Direct Simulation

Ieuan David Vinluan

August 2024

Implementation Problems

Some problems pose situations that we can implement directly through code, such as...

Implementation Problems

Some problems pose situations that we can implement directly through code, such as...

- outputting the result of an algorithm after it is done this many times

Implementation Problems

Some problems pose situations that we can implement directly through code, such as...

- outputting the result of an algorithm after it is done this many times
- saying which player will win given a setup on a board game, some rules, and a set of moves played

Implementation Problems

Some problems pose situations that we can implement directly through code, such as...

- outputting the result of an algorithm after it is done this many times
- saying which player will win given a setup on a board game, some rules, and a set of moves played
- printing the resulting string after some operations are applied to a given string

Implementation Problems

If the given instructions are simple enough, then we can just do as instructed, assuming the constraints of the problem are small. After all, using brute force to solve CompProg problems is always a correct approach, though not always an efficient one!

Implementation Problems

If the given instructions are simple enough, then we can just do as instructed, assuming the constraints of the problem are small. After all, using brute force to solve CompProg problems is always a correct approach, though not always an efficient one!

A source of difficulty with these problems is that while the solution may be simple enough, implementing it through code and debugging can be challenging.

Implementation Problems

If the given instructions are simple enough, then we can just do as instructed, assuming the constraints of the problem are small. After all, using brute force to solve CompProg problems is always a correct approach, though not always an efficient one!

A source of difficulty with these problems is that while the solution may be simple enough, implementing it through code and debugging can be challenging.

Also, it's important to ensure that the time complexity of the algorithm is good enough for the given constraints!

Implementation Problems (Example)

Let's look at this AtCoder problem: Langton's Takahashi - ABC339 B

Problem Statement

There is a grid with H rows and W columns; initially, all cells are painted white. Let (i, j) denote the cell at the i-th row from the top and the j-th column from the left.

This grid is considered to be toroidal. That is, $(i, 1)$ is to the right of (i, W) for each $1 \leq i \leq H$, and $(1, j)$ is below (H, j) for each $1 \leq j \leq W$.

Takahashi is at $(1, 1)$ and facing upwards. Print the color of each cell in the grid after Takahashi repeats the following operation N times.

- If the current cell is painted white, repaint it black, rotate 90° clockwise, and move forward one cell in the direction he is facing. Otherwise, repaint the current cell white, rotate 90° counterclockwise, and move forward one cell in the direction he is facing.

Constraints

- $1 \leq H, W \leq 100$
- $1 \leq N \leq 1000$
- All input values are integers.

Implementation Problems (Example)

We can observe that the number of operations N is small $(1 \leq N \leq 1000)$.

Implementation Problems (Example)

We can observe that the number of operations N is small $(1 \leq N \leq 1000)$.

Since each operation can be completed in constant time, an $O(N)$ solution where we go through each operation one by one will be fast enough!

Implementation Problems

Thus, our solution can look like this:

Implementation Problems

Thus, our solution can look like this:

```
int h, w, n;
cin >> h >> w >> n;
vector<string> g(h, string(w, '.'));

for (int i = 0; i < n; i++) {
    // process operation here
}

for (int i = 0; i < h; i++) cout << g[i] << endl;
```

Implementation Problems (Example)

Now, how would one go about cleanly implementing this?

Implementation Problems (Example)

Now, how would one go about cleanly implementing this?

- Takahashi rotates depending on the color of the square he's standing on, and then moves forward

Implementation Problems (Example)

Now, how would one go about cleanly implementing this?

- Takahashi rotates depending on the color of the square he's standing on, and then moves forward
- Opposite sides of the grid are connected

Implementation Problems (Example)

- We can define two arrays `dx` and `dy` that describe a step forward. `dx` describes the change in horizontal position, while `dy` describes the change in vertical position.

Implementation Problems (Example)

- We can define two arrays `dx` and `dy` that describe a step forward. `dx` describes the change in horizontal position, while `dy` describes the change in vertical position.
- For example, if we move up: `dx = 0` and `dy = -1` (remember that $(1, 1)$ is the top-left cell and (H, W) is the bottom-right cell)

Implementation Problems (Example)

- We can define two arrays `dx` and `dy` that describe a step forward. `dx` describes the change in horizontal position, while `dy` describes the change in vertical position.
- For example, if we move up: `dx = 0` and `dy = -1` (remember that $(1, 1)$ is the top-left cell and (H, W) is the bottom-right cell)
- Then, if we arrange the elements of `dx` and `dy` such that counterclockwise and clockwise movements are next to each other: `dx = {1, 0, -1, 0}` and `dy = {0, 1, 0, -1}`.

Implementation Problems (Example)

Now, assuming we are facing up, how can we define a clockwise or counterclockwise turn?

Implementation Problems (Example)

Now, assuming we are facing up, how can we define a clockwise or counterclockwise turn?

- If we are facing up, we are currently at index 3 (`dx[3] = 0` and `dy[3] = -1`)

Implementation Problems (Example)

Now, assuming we are facing up, how can we define a clockwise or counterclockwise turn?

- If we are facing up, we are currently at index 3 (`dx[3] = 0` and `dy[3] = -1`)
- If we turn clockwise, we will face right; if we turn counterclockwise, we will face left

Implementation Problems (Example)

Now, assuming we are facing up, how can we define a clockwise or counterclockwise turn?

- If we are facing up, we are currently at index 3 (`dx[3] = 0` and `dy[3] = -1`)
- If we turn clockwise, we will face right; if we turn counterclockwise, we will face left
- Right and left are denoted by indices 0 and 2 respectively

Implementation Problems (Example)

Now, assuming we are facing up, how can we define a clockwise or counterclockwise turn?

- If we are facing up, we are currently at index 3 (`dx[3] = 0` and `dy[3] = -1`)
- If we turn clockwise, we will face right; if we turn counterclockwise, we will face left
- Right and left are denoted by indices 0 and 2 respectively
- Thus, clockwise turns are defined by adding 1 to our current index; counterclockwise turns are defined by subtracting 1

Implementation Problems (Example)

We can now code our solution. Remember that since our indices loop around, we can use modular arithmetic!

Implementation Problems (Example)

We can now code our solution. Remember that since our indices loop around, we can use modular arithmetic!

```
int h, w, n;
cin >> h >> w >> n;
vector<string> g(h, string(w, '.'));
int dx[4] = {1, 0, -1, 0}, dy[4] = {0, 1, 0, -1};
auto add = [&](int a, int b, int MOD) {
    return (a % MOD + b % MOD + MOD) % MOD;
};
```

(Cont.)

Implementation Problems (Example)

```
int x = 0, y = 0, direction = 3; // facing up!
for (int i = 0; i < n; i++) {
    if (g[y][x] == '.') {
        g[y][x] = '#';
        direction = add(direction, 1, 4);
    } else {
        g[y][x] = '.';
        direction = add(direction, -1, 4);
    }
    x = add(x, dx[direction], w);
    y = add(y, dy[direction], h);
}
for (int i = 0; i < h; i++) cout << g[i] << endl;
```

Implementation Problems (Example)

Next, let's look at the following Codeforces problem: Collatz Conjecture (1982B).

B. Collatz Conjecture

time limit per test: 1 second
memory limit per test: 256 megabytes

Recently, the first-year student Maxim learned about the Collatz conjecture, but he didn't pay much attention during the lecture, so he believes that the following process is mentioned in the conjecture:

There is a variable x and a constant y. The following operation is performed k times:

- increase x by 1, then
- while the number x is divisible by y, divide it by y.

Note that both of these actions are performed sequentially within one operation.
For example, if the number $x = 16$, $y = 3$, and $k = 2$, then after one operation x becomes 17, and after another operation x becomes 2, because after adding one, $x = 18$ is divisible by 3 twice.

Given the initial values of x, y, and k, Maxim wants to know what is the final value of x.

Input

Each test consists of multiple test cases. The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases. Then follows the description of the test cases.

The only line of each test case contains three integers x, y, and k ($1 \le x, k \le 10^9$, $2 \le y \le 10^9$) — the initial variable, constant and the number of operations.

Output

For each test case, output a single integer — the number obtained after applying k operations.

Implementation Problems (Example)

The most obvious solution would be to just use a `for` loop: do as instructed!

Implementation Problems (Example)

The most obvious solution would be to just use a `for` loop: do as instructed!

```
int t, x, y, k;
cin >> t;
for (int i = 0; i < t; i++) {
    cin >> x >> y >> k;
    for (int j = 0; j < k; j++) {
        x += 1;
        while (x % y == 0) {
            x /= y;
        }
    }
    cout << x << endl;
}
```

Time complexity: $O(kt)$

Implementation Problems (Example)

Obviously, this solution is too slow! Observe that $k \leq 10^9$ and $t \leq 10^4$, so for the test cases with higher values of k and t, we definitely can't run this within 1 second.

Implementation Problems (Example)

Obviously, this solution is too slow! Observe that $k \leq 10^9$ and $t \leq 10^4$, so for the test cases with higher values of k and t, we definitely can't run this within 1 second.

Thus, with implementation problems, it's important to observe patterns that will let us simplify the time complexity of our solution.

Implementation Problems (Example)

In this problem, two important observations can be made:

Implementation Problems (Example)

In this problem, two important observations can be made:

- We won't be dividing x by y every time: in most operations, we will only be adding 1 to x. We will only divide when we add enough to x such that it is divisible by y.

Implementation Problems (Example)

In this problem, two important observations can be made:

- We won't be dividing x by y every time: in most operations, we will only be adding 1 to x. We will only divide when we add enough to x such that it is divisible by y.
- For sufficiently large k, we will reach a point where $1 \leq x < y$. From here, we can determine the final value of x with the remaining k operations in constant time because we are sure that x will loop through values ranging from 1 to $y - 1$ in a cycle.

Implementation Problems (Example)

So, what improvements can we make to our brute-force algorithm?

Implementation Problems (Example)

So, what improvements can we make to our brute-force algorithm?

- We can combine all of the operations where we're only adding 1. In one step, we can add $\min(k, y - x \bmod y)$ and then divide x by y as much as possible

Implementation Problems (Example)

So, what improvements can we make to our brute-force algorithm?

- We can combine all of the operations where we're only adding 1. In one step, we can add $\min(k, y - x \bmod y)$ and then divide x by y as much as possible
- Once we reach a point where $x < y$, we can instantly compute the answer as follows:

Implementation Problems (Example)

So, what improvements can we make to our brute-force algorithm?

- We can combine all of the operations where we're only adding 1. In one step, we can add $\min(k, y - x \bmod y)$ and then divide x by y as much as possible
- Once we reach a point where $x < y$, we can instantly compute the answer as follows:

$$x_{\text{answer}} = \begin{cases} x, & \text{if } k \bmod (y-1) = 0 \\ 1 + (x + k) \bmod y, & \text{if } x + (k \bmod (y-1)) \geq y \\ x + k, & \text{otherwise} \end{cases}$$

Implementation Problems (Example)

We can implement this in code as such:

Implementation Problems (Example)

We can implement this in code as such:

```
int t = 1;
cin >> t;
for (int i = 0; i < t; i++) {
    int x, y, k;
    cin >> x >> y >> k;
    while (x >= y and k > 0) {
        int dx = min(k, y - x % y);
        x += dx;
        k -= dx;
        while (x % y == 0) x /= y;
    }
    k %= y - 1;
    if (k == 0) cout << x << endl;
    else if (x + k >= y) cout << 1 + (x + k) % y << endl;
    else cout << x + k << endl;
}
```

Implementation Problems (Example)

What is the new time complexity of our improved solution?

Implementation Problems (Example)

What is the new time complexity of our improved solution?

- Because it will take $\log_y(x)$ divisions for to reach new values of x where $x < y$, the time complexity is $O(t \log_y(x))$

Implementation Problems (Example)

What is the new time complexity of our improved solution?

- Because it will take $\log_y(x)$ divisions for to reach new values of x where $x < y$, the time complexity is $O(t \log_y(x))$
- This will be fast enough given our constraints!

Other Notes

- If you find a problem too difficult, try using the brute-force approach and see if you can find any key observations! For some contests (like NOI), using brute-force solutions to get partial points can help greatly!

Other Notes

- If you find a problem too difficult, try using the brute-force approach and see if you can find any key observations! For some contests (like NOI), using brute-force solutions to get partial points can help greatly!
- Find ways to cleanly implement the algorithm being described. Use every data structure at hand. You can even switch programming languages if you think it'll be easier to implement in other languages.

Homework :3

As always, check the Reboot website for homework problems :D