第12届山东大学ACM/ICPC校赛

高年级组

Problem Set

April 14th, 2018

Contents

0 注意事项 **2**

1 Problem A: Aviation trouble **3**

2 Problem B: Boy next door **5**

3 Problem C: Contest Arrangement **6**

4 Problem D: Dijkstra **7**

5 Problem E: EnTrance **8**

6 Problem F: Funky Number **9**

7 Problem G: Gallery Exhibition **10**

8 Problem H: Hogs **11**

9 Problem I:Indifferent transformation **12**

10 Problem J: Je ne sais quoi **13**

11 Problem K: Kennel Sequence **14**

0 注意事项

- 若对题目描述有任何疑问，请通过使用OJ的讨论功能提问。若设备出现任何问题，请举手咨询现场工作人员。
- 任何妨碍比赛或评测进行的行为均可能导致取消资格。
- 各语言的编译器版本以及所用的编译命令为：
  - C(gcc 7.3.0-13ubuntu1):
    `gcc <src> -o <exec> -fno-asm -Wall -lm -std=c99 -DONLINE_JUDGE -O2`
  - C++(g++ 7.3.0-13ubuntu1):
    `g++ <src> -o <exec> -fno-asm -Wall -lm -std=c++17 -DONLINE_JUDGE -O2`
  - Java(OpenJDK 1.8.0_161):
    `javac Main.java`
- 对于C/C++选手，使用printf输出64位整型时应使用的格式为`"%lld"`或`"%llu"`。
- 某些题目数据量较大，请考虑使用快速IO。
- Java用快速IO：
  将以下内容放在公共类中：

```
static StreamTokenizer in = new StreamTokenizer(new BufferedReader(new
                                InputStreamReader(System.in)));
static PrintWriter out = new PrintWriter(new OutputStreamWriter(System.out));
static int nextInt() throws IOException {
        in.nextToken();
        return (int) in.nval;
}
```

  使用示例：

```
int n=nextInt();
out.println(n);
out.flush();
```

  其中`out.flush();`在所有输出进行完毕后调用一次即可。同时注意所有调用`nextInt()`的方法在其说明后都要加上`throws IOException`。

1 Problem A: Aviation trouble

Time Limit: 1000ms

Description

The Troublesome Airline is recently suffering from balance issues: not about currency balance, but balancing the aircraft. So they came up with a new seat assigning algorithm, trying to make the aircraft as balanced as possible. However the algorithm is so complicated that no programmer in the airline is willing to implement it.

The aircraft has $r+3$ rows, 3 of which are used for access to the exit thus not containing any seat. The exit rows are located at the 1st row, the $(r+3)/2+1$-th row and the $r+3$-th row. Each of the remaining r rows contains 3 groups of 3 seats, separated by passenger aisles. Therefore a seat row can be represented in the following form: `ABC.DEF.GHI`. The algorithm is described as follows:

1. If an empty seat exists directly after an exit row, all other rows are ignored in the second step. (Selected row in the second step is always directly behind an exit row in this case.)
2. Select a seat row with the largest number of empty rows. If multiple candidates exist, select the one closest to an exit row. If there are still multiple such rows, select the one with the lowest number.
3. Select an empty seat within the selected row. The seat with the highest priority is selected. The priority of a seat is defined as follows: `[D,F]>[C,G]>[A,I]>E>[B,H]`, using the representation above.
4. If two empty seats with the same highest priority exist, consider the balance of the **entire** aircraft. Left side seats consists of all seats with letters `ABCD`. Right side seats consists of all seats with letters `FGHI`. The empty seat in the side with more empty seats is selected. If both sides have the same number of empty seats, we select the seat in the left side.

Implement the algorithm for Troublesome Airline and you might get hired!

Note: some seats may have already been assigned, possibly not using the algorithm above. These seats are shown as "occupied" in the input. You have to assign seats for the next n passengers using this algorithm.

Input Format

The first line consists of a single integer T denoting the number of test cases.

For each test case, the first line contains two integer r and n, denoting the number of seat rows and passenger purchasing tickets.

Each of the following $r+3$ lines contains 11 characters. The i-th line describes the initial layout of the i-th row: `'.'` denotes aisles or exit rows; `'-'` denotes empty seats and `'#'` denotes an occupied seat.

Output Format

For each test case, print $r+3$ lines using the same representation as the input, replacing newly assigned seats (`'-'`) with letters. The i-th passenger is represented using the i-th lowercase letter.

Print a blank line between two test cases.

Sample Input

```
2
2 17
...........
---.#--.---
...........
---.---.---
...........
6 26
...........
---.---.###
#-#.---.---
---.###.---
...........
```

```
---.###.---
#--.#-#.--#
#--.--#.#-#
...........
```

Output for Sample Input

```
...........
hnd.#lb.fpj
...........
kqg.cma.eoi
...........

...........
gke.aic.###
#-#.mzo.r-v
x-p.###.n-t
...........
fjb.###.dlh
#-s.#-#.w-#
#-u.qy#.#-#
...........
```

Constraints

$2 \leq r \leq 50, r \equiv 0 \pmod 2$, $1 \leq n \leq 26$
You may assume that there are at least n empty seats in the initial layout.

2 Problem B: Boy next door

Time Limit: 1000ms

Description

Your gang is holding a secret conference in a bunker. All of a sudden you got the information that a hostile clan is nearby. A conflict is almost sure to outbreak.

You got the exact number of members in both clans. The strength of a conflict is said to be the sum of the number of members in the two clans if they are equal (such conflict is called "balanced"), otherwise it is an "imbalanced" conflict, of which the strength is twice the number of members in the larger clan.

As a member of the gang, you decide to write a program to calculate the strength of a conflict to flatter the boss.

Input Format

The first line consists of a single integer T denoting the number of test cases.

Each of the following T lines contains two integers a and b, denoting the number of members of the two conflicting clans.

Output Format

For each test case, print one line containing the type of the conflict and its strength, separated by a space. The type of conflict is either `balanced` or `imbalanced`.

Sample Input

```
2
1 1
999 9
```

Output for Sample Input

```
balanced 2
imbalanced 1998
```

Constraints

$T \leq 100$
$1 \leq a, b < 2^{30}$

3 Problem C: Contest Arrangement

Time Limit: 3000ms

Description

You are arranging a programming contest. The jury has come up with N problems to be included in the contest.

Unlike conventional programming contests though, you decided that problems do not have to be arranged in a list. You can arrange them in the shape of a tree, that is, $N-1$ pairs of problem has adjacency relationship. You also find out that if problem a and problem b are adjacent in the problem tree, they contribute $E_{a,b}$ points to the contestants' enjoyment of the contest.

However, considering the fact that an easy problem followed by another easy problem may ruin the contest, you decide to limit the number of such *bad arrangements*. You want to maximize the enjoyment of the contestants, while keeping the number of bad arrangements used no more than K.

Input Format

The first line consists of a single integer T denoting the number of test cases.

For each test case, the first line contains 3 integers N, M and K, N and K meaning as above.

Each of the following M lines describes a possible arrangement with 4 integers: a, b, $E_{a,b}$, $GOOD$. If this arrangement is a *bad arrangement*, $GOOD$ equals to 0. Otherwise it equals to 1.

Output Format

For each test case, print one line with a single integer denoting the maximum enjoyment of the contestants.

Sample Input

```
3 3 1
1 2 1 1
1 3 2 0
2 3 2 0
```

Output for Sample Input

```
3
```

Constraints

$T \leq 5$
$2 \leq N \leq 1000$, $N-1 \leq M \leq \frac{N \times (N-1)}{2}$
$1 \leq E_{a,b} \leq 1000$

4 Problem D: Dijkstra

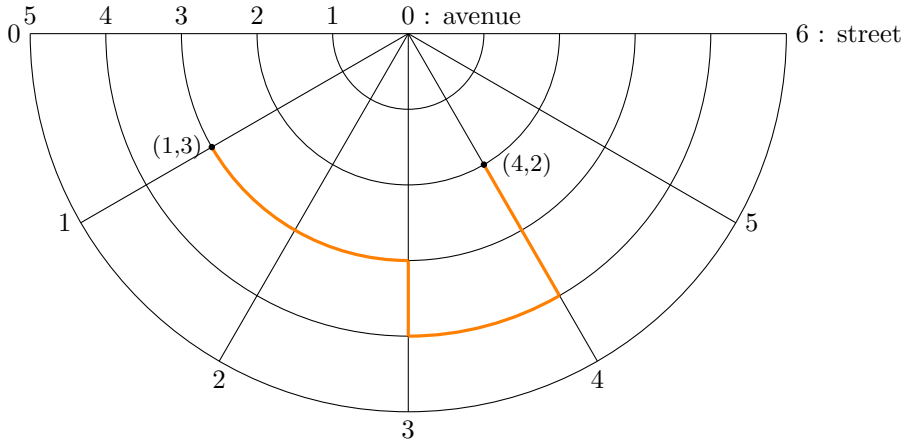
Time Limit: 1000ms

Description

There is a city named after Edsger W. Dijkstra in Byteland. The residents there are all born with mastery of Dijkstra's shortest path algorithm so they always use the shortest route when they commute.

The company you work for is working on a game called "Byteland Simulator". Dijkstra city is of course one of the elements of the game. You are given the work to simulate a Dijkstra-ian's behavior. Unfortunately, you don't understand Dijkstra's great algorithm very well.

The shape of Dijkstra city resembles a half disc. Streets radiate from the center while avenues run along the arcs. The figure below gives you an intuitive example.



An example of Dijkstra city, with coordinates and a route planned by a dumb

Dijkstra-ians use a $(street, avenue)$ denotion for coordinates. Given a map of dijkstra city and the coordinates of a Dijkstra-ian's commute endpoints, find the euclidean distance the Dijkstra-ian has to walk.

Input Format

The first line consists of a single integer T denoting the number of test cases.

For each test case, the first line contains 2 integers M and N, denoting the maximum index of streets and avenues respectively, and a floating point number R, which is the radius of the half disc. The second line contains 4 integers a_x a_y b_x b_y, denoting the endpoints of the commute.

Output Format

For each test case, print one line containing a single floating point number. Your answer will be accepted if its absolute or relative error does not exceed 10^{-6}.

Sample Input

```
2
6 5 2
1 3 4 2
9 9 9
1 0 8 0
```

Output for Sample Input

```
1.65663706143592
0
```

Constraints

$T \leq 1000$

$1 \leq N, M \leq 100$

$0 \leq a_x, b_x \leq M$, $0 \leq a_y, b_y \leq N$

Although Dijkstra-ians master all Dijkstra's algorithms, they can't fly. So they may only move along streets and avenues.

5 Problem E: EnTrance

Time Limit: 3000ms

Description

You got addicted to a game called "EnTrance" recently. The game is played in a $N \times M$ grid, where each cell may contain an arrow pointing to one of the four directions, or nothing at all. You start in a cell with an arrow in it. Then you jump to the first cell with an arrow in the direction of the arrow in the current cell (if there's no cell with an arrow in that direction, you just jump out of the grid). Each time you leave a cell, you score a point, taking the arrow in that cell (thus there would be no arrow left in the cell after that). The game terminates when you jump out of the grid. Given an initial grid, you are interested in the highest possible score you can obtain in the grid and the number of ways to get it.

Input Format

The first line consists of a single integer T denoting the number of test cases.

For each test case, the first line contains two integers N and M. The initial grid follows. Each of the following N lines is a string of length M decribing the grid. Each character of the string is one of the following: `L` for left, `R` for right, `U` for up, `D` for down, `.` for nothing.

Output Format

For each test case, print one line with two integers denoting the hightest possible score and the number of ways to get it.

Sample Input

```
1
4 4
DRLD
U.UL
.UUR
RDDL
```

Output for Sample Input

```
10 1
```

Constraints

$T \leq 20$, $1 \leq N \times M \leq 5000$

6 Problem F: Funky Number

Time Limit: 1000ms

Description

Let f be a function applied on any natural number defined as follows:

$f(n)$ equals to the number obtained by summing up the squares of the digits of n in decimal.

For instance, $f(35) = 3^2 + 5^2 = 34$.

Soon you found out that applying the function repeatly on some initial number, the result may eventually become 1. These numbers are called "Funky numbers". For example:

$$f(91) = 9^2 + 1^2 = 82$$
$$f(82) = 8^2 + 2^2 = 68$$
$$f(68) = 6^2 + 8^2 = 100$$
$$f(100) = 1^2 + 0^2 + 0^2 = 1$$

Give a natural number, determine whether the number is funky or not. If it is, find the times the function has been applied before it become 1.

Input Format

The first line consists of a single integer T denoting the number of test cases.

For each test case, there's only a line containing a single natural number.

Output Format

For each test case, print the number of times applying the function before the number became 1. If the number is not a funky number, print `"INF"` instead (quotes excluded).

Sample Input

```
2
19
24
```

Output for Sample Input

```
4
INF
```

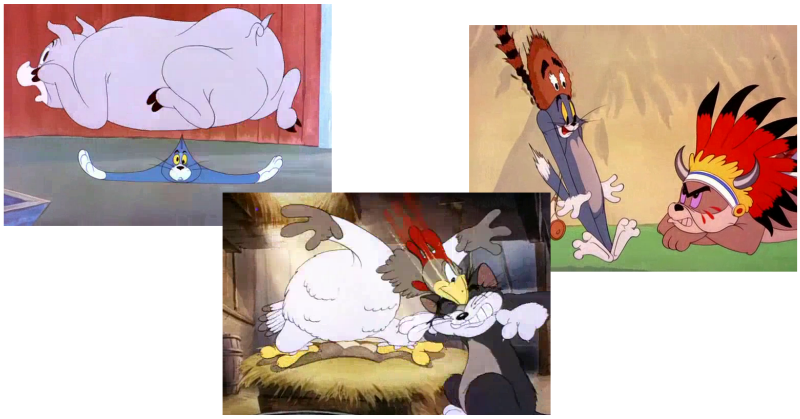
Constraints

The given number does not exceed 10^9.

7 Problem G: Gallery Exhibition

Time Limit: 1000ms

Description



Example of fine art from the ACM Lab

The ACM Lab is going to hold an exhibition of the collection of fine art created by pigs, dogs and hens. There are N pieces of artwork, which will be displayed in two exhibition sites. We've decided to hold multiple exhibitions so that **every pair of pieces would have been displayed in different sites**. To be more specific, for every exhibition we split the artwork into two groups. After a couple of exhibitions, every pair of art should have been in different groups at least once. As holding an exhibition costs, and has the risk of damaging the artwork, we want to hold the least number of exhibitions while still achieving the goal. Can you help determine the number of exhibitions we should hold?

Input Format

The first line consists of a single integer T, denoting the number of test cases.
For each test case, there's a single line containing a single integer N, meaning as above.

Output Format

For each test case, print a single integer, denoting the minimum number of exhibitions to be held.

Sample Input

```
1
2
```

Output for Sample Input

```
1
```

Constraints

$2 \leq n \leq 10^5$

8 Problem H: Hogs

Time Limit: 2000ms

Description

Hogs are domestic pigs raised to over 60 kg for slaughtering. Today N hogs are ready to be slaughtered, but they were allowed to play before going into the slaughter house.

Each hog has a initial number of tokens and a maximum number of tokens it can keep. A hog can pass its tokens to another hog, either topping the receiver's limit or emptying its own stock. As you found this extremely boring and can't wait to send them to the slaughter house, you want to simulate this process using a program.

Input Format

The first line of input consists of a single integer T denoting the number of test cases.

For each test case, the first line contains two integers N and M.

The second line contains N integers denoting the initial number of tokens each hog has.

The third line contains N integers denoting the maximum number of tokens each hog can keep.

The following M lines describes the sequence of operations, each containing a pair of integers u v, meaning hog $\#u$ try to pass tokens to hog $\#v$ using the rules above.

Output Format

For each test case, print a line with N integers denoting the final number of tokens each hog owns.

Sample Input

```
1
3 1
3 3 3
5 5 5
1 3
```

Output for Sample Input

```
1 3 5
```

Constraints

$T \leq 10$, $n, m \leq 10000$

Number of tokens each hog can keep does not exceed 10000.

9 Problem I:Indifferent transformation

Time Limit: 1000ms

Description

You have learned linear algebra long time ago but you have forgotten most part of it. One day you encountered some type of matrix transformation called the "indifferent transformation". The transformation is performed as follows: for each element in the matrix, the result value is the maximum difference between the original value of the current element and its (maximum 8) surrounding elements.

1	1	1	5	5
5	5	5	5	5
5	5	5	5	3
3	3	2	2	2
2	1	1	1	1

$\Rightarrow$

4	4	4	4	0
4	4	4	4	2
2	3	3	3	2
2	2	3	3	3
1	2	2	1	1

An example of the transformation

You thought the transformation is easy peasy, however soon you found out that this transformation is usually performed on a huge matrix. You want to implement it anyways.

Input Format

The first line of input consists of a single integer T denoting the number of test cases.

For each test case, the first line contains two integers M and C. M is the number of columns of the matrix.

Each of the following C lines contains two integers a and l. The next l elements of the matrix is filled with value a. The matrix is filled from left to right, from top to bottom. Once a row is completely filled, it just wraps around and starts filling the next row. The last row of the matrix is guaranteed to be filled completely.

Output Format

For each test case, print a single integer C_r in the first line. Then C_r lines follow, each consisting of two integers a and l. The matrix should be encoded using the same way as the input.

Sample Input

```
1
5 5
1 3
5 11
3 3
2 4
1 4
```

Output for Sample Input

```
10
4 4
0 1
4 4
2 2
3 3
2 3
3 3
1 1
2 2
1 2
```

Constraints

$T \leq 5$, $2 \leq M \leq 10^9$, $C \leq 2000$

The matrix contains no more than 2×10^9 elements. Values in the original matrix are within the range $[0, 256)$.

10 Problem J: Je ne sais quoi

Time Limit: 1000ms

Description

Once there were two dendrophiles. They love trees. However they like different kinds of trees: one likes trees with larger weight, the other likes the opposite. One day they saw a huge tree. The tree had a lot of nodes on it. They tried to optimize the tree according to their own preference by taking turns to chop the tree into two parts and throwing one part away. Soon they found out that eventually the tree would have only one node left. So they just want to maximize or minimize the weight of that node. The one who likes trees with larger weight took the first step. However the tree was so huge that they haven't finished chopping the wood yet. You, as a bystander, want to predict the final node left.

Input Format

The first line consists of a single integer T denoting the number of test cases.
For each test case, the first line contains a single integer N denoting the number of nodes in the tree.
The following line contains N integers $w_1..w_N$ denoting the weight of nodes.
$N-1$ lines follow. Each containing 2 integers x y, meaning that node x and y are adjacent.

Output Format

For each test case, print one line with a single integer denoting the weight of the only node left.

Sample Input

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

Output for Sample Input

```
3
```

Constraints

$T \leq 100$, $1 \leq N \leq 1000$, $w_i \leq 1000$

11 Problem K: Kennel Sequence

Time Limit: 10000ms

Description

Kennel is a messy place. A kennel sequence is a messy sequence. It consists of N positive integers between 1 and 10 $k_0..k_{N-1}$. For a given tuple a, b, c, it should satisfy:

$$\exists 0 \le x < y < z < w \le N, \text{ so that}$$
$$\sum_{i=x}^{y-1} k_i = a, \sum_{i=y}^{z-1} k_i = b, \sum_{i=z}^{w-1} k_i = c$$

Given N, a, b, c, find the number of kennel sequences of length N subjected to parameters (a, b, c). As the result might be very large, just print the answer modulo $10^9 + 7$.

Input Format

The first line consists of a single integer T denoting the number of test cases.
For each test case, there's a single line with 4 integers N, a, b, c, meaning as above.

Output Format

For each test case, print the answer in one line.

Sample Input

```
1
37 4 2 3
```

Output for Sample Input

```
863912418
```

Constraints

$3 \le N \le 40$, $1 \le a, b, c \le 6$