Bytecode 2011

Problems and Solutions

- P. Suhash Venkatesh
  Vikas Kumar Mall
  Venkatesh Basker

Problem A: Traversing Grid

Given 2 points in 2 dimensional space (xs,ys) and (xd,yd), your task is to find whether (xd,yd) can be reached from (xs,ys) by making a sequence of zero or more operations.
From a given point (x, y), the operations possible are:

a) Move to point (y, x)
b) Move to point (x, -y)
c) Move to point (x+y, y)
d) Move to point (2*x, y)

Input

The first line of input contains 'T', the number of test cases. 'T' lines follow, one for each test case. For each test case, the input contains one line denoting the 4 integers xs, ys, xd, yd.

Output

Output 'T' lines, one for each test case. For each test case, output "YES" if (xd,yd) is reachable from (xs,ys) and "NO" otherwise (quotes for clarity).

Example

Input:
1
1 1 2 2

Output:
YES

Constraints:
T <= 25
-10^10 <= xs, ys, xd, yd <= 10^10
Note that, although the input values are constrained by the above inequality, the coordinates of the points at the intermediate steps need not be.

Explanation:

Test case 1: We can move in the following manner: (1,1) -> (2,1), using the operation (x,y) -> (2*x,y). Then, move from (2,1) -> (1,2), using the operation (x,y) -> (y,x). Finally use the operation (x,y) -> (2*x,y) to move from (1,2) -> (2,2).

Time limit: 1s

Solution:

Firstly, note that the operations a, b and c do not change the gcd, and operation d either leaves the gcd as the same or multiplies the gcd by 2.

Therefore, (xd,yd) can be reached from (xs,ys) only if gcd(xd,yd)=gcd(xs,ys) * 2^k, for some k>=0.

Now, to prove that this is indeed a sufficient condition, we have to prove that (xd,yd) can be reached from (xs,ys) if the above condition is satisfied. For this, we will give a construction for the solution:

Keep performing operation d (combined with operation a) till we get gcd(xs,ys)=gcd(xd,yd). Now we can prove that we can reach (xd,yd) from (xs,ys) using only operations a,b,c.

Let g=gcd(xs,ys)=gcd(xd,yd)

Using the same method of repetitive subtraction for finding the gcd of 2 numbers, we can reduce both pairs of numbers to (g,0). But now, note that each operation we performed is reversible, that is if we can move from point P1 to P2, then we can move from point P2 to P1 too. Hence, this proves that we can reach (xd,yd) from (xs,ys).

Complexity: O(log MAX_COORD)

Problem B : Finding Minimum

You are given 'n' integers k_1, k_2, ..., k_n and an integer 'x', which satisfy the equation $x_1^{k_1} * x_2^{k_2} * ..... * x_n^{k_n} = x$. You are also given values a_1, a_2, ..., a_n and y_1, y_2, ..., y_n. Your task is to find the least positive value 'v', that can be taken by the expression: $a_1*x_1^{y_1} + a_2*x_2^{y_2} + ... + a_n*x_n^{y_n}$. Note that x_1, x_2, x_3, ... x_n are some variables (not necessarily integers), which can only take positive values.

Input

The first line of input contains a single integer 't', denoting the number of test cases.
The first line of each testcase contains two space separated integers 'n' and 'x'.
Next line contains 'n' integers k_1, k_2, ..., k_n.
Next line contains 'n' integers a_1, a_2, ..., a_n.
Next line contains 'n' integers y_1, y_2, ..., y_n.

Output

For each testcase output the least positive value 'v' that can be taken by the expression. To avoid floating point errors, round it off to the nearest integer.

For example, 12.6 is rounded off to 13, and 12.4 is rounded off to 12. To avoid ambiguity, there will be no test case for which the fractional part of the answer equals 0.5.

Example

Input:
2
1 4
2
3
3
2 6
1 1
1 1
1 1

Output:
24
5

Constraints:
t <= 25
1 <= n <= 20
1 <= x <= 1000000
1 <= k_i, a_i, y_i <=20
$x_i > 0$

Explanation:

Test case 1: $x_1^2 = 4$. Therefore, $x_1 = 2$ and $3*x_1^3 = 24$.
Test case 2: $x_1*x_2 = 6$. Minimum value of $x_1 + x_2$ is 2*sqrt(6) = 4.89897. x_1 = sqrt(6) and x_2 = sqrt(6) gives this solution. Answer is 4.89897, which when rounded off to the nearest integer equals 5.

Time limit: 1s

Solution:

This problem exploits the following property: If the product 'n' quantities is a constant, then their sum is minimum when all the 'n' quantities are equal (when the quantities are positive). Now, we are given an equation of the form: $x_1^{k_1} * x_2^{k_2} * ..... * x_n^{k_n} = x$. By making some clever substitutions of the form $x_1^{k_1} = X_1$, $x_2^{k_2} = X_2$, ..., $x_n^{k_n} = X_n$ etc, and by using the above property, the problem can be solved easily.

Complexity: O(n)

Problem C : Fun With Inequalities

You are given 'n' inequalities. Each inequality is of one of the following 4 types:

Type 1: x > v
Type 2: x < v
Type 3: x = v
Type 4: x != v

where 'x' is a variable which can only take non-negative integral values.

Your task is to find the maximum number of inequalities which are satisfied for some value of 'x'. You are also required to find the minimum value of 'x' for which the maximum number of inequalities are satisfied.

Input

The first line of input contains a single integer 'n', denoting the total number of inequalities. Each of the next 'n' lines contain 2 space separated integers t_i and v_i. t_i denotes the type of inequality and v_i denotes the value on the right hand side of the inequality.

Output

Output two space separated integers, the first integer denoting the maximum number of inequalities which are satisfied for some value of 'x', and the second integer denoting the minimum value of 'x' for which the maximum number of inequalities are satisfied.

Example

Input:
4
1 10
2 9
3 7
4 4

Output:
3 7

Constraints:
1 <= n <= 100000
1 <= t_i <= 4
1 <= v_i <= 10^18

Explanation:

The given inequalities are: 1) x > 10, 2) x < 9, 3) x = 7, 4) x != 4. For x=7, the inequalities 2), 3) and 4) are satisfied.

Time limit: 3s

Solution:

In this question, we are given many inequalities and we are to find the value of x which satisfies the maximum number of inequalities.

Firstly, for each inequality (of the form x operation v), we only need to consider v-1, v, v+1 as event points (along with 0).

Now, traverse once from left to right, simultaneously processing each inequality of the form x>v, and update the corresponding event points.

Similarly, we can do 3 more traversals for the other 3 types of inequalities. If the maximum value of v was small, then this could simulated using an array. The values of v can be really large, but for this problem, there can be at-most 3*n different values which can be taken by v. Since the value of 'n' is considerably small, we can map the possible values of v to smaller integers and then simulate.

Complexity: O(N*log N) (Assuming mapping the values takes O(log N) time).

Problem D : Maximum Profit

Chakra is a young and dynamic entrepreneur, who is developing rapidly as a successful hotelier. He owns the Quickbyte chain of restaurants, 'M' of which are fully functional now. He divides each day into 'N' time slots. For each time slot 'j', in every restaurant 'i', there are A_{ij} waiters and B_{ij} customers. Being a quality conscious person, he wants each waiter to handle atmost one customer in a given time slot. Since he is really busy, in a day each restaurant is open only during one of the time slots. Since the hunger and demand for food varies during the day, the price which the customer is willing to pay varies, and is given by C_{ij} for a restaurant 'i' during a time slot 'j'.

Given the values of A_{ij}, B_{ij} and C_{ij}, find the maximum profit which Chakra can make in a day.

Input

The first line of input contains an integer 't', denoting the number of test cases.
For each testcase, the first line contains 2 space separated integers 'M' and 'N'.
Each of the next 'M' lines contains 'N' integers. The j^{th} integer on the i^{th} line denotes the value of A_{ij}

Each of the next 'M' lines contains 'N' integers. The j^{th} integer on the i^{th} line denotes the value of B_{ij}

Each of the next 'M' lines contains 'N' integers. The j^{th} integer on the i^{th} line denotes the value of C_{ij}

Output

For each test case output one value, denoting the maximum profit which Chakra can make in a day.

Example

Input:
1
2 3
1 2 3
3 2 1
3 2 1
1 2 3
4 5 2
3 1 6

Output:
16

Constraints:
t <= 50
1 <= M,N <= 100
1 <= A_{ij}, B_{ij} <= 5000
0 <= C_{ij} <= 10^9

Explanation:

Test case 1: By opening the first restaurant at time slot 2 and second restaurant at time slot 3, Chakra makes a profit = 2*5 + 1*6 = 16. Note that although there are 3 customers for the second restaurant at time slot 3, since there is only 1 waiter, only 1 customer can be served.

Time limit: 3s

Solution:

For each restaurant, greedily pick the slot which gives the maximum profit. For slot (i,j), profit=C(i,j)*min(A(i,j)*B(i,j)).

Complexity: O(M*N)

Problem E: Recover Polynomials

Venkatesh is an expert in mathematics, and loves playing around with polynomials during his free time. His favourite mathematical equation is pretty obviously: $f(x) = a_n*x^n + a_{n-1}*x^{n-1} + ... + a_1*x + a_0$. His friend Suhash loves posing challenges to Venkatesh. Once they were discussing a particular problem at Snacky, which goes as follows:

Suhash would choose an integer 'n' as the degree of the polynomial and give Venkatesh the value of the polynomial at 'n+1' equally spaced points, i.e he gives Venkatesh integers 'n', 'x_0', 'd' and g_0, g_1, g_2, ..., g_n such that: $f(x_0) = g_0$, $f(x_0+d) = g_1$, $f(x_0+2*d) = g_2$, ... , $f(x_0+n*d) = g_n$. Now, Venkatesh is required to find the polynomial. Since he hates floating point values, he decides to find the polynomial in coefficient form, modulo a prime number. Can you help Venkatesh find the polynomial?

Input

The first line of input contains an integer 't', denoting the number of test cases.
For each test case, the first line contains 3 space separated integers 'n', 'x_0', 'd'. The next line contains 'n+1' space separated integers g_0, g_1, g_2, ... ,g_n.

Output

For each test case output 'n+1' integers, denoting the coefficients of the polynomial a_0, a_1, a_2,..... a_n. All the coefficients that are printed should be non-negative and should be less than 1000000007.

You are required to find coefficients of the polynomial a_0, a_1, a_2,.... ,a_n, which satisfy the equations: $f(x_0)\%1000000007 = g_0$, $f(x_0+d)\%1000000007 = g_1$, ..., $f(x_0+n*d)\%1000000007 = g_n$. It is guarenteed that there is a unique solution for every test case.

Example

Input:
1
3 1 1
10 26 58 112

Output:
4 3 2 1

Constraints:
t <= 25
1 <= n <= 1000
0 <= x_0 <= 100000
0 < d <= 10000
0 <= g_i <= 10^9

Explanation:

Test case 1: It can be seen that the polynomial $f(x) = x^3 + 2*x^2 + 3*x + 4$ satisfies the above input.

Time limit: 15s

Solution:

There were 2 approaches to solve this problem:

Solution idea 1: (doesn't use the fact that the x coordinates given are in AP):

From Lagrange's theorem , we have got a formula to find the polynomial given n+1 points. Since we are dealing with a prime field, we can replace the division by modulo inverse. But straightforward implementation of this will lead to an O(N^3) algorithm.

To improve on this, we can initially compute the polynomial $(x-x_0)*(x-x_1)*...*(x-x_n)$. Now to get the numerator of each term present we just need to perform 1 divsion. This leads to an O(N^2) solution.

Solution idea 2: (works only when the x coordinates given are in AP):

The given set of equations can be written in matrix form as: P A = G, where P in a (n+1) * (n+1) matrix where $P[i][j] = (x0+(n-i)*d)^j$, A is a (n+1)*1 matrix where A[i][0] = ai, and G is also a (n+1)*1 matrix where G[i][0] = gi.

Now, after applying a series of row transformations, we end up with the equation (in matrix form): M A = G'.

G'[i][0] = Gi, where Gi is the new rhs value after applying the row transformations.

M[i][j]=0 (if i<j), sum(k=0 to i, $(-1)^k$ * choose(i,k) * $(x0+(n-k)*d)^j$) (if i>=j).

By using some mathematics, we can get the following relation for M[i][j]:

M[i][j] = 0 (if i<j), i! * d^i (if i=j), (x0+n*d-i*d) * M[i][j-1] + i*d*M[i-1][j-1] (if i>j).

Now, we can polpulate the table for M in $O(n^2)$, after which we can solve easily for the coefficients in $O(n^2)$.

Overall complexity: $O(n^2)$

Problem F: Life Game

Gobo and Muku were really bored of working and decided to play a game on their respective laptops - the game of life. It is a one player game which consists of an M*N rectangular grid. Each cell of the grid contains exactly one magical potion. The potion at the j^{th} column of the i^{th} row of the grid increases the player's current health by V_{ij}.(This value can be negative, in which case the player's health decreases). At any point of time, the health of a player can be negative too (i.e. He does not die). From a cell(i,j), the player can move to cells (i+1,j-1) or (i+1,j) or (i+1,j+1), as long as these cells exist in the grid. Initially, the player has a health of 0. He can start from any column on the first row (1,j). If he chooses to enter a cell, then he is forced to drink the potion in that cell. The game is completed when any column of the last row is reached. There are 2 modes in which the game can be played: the "min" mode and the "max" mode. In "max" mode, the aim is to finish the game with maximum health H_{max} satisfying the condition A <= H_{max} <= B. Similarly, in "min" mode the aim is finish the game with minimum health H_{min}, satisfying the conditions A <= H_{min} <= B. Now, Gobo decides to play the game in "max" mode on his laptop, and Muku decides to play the game in "min" mode on his laptop. Can you help Gobo and Muku finish with maximum and minimum health respectively, satisfying the above conditions?

Input

The first line of input contains an integer 't', denoting the number of test cases.
For each test case, the first line contains 2 space separated integers 'M' and 'N'. The next line contains 2 space separated integers 'A' and 'B'. Each of the next 'M' lines contain 'N' integers. The j^{th} integer on the i^{th} line denotes the value V_{ij}.

Output

Output 2 space separated integers H_{min} and H_{max}, the minimum and maximum health with which Gobo and Muku can finish the game. H_{max} and H_{min} should satisfy A <= H_{max}, H_{min} <= B. If it is not possible to achieve such a health, output "NO" (quotes for clarity).

Gobo and Muku start playing on 2 different instances of the same game independently. i.e the values of A,B and initial values of V_{ij} are same for both grids.

Example

Input:
2
3 3
5 10
2 5 10
-1 -10 3
-3 6 -2
2 3
8 11
2 5 10
-1 -10 2

Output:
6 10
NO NO

Constraints:

$t <= 10$
$1 <= M,N <= 25$
$-1000 <= A <= B < 1000$
$-25 <= V_{ij} <= 25$

Explanation:

Test case 1: Take the path (1,2) -> (2,1) -> (3,2), to get a value 5-1+6=10. Take the path (1,2) -> (2,3) -> (3,3), to get a value 5+3-2=6.
Test case 2: There is no valid path which satisfies the above conditions.

Time limit: 1s

Solution:

Let dp[i][j] store the set of all possible values which you can get if you start from some column in row 1. Now, there are only 3 ways to reach (i,j), namely from (i-1,j-1), (i-1,j) and (i-1,j+1). So, with the sets of values which can be obtained for these locations, we can find the set of value obtainable for the current location.

Also, note that since -25 <= vij <= 25, only a maximum of 25*51 (~1300) values are obtainable, which is pretty small, so this solution will run in time.

Overall Complexity: O(M*N*1300*log(1300))

Problem G: Coloring Trees

Nivash and Bhoopathi play a game of memory, which goes as follows: There is a tree containing 'N' nodes, all of which are initially uncoloured. In the game, Nivash has 2 moves:

1) Command: Color a particular node with a given color.
2) Query: Ask Bhoopathi if the path from node 'a' to node 'b' (both inclusive), is monochromatic or not.(i.e Whether all nodes on the path have the same color).

Nivash can do these steps in any order he wishes and he colors each node atmost once. Whenever Nivash puts forth a 'Query' at Bhoopathi, Bhoopathi has to recollect the colouring of the tree and reply either "YES" or "NO". Can your help Bhoopathi answer these queries?

Input

The first line of input contains an integer 'N', denoting the number of nodes in the tree. The next 'N-1' lines contain 2 space separated integers 'u' and 'v', denoting an edge between vertex 'u' and vertex 'v'.
The next line contains an integer 'Q', denoting the number of inputs (commands and queries) which Nivash wants to give. The next 'Q' lines contain 3 space separated integers 'x', 'a', 'b'. If 'x' is 1, it denotes a command to color node 'a' with a color 'b'. If 'x' is 2, it denotes a query and Bhoopathi should answer if the path from node 'a' to node 'b' (both inclusive), is monochromatic or not.

All vertices of the tree are 0 based.

Output

For each query, output "YES" or "NO" (quotes for clarity), denoting whether the path from node 'a' to node 'b' (both inclusive), is monochromatic or not.

Output "NO", even if all nodes on the path from node 'a' to node 'b' (both inclusive) are uncolored.

Example

Input:
3
0 1
1 2
7
1 0 11
2 0 1
2 0 2
1 2 12
1 1 11
2 0 1
2 0 2

Output:
NO
NO
YES
NO

Constraints:
1 <= N <= 100000

1 <= Q <= 200000
1 <= color value <= 30.

Explanation:

Initially node '0' is colored with color '11', so path between node '0' and node '1' is not monochromatic. Hence, the answer is "NO". The same explanation holds for the path between node '0' and node '2'. Then node '2' is colored with color '12' and node '1' with color '11'. Now, all nodes on the path between node '0' and node '1' are colored with only one color ('11'), so the answer is "YES". The path between node '0' and node '2' has 2 colors ('11' and '12'), hence the answer is "NO".

Time limit: 2s

Solution:

We can use a disjoint set data structure to solve this question.
When a command arrives to color a given node u, check all the neighbors v of the node. If the color of v = color of u, then merge the sets containing the nodes u and v.
When a query arrives for 2 nodes u and v, we just need to check whether both u and v are in the same set.

Complexity: O(N+Q) (assuming that the operations of disjoint set can be done in O(1))

Problem H: Trie Expectation

What is the expected number of nodes in a trie when 'N' words, each of length 'L' are inserted into it. The words are made up only of 0's and 1's. The words may be repeated and all possible permutations of words are equally likely. Initially the trie consists of only one node (root node).

Input

The first line of input contains an integer 't', denoting the number of test cases. Each of the next 't' lines contain 2 space separated integers 'N' and 'L'.

Output

For each test case, output one floating point value denoting the expected number of nodes in the trie. Output the values rounded off to 2 decimal places. Always print 2 digits after the decimal point.

To know more about tries visit here.

Example

Input:
2
1 3
2 2

Output:
4.00
4.25

Constraints:
t <= 25
1 <= N <= 300
1 <= L <= 300

Explanation:

Test case 1: There are 8 possible words of length 3. Which ever word is inserted into the trie, we get only 4 nodes.

Time limit: 3s

Solution:

Let dp[n][l] store the expected number of nodes in a trie if there are n words each of length l are inserted into it.

Now, suppose i words have a 0 at the beginning and n-i words have a 1 at the beginning. This leaves us with subproblems dp[i][l-1] and dp[n-i][l-1] which we would have already computed if we process in the increasing order of l.

Note that the probability of i words having a 0 at the beginning at n-i nodes having a 1 at the beginning is just choose(n,i)/(2^n). So, we will have to precompute choose(n,i) too initially.

Overall Complexity: O(N*N + N*L)

Problem I: Permutation Game

Harsha is given 9 integers a_1, a_2, a_3, ... a_9. This denotes that he is given a_1 1's, a_2 2's,.....a_9 9's. Let 'x' = (a_1 + a_2 + ...a_9). Now, Harsha makes all possible 'x' digit numbers by using these given digits. Let S be the set of all such numbers which he makes. Now he constructs a directed graph containing |S| nodes, in which each node denotes a unique number from the set. For all numbers u,v belonging to S, there is a directed edge from node 'u' to node 'v in the graph iff u>v. It is easy to note that we obtain a directed acyclic graph. Whats more, the edges of the graph are weighted. The weight of an edge joining node 'u' and node 'v' is equal to u+v. Now, Deepak decides to test Harsha's memory and gives him 'Q' queries. Each query consists of two numbers 'u', 'v' (u>v, both belonging to the set S). For each query Harsha must provide the following answers:

1) How many distinct paths are there from node 'u' to node 'v' in the graph.
2) For each distinct path 'i' from node 'u' to node 'v', let S_i denote the sum of weights of all edges on this path. Calculate the value of sum(S_i), for every distinct path 'i' from node 'u' to node 'v'.

Input

The first line of input contains 9 integers a_1, a_2, ...a_9. The second line contains a single integer 'Q', denoting the number of queries. Each of the next 'Q' lines contain 2 numbers 'u' and 'v'.

Output

For each query, output 2 space separted integers denoting the number of distinct paths and sum of weights of all paths respectively. Since the output can be large, output these quantities modulo 1000000007.

Two paths (v_1, v_2, ... v_m) and (u_1, u_2, ... u_n) are distinct if:

1) m != n
2) m = n, there exists some index 'k' (1 <= k <= m) such that v_k != u_k

Example

Input:
2 0 1 0 0 0 0 0 0
1
311 113

Output:
2 1110

Constraints:
1 <= (a_1 + a_2 + ... a_9) <= 500
1 <= Q <= 20
a_i >= 0

Explanation:

Test case 1: The set S for the above problem is {311, 113, 131}. The edges of the graph are 311->131, 311->113, 131->113. There are 2 distinct paths from 311 to 113, namely (311->131->113) and (311->113). The sum of weights of edges on path-1 = (311+131)+(131+113) = 686. For path-2, the sum of weights of edges = (311+113) = 424. Therefore, answer = 686 + 424 = 1110.

Time limit: 15s

Solution:

Let 'u' and 'v' be some given numbers from the set S ('u' > 'v'). Let x(i) denote the node having a rank 'i'. Let i_r denote the rank of node 'i'. It can be seen that the number of distinct paths from 'u' to 'v' is $2^{(u_r-v_r-1)}$.

Proof is as follows: Let, $f(x(u_r))$ be the the number of distinct paths from node 'u' to node 'v'. We can write it as: $f(x(u_r)) = f(x(u_r-1)) + f(x(u_r-2)) + ... f(x(v_r))$. This reduces to $f(x(u_r)) = 2*f(x(u_r-1))$.

We know that $2^x \equiv 2^{(x \bmod \varphi(p))} \pmod p$.

To find $2^{(u_r-v_r-1)} \% p$, first find $(u_r-v_r-1) \% \varphi(p)$. Here, p=1000000007 and $\varphi(p) = p-1 = 1000000006 = 2 * 500000003$. The calculation of rank of a permutation involves evaluation of multinomial coefficients. Let p1=2 and p2=500000003. To calculate rank of a permutation mod (p1*p2), first calculate rank mod p1, then rank mod p2. After that we can apply chinese remainder theorem to get rank mod (p1*p2).

Similary, for the second part of the problem, answer = $2^{(u_r-v_r-1)} *$ (sum of all numbers belonging to S, which are >= u and <= v). Let, $g(x(u_r))$ deonte the sum of weights of all distinct paths from node 'u' to node 'v'.

We have, $g(x(u_r)) = g(x(u_r-1)) + (x(u_r) + x(u_r-1)) * f(x(u_r-1)) + g(x(u_r-2)) + (x(u_r) + x(u_r-2)) * f(x(u_r-2)) + ... + g(x(v_r)) + (x(u_r) + x(v_r)) * f(x(v_r))$. On simplification, this gives $g(x(u_r)) = f(x(u_r)) * (x(u_r) + x(u_r-1) + ... + x(v_r))$.

Let $h(x(u_r))$ = sum of all numbers belonging to S, which are <= v. So, our required answer for the second part of the problem is $2^{(u_r-v_r-1)} * (h(x(v_r))-h(x(u_r))+u)$. Let 'n' = $a_1+a_2+...+a_9$. $h(x(u_r))$ can be easily calculated in O(n*100).

Overall complexity: O(n*100*Q)

Problem J: Grid Tiling

Vikas is the chief interior designer incharge of the Taj Palace, Mumbai. He wants to make an impressive and colourful pattern in the courtyard. Importing exotic tiles has become very difficult after the Mumbai terror attacks, and therefore Vikas has only 4 kinds of tiles to choose from:

```
A    B   C    D
==   ==  ==   ==
XX   X   X    X
XX   X   XX
```

Any rotation of above tiles is also permitted.

Each tile is available in 'k' different colors, and there's an infinite supply of all tiles. The courtyard has dimensions 2 * 'n'. Vikas wants to tile the courtyard in such a way that no two adjacent tiles have the same color. Two tiles are considered adjacent if they share a common side. Two tilings are considered different if:

1) The arrangement of tiles is different.
2) There is atleast 1 position (1*1 square) which has different colors in the two arrangements.

Can you help Vikas find the number of different ways in which he can tile the courtyard, subject to the above conditions?

Input

The first line of input contains a single integer 't', denoting the number of test cases. Each of the next 't' lines contains 2 space separated integers 'n' and 'k'.

Output

For each test case output one integer, denoting the number of different ways in which the courtyard can be tiled.

Two tiles are considered adjacent if they share an edge. Two tiles which just share a common point are not considered adjacent.

Example

Input:
2
1 1
1 2

Output:
1
4

Constraints:
t <= 1000
1 <= n <= 10^9
1 <= k <= 1000

Explanation:

Test case 1: There is only 1 way to tile the courtyard, by using a 2*1 tile.
Test case 2: Let '1' and '2' be the available colors. The grid can be tiled in 4 ways - 1) place one 2*1

tile of color '1', 2) place one 2*1 tile of color '2', 3) Place two 1*1 tiles (color '1' above and color '2' below), 4) Place two 1*1 tiles (color '2' above and color '1' below)

Time limit: 2s

Solution:

Let f_n be the number of ways of tiling a 2*n room in which the 2 positions to the right of the last tile have the same color. Let g_n be the number of ways of tiling a 2*n room in which the 2 positions to the right of the last tile have different colors. Let the definitions of hn and wn be similar to that of fn and gn, except that they represent a 2*n room in which a 1*1 tile is missing in the last column.

We can write the following recurrence relations:

$h_i = (k-1)*g_{i-1} + (k-1)*w_{i-1} + (k-1)*f_{i-2}$

$w_i = (k-2)*g_{i-1} + (k-2)*w_{i-1} + (k-2)*f_{i-2}$

$g_i = (k-2)*f_{i-1} + (k-2)*f_{i-2} + (2*k-4)*w_i + (2*k-4)*h_{i-1} + (k*k-3*k+3)*g_{i-2} + 2*h_i - (k*k-3*k+3)*g_{i-1}$

$f_i = (k-1)*f_{i-1} + (k-1)*f_{i-2} + (2*k-2)*w_i + (2*k-2)*h_{i-1} + (k*k-3*k+2)*g_{i-2} - (k*k-3*k+2)*g_{i-1}$

Be careful with boundary conditions.

Answer is $2*k*(k-1)*w_{n-1} + 2*k*(k-1)*f_{n-2} + k*(k-1)*g_{n-1} + k*f_{n-1} + k*f_{n-2} + 2*k*h_{n-1} + k*(k-1)*g_{n-2}$

We can use matrix exponentiation on a 6 * 6 matrix to solve the above recurrence relations.

Overall complexity: O(6^3 * log n)

Problem K: Array Sorting

Sumit specialises in sorting algorithms, and Abhishek decides to test Sumit's coding skills. An array of 'n' numbers a[0], a[1], a[2], ..., a[n-1] is given. Abhishek gives a sequence of inputs of the form "v i j". Each input is either a query or an update (query if 'v' is 0, update otherwise).

For any input of the form "0 i j" ,Sumit's output should be as follows:
If the subarray a[i], a[i+1], ... a[j] is unsorted, output 0.
If the subarray a[i], a[i+1], ... a[j] is sorted in non-descending order, output 1.
If the subarray a[i], a[i+1], ... a[j] is sorted in non-ascending order, output 2.
If the subarray a[i], a[i+1], ... a[j] is sorted in both non-ascending and non-descending order (i.e, if all the elements in the range are equal), output 3.

Any other input "v i j" (v!=0) should be treated as an update, as follows:
For each element in the subarray a[i], a[i+1], ... a[j], Sumit has to replace the element a[k] with v-a[k].

Sumit is really tired and does not want to write a program. Can you write a program for Sumit, which responds to Abhishek's instructions?

Input

The first line of input contains 2 space separated integers 'n' and 'q'. The second line contains 'n' integers a[0], a[1], ..., a[n-1]. Each of the next 'q' lines contain 3 integers 'v', 'i', 'j'.

Output

For each query, output a single integer 0, 1, 2 or 3, denoting the answer.

Example

Input:
4 5
3 -2 -5 1
1 1 3
0 0 3
0 0 2
0 2 3
0 0 1

Output:
0
1
2
3

Constraints:
1 <= n <= 100000
1 <= q <= 100000
-5000 <= a[i] <= 5000
-5000 <= v <= 5000

Explanation
Initial array is {3, -2, -5, 1}. After first update, the array will be {3, 3, 6, 0}. Now, from indices '0' to '3', it is unsorted. From indices '0' to '2', it is sorted in non-descending order. From indices '2' to '3', it is sorted in non-ascending order. Between indices '0' and '1', the values are equal.

Time limit: 2s

Solution:

This problem can be solved by segment trees, using lazy propagation technique. The problem would be simpler if updates were of the form v+A[i], instead of v-A[i]. For updates of the form v-A[i], if multiple updates are performed on A[i], the value of A[i] depends on the order in which the updates are performed. This makes the problem harder.

Let each node of the segment tree store the following details: whether the range is ascending/descending/both, what is the leftmost value in the range and rightmost value in the range. Apart from these, we store 2 more parameters indicating a value(v_1) and sign(s_1) for a node (These values are propagated from a node to its children during updation). By default, all nodes have value(v_1) = 0, and sign(s_1) = 1. Building the segment tree is simple.

To update the segment tree, have 2 more additional parameters which indicate the value(v_2) and the sign(s_2) propagated to a node from it's ancestors. When a node receives a value(v_2) and a sign(s_2), the current details of the node are updated accordingly and a new value(v_2^1) and sign(s_2^1) are propagated to both its children. v_2^1, s_2^1 depend on v_1,v_2,s_1,s_2 ($v_2^1=v_2+s_2*v_1$, $s_2^1=s_2*s_1$) When the current range lies completely within the range we have to update, we stop moving down the tree.

Querying the segment tree can be achieved in a similar fashion, by propagating values. Each query and update take O(log n). For a clear implementation, see the judge solution.

Overall complexity: O(n * log n)