ACM International Collegiate Programming Contest, Asia-Amritapuri Site, 2011

Problem A: MAGRID

Thanks a lot for helping Harry Potter in finding the Sorcerer's Stone of Immortality in October. Did we not tell you that it was just an online game ? uhhh! now here is the real onsite task for Harry. You are given a magrid S (a magic grid) having R rows and C columns. Each cell in this magrid has either a Hungarian horntail dragon that our intrepid hero has to defeat, or a flask of magic potion that his teacher Snape has left for him. A dragon at a cell (i,j) takes away |S[i][j]| strength points from him, and a potion at a cell (i,j) increases Harry's strength by S[i][j]. If his strength drops to 0 or less at any point during his journey, Harry dies, and no magical stone can revive him.

Harry starts from the top-left corner cell (1,1) and the Sorcerer's Stone is in the bottom-right corner cell (R,C). From a cell (i,j), Harry can only move either one cell down or right i.e., to cell (i+1,j) or cell (i,j+1) and he can not move outside the magrid. Harry has used magic before starting his journey to determine which cell contains what, but lacks the basic simple mathematical skill to determine what minimum strength he needs to start with to collect the Sorcerer's Stone. Please help him once again.

Input (STDIN):

The first line contains the number of test cases T. T cases follow. Each test case consists of R C in the first line followed by the description of the grid in R lines, each containing C integers. Rows are numbered 1 to R from top to bottom and columns are numbered 1 to C from left to right. Cells with S[i][j] < 0 contain dragons, others contain magic potions.

Output (STDOUT):

Output T lines, one for each case containing the minimum strength Harry should start with from the cell (1,1) to have a positive strength through out his journey to the cell (R,C).

Constraints:

$1 \le T \le 5$
$2 \le R, C \le 500$
$-10^3 \le S[i][j] \le 10^3$
$S[1][1] = S[R][C] = 0$

Time Limit : 3 s
Memory Limit : 64 MB

Sample Input:

```
3
2 3
0 1 -3
1 -2 0
2 2
0 1
2 0
3 4
0 -2 -3 1
```

```
-1 4 0 -2
1 -2 -3 0
```

Sample Output:

```
2
1
2
```

Explanation:

Case 1 : If Harry starts with strength = 1 at cell (1,1), he cannot maintain a positive strength in any possible path. He needs at least strength = 2 initially.
Case 2 : Note that to start from (1,1) he needs at least strength = 1.

ACM International Collegiate Programming Contest, Asia-Amritapuri Site, 2011

Problem B: Save the Students!

Hogwarts is under attack by the Dark Lord, He-Who-Must-Not-Be-Named. To protect the students, Harry Potter must cast protective spells so that those who are protected by the spells cannot be attacked by the Dark Lord.

Harry has asked all the students to gather on the vast quidditch sports field so that he can cast his spells. The students are standing in a 2D plane at all grid points - these are the points (x,y) such that both x and y are integers (positive, negative or 0). Harry's spell can take the shapes of triangle, circle or square, and all who fall within that shape (including its boundaries) are protected.

Given the types of spells and the details regarding where Harry casts the spell, output the number of people saved by Harry's spells.

Input (STDIN):

The first line contains the number of test cases T. T test cases follow.
Each case contains an integer N on the first line, denoting the number of spells Harry casts. N lines follow, each containing the description of a spell.
If the ith spell is a triangle, then the line will be of the form "T x1 y1 x2 y2 x3 y3". Here, (x1,y1), (x2,y2) and (x3,y3) are the coordinates of the vertices of the triangle.
If the ith spell is a circle, then the line will be of the form "C x y r". Here, (x,y) is the center and r is the radius of the circle.
If the ith spell is a square, then the line will be of the form "S x y l". Here, (x,y) denotes the coordinates of the bottom-left corner of the square (the corner having the lowest x and y values) and l is the length of each side.

Output (STDOUT):

Output T lines, one for each test case, denoting the number of people Harry can save.

Constraints:

All numbers in the input are integers between 1 and 50, inclusive.
The areas of all geometric figures will be > 0.
Time Limit: 3 s
Memory Limit: 32 MBytes

Sample Input:

```
4
1
C 5 5 2
1
S 3 3 4
1
T 1 1 1 3 3 1
3
C 10 10 3
```

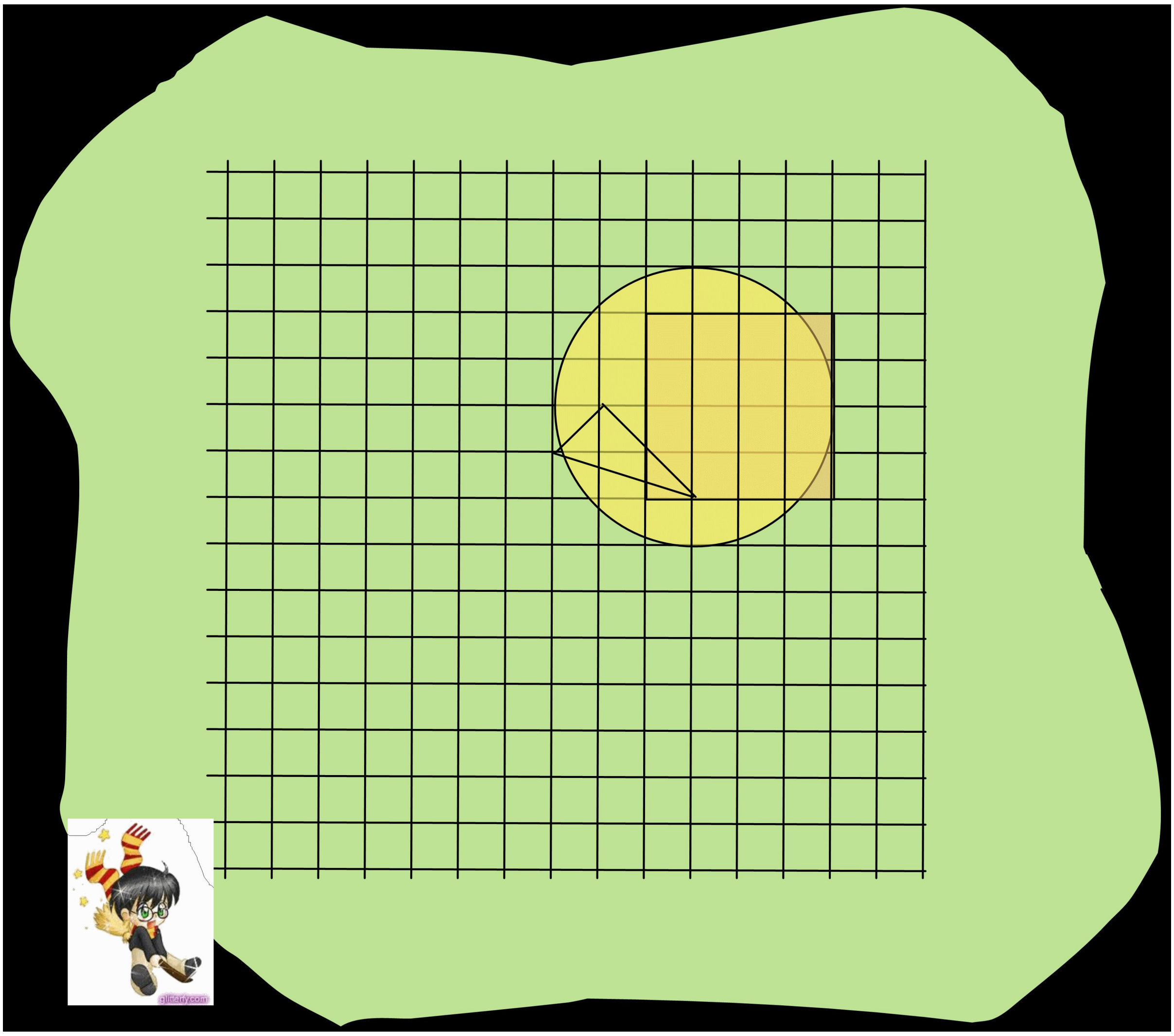
```
S 9 8 4
T 7 9 10 8 8 10
```

Sample Output:

```
13
25
6
34
```

Notes/Explanation of Sample Input:

Illustration of Testcase 4.



ACM International Collegiate Programming Contest, Asia-Amritapuri Site, 2011

Problem C: Robbing Gringotts

The Death Eaters are low on funds, and their leader Voldemort has asked them to get more money quickly, or face his wrath. They decide that the best way is to rob Gringotts Wizarding Bank.

By threatening one of the goblins in charge of the bank, the 'M' Death Eaters have discovered that Gringotts has 'N' vaults, with vault number 'i' containing X[i] gold items. The weights of the gold items in the ith vault are g[i][1],g[i][2],...,g[i][X[i]]. But as soon as a vault is robbed, the magical wards go off and alarm bells ring. Thus they have enough time to rob just one vault each, and all robberies have to take place at the same time. The Death Eaters have decided that no two of them will rob the same vault as this increases the chances of them being caught.

Death Eater j has a bag which can hold weight v[j]. They are a greedy and cowardly lot, so a Death Eater will only agree to rob a vault if he can fill up his bag completely to its capacity by taking some subset of the objects present in that vault. Note that it is not possible for a Death Eater to break any gold item; either it should be taken fully, or not be taken.

Find the maximum weight of gold they can take away by planning their strategy correctly.

Input (STDIN):

The first line contains the number of test cases T. T test cases follow. Each test case contains integers M and N on the first line. The next line contains M integers, the jth of which is v[j], which is the maximum weight of gold the jth Death Eater's bag can hold. The following N lines describe the gold items in the vaults. The ith line contains X[i], the number of gold items present in the ith vault, followed by X[i] integers g[i][1]..g[i][X[i]], denoting the weights of the each of the items present in the ith vault. There is a blank line after each test case.

Output (STDOUT):

Output one line for each test case, containing the maximum total weight of gold that the Death Eaters can rob.

Constraints:

1 <= T <= 10
1 <= N,M <= 50
1 <= X[i] <= 25
1 <= v[i],g[i][j] <= 10,000,000
Time Limit: 10 s
Memory Limit: 64 MB

Sample Input:

```
2
2 3
3 9
2 4 5
2 3 9
1 10

4 4
3 7 8 10
3 1 2 4
2 3 7
4 2 2 4 4
1 3
```

Sample Output:

```
12
28
```

ACM International Collegiate Programming Contest, Asia-Amritapuri Site, 2011

Problem D: Wizarding Duel

You are the organizer of a Wizarding Duel tournament at Hogwarts. N players participated in the tournament and each player played with every other player exactly once, and each such game resulted in a winner and a loser (no drawn games). It is now time to declare the results. Each player's score is the number of games the player won. However, you realize that something possibly went wrong with the scoring system, and the scores that you have noted down might be wrong. In fact, the scores you have noted down could simply not be possible. For example, suppose there are 3 players and the scores that you noted are 0,0 and 2. Clearly this is not possible as the game between the first two players must have had a winner and thus both the players cannot have score 0.

While you have no way to figure out the correct scores of each player, you've decided to set the scores right by adjusting the scores of some players. The new set of scores should be possible to have arisen in the tournament, otherwise there would be ample proof that the scoring is wrong. However, making drastic changes might cause suspicion. So you've decided to adjust the scores so that the sum of the absolute differences between the old and the new scores of each player is minimized. In other words, if the original scores which you noted are a1,..,aN, you must change them to the series of possible scores b1,...bN such that the sum |ai - bi| is minimized.

Input (STDIN):

The first line contains the number of test cases T. T test cases follow. Each case contains an integer N on the first line, followed by the set of scores which you have noted down: a1..aN.

Output (STDOUT):

Output T lines, one for each test case, containing the minimum sum of absolute values in order to make the scorecard a valid one.

Constraints:

1 <= T <= 20
2 <= N <= 50
0 <= ai <= 100
Time Limit: 3 seconds
Memory Limit: 64 MB

Sample Input:

```
2
3
0 0 2
5
5 3 2 1 4
```

Sample Output:

```
1
5
```

ACM International Collegiate Programming Contest, Asia-Amritapuri Site, 2011

Problem E: Distinct Primes

Arithmancy is Draco Malfoy's favorite subject, but what spoils it for him is that Hermione Granger is in his class, and she is better than him at it. Prime numbers are of mystical importance in Arithmancy, and Lucky Numbers even more so. Lucky Numbers are those positive integers that have at least three distinct prime factors; 30 and 42 are the first two. Malfoy's teacher has given them a positive integer *n,* and has asked them to find the *n*th lucky number. Malfoy would like to beat Hermione at this exercise, so although he is an evil git, please help him, just this once. After all, the know-it-all Hermione does need a lesson.

Input (STDIN):

The first line contains the number of test cases T. Each of the next T lines contains one integer n.

Output (STDOUT):

Output T lines, containing the corresponding lucky number for that test case.

Constraints:

1 <= T <= 20
1 <= n <= 1000
Time Limit: 2 s
Memory Limit: 32 MB

Sample Input:

```
2
1
2
```

Sample Output:

```
30
42
```

ACM International Collegiate Programming Contest, Asia-Amritapuri Site, 2011

Problem F: Magical Bridges

Hogwarts School of Witchcraft and Wizardry has a circular lane having N towers numbered 1 to **N**. Towers *i* and *i*+1 are adjacent to each other for $1 \leq i < N$ and also towers 1 and N are adjacent to each other. Each of these towers has exactly **F** number of floors, numbered 1,2,3,..,F-1,F from bottom to top. Floors *i* and *i*+1 in a tower are adjacent to each other and it takes one second to move between them. It also takes one second to move between floor 1 of a tower and floor 1 of its adjacent tower. Apart from these, there are **M** bridges designed for a quick escape in case of a Death Eater attack, each of which connects two floors of different towers. Each of these bridges is given as *bi fi bj fj t*, meaning it takes *t* seconds to move along this bridge that connects the floor *fi* of tower *bi* and the floor *fj* of tower *bj*. All ways are bidirectional.

Given *(qbi,qfi)* and *(qbj,qfj)*, find the minimum time in seconds it takes to reach floor qfj of tower qbj, starting from floor qfi of tower qbi. You have to answer a lot of such queries.

Input (STDIN):

The first line contains the number of test cases T. T cases follow. Each test case consists of N F M in the first line. N is the number of towers, F is the number of floors in each tower and M is the number of bridges. M lines follow, each of the form *bi fi bj fj t*, as mentioned in the problem statement. Next line contains Q, the number of queries and Q lines follow, each of the form *qbi qfi qbj qfj.*

Output (STDOUT):

For each query of the form *qbi qfi qbj qfj*, output one line denoting the minimum time in seconds it takes to reach the floor *qfj* of tower *qbj*, starting from the floor *qfi* of tower *qbi.*

Constraints:

$1 \leq T \leq 3$
$2 \leq N, M \leq 100$
$2 \leq F \leq 1{,}000{,}000$
$1 \leq t \leq 1{,}000{,}000$
$1 \leq Q \leq 100{,}000$
$1 \leq bi, bj, qbi, qbj \leq N$
$1 \leq fi, fj, qfi, qfj \leq F$

Time Limit : 5s
Memory Limit : 64 MB

Sample Input:

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

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

Sample Output:

```
4
5
4
6
1
```

ACM International Collegiate Programming Contest, Asia-Amritapuri Site, 2011

Problem G: Here Be Dragons!

The Triwizard Tournament's third task is to negotiate a corridor of many segments, and reach the other end. The corridor is N segments long. The ith segment is either empty or has a dragon. Harry cannot pass the dragon and will have no option but to retreat if he encounters one. Is it possible for him to reach the exit starting from the entrance?

Input (STDIN):

The first line contains the number of test cases T.
Each of the next T lines contains a string describing the corridor. The ith character is either a '.' if the segment is empty, or a 'D' if the segment contains a dragon.

Output (STDOUT):

Output T lines, each containing either the string "Possible" if you can reach the exit, and "You shall not pass!" if it is not possible to reach the exit.

Constraints:

1 <= T <= 50
1 <= N <= 50
Time Limit: 1 s
Memory Limit: 32 MB

Sample Input:

```
3
..
..D.
D..D
```

Sample Output:

```
Possible
You shall not pass!
You shall not pass!
```

ACM International Collegiate Programming Contest, Asia-Amritapuri Site, 2011

Problem H: Array Diversity

Enough with this Harry Potter, please! What are we, twelve-year olds? Let's get our teeth into some real pumpkin pasties -- oops, programming problems!

Here we go!
Let's define the diversity of a list of numbers to be the difference between the largest and smallest number in the list.
For example, the diversity of the list (1, -1, 2, 7) = 7 - (-1) = 8.

A substring of a list is considered a non-empty sequence of contiguous numbers from the list. For example, for the list (1,3,7), the substrings are (1), (3), (7), (1,3), (3,7), (1,3,7). A subsequence of a list is defined to be a non-empty sequence of numbers obtained by deleting some elements from the list. For example, for the list (1,3,7), the subsequences are (1), (3), (7), (1,3), (3,7), (1,7), (1,3,7).

Given a list of length N find the number of substrings and subsequences in this list with the maximum diversity. If a substring/subsequence having maximum diversity occurs multiple times in the list, each of its occurances adds towards the answer. And tell Harry Potter your answer.

Input (STDIN):

The first line contains T, the number of test cases. Then follow T test case blocks.
Each blocks starts with the first line containing the number N.
The second line contains a list of numbers in this list.

Output (STDOUT):

For each test case, output the number of substrings and the number of subsequences in this list with the maximum diversity.
Since the answers maybe very large, output them modulo 1000000007.

Constraints:

T <= 10
N <= 100,000
Each number in the list is between 1 and 100,000 inclusive.
Time Limit: 2 s
Memory Limit: 32 MB

Sample Input:

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

Sample Output:

```
1 2
3 6
1 2
```

ACM International Collegiate Programming Contest, Asia-Amritapuri Site, 2011

Problem I: Generations

Harry's friend Charlie Weasley has partnered with an old-time breeder of dragons in Romania. When Harry visited Charlie over the summer, Charlie showed him breeding records stretching back centuries. It had started out with just one female dragon named Abraxia that had then reproduced to give many generations of dragons. The breeding record showed a family tree of dragons, starting with Abraxia and showing each descendant* (only females' descendants were shown), the year each was hatched from its egg and the year each died. Only already dead dragons were included. Charlie pointed out that a dragon matured early, and once hatched, could herself lay and hatch an egg in the very year it was born. Dragon eggs did not need a mother's care to hatch -- the breeders simply used the warmth of a blowtorch to keep the egg warm and hatch it, sometimes long after the mother dragon was dead.
Harry noticed that sometimes many generations of dragons were alive at the same time. He was curious to figure out, for each dragon when it was alive, what was the maximum generational difference (absolute value) between it and its coeval** descendants. Can you help him? Assume that if a dragon dies the year another is hatched, they were coeval.

*A descendant is a child, grandchild, great-grandchild etc.
**coeval: Alive at the same time.

Input (STDIN):

The first line contains the number of test cases T.
The first line of each test case starts with an integer N - the number of dragons.
It is followed by N lines. The ith line (0 indexed) contains 3 integers, p_i, hatchyear_i and deathyear_i. p_i is the parent of ith dragon and its interval is hatchyear_i to deathyear_i. The dragon with index 0 is Abraxia, and a mother's index is smaller than all its descendants'.

Output (STDOUT):

Output T lines.
Each line contains a space-separated list of N integers, the ith of them denoting the required answer for the ith dragon. If the ith dragon's life does not overlap with any descendant's, output 0 for that dragon.

Constraints:

1 <= T <= 3
1 <= N <= 70000
0 <= hatchyear <= deathyear <= 10^9
p_0 = -1
For all i > 0, 0 <= p_i < i
hatchyear of dragon >= hatchyear of its mother

Time Limit: 3s
Memory Limit: 64 MB

Sample Input:

```
2
5
-1 0 10
0 1 5
0 2 8
0 2 5
3 10 12
9
-1 0 10
0 1 1
0 2 2
1 2 3
1 3 4
2 2 3
2 2 4
6 10 11
6 20 30
```

Sample Output:

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

ACM International Collegiate Programming Contest, Asia-Amritapuri Site, 2011

Problem J: Goblin Wars

The wizards and witches of Hogwarts School of Witchcraft found Prof. Binn's *History of Magic* lesson to be no less boring than you found your own history classes. Recently Binns has been droning on about Goblin wars, and which goblin civilization fought which group of centaurs where etc etc. The students of Hogwarts decided to use the new-fangled computer to figure out the outcome of all these wars instead of memorizing the results for their upcoming exams. Can you help them?

The magical world looks like a 2-D R*C grid. Initially there are many civilizations, each civilization occupying exactly one cell. A civilization is denoted by a lowercase letter in the grid. There are also certain cells that are uninhabitable (swamps, mountains, sinkholes etc.) - these cells are denoted by a '#' in the grid. All the other cells - to which the civilizations can move - are represented by a '.' in the grid.

A cell is said to be adjacent to another cell if they share the same edge - in other words, for a cell (x,y), cells (x-1, y), (x, y-1), (x+1, y), (x, y+1) are adjacent, provided they are within the boundaries of the grid. Every year each civilization will expand to all unoccupied adjacent cells. If it is already inhabited by some other civilization, it just leaves the cell alone. It is possible that two or more civilizations may move into an unoccupied cell at the same time - this will lead to a battle between the civilizations and the cell will be marked with a '*'. Note that the civilizations fighting in a particular cell do not try to expand from that cell, but will continue to expand from other cells, if possible.

Given the initial grid, output the final state of the grid after no further expansion by any civilization is possible.

Input (STDIN):

The first line contains T, the number of cases. This is followed by T test case blocks.
Each test case contains two integers, R, C.
This is followed by R lines containing a string of length C. The j-th letter in the i-th row describes the state of the cell in year 0.
Each cell is either a
1. '.' which represents an unoccupied cell
2. '#' which represents a cell that cannot be occupied
3. A civilization represented by a lowercase letter ('a' - 'z')

Output (STDOUT):

For each test case, print the final grid after no expansion is possible. Apart from the notations used in the input, use '*' to denote that a battle is being waged in that particular cell.
Print a blank line at the end of each case.

Constraints:

1 <= R, C <= 500
1 <= T <= 5

Time Limit: 3 s
Memory Limit: 64 MB

Sample Input:

```
5
3 5
#####
a...b
#####
3 4
####
a..b
####
3 3
#c#
a.b
#d#
3 3
#c#
...
a.b
3 5
.....
.#.#.
a...b
```

Sample Output:

```
#####
aa*bb
#####

####
aabb
####

#c#
a*b
#d#

#c#
acb
a*b

aa*bb
a#.#b
aa*bb
```