SKOCIMIS

Suppose kangaroo A is farther from B than B is from C. Then the three kangaroos will play longest if C jumps first, onto position A+1 or B−1. They can repeat this until they are standing next to each other.

The total number of jumps will be B−A−1 or C−B−1, depending on which pair of kangaroos was initially farther. The solution is the value of the expression max{B−A, C−B} − 1.

PTICE

The easiest way to solve this problem is to generate the three sequences of answers given by the boys and compare them to the correct sequence. First we count the maximum number of correct answers, then output which of the boys had that number.

MRAVOJED

The first step is to calculate, for each cell, how many consecutive characters 'x' there are starting with that cell, in each of the four cardinal directions.

We can recognize all cells that are corners of some square – these are cells with 'x' in them such that at least two neighbouring squares are empty. This is important because, except in the special case when one square is completely contained in the other, each of the two squares will have at least one corner recognizable that way.

Once we have recognized a corner, we can find the size of the square: check the numbers of consecutive squares 'x' in the two directions it extends in – the smaller of the two is the size of the square.

From the location of the corner and the size of the square, we can calculate the position of the upper-left corner. If we find only one such upper-left corner on the entire board, that means that one of the squares is inside the other so output any cell as the other square.

JEZ

There are three parts to the solution:

1. Find the number of complete diagonals the hedgehog walks on and also how many cells on the last diagonal.
2. For every row:
    a. Calculate how many cells in the row the hedgehog visits;
    b. Calculate how many of those cells are grey.

For the first part, it suffices to add diagonals one by one, stopping when the total number of cells visited is more than K.

The number of visited cells in a row can be calculated from R, S and the number of diagonals traversed.

What remains is to calculate, for two integers x and ymax, how many integers y between 1 and ymax there are such that x&y is zero, where & is the "bitwise and" operator. Consider the binary representation of ymax and let k be the position of the most significant binary one. First we calculate how many integers y there are with a binary zero in position k (all these numbers are less than ymax) sharing no binary ones with x. Then we calculate the same for y with a binary zero in position k. If x has a binary one in position k, then there are more y such that x&y=0 and we are done. Otherwise, look for the next highest binary one in ymax and repeat.

For example, if ymax=10110, then for k=4 we count all y between 00000 and 01111. For k=2 we will count all y between 10000 and 10011 and for k=1 all y between 10100 and 10101.

SKAKAVAC

Suppose first that the grasshopper can jump to any flower in the neighbouring rows and columns.

We can use dynamic programming to solve the problem. First sort the flowers by numbers of petals. For each flower F, we calculate the best(F), the largest number of flowers the grasshopper can visit if it starts at flower F. We calculate this by considering all flowers G in the neighbouring rows and columns; then best(F) = max{best(G)} + 1.

To calculate best(F) more quickly, we can keep track of the largest best(G) value for each row and column.

If we reintroduce the restriction from the problem statement on which flowers the grasshopper can jump to, this last improvement no longer works, since the grasshopper may not be allowed to jump to the best flower in a row or column. However, keeping track of the four best flowers in a row or column will do because at most three flowers in a row or column are forbidden.

KRTICA

Kako je broj bridova koje krtica može ukloniti linearan, moguće je iterirati kroz sve bridove kako bismo izračunali koji brid je potrebno dodati ako uklonimo taj brid.

Suppose we remove edge e = (a, b) and that doing so separates the tree into subtrees A and B. Now we need to create a new edge e' = (a', b') where vertex a' is in subtree A and vertex b' is in subtree B. There are many ways to do this to check them all.

Notice that the diameter of the new tree will be the largest of:

- The diameter of subtree A, of length d(A);
- The diameter of subtree B, of length d(B);
- The length of the longest path in subtree A rooted in a' plus the length of the longest path in subtree B rooted in b' plus one (for e').

The first two of these depend only on edge e, the last on edge e'. For this path to be as short as possible, we choose a' to be the halfway point on the diameter of subtree A and b' to be the halfway point on the diameter of subtree B.

The length of the new path is then:

$$\left\lceil \frac{d(A)}{2} \right\rceil + 1 + \left\lceil \frac{d(B)}{2} \right\rceil .$$

For each choice of edge e, we can calculate the shortest diameter we can achieve if we remove that edge, and it is exactly:

$$\min\left\{ d(A), d(B), \left\lceil \frac{d(A)}{2} \right\rceil + 1 + \left\lceil \frac{d(B)}{2} \right\rceil \right\}$$

So we need to calculate the diameters of both subtrees. The total number of subtrees we need to do this for is $2\cdot(N-1)$, so we cannot afford a linear search for every subtree.

If we root the tree in any vertex, then subtrees isolated by removing edges from the tree can be divided into two groups – those not containing the root (call these subtrees "hanging") and those containing the root (call these "trimmed").

We can use dynamic programming to calculate the diameters of hanging subtrees. Let d1(v) and d2(v) be the two longest paths from vertex v to leafs in the tree. Then the diameter d(v) of the subtree rooted at v is easily calculated as:

$$d(v) = \max\{d_1(v) + d_2(v), \max\{d(w); \text{for each child } w\}\}$$

The diameters of trimmed trees can also be found using dynamic programming. This time the recursive relation is more complex and calculated in the opposite order – from root to leaves.

Once we have calculated the diameteres we can easily find which edge to remove. After this, we can find the halfway points on the diameters of subtrees to find which edge to add.

The overall complexity of the solution is O(N).