MJEHURIC

The problem statement describes the bubble sort algorithm. It is required to implement the described algorithm and output the sequence after every swap. Using loops simplifies the code.

DATUM

The first example reveals that January 1, 2009 was a Thursday. Knowing how many days there are in every month, we can calculate for a given date which day it is in the year, and from that which day it is of the week.

ROT

Rotating by 90 degrees is not hard, we just need to figure out the formula to move elements: element (y, x) in the rotated table was element (C−x+1, y) in the original table.

With this we can rotate the table by 90 degrees until the sum of angles we rotated by is equal to K, or so that we need another rotation by 45 degrees.

Rotating by 45 degrees requires careful formatting of output, so that it looks slanted as in the examples.

SLIKAR

The solution calculates three numbers for every square S: the difference if we colour S all white, the difference if we colour S all black, and the difference if we recursively use the procedure from the problem statement (dividing into four sub-squares) on S.

To calculate this last number, we use dynamic programming. First separate S into four sub-squares and calculate all three numbers for each of the sub-squares. From the calculated numbers for sub-squares, it is trivial to calculate the first and second numbers for K. The third is calculated by considering all ways of choosing which sub-square is coloured white, which one is coloured black and which two we apply the recursive algorithm on. There are a couple ways to do this and we will of course choose the one that minimizes the overall difference.

TREZOR

We say that vaults with a common x-coordinate form a column. There are A+B+1 columns and the limits on A and B are relatively small, so we will solve the problem by counting vaults each of the guards sees in a particular column.

Let:

- a(x) be the total number of vaults in column x seen by the guard at (−A, 0);
- b(x) be the total number of vaults in column x seen by the guard at (B, 0);
- ab(x) be the total number of vaults in column x seen by both guards.

Knowing these numbers, we can calculate:

- The number of super-secure vaults in column x as ab(x);
- The number of secure vaults in column x as a(x) + b(x) − 2·ab(x);
- The number of insecure vaults as L − a(x) − b(x) + ab(x).

How do we calculate the number of vaults in a column? Let f(dx) be the number of vaults the guard can see in the column at distance dx from his position. For some y, the guard can see the vault at (x+dx, y) only if y and dx are relatively prime. If the greatest common divisor between y and dx is greater than 1, then y and dx can be divided by the divisor to get the coordinates of another point blocking the view.

The number of integers relatively prime with dx that are at most L can be calculated by finding the prime factors of dx and using the inclusion-exclusion principle. Take for example L=15 and dx=6. Then the number of vaults the guard can see in a column at distance 6 from his position is:

$$f(6) = \left\lfloor \frac{15}{1} \right\rfloor - \left\lfloor \frac{15}{2} \right\rfloor - \left\lfloor \frac{15}{3} \right\rfloor + \left\lfloor \frac{15}{6} \right\rfloor = 15 - 7 - 5 + 2 = 5$$

A special case is f(0) = 1, because the guard can only see the vault directly ahead of him in his column.

Now it is obvious that a(x) = f(x−A) i b(x) = f(B−x). It is less obvious that we can calculate ab(x) by applying the inclusion-exclusion principle to the union of prime factors of x−A and B−x.

PERIODNI

The table in the problem is called a histogram. We need to count the number of ways to separate the gases into K cells in the histogram H, according to the rules outlined in the problem statement. Let that number be dp(H, K). We can calculate it recursively.

Find the height of the lowest column in H and call it h.

There are three cases:

1. All columns are of height 0 – the histogram is empty. Therefore:

$$dp(H,K)=\begin{cases}1; \text{for } K=0\\ 0; \text{for } K>0\end{cases}$$

2. h = 0. There is a column of height zero, which means that we can split the histogram into two sub-histograms H_L and H_R (left and right of the zero-height column). We can independently calculate the number of ways to put gases into the two histograms, combining the results:

$$dp(H,K)=\sum_{k=0}^{K} dp(H_L,k)\cdot dp(H_R,K-k)$$

3. h > 0. There is a rectangle R of height h that is as wide as the histogram (call this width w). We will solve this case by counting how many ways we can put k gases into rectangle R, the remaining K−k gases going into the sub-histogram H_U (which is H without R). We can choose the rows into which to put gases in $\binom{h}{k}$ ways and permute them in k! ways. Then we can choose the columns in $\binom{w-(K-k)}{k}$ ways, because K−k columns are already occupied by gases above, in sub-histogram H_U.

$$dp(H,K)=\sum_{k=0}^{K}\frac{h!}{(h-k)!}\cdot\binom{w-(K-k)}{k}\cdot dp(H_U,K-k)$$

The total number of different histograms that will occur as the parameter in the recursion is in linear proportion to the width of the histogram: a recursive call in case 3 always yields a case 2 histogram, which reduced the histogram in width.

For a full score, it is also required to efficiently calculate the binomial coefficients, as in the example source code.