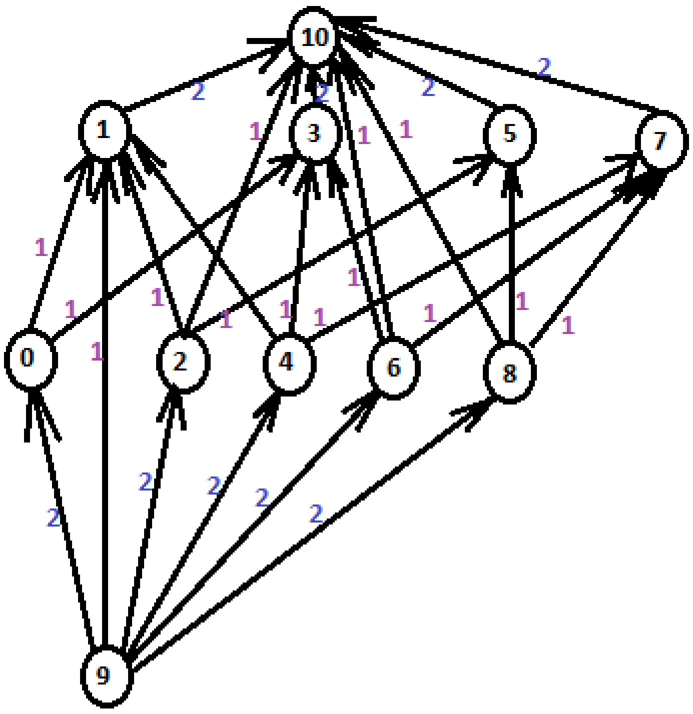
7. Even vertex $v1$ connected with odd vertex $v2$ by edge iff associated squares of $v1$ and $v2$ intersect in exactly two neighbouring squares which are not separated by gray line; capacity of each edge is 1;

8. Even vertex has an additional edge, capacity is 1, to the sink iff the associated square of this vertex is "on bound"of big $N \times N$ square, and two on-bound neighbouring (and painted in input data) cells in this square are different; if there are two such pairs in one 2×2-square, so there are two such edges;

9. Odd vertex has an additional edge, capacity is 1, from the source iff the same constraints as in previous item are satisfied.

On the following picture you can find an example:





After building of the network, calculate maximal flow. If this flow saturates all edges from the source and to the sink so the answer is "YES"; otherwise the answer is "NO".

7 Problem G. TRAX

In this problem, we are to place tiles in cells of the grid so there are three constraints are satified:

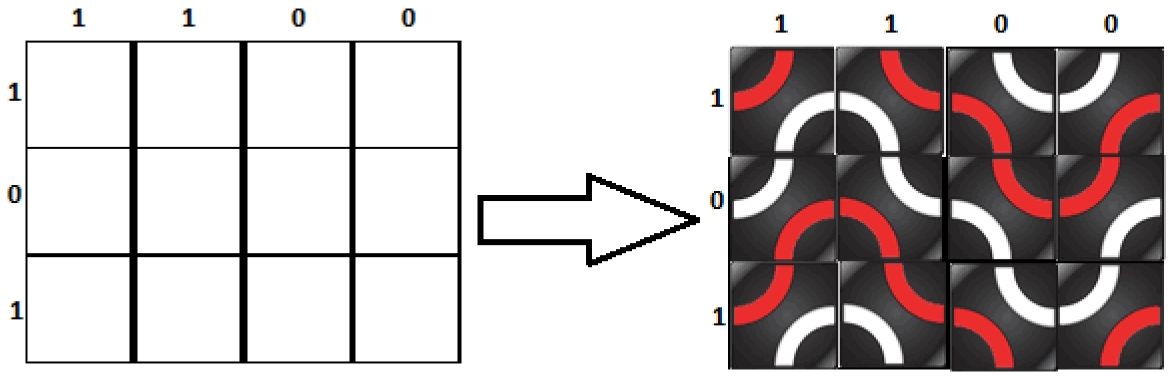
1. A red arc must not touch a white arc;

2. The grid must not contain cycles (see the examples below);

3. For each $i(1 \leq i \leq N)$, in the cell that is R_i-th from the top and C_i-th from the left, the orientation of the tile must be D_i;

Let's enumerate red color to be 1 and white to be 0.

One of the most important observation is that in given tile, any two opposite ends have different colors. It implies the fact that every picture satisfying the first condition can be built in a unique way from the colors of arcs touching upper and left boundaries of table; let it be functions $colory[y]$ and $colorx[x], 1 \leq x \leq H, 1 \leq y \leq W$ (for example, on the following picture $colorx = \{1, 0, 1\}, colory = 1, 1, 0, 0$):

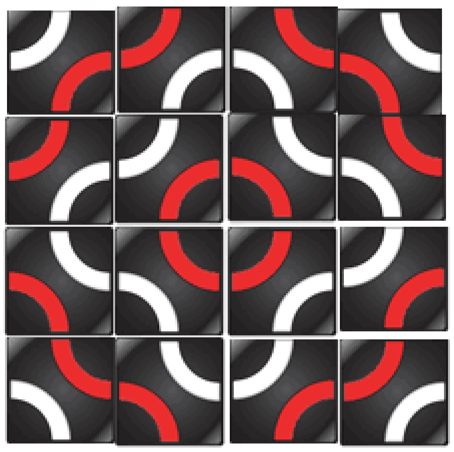


It means that the number of placement satisfying the first and third conditions are 2^{A+B}, where A and B are numbers of rows and columns which are free of (R_i, C_i) for any $i \in [1, N]$. But how to satisfy the second requirement?

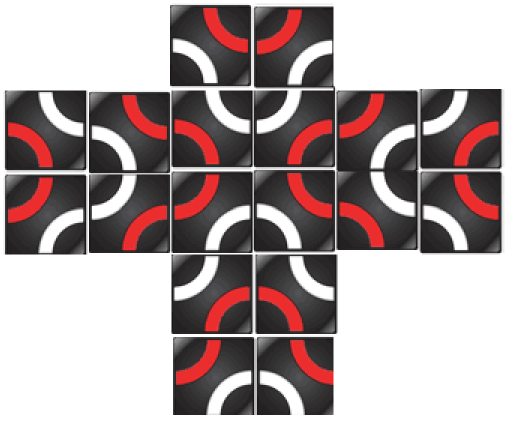
For further analysis, define a function $parx(x) := (x + colorx[x])\ mod\ 2, pary(y) := (y + colory[y])\ mod\ 2$. It can be observed that if $parx[x] = parx[x+1] \forall x \in [1, R)$ then all lines will move "from left to right"(as on the previous picture), without a possibility to build a cycle; so, the condition of "$parx$ equality"is sufficient, so do "$pary$ equality and it's easy to compute the number of placements with both "equalities"(and subtract placements where both "equalities"exist) in $O(R + C)$.

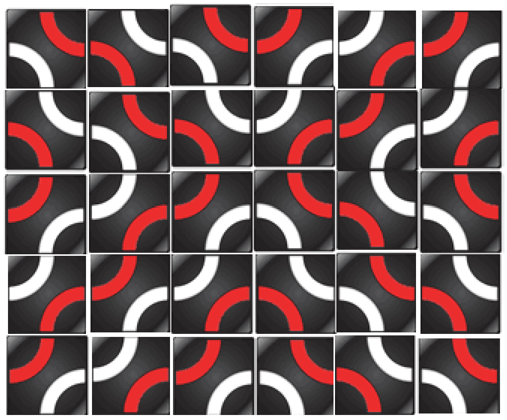
The remaining part are the placements in which both $parx$ and $pary$ has some inequalities. Let x_0, y_0 are the numbers so that $parx[x] \neq parx[x+1], pary[y] \neq pary[y+1]$. There are two cases:

1. $parx[x] = pary[y]$. In this case, a cycle occurs in cells $(x, y), (x+1, y), (x+1, y+1), (x, y+1)$. in the following picture, $parx[1] = pary[1] = 1 - parx[2] = 1 - pary[2] = 0$:



2. $parx[x] \neq pary[y]$. In this case there are a unique way to avoid cycles. To understand it better, consider some fixed situation when $parx[1] = 1 - parx[2] = 0, pary[2] = 1 - parx[3] = 1$:





The unique way to avoid cycles is pictured. The fact is that $parx = px, px, px, ..., px, 1 - px, 1 - px, ..., 1 -$ $py, py, py, ..., py, 1 - py, 1 - py, ..., 1 - py$. where $px = parx[x], py = pary[y]$; this fact gives an (harder that in previous parts, but nevertheless) easy way to calculate the number of such situations in the same $O(R + C)$.

So, the whole problem can be solved in $O(R + C + N)$; the details of implementation are left to the reader.

8 H. Pyramid Decoration

Let black color to be color 1, white - 0. Then, if $N = 0$ then the answer would be $1 - L \ mod\ 2$.

So, assume some configuration S of the bottommost layer; let $ans(S)$ be the color of the topmost stone. Then, change color of stone with coordinates (L, j_0, k_0); let new configuration be S_{new}. It can be observed from the properties of coloring algorithm, ans_{new} is equal to $(ans_{new} + f(1, 1, 1, L, j_0, k_0)\ mod\ 2$, where $f(i, j, k, j_0, k_0)$ can be calculated in following (very slow) way:

```
long long long int f(int i, int j, int k, int L, int j0, int k0) {
if (i == L)
    return (j == j0 && k == k0);
else
return f(i+1, j, k, L, j0, k0) + f(i+1, j+1, k, L, j0, k0) + f(i+1, j+1, k+1, L, j0, k0);
}
```

It can be proved by induction that:

1. parity of $f(i, j, k, L, j_0, k_0)$ is equal to 1 iff color of stone (i, j, k) changes with changing of color of (L, j_0, k_0);
2. $f(i, j, k, L, j_0, k_0)$ is the number of ways from the stone (i, j, k) to (L, j_0, k_0) using only "edges"$(i, j, k) \to (i + 1, j, k), (i, j, k) \to (i + 1, j + 1, k), (i, j, k) \to (i + 1, j + 1, k + 1)$.

So, we should find the parity of the number of ways from the stone $(1, 1, 1)$ to the stone (L, j, k). To achieve the goal we should make $k - 1$ steps of type $(i, j, k) \to (i + 1, j + 1, k + 1)$, $j - k$ steps of type $(i, j, k) \to (i + 1, j + 1, k)$, and $L - j$ steps of type $(i, j, k) \to (i + 1, j, k)$; the number of ways to do it is $C_{L-1}^{k-1} * C_{L-k}^{j-k}$. But the problem is that L can be 10^9; how to calculate the parity of number like that?

Let degree2(x) be some nonnegative integer $z = z(x)$ so that $x = 2^z * k, k$ is odd. Also, let $factorialDegree2(x)$ be $degree2(factorial(x))$. Then, $factorialDegree2(x) = [x/2] + [x/4] + [x/8] + ...$ and can be computed in $O(logx)$. As a consequence, $degree2(C_n^k) = factorialDegree2(n) - factorialDegree2(n-k) - factorialDegree2(k)$. So, changing the color of some stone (L, j, k), we can calculate if the stone $(1, 1, 1)$ was changed in $O(logL)$. To solve given problem it is sufficient to summarise (modulo 2) parities of $1 - L \ mod \ 2$ and $f(1, 1, 1, L, A_i, B_i)$ for all $i, 1 \leq i \leq N$. The time complexity is $O(NlogL)$.

9 Problem I

Let's apply a transformation f to the coordinates of stones so that $f(i, j) = (i - j, j)$. Then, for given, after transformation, A, B, C, D, we should delete a minimal number of stones so that

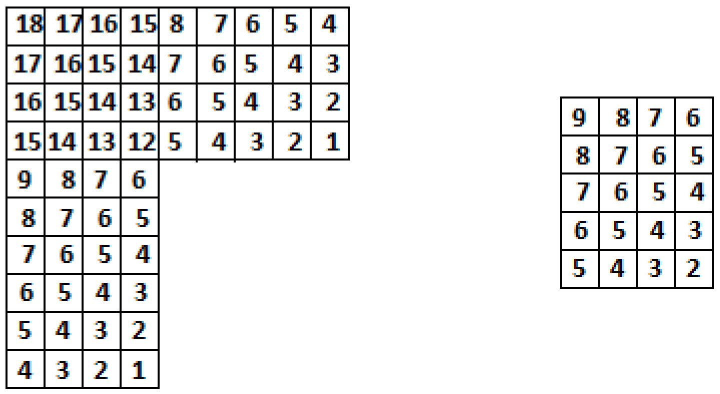
1. stones (A, B) and (C, D) are deleted;
2. stone (i, j) can be deleted only after $(i - 1, j)$ and $(i, j - 1)$.

This task can be solved using hook-length formula:

$$C = \frac{n!}{\prod_{(i,j)|(i \leq a \cap j \leq b) \cup (i \leq c \cap j \leq d)} h_\lambda(i,j)},$$

where $h_{lambda}(i, j)$ is the number of removing stones (x, y) so that $(x = i \cap y \geq j) \cup (y == j \cap x \geq i)$ (see details in https://en.wikipedia.org/wiki/Hook_length_formula). It's guaranteed that $n \leq 10^6$, so we can precalculate all factorials (or copy from solutions of some previous problems); but how to calculate the product of all h_λ?

For better understanding look at the picture:



18	17	16	15	8	7	6	5	4
17	16	15	14	7	6	5	4	3
16	15	14	13	6	5	4	3	2
15	14	13	12	5	4	3	2	1
9	8	7	6					
8	7	6	5					
7	6	5	4					
6	5	4	3					
5	4	3	2					
4	3	2	1					

9	8	7	6
8	7	6	5
7	6	5	4
6	5	4	3
5	4	3	2

On the left side of the picture typical situation of this task is pictured; it's obvious that calculating of h_λ can be reduced to no more than three query for calculating function $g(A, B, s) = \prod_{0 \leq i < A, 0 \leq j < B}(s + i + j)$. To calculate g, precalculate in $O(10^6)$ time two functions:

- $fact(n)$: $fact(-1) = fact(0) = 1, fact(i) = fact(i - 1) * i for 0 < i \leq 2 * 10^6$;
- $fact2(n)$: $fact2(-1) = fact2(0) = 1, fact2(i) = fact2(i - 1) * fact(i) for 0 < i \leq 2 * 10^6$.

In terms of these functions, $g(A, B, s) = \prod_{0 \leq i < A, 0 \leq j < B}(s + i + j) = \prod_{0 \leq i < A} fact(s + i + B - 1)/fact(s + i - 1) = \frac{fact2(s+A+B-2)/fact2(s+B-2)}{fact2(s+A-2)/fact2(s-2)}$; so, using precalculations, g can be calculated in $O(1)$ time. So, answer on every query can be computed in $O(1)$.

10 Problem J. Foot Game

It is one of the simplest problems in this contest. Obviously, minimum possible X is the maximum of buttons pressed during some fixed period of time; we can find it in $O(MAXTlogMAXT)$, where $MAXT$ is maximal moment of time in input data, using segment tree of in $O(MAXT)$ using method of differences.

After erasing of one of buttons, answer decreases by 1 iff there exists a button with time segment "covering"all moments of time with maximal number of buttons (otherwise, the answer is not changed); minimal segment containing all such moments can be calculated in $O(MAXT)$; after that, each button can be checked on "coveringness"in $O(1)$. So, we have a solution with time complexity $O(MAXT + N)$.

11 Problem K. Pyramid Game

Let S be $A_1 + A_2 + ... + A_N$, and $a = min(A_1, A_2, ..., A_N)$. Notice that if N is odd, then parity of S is changed after every turn; then, first player (Iori) wins iff at the beginning of the game, S is odd.

If N is even, then player can save the parity by taking one stone from all piles. This implies the fact that if $a = 1$ then first playes wins (if S is even, this player remove 1 from all piles; otherwise he takes 1 from ith pile; in both variants, second player gets a position in which S is even, and all future moves will change the parity of S; so, first player wins this game).

But what if N is even, and $a \geq 2$? Notice that if some $a = 2$, then it is bad way for every player to make a non-changing-parity turn (because in this case, minimum of A_i is becomes 1); so, players will make a turns removing only 1 stone until all A_i becomes 2. It implies that in this case, winner is first iff parity of S is 1. Repeating arguments for $a = 3$ and 4, 5 and 6 etc. and uniting all cases, we get a $O(N)$-time solution:

- if N is odd then first wins iff S is odd;
- if N is even then first wins iff a is odd or S is odd.