



# South African Computer Olympiad

## Third Round 2009

### Day 2



## Overview

Problem	trucount	cunning	tfred
Source	trucount.java trucount.py trucount.c trucount.cpp	cunning.java cunning.py cunning.c cunning.cpp	tfred.java tfred.py tfred.c tfred.cpp
Input file	stdin	stdin	stdin
Output file	stdout	stdout	stdout
Time limit	0.5 seconds	2 seconds	0.2 seconds
Memory limit	128MiB	128MiB	128MiB
Number of tests	10	10	10
Points per test	10	10	10
Detailed feedback	Yes	Yes	No
<b>Total points</b>	<b>100</b>	<b>100</b>	<b>100</b>

The maximum total score is 300 points.

<http://olympiad.cs.uct.ac.za/contest.html>



# South African Computer Olympiad

## Third Round 2009

### Day 2



## Counting Truths

### Introduction

You are a prisoner of the Evil Logicians, whose tricky riddles have become infamous around the world.

You are guarded by a number of Logician Guards, and they have given you a single chance to escape. Each guard will make a statement, which may or may not be true. If you work out the minimum number which are true, the guards will let you go.

### Task

There are  $N$  guards.

Guard  $i$  will make a statement of the form “At least  $K_i$  guards (including myself) are telling the truth.”

At least one guard will be telling the truth. You must determine the minimum number of guards who could be telling the truth.

### Example

Guards 1 and 2 say, “At least 1 guard is telling the truth.”  
Guard 3 says, “At least 3 guards are telling the truth.” At least two guards must be telling the truth: specifically, we can determine that it is guards 1 and 2 that must be telling the truth, but we cannot be sure about the truthfulness of guard 3.

### Input (stdin)

The first line of input contains an integer,  $N$ . The next  $N$  lines each contain  $K_i$  for  $i = 1 \dots N$ .

### Sample input

```
3
1
1
3
```

### Output (stdout)

The output consists of a single line containing a single integer: the smallest number of guards who must be telling the truth.

### Sample output

```
2
```

### Constraints

- $1 \leq N \leq 100\,000$
- $1 \leq K_i \leq N$

Additionally, in 50% of the test cases:

- $1 \leq N \leq 300$

### Time limit

0.5 seconds. Python multiplier: 10.

### Detailed feedback

Detailed feedback is enabled for this problem.

### Scoring

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



# South African Computer Olympiad

## Third Round 2009

### Day 2



## As Cunning as a Fox

### Introduction

Bruce and Carl are once again trying to outdo each other. This time, they are having a bet to see who can predict the outcome of their latest trivial game, Fox Wars. Help Bruce become the ultimate victor.

In this game there is a  $N \times N \times N$  cube formed by stacking cages on top of each other. The cages are numbered from  $(1, 1, 1)$  to  $(N, N, N)$ . Within each cage there is a fox which wants to get to the treats in cage  $(1, 1, 1)$ , but, as these foxes are not ordinary foxes, this could be a slight problem.

Each fox takes a certain number of turns to get to the exit. If there are multiple paths, they will always take the path with the smallest number of jumps. A fox can jump in the *positive or negative*  $x, y$  or  $z$  direction by any power of 2 ( $\dots, -8, -4, -2, -1, 1, 2, 4, 8, \dots$ ) but no other way.

A cage may contain any number of foxes at the same time. Foxes may not leave the cube of cages.

Note: No foxes were harmed in the setting of this problem.

### Task

Write a program to calculate  $T$ : the total number of jumps required by all foxes to get to cage  $(1, 1, 1)$ . In addition, you must determine the number of jumps required for several individual foxes to get to cage  $(1, 1, 1)$ .

### Example

We have a  $3 \times 3 \times 3$  cube. The fox at  $(2, 1, 2)$  can jump to  $(1, 1, 2)$  and then  $(1, 1, 1)$ ; from  $(2, 3, 3)$  the fox jumps to  $(1, 3, 3)$ ,  $(1, 1, 3)$  and then  $(1, 1, 1)$ . The fox at  $(1, 1, 1)$  is already at the treats. The fox at  $(3, 3, 3)$  jumps to  $(1, 3, 3)$ ,  $(1, 1, 3)$  and then  $(1, 1, 1)$ . There are other routes which take the same number of jumps.

The total number of jumps for all 9 foxes is 54.

### Input (stdin)

The first line of the input contains two space-separated integers:  $N$  and  $K$ . The next  $K$  lines each contain three space-separated integers:  $a, b$  and  $c$ .

### Sample input

```
3 4
2 1 2
2 3 3
1 1 1
3 3 3
```

### Output (stdout)

The first line of output should contain a single integer  $T$ , the number of jumps taken by all foxes in total. The next  $K$  lines each contain the number of jumps made by the fox starting in cage  $(a, b, c)$  as specified by the  $K$ -th query.

### Sample output

```
54
2
3
0
3
```

### Constraints

- $1 \leq a, b, c \leq N \leq 10\,000$
- $0 \leq K \leq 10\,000$

Additionally, in 50% of the test cases  $N$  will be smaller than 100.

### Time limit

2 seconds. Python multiplier: 10.

### Scoring

The sum and queries will be marked separately, the sum being worth 8 points and the queries being worth 2. A correct value for  $T$  will score you 8. If  $P$  of the queries are answered correctly you will score  $\lfloor 2 \frac{P}{K} \rfloor$  for the queries.



# South African Computer Olympiad

## Third Round 2009

### Day 2



## Travelling Fred

### Introduction

After many years of assistance from the SACO contestants, Fred the Manic Storekeeper has amassed a small fortune. He has decided to charter a small plane from Bob's Eccentric Airlines to visit cities he has never been to before.

Bob does not fly directly to every city, so Fred may have to visit auxiliary cities along the way.

### Task

Fred is given a list of  $F$  possible *one-way* flights between  $C$  cities. He wants to visit the first  $T$  cities (numbered  $0, 1, 2, \dots, T-1$ ). Your task is to minimise the number of times Fred travels between cities.

Fred lives in city 0, so his trip must start and end at city 0. Fred may also visit a city more than once.

It is guaranteed that you can always find a path which starts and ends at city 0, and visits cities  $0, 1, 2, \dots, T-1$ .

### Example

In the sample input there are 7 cities, but Fred only wants to visit cities 0, 1, 2 and 3. If he visits

$$0 \rightarrow 4 \rightarrow 3 \rightarrow 6 \rightarrow 2 \rightarrow 6 \rightarrow 1 \rightarrow 0$$

he only travels 7 times. This is the shortest possible route which visits cities 0 to 3.

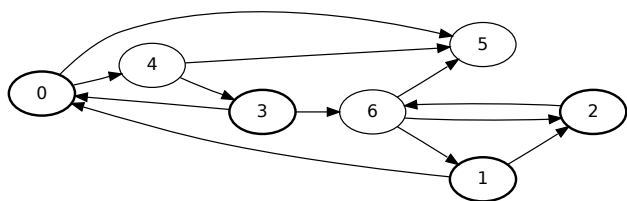


Figure 1: *The sample input. The circles with numbers represent cities. The arrows represent a flight path from one city to another.*

### Input (stdin)

The first line of input contains three space-separated integers:  $C$ ,  $T$  and  $F$ . The next  $F$  lines contain two space-

separated integers:  $a$  and  $b$ . This represents a flight from city  $a$  to city  $b$ .

### Sample input

```
7 4 12
0 5
0 4
1 0
1 2
2 6
3 0
3 6
4 3
4 5
6 1
6 2
6 5
```

### Output (stdout)

The output should contain a single integer: the smallest number of times Fred needs to travel between cities in order to visit cities  $0, 1, 2, \dots, T$ .

### Sample output

```
7
```

### Constraints

- $3 \leq T \leq 8$
- $T \leq C \leq 5\,000$
- $C \leq F \leq 100\,000$

Additionally, in 60% of the test cases:

- $C \leq 80$

Additionally, in 30% of the test cases:

- $T \leq 5$
- $C \leq 10$

### Time limit

0.2 seconds. Python multiplier: 10.

### Scoring

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