South African Computer Olympiad
Third Round 2008
Day 1

Overview

Author(s)	Marco Gallotta and Mark Danoher	Richard Baxter and Julian Kenwood	Harry Wiggins and Keegan Carruthers-Smith
Problem	**cleaning**	**hanoi**	**grandma**
Source	cleaning.java cleaning.py cleaning.c cleaning.cpp cleaning.pas	hanoi.java hanoi.py hanoi.c hanoi.cpp hanoi.pas	grandma.java grandma.py grandma.c grandma.cpp grandma.pas
Input file	stdin	stdin	stdin
Output file	stdout	stdout	stdout
Time limit	1 second	1 second	1 second
Number of tests	20	10	10
Points per test	5	10	10
Detailed feedback	Yes	No	Yes
Total points	**100**	**100**	**100**

The maximum total score is 300 points.

Spring Cleaning

Marco Gallotta and Mark Danoher

Introduction

Fred the manic storekeeper has a large storeroom filled with piles of boxes of explosives that he has been unable to sell. Spring has come and Fred needs to clear out his storeroom so he has space for the next popular product. He will do this by detonating the boxes and he wants you to help him minimise the cost to do so.

Task

Due to the odd shape of his storeroom, Fred has stored the boxes in N piles, which have been arranged in a single straight line. Each time a box is detonated, it is destroyed and the explosion spreads sideways, destroying the top box of any neighbouring piles *of the same height* (all boxes are of identical size and shape). The explosion continues spreading in both directions, until it reaches a pile of a different height. For safety reasons, only the top box of a pile can be exploded (either by being detonated or by its neighbour exploding).

Fred will only consider his storeroom clear when all boxes have been destroyed. He would like you to calculate the minimum number of detonations required to achieve this so that he can budget accordingly. Fred isn't good at working with large numbers so he would like you to give him only the last six digits of the number of detonations, i.e. the remainder after division by 1 000 000.

Example

Suppose there are four piles, containing 1, 3, 2 and 5 boxes respectively. One way to destroy all the boxes in the minimum number of detonations would be:

- Detonate the fourth pile three times, destroying the top three boxes. (three detonations)
- Detonate the second pile once, destroying the top box. (one detonation)
- Detonate the fourth pile once, destroying the top box of the fourth pile and then the top boxes of the third and second piles. (one detonation)
- Detonate any pile, destroying the last box in that pile and then the last box in all of the other piles. (one detonation)

This would clear out the storeroom in six detonations.

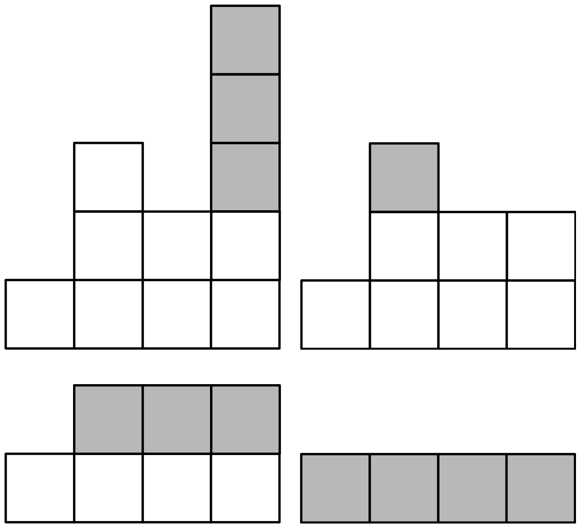

Figure 1: *The sequence of detonations. The grey boxes are to be exploded next. Note that the first step requires three detonations.*

Input (stdin)

The first line of input contains a single integer, N, the number of piles. The next N lines each contain a single integer, H_i, the number of boxes in the i^{th} pile.

Sample input

```
4
1
3
2
5
```

Output (stdout)

Output a single integer: the remainder when the minimum number of detonations is divided by 1 000 000.

Sample output

```
6
```

Constraints

- $1 \leq N \leq 750\,000$
- $0 \leq H_i \leq 1\,000\,000$ for all i

Additionally, in 70% of the test cases:

- $N \leq 200\,000$

Additionally, in 25% of the test cases:

- $N \leq 1\,000$
- $H_i \leq 1\,000$ for all i

Time limit

1 second. Python: 10 seconds.

Detailed feedback

Detailed feedback is enabled for this problem.

Scoring

A correct solution will score 100%, while an incorrect solution will score 0%.

Hanoi Reconstruction

Richard Baxter and Julian Kenwood

Introduction

The legend of the Towers of Hanoi says that a group of monks was given three pegs. The first peg had 64 discs, of different sizes, stacked from largest to smallest. The monks had to move all the discs to another peg, one at a time. However, they could only move a disc either onto an empty peg or onto a larger disc.

The optimal recursive solution to the puzzle is:

```
hanoi(NUM, FROM, HELPER, TO):
    if NUM == 1:
        Move disc from peg FROM to peg TO
    else:
        hanoi(NUM-1, FROM,   TO,     HELPER)
        hanoi(1,     FROM,   HELPER, TO)
        hanoi(NUM-1, HELPER, FROM,   TO)
hanoi(N, 1, 2, 3)
```

Task

The Guji tribe are attempting to solve a smaller version of the puzzle with N discs, *using the above solution*. After many long days of solving the puzzle, they took a well-deserved rest.

Upon returning to the puzzle, they discovered that they had been raided by their enemy, the Burji tribe, who moved all the discs back to the first peg. They really did not feel like starting all over again, but could not remember which discs were on which pegs. The only information they remembered was, T, the number of moves they had made, where a "move" is the act of taking the upper disk from one of the pegs and putting it onto another peg. They have asked you to determine on which peg each disc was before they took their rest, using N and T provided.

Example

The Guji tribe had made four moves ($T = 4$) in the puzzle with three discs before they were raided. Working from the start, with Disc 1 the largest, you determine the four moves were:

1. Move Disc 3 onto Peg 3
2. Move Disc 2 onto Peg 2

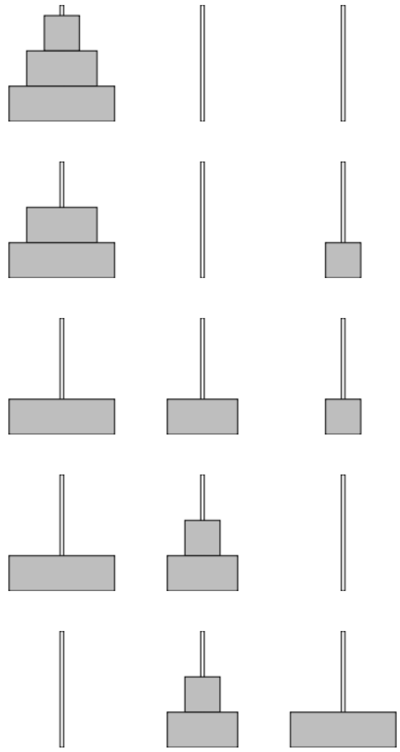

Figure 2: *The initial configuration and first four moves of the puzzle with $N = 3$ discs. The leftmost peg is Peg 1.*

3. Move Disc 3 onto Peg 2
4. Move Disc 1 onto Peg 3

The three discs were therefore on pegs 3, 2 and 2 respectively. This example is illustrated in Figure 2.

Input (stdin)

The input contains two space-separated integers, N and T.

Sample input

```
3 4
```

Output (stdout)

Output N lines, with line i containing a single integer representing the current peg on which the i^{th} disc is on (where disc 1 is the largest and N the smallest). Each number should be one of:

- 1, if the disc is on the first peg
- 2, if the disc is on the middle (helper) peg

- 3, if the disc is on the last peg

Sample output

```
3
2
2
```

Constraints

- $1 \leq N \leq 30$
- $0 \leq T < 2^N$

Additionally, in 50% of the test cases:

- $1 \leq N \leq 16$

Time limit

1 second. Python: 10 seconds.

Scoring

A correct solution will score 100%, while an incorrect solution will score 0%.

Visiting Grandma

Harry Wiggins and Keegan Carruthers-Smith

Introduction

Bruce plans to visit his beloved grandmother whom he has not seen in ages. The petrol price is high and the roads are dangerous, so he wants to travel the shortest possible distance. He also wants to know how much choice he has for the route, as he wants to visit his grandmother more often and being Bruce, he loves change.

Bruce wants to surprise his grandmother with a box of cookies every time he visits. Bruce does not know how to make real cookies (digital ones are easy, however!), so he will have to go past a cookie store along the way, possibly increasing the length of the shortest distance.

Task

Bruce will start off in his home town, numbered 1, and his grandmother lives in town N. Given the distance between each town and the location of the cookie stores, your task is to determine the length of the shortest route and the number of routes having that length. All routes must visit a town with a cookie store. Bruce is only interested in the last six digits of the number of routes, i.e. the remainder after division by 1 000 000.

Example

In the example input there are 5 towns, with Towns 2 and 4 containing cookie stores. Bruce, who lives in Town 1, wants to visit his grandma in Town 5. There are four routes having a distance of 3:

1. Bruce goes from Town $1 \to 2 \to 5$ (purchasing cookies at 2)

2. Bruce goes from Town $1 \to 4 \to 1 \to 5$ (purchasing cookies at 4)

3. Bruce goes from Town $1 \to 4 \to 5$ (purchasing cookies at 4)

4. Bruce goes from Town $1 \to 5 \to 2 \to 5$ (purchasing cookies at 2)

Notice that travelling from Town $1 \to 5$ directly has a distance of only 1. However, Bruce would then arrive at his grandma without any cookies!

Input (stdin)

The first line contains a single integer, N, representing the number of towns (numbered 1 to N). The next N lines each contain N space-separated integers. The j^{th} integer on the i^{th} line, d_{ij}, is the distance between town i and town j. Following this is a line containing a single integer M, the number of cookie stores. The last line contains M space-separated integers, each representing a town t_j that has a cookie store.

Sample input

```
5
0 2 2 1 1
2 0 1 2 1
2 1 0 2 1
1 2 2 0 2
1 1 1 2 0
2
2 4
```

Output (stdout)

Output two space-separated integers,

- the length of the shortest route, followed by
- the remainder when the number of route having this length is divided by 1 000 000.

Sample output

```
3 4
```

Constraints

- $1 \leq N \leq 700$
- $1 \leq M, t_j \leq N - 1$
- $1 \leq d_{ij} \leq 1\,000$ for all $i \neq j$
- $d_{ii} = 0$ (i.e. distance from a town to itself is zero)
- $d_{ij} = d_{ji}$ (i.e. distance from town i to town j is the same as the distance from town j to town i)

Additionally, in 30% of the test cases:

- $1 \leq N \leq 10$

Detailed feedback

Detailed feedback is enabled for this problem.

Time limit

1 second. Python: 10 seconds.

Scoring

A solution which correctly outputs the shortest route length and number of shortest routes will score 100% for the test run. If only the shortest route length is correct, you will score 50% for the test run. Otherwise you will score 0% for the test run.