107 學年度
全國大專電腦軟體設計競賽
台大校內初賽

National Taiwan University

2018/09/29

Language	Version	Compile Flags	Extensions
C	gcc 5.4.0	`-g -O2 -std=gnu99 -static -lm`	`.c`
C++	g++ 5.4.0	`-g -O2 -std=gnu++11 -static -lm`	`.cc, .cpp`
Java	javac 1.8.0	`-encoding UTF-8 -sourcepath . -d .`	`.java`

Problem	Problem Name	Time Limit	Memory Limit
A	Clock	1 s	256 MB
B	Rectangle Donut	1 s	256 MB
C	Diff-prime Pairs	1 s	256 MB
D	Annoying Path	1 s	256 MB
E	Expected Number of Nodes	1 s	256 MB
F	Sum Of Digit	8 s	256 MB
G	Prefix Sum	3 s	256 MB
H	Steiner Tree	1 s	256 MB
I	Maximum Mode	1 s	256 MB
J	SCC	10 s	256 MB

This page is intentionally left blank.

A. Clock

Eddy bought a magic clock. There's only one hand on that clock.

The clock can be represented on the 2D coordinate system. The hand is fixed on the origin $(0, 0)$ and pointing to (X, Y). After each seconds, the hand will move clockwise by one degree.

Eddy is now wondering where will the hand point after T seconds.

Input

The first line of input contains an integer Q indicating the number of testcases. Following Q lines each contains three space-separated integers X, Y, T mentioned above.

- $1 \leq Q \leq 1000$
- $0 \leq |X|, |Y| \leq 10^9$
- $X^2 + Y^2 > 0$
- $0 \leq T \leq 10^9$

Output

For each testcase, output one line containing two space-separated number X', Y' indicating that the hand will point to (X', Y') after T seconds.

The answer will be considered correct if the relative or absolute error is less than 10^{-6}.

Sample Input 1	Sample Output 1
3 1 0 90 0 1 180 1 0 21	0.000000000 -1.000000000 0.000000000 -1.000000000 0.933580426 -0.358367950

This page is intentionally left blank.

B. Rectangle Donut

Eddy likes to eat donut no matter what shape it is! There are lots of shape of donuts. For example, the most usual one is the circle donuts. However, the one most liked by Eddy is the rectangle donut.

The rectangle donut can be made as follow:

1. Bake a rectangle bread.
2. Cut out a center portion of the bread, where the center of the cut-out part must be the same as the center of the bread.

Eddy is not good at baking bread. Thus, he just bought one from the bakery. The baked bread is in the form of rectangle and it can be represented as $N \times M$ equal-sized square portions. For each square portion, it may have different flavor.

Eddy is going to cut out some $P \times Q$ square portions in the shape of rectangle and centered at the same center as the whole bread.

Eddy considers a rectangle donut delicious if the flavors on the remaining part is symmetric(both horizontally and vertically).

Since there are lots of choice of P and Q, Eddy is wondering how many way to choose P and Q such that the resulting rectangle donut is delicious.

Input

The first line of input contains two space-separated integers N, M indicating the size of the rectangle bread. Following N lines each contains M characters. The j-th character of i-th line c_{ij} indicates the flavor of the square portion of the i-th row and j-th column.

- $1 \leq N, M \leq 2000$, N, M are even
- c_{ij} will be one of the lowercase English letters(`"a-z"`).

Output

Output one line contains an integer indicating the number of way to choose P, Q such that the resulting rectangle donut is delicious.

Note that P, Q should be positive and satisfies $P < N, Q < M$.

Sample Input 1

```
4 6
aaaaaa
aaaaaa
aaaaaa
aaaaaa
```

Sample Output 1

```
2
```

Sample Input 2

```
4 4
abba
baab
baab
abba
```

Sample Output 2

```
1
```

Sample Input 3

```
6 8
acbbbbca
dcaccacd
cdaddadc
cdaddadc
dcaccacd
acbbbbca
```

Sample Output 3

```
6
```

C. Diff-prime Pairs

Eddy has solved lots of problem involving calculating the number of coprime pairs within some range. This problem can be solved with inclusion-exclusion method. Eddy has implemented it lots of times. Someday, when he encounters another coprime pairs problem, he comes up with **diff**-prime pairs problem. **Diff**-prime pairs problem is that given N, you need to find the number of pairs (i, j), where $\frac{i}{gcd(i,j)}$ and $\frac{j}{gcd(i,j)}$ are both prime and $i, j \leq N$. $gcd(i, j)$ is the greatest common divisor of i and j. Prime is an integer greater than 1 and has only 2 positive divisors.

Eddy tried to solve it with inclusion-exclusion method but failed. Please help Eddy to solve this problem.

Note that pair (i_1, j_1) and pair (i_2, j_2) are considered different if $i_1 \neq i_2$ or $j_1 \neq j_2$.

Input

Input has only one line containing a positive integer N.

- $1 \leq N \leq 10^7$

Output

Output one line containing a non-negative integer indicating the number of **diff**-prime pairs (i, j) where $i, j \leq N$

Sample Input 1	Sample Output 1
3	2

Sample Input 2	Sample Output 2
5	6

This page is intentionally left blank.

D. Annoying Path

Eddy lives in Tien-Long country. There are N cities in the Tien-Long country and $N-1$ bidirectional roads. It's possible for any city to go to any other city by those roads.

For each road, there's a difficulty d_i indicating how difficult to drive through this road. For example, there may some holes or some puddles on the road, which will increase the difficulty. Nevertheless, people in Tien-Long country have be trained so well that it's easy for them to drive through any road regardless its difficulty.

But, it's annoying when you drive through a road with high difficulty followed by a road with low difficulty, or vise versa. Since you need to adjust your driving tempo from one to the other.

Thus, Eddy defines the annoying value for the shortest path from u_i to v_i going through roads $r_1, r_2, \ldots, r_k$ as $\sum_{j=1}^{k-1}(d_{r_j} - d_{r_{j+1}})^2$.

Now, Eddy is wondering for each city, what is the maximum annoying value for the shortest path starting from the city and ending at any other city.

Input

The first line of input contains an integer N indicating the number of cities in the Tien-Long country. Following $N-1$ lines each contains three space-separated integers u_i, v_i, d_i indicating there is a road between city u_i and v_i with difficulty d_i.

- $1 \leq N \leq 10^5$
- $1 \leq u_i, v_i \leq N$
- $1 \leq d_i \leq 10^5$

Output

Output N lines. For the i-th line, output an integer indicating the maximum annoying value for the path starting from city i.

Sample Input 1	Sample Output 1
2 1 2 1	0 0

Sample Input 2	Sample Output 2
4 1 2 1 2 3 2 3 4 1	2 1 1 2

E. Expected Number of Nodes

Eddy likes to play with tree. Of course, it's not those green and natural trees, but those trees containing N vertices and $N-1$ edges and miraculously remaining connected. However, it's difficult for Eddy to remember every tree he likes. Some tree may be too large while some tree may have strange structure which is hard to be encoded in memory(in Eddy's brain). Thus, Eddy comes up with following way to contract a tree:

1. Select a subset of vertices.
2. While there's an unselected vertex with degree 1, delete it.
3. While there's an unselected vertex with degree 2, delete it and connect its neighbors.
4. Output the remaining tree.

However, choosing which subset in step 1 is a hard choice for Eddy. He decides to choose some subset of vertices containing exactly k vertices. From all possible choices, He will uniformly randomly choose one of them.

Now, you are wondering what would be the expected number of vertices after contracting. But, you don't know Eddy chooses which k in the first step. Then, you wants to find out all the answer for each k from 1 to N(total number of vertices).

Input

First line of input contains a positive integer N indicating the number of vertices of the tree. For each following $N-1$ lines, each contains two space-separated positive integer u_i, v_i indicating that there's an edge between u_i and v_i.

- $1 \leq N \leq 5000$
- $1 \leq u_i < v_i \leq N$
- It's guaranteed that the given input is a tree.

Output

Please output N lines. For i-th line, output one integer Z indicating the expected number of vertices module 1000000007(10^9+7) after contracting when $k = i$. One can show that the expected

number of vertices can be represented as $\frac{P}{Q}$, then Z will be $P \times Q^{-1}$ module $1000000007(10^9+7)$

Sample Input 1	Sample Output 1
1	1

Sample Input 2	Sample Output 2
4 1 2 2 3 3 4	1 2 3 4

Sample Input 3	Sample Output 3
4 1 2 1 3 1 4	1 2 250000005 4

F. Sum Of Digit

Eddy likes to play with digits. However, as you may know, Eddy is a programmer not a normal human. Thus, he likes to play with hexadecimal digits(base 16) instead of decimal digits(base 10). One day, he found that sum of digits(SOD) is very interesting. Then, he invents following function.

```
def SOD(v):
  if v<16: return v
  return SOD(sum of digits of v in hexadecimal)
```

After playing with SOD several times, Eddy found that for one integer, the computation is too easy to make him happy. Thus, Eddy generates a string of hexadecimal digits S, and takes some subsegment(consecutive digits) of it. Then, Eddy takes all the non-empty subsequence(not necessary consecutive digits) from the subsegment as the inputs of the SOD function. It becomes a little challenging for Eddy now. But, Eddy is still not satisfied. He wants to change the string sometimes and keeps taking some subsegments as queries. Now, it's really a problem for Eddy. You, as one of the friends of Eddy, come to rescue him and are going to compute the answer for him.

Since the number of outputs would be too many(which will be equal to the number of non-empty subsequences), you are only required to compute the number of each output and report the number $(\sum(\text{number of outputs being } i) \times 1021^i) \mod 10^9+7(1000000007)$ to Eddy.

For example, the hexadecimal string S equals 12345. Eddy takes the subsegment $[1,1]$ which is 1. All the non-empty subsequence is [1]. Thus, the answer will be $\sum 1 \times 1021^1 \mod 10^9+7 = 1021$.

If Eddy takes the subsegment $[1,3]$ which is 123. All the non-empty subsequence is $[1, 2, 3, 12, 13, 23, 123]$. Then, the answer will be 267411465.

Input

First line of input contains two space-separated integer N, Q indicating the length of hexadecimal digits S and number of operations Eddy will take.

Second line of input contains a string S indicating the hexadecimal string Eddy generates.

Following Q lines, each line will be one of following form:

- 1 p c: changing p-th digit of S into c.
- 2 l r: taking the subsegment $[l, r]$ and compute the answer.

- $1 \leq N, Q \leq 10^5$
- $|S| = N$, length of S will be N
- character of S will be hexadecimal digit(`0123456789ABCDEF`)
- $1 \leq p \leq N$, c will be hexadecimal digit
- $1 \leq l \leq r \leq N$

Output

For each second type operation(2 l r), output one line indicating the corresponding answer.

Sample Input 1

```
5 2
12345
2 1 1
2 1 3
```

Sample Output 1

```
1021
267411465
```

Sample Input 2

```
5 3
12345
2 1 5
1 1 A
2 1 5
```

Sample Output 2

```
930616025
659780022
```

G. Prefix Sum

As title, this is another simple problem about prefix sum.

Really? Haven't you heard about prefix sum. Prefix sum array S is associated with an array A. The i-th element of S is the sum of first i elements of A. That is, $S_i = \sum_{j=1}^{i} A_j$.

Now, you know how simple the prefix sum is! Since prefix sum array is also an array, we can compute its prefix sum again! Formally, we can define them as follow:

- $S_{1i} = \sum_{j=1}^{i} A_j$
- $S_{xi} = \sum_{j=1}^{i} S_{x-1j}$, for all $x > 1$

Though it's simple, Eddy cannot compute it efficiently. But, he still want to play with prefix sum. As the best friend of him, you come to help Eddy compute those numbers!

Initially, all the elements of the array A will be zero. Then, Eddy will have M operations to perform:

- `0 x y`: increase $A[x]$ by $y (A[x] += y)$
- `1 x`: query the value of S_{Kx}

For each query operation, please tell Eddy the value he is interested.

Input

The first line of the input contains three space-separated integers N, M, K indicating the length of A, the number of operations, and the parameter in the query operation. Following M lines each contains an operation in the form of `"0 x y"` or `"1 x"`.

- $1 \leq N, M \leq 10^5$
- $1 \leq K \leq 40$
- $1 \leq x \leq N$
- $0 \leq y < 10^9 + 7$

Output

For each query operation, you should output one line containing the value. Since the value may be too large, please output it module 1000000007($10^9 + 7$).

Sample Input 1	Sample Output 1
4 11 3 0 1 1 0 3 1 1 1 1 2 1 3 1 4 0 3 1 1 1 1 2 1 3 1 4	1 3 7 13 1 3 8 16

H. Steiner Tree

Currently, Eddy studies about Steiner tree.

Steiner tree is very similar to minimum spanning tree. But, instead of making all the vertices remain connected, Steiner tree only requires some subset of vertices remain connected with minimum total weight.

As a homework, Eddy takes an undirected connected graph and tries to find a Steiner tree of it. However, it turns out that the Steiner tree may not be unique.

Thus, Eddy comes up with following problem: Given an undirected connected graph, how many Steiner tree of first K vertices are there? That's, how many way to choose minimum number of edges to make first K vertices remain connected.

Eddy takes this problem to challenge you. Find out the answer to beat Eddy!

Input

The first line of input contains three space-separated integers N, M, K indicating the number of vertices, number of edges, and the parameter for the Steiner tree. Following M lines each contains two space-separated integers u_i, v_i indicating there is an edge connecting u_i and v_i.

- $1 \leq N \leq 50$
- $N - 1 \leq M \leq min(\frac{N(N-1)}{2}, 1000)$
- $1 \leq K \leq min(N, 12)$
- $1 \leq u_i \neq v_i \leq N$
- It's guaranteed that there is no multiple edges.

Output

Output one line containing an integer indicating the number of way to choose the Steiner tree. Since the number may be too large, please output it module $1000000007(10^9 + 7)$.

Sample Input 1	Sample Output 1
4 4 2 1 3 1 4 2 3 2 4	2

Sample Input 2	Sample Output 2
4 3 3 1 4 2 4 3 4	1

Sample Input 3	Sample Output 3
3 2 2 1 3 2 3	1

I. Maximum Mode

Eddy likes to play with integers. Today, He learns that the mode of an integer sequence is the element that appears the most times.

Eddy collects N integers and wants to compute the mode of them. He found that it's too simple and boring. Then, Eddy tries to remove some integers and computes the mode again. Surprisingly, removing different integers may result in different mode. Thus, Eddy is now wondering how large could the mode be after removing M elements from those N integers.

However, if there are multiple modes after removing M elements, Eddy will feel confused and can't handle it. As the best friend of Eddy, please find the maximum possible unique mode after removing M elements or tell poor Eddy that it's impossible to make mode unique.

Input

The first line of input contains two space-separated integers N, M indicating the number of integers Eddy collected and the number of elements going to be removed.

The second line of input contains N space-separated integers a_i indicating the collected integers.

- $1 \leq N \leq 10^5$
- $0 \leq M < N$
- $1 \leq a_i \leq 10^9$

Output

Output one line containing an integer indicating the maximum possible unique mode or -1 if it's impossible.

Sample Input 1	Sample Output 1
5 0 2 2 3 3 4	-1

Sample Input 2	Sample Output 2
5 1 2 2 3 3 4	3

Sample Input 3	Sample Output 3
5 4 2 2 3 3 4	4

J. SCC

Given a graph $G(V, E)$, V is the set of vertices and E is the set of edges.

For a subset E' of E, let $CC(G(V, E'))$ be the number of connected component of $G(V, E')$

Let define $SCC(G(V, E))$ be $\sum_{E' \subseteq E} CC(G(V, E'))^{CC(G(V,E'))-1}$.

Please compute the $SCC(G(V, E))$ for the given graph.

Input

The first line of input contains two space-separated integers N, M indicating the number of vertices and the number of edges in the graph G. Following M lines each contains two space-separated integers u_i, v_i indicating that there is an edge between u_i and v_i.

- $1 \leq N \leq 19$
- $0 \leq M \leq \frac{N(N-1)}{2}$
- $1 \leq u_i \neq v_i \leq N$
- It's guaranteed that there is no multiple edges.

Output

Output the required value. Since the value may be too large, please output it module 998244353.

Sample Input 1	Sample Output 1
3 3 1 2 2 3 1 3	19

This page is intentionally left blank.