The Coin Change Problem

You have m types of coins available in infinite quantities where the value of each coin is given in the array $C = [c_0, c_1, \ldots, c_{m-1}]$. Can you determine the number of ways of making change for n units using the given types of coins? For example, if $m = 4$, and $C = [8, 3, 1, 2]$, we can make change for $n = 3$ units in three ways: $\{1, 1, 1\}$, $\{1, 2\}$, and $\{3\}$.

Given n, m, and C, print the number of ways to make change for n units using any number of coins having the values given in C.

Input Format

The first line contains two space-separated integers describing the respective values of n and m.
The second line contains m space-separated integers describing the respective values of $c_0, c_1, \ldots c_{m-1}$ (the list of distinct coins available in infinite amounts).

Constraints

- $1 \leq c_i \leq 50$
- $1 \leq n \leq 250$
- $1 \leq m \leq 50$
- Each c_i is guaranteed to be distinct.

Hints

- *Solve overlapping subproblems using Dynamic Programming (DP):*
  You can solve this problem recursively but will not pass all the test cases without optimizing to eliminate the overlapping subproblems. Think of a way to store and reference previously computed solutions to avoid solving the same subproblem multiple times.
- Consider the degenerate cases:
  - How many ways can you make change for 0 cents?
  - How many ways can you make change for > 0 cents if you have no coins?
- If you're having trouble defining your solutions store, then think about it in terms of the base case $(n = 0)$.
- The answer may be larger than a 32-bit integer.

Output Format

Print a long integer denoting the number of ways we can get a sum of n from the given infinite supply of m types of coins.

Sample Input 0

```
4 3
1 2 3
```

Sample Output 0

```
4
```

Explanation 0

There are four ways to make change for $n = 4$ using coins with values given by $C = [1, 2, 3]$:

1. $\{1, 1, 1, 1\}$
2. $\{1, 1, 2\}$
3. $\{2, 2\}$
4. $\{1, 3\}$

Thus, we print 4 as our answer.

Sample Input 1

```
10 4
2 5 3 6
```

Sample Output 1

```
5
```

Explanation 1

There are five ways to make change for $n = 10$ units using coins with values given by $C = [2, 5, 3, 6]$:

1. $\{2, 2, 2, 2, 2\}$
2. $\{2, 2, 3, 3\}$
3. $\{2, 2, 6\}$
4. $\{2, 3, 5\}$
5. $\{5, 5\}$

Thus, we print 5 as our answer.

Equal

Christy is interning at HackerRank. One day she has to distribute some chocolates to her colleagues. She is biased towards her friends and may have distributed the chocolates unequally. One of the program managers gets to know this and orders Christy to make sure everyone gets equal number of chocolates.

But to make things difficult for the intern, she is ordered to equalize the number of chocolates for every colleague in the following manner,

For every operation, she can choose one of her colleagues and can do one of the three things.

1. She can give one chocolate to every colleague other than chosen one.
2. She can give two chocolates to every colleague other than chosen one.
3. She can give five chocolates to every colleague other than chosen one.

Calculate minimum number of such operations needed to ensure that every colleague has the same number of chocolates.

Input Format

First line contains an integer T denoting the number of testcases. T testcases follow.
Each testcase has 2 lines. First line of each testcase contains an integer N denoting the number of colleagues. Second line contains N space separated integers denoting the current number of chocolates each colleague has.

Constraints

$1 <= T <= 100$
$1 <= N <= 10000$
Number of initial chocolates each colleague has < 1000

Output Format

T lines, each containing the minimum number of operations needed to make sure all colleagues have the same number of chocolates.

Sample Input

```
1
4
2 2 3 7
```

Sample Output

```
2
```

Explanation

1st operation: Christy increases all elements by 1 except 3rd one
2 2 3 7 -> 3 3 3 8
2nd operation: Christy increases all element by 5 except last one
3 3 3 8 -> 8 8 8 8

Sherlock and Cost

In this challenge, you will be given an array B and must determine an array A. There is a special rule: For all i, $A[i] \leq B[i]$. That is, $A[i]$ can be any number you choose such that $1 \leq A[i] \leq B[i]$. Your task is to select a series of $A[i]$ given $B[i]$ such that the sum of the absolute difference of consecutive pairs of A is maximized. This will be the array's *cost*, and will be represented by the variable S below.

The equation can be written:

$$S = \sum_{i=2}^{N} |A[i] - A[i-1]|$$

For example, if the array $B = [1, 2, 3]$, we know that $1 \leq A[1] \leq 1$, $1 \leq A[2] \leq 2$, and $1 \leq A[3] \leq 3$. Arrays meeting those guidelines are:

```
[1,1,1], [1,1,2], [1,1,3]
[1,2,1], [1,2,2], [1,2,3]
```

Our calculations for the arrays are as follows:

```
|1-1| + |1-1| = 0 |1-1| + |2-1| = 1 |1-1| + |3-1| = 2
|2-1| + |1-2| = 2 |2-1| + |2-2| = 1 |2-1| + |3-2| = 2
```

The maximum value obtained is 2.

Function Description

Complete the *cost* function in the editor below. It should return the maximum value that can be obtained.

cost has the following parameter(s):

- *B*: an array of integers

Input Format

The first line contains the integer t, the number of test cases.

Each of the next t pairs of lines is a test case where:
- The first line contains an integer n, the length of B
- The next line contains n space-separated integers $B[i]$

Constraints

- $1 \leq t \leq 20$
- $1 < n \leq 10^5$
- $1 \leq B[i] \leq 100$

Output Format

For each test case, print the maximum sum on a separate line.

Sample Input

```
1
5
10 1 10 1 10
```

Sample Output

```
36
```

Explanation

The maximum sum occurs when A[1]=A[3]=A[5]=10 and A[2]=A[4]=1. That is
$|1-10|+|10-1|+|1-10|+|10-1|=36$.

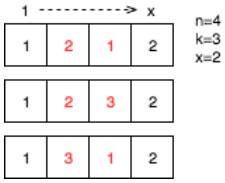
Construct the Array

Your goal is to find the number of ways to construct an array such that consecutive positions contain different values.

Specifically, we want to construct an array with n elements such that each element between 1 and k, inclusive. We also want the first and last elements of the array to be 1 and x.

Given n, k and x, find the number of ways to construct such an array. Since the answer may be large, only find it modulo $10^9 + 7$.

For example, for $n = 4$, $k = 3$, $x = 2$, there are 3 ways, as shown here:



Complete the function $countArrayFill$ which takes input n, k and x. Return the number of ways to construct the array such that consecutive elements are distinct.

Constraints

- $3 \leq n \leq 10^5$
- $2 \leq k \leq 10^5$
- $1 \leq x \leq k$

Subtasks

- For 20% of the maximum score, $n \leq 10^3$ and $k \leq 10^2$

Sample Input

jnj

Sample Output

hbj

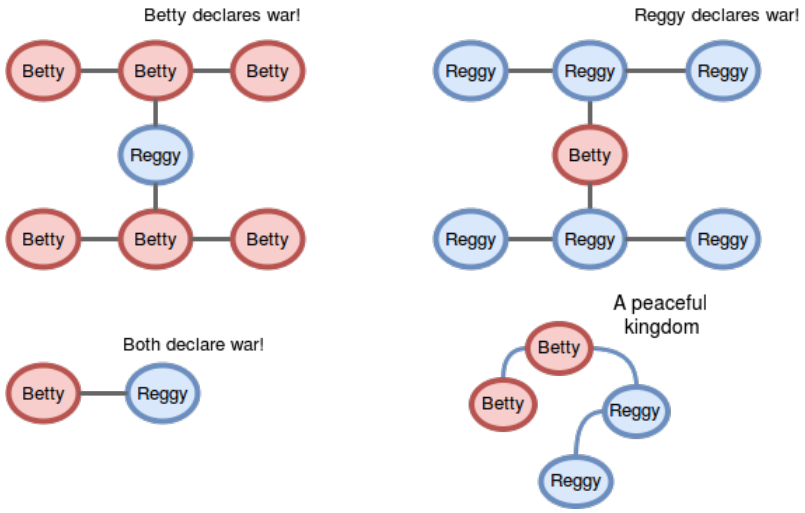
Explanation

vhb

Kingdom Division

King Arthur has a large kingdom that can be represented as a tree, where nodes correspond to cities and edges correspond to the roads between cities. The kingdom has a total of n cities numbered from 1 to n.

The King wants to divide his kingdom between his two children, Reggie and Betty, by giving each of them 0 or more cities; however, they don't get along so he must divide the kingdom in such a way that they will not invade each other's cities. The first sibling will invade the second sibling's city if the second sibling has no other cities directly connected to it. For example, consider the kingdom configurations below:



Given a map of the kingdom's n cities, find and print the number of ways King Arthur can divide it between his two children such that they will not invade each other. As this answer can be quite large, it must be modulo $10^9 + 7$.

Input Format

The first line contains a single integer denoting n (the number of cities in the kingdom).
Each of the $n - 1$ subsequent lines contains two space-separated integers, u and v, describing a road connecting cities u and v.

Constraints

- $2 \leq n \leq 10^5$
- $1 \leq u, v \leq n$
- It is guaranteed that all cities are connected.

Subtasks

- $2 \leq n \leq 20$ for 40% of the maximum score.

Output Format

Print the number of ways to divide the kingdom such that the siblings will not invade each other, modulo $10^9 + 7$.

Sample Input

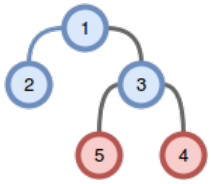
```
5
1 2
1 3
3 4
3 5
```

Sample Output

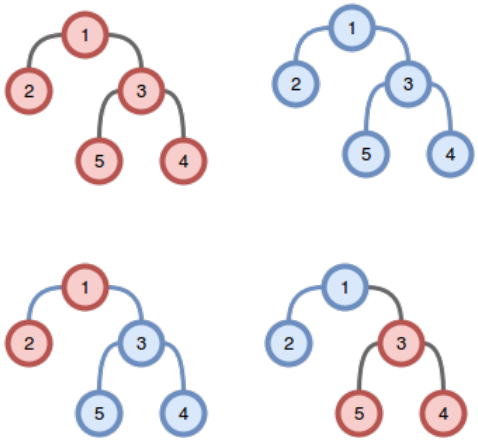
```
4
```

Explanation

In the diagrams below, *red* cities are ruled by Betty and *blue* cities are ruled by Reggie. The diagram below shows depicts a division of the kingdom that results in war between the siblings:



Because cities 5 and 4 are not connected to any other *red* cities, *blue* city 3 will cut off their supplies and declare war on them. That said, there are four valid ways to divide the kingdom peacefully:



We then print the value of $4 \bmod (10^9 + 7) = 4$ as our answer.

Candies

Alice is a kindergarten teacher. She wants to give some candies to the children in her class. All the children sit in a line and each of them has a rating score according to his or her performance in the class. Alice wants to give at least 1 candy to each child. If two children sit next to each other, then the one with the higher rating must get more candies. Alice wants to minimize the total number of candies she must buy.

For example, assume her students' ratings are *[4, 6, 4, 5, 6, 2]*. She gives the students candy in the following minimal amounts: *[1, 2, 1, 2, 3, 1]*. She must buy a minimum of *10* candies.

Function Description

Complete the *candies* function in the editor below. It must return the minimum number of candies Alice must buy.

candies has the following parameter(s):

- *n*: an integer, the number of children in the class
- *arr*: an array of integers representing the ratings of each student

Input Format

The first line contains an integer, n, the size of arr.
Each of the next n lines contains an integer $arr[i]$ indicating the rating of the student at position i.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq arr[i] \leq 10^5$

Output Format

Output a single line containing the minimum number of candies Alice must buy.

Sample Input 0

```
3
1
2
2
```

Sample Output 0

```
4
```

Explanation 0

Here 1, 2, 2 is the rating. Note that when two children have equal rating, they are allowed to have different number of candies. Hence optimal distribution will be 1, 2, 1.

Sample Input 1

```
10
2
4
2
6
1
```

```
7
8
9
2
1
```

Sample Output 1

```
19
```

Explanation 1

Optimal distribution will be $1, 2, 1, 2, 1, 2, 3, 4, 2, 1$

Sample Input 2

```
8
2
4
3
5
2
6
4
5
```

Sample Output 2

```
12
```

Explanation 2

Optimal distribution will be 12121212.

Sam and substrings

Samantha and Sam are playing a numbers game. Given a number as a string, no leading zeros, determine the sum of all integer values of substrings of the string. For example, if the string is 42, the substrings are $4, 2$ and 42. Their sum is 48.

Given an integer as a string, sum all of its substrings cast as integers. As the number may become large, return the value modulo $10^9 + 7$.

Input Format

A single line containing an integer as a string without leading zeros.

Constraints

- $1 \leq n \leq 2 \times 10^5$

Output Format

A single line which is sum of the substrings, $T\%(10^9 + 7)$

Sample Input 0

```
16
```

Sample Output 0

```
23
```

Explanation 0

The substring of number `16` are `16, 1` and `6` which sums to `23`.

Sample Input 1

```
123
```

Sample Output 1

```
164
```

Explanation 1

The sub-strings of `123` are `1, 2, 3, 12, 23, 123` which sums to `164`.

Fibonacci Modified

We define a *modified* Fibonacci sequence using the following definition:

Given terms t_i and t_{i+1} where $i \in [1, \infty)$, term t_{i+2} is computed using the following relation:

$$t_{i+2} = t_i + (t_{i+1})^2$$

For example, if term $t_1 = 0$ and $t_2 = 1$, term $t_3 = 0 + 1^2 = 1$, term $t_4 = 1 + 1^2 = 2$, term $t_5 = 1 + 2^2 = 5$, and so on.

Given three integers, t_1, t_2, and n, compute and print term t_n of a *modified Fibonacci sequence*.

Note: The value of t_n may exceed the range of a 64-bit integer. Many submission languages have libraries that can handle such large results but, for those that don't (e.g., C++), you will need to be more creative in your solution to compensate for the limitations of your chosen submission language.

Input Format

A single line of three space-separated integers describing the respective values of t_1, t_2, and n.

Constraints

- $0 \leq t_1, t_2 \leq 2$
- $3 \leq n \leq 20$
- t_n may exceed the range of a 64-bit integer.

Output Format

Print a single integer denoting the value of term t_n in the modified Fibonacci sequence where the first two terms are t_1 and t_2.

Sample Input

```
0 1 5
```

Sample Output

```
5
```

Explanation

The first two terms of the sequence are $t_1 = 0$ and $t_2 = 1$, which gives us a modified Fibonacci sequence of $\{0, 1, 1, 2, 5, 27, \ldots\}$. Because $n = 5$, we print term t_5, which is 5.

Abbreviation

You can perform the following operations on the string, a:

1. Capitalize zero or more of a's lowercase letters.
2. Delete all of the remaining lowercase letters in a.

Given two strings, a and b, determine if it's possible to make a equal to b as described. If so, print `YES` on a new line. Otherwise, print `NO`.

For example, given $a = \mathbf{AbcDE}$ and $b = \mathbf{ABDE}$, in a we can convert $\mathbf{b}$ and delete $\mathbf{c}$ to match b. If $a = \mathbf{AbcDE}$ and $b = \mathbf{AFDE}$, matching is not possible because letters may only be capitalized or discarded, not changed.

Function Description

Complete the function $abbreviation$ in the editor below. It must return either YES or NO.

abbreviation has the following parameter(s):

- *a*: the string to modify
- *b*: the string to match

Input Format

The first line contains a single integer q, the number of queries.

Each of the next q pairs of lines is as follows:
- The first line of each query contains a single string, a.
- The second line of each query contains a single string, b.

Constraints

- $1 \leq q \leq 10$
- $1 \leq |a|, |b| \leq 1000$
- String a consists only of uppercase and lowercase English letters, ascii[A-Za-z].
- String b consists only of uppercase English letters, ascii[A-Z].

Output Format

For each query, print `YES` on a new line if it's possible to make string a equal to string b. Otherwise, print `NO`.

Sample Input

```
1
daBcd
ABC
```

Sample Output

```
YES
```

Explanation

We have $a =$ daBcd and $b =$ ABC. We perform the following operation:

1. Capitalize the letters a and c in a so that $a =$ dABCd.
2. Delete all the remaining lowercase letters in a so that $a =$ ABC.

Because we were able to successfully convert a to b, we print YES on a new line.

Prime XOR

Penny has an array of n integers, $[a_0, a_1, \ldots, a_{n-1}]$. She wants to find the number of unique multisets she can form using elements from the array such that the bitwise XOR of all the elements of the multiset is a prime number. Recall that a *multiset* is a set which can contain duplicate elements.

Given q queries where each query consists of an array of integers, can you help Penny find and print the number of valid multisets for each array? As these values can be quite large, modulo each answer by $10^9 + 7$ before printing it on a new line.

Input Format

The first line contains a single integer, q, denoting the number of queries. The $2 \cdot q$ subsequent lines describe each query in the following format:

1. The first line contains a single integer, n, denoting the number of integers in the array.
2. The second line contains n space-separated integers describing the respective values of $a_0, a_1, \ldots, a_{n-1}$.

Constraints

- $1 \leq q \leq 10$
- $1 \leq n \leq 100000$
- $3500 \leq a_i \leq 4500$

Output Format

On a new line for each query, print a single integer denoting the number of unique multisets Penny can construct using numbers from the array such that the bitwise XOR of all the multiset's elements is prime. As this value is quite large, your answer must be modulo $10^9 + 7$.

Sample Input

```
1
3
3511 3671 4153
```

Sample Output

```
4
```

Explanation

The valid multisets are:

- $\{3511\} \rightarrow 3511$ is prime.
- $\{3671\} \rightarrow 3671$ is prime.
- $\{4153\} \rightarrow 4153$ is prime.
- $\{3511, 3671, 4153\} \rightarrow 3511 \oplus 3671 \oplus 4153 = 5081$, which is prime.

Because there are four valid multisets, we print the value of $4 \% (10^9 + 7) = 4$ on a new line.

Decibinary Numbers

Let's talk about *binary numbers*. We have an n-digit binary number, b, and we denote the digit at index i (zero-indexed from right to left) to be b_i. We can find the *decimal* value of b using the following formula:

$$(b)_2 \Rightarrow b_{n-1} \cdot 2^{n-1} + \ldots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0 = (?)_{10}$$

For example, if binary number $b = 10010$, we compute its decimal value like so:

$$(10010)_2 \Rightarrow 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = (18)_{10}$$

Meanwhile, in our well-known decimal number system where each digit ranges from 0 to 9, the value of some decimal number, d, can be expanded in the same way:

$$d = d_{n-1} \cdot 10^{n-1} + \ldots + d_2 \cdot 10^2 + d_1 \cdot 10^1 + d_0 \cdot 10^0$$

Now that we've discussed both systems, let's combine decimal and binary numbers in a new system we call *decibinary*! In this number system, each digit ranges from 0 to 9 (like the decimal number system), but the *place value* of each digit corresponds to the one in the binary number system! For example, the decibinary number 2016 represents the decimal number 24 because:

$$(2016)_{decibinary} \Rightarrow 2 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 6 \cdot 2^0 = (24)_{10}$$

Pretty cool system, right? Unfortunately, there's a problem; two different decibinary numbers could evaluate to the same decimal value! For example, the decibinary number 2008 also evaluates to the decimal value 24:

$$(2008)_{decibinary} \Rightarrow 2 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 8 \cdot 2^0 = (24)_{10}$$

This is a major problem because our new number system has no real applications beyond this challenge!

Consider an infinite list of non-negative decibinary numbers that is sorted according to the following rules:

- The decibinary numbers are sorted in increasing order of the decimal value that they evaluate to.
- Any two decibinary numbers that evaluate to the same decimal value are ordered by increasing decimal value, meaning the equivalent decibinary values are strictly interpreted and compared as decimal values and the smaller decimal value is ordered first. For example, $(2)_{decibinary}$ and $(10)_{decibinary}$ both evaluate to $(2)_{10}$. We would order $(2)_{decibinary}$ before $(10)_{decibinary}$ because $(2)_{10} < (10)_{10}$.

Here is a list of first few decibinary numbers properly ordered:

x	Decibinary	Decimal
1	0	0
2	1	1
3	2	2
4	10	2
5	3	3
6	11	3
7	4	4
8	12	4
9	20	4
10	100	4
.	...	...
20	110	6

You will be given q queries in the form of an integer, x. For each x, find and print the the x^{th} decibinary number in the list on a new line.

Input Format

The first line contains an integer, q, denoting the number of queries.
Each line i of the q subsequent lines contains a single integer, x, describing a query.

Constraints

- $1 \leq q \leq 10^5$
- $1 \leq x \leq 10^{16}$

Subtasks

- $1 \leq x \leq 50$ for 10% of the maximum score
- $1 \leq x \leq 9000$ for 30% of the maximum score
- $1 \leq x \leq 10^7$ for 60% of the maximum score

Output Format

For each query, print a single integer denoting the the x^{th} decibinary number in the list. Note that this must be the actual decibinary number and *not* its decimal value.

Sample Input 0

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

Sample Output 0

```
0
1
2
10
100
```

Explanation 0

For each x, we print the x^{th} decibinary number on a new line. See the figure in the problem statement.

Sample Input 1

```
7
8
23
19
16
26
7
6
```

Sample Output 1

```
12
23
102
14
```

```
111
4
11
```

Sample Input 2

```
10
19
25
6
8
20
10
27
24
30
11
```

Sample Output 2

```
102
103
11
12
110
100
8
31
32
5
```

Fair Cut

Li and Lu have n integers, $a_1, a_2, \ldots, a_n$, that they want to divide fairly between the two of them. They decide that if Li gets integers with indices $I = \{i_1, i_2, \ldots, i_k\}$ (which implies that Lu gets integers with indices $J = \{1, \ldots, n\} \setminus I$), then the measure of unfairness of this division is:

$$f(I) = \sum_{i \in I} \sum_{j \in J} |a_i - a_j|$$

Find the minimum measure of unfairness that can be obtained with some division of the set of integers where Li gets exactly k integers.

Note $A \setminus B$ means Set complement

Input Format

The first line contains two space-separated integers denoting the respective values of n (the number of integers Li and Lu have) and k (the number of integers Li wants).
The second line contains n space-separated integers describing the respective values of $a_1, a_2, \ldots, a_n$.

Constraints

- $1 \leq k < n \leq 3000$
- $1 \leq a_i \leq 10^9$
- For 15% of the test cases, $n \leq 20$.
- For 45% of the test cases, $n \leq 40$.

Output Format

Print a single integer denoting the minimum measure of unfairness of some division where Li gets k integers.

Sample Input 0

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

Sample Output 0

```
6
```

Explanation 0

One possible solution for this input is $I = \{2, 4\}$; $J = \{1, 3\}$.
$|a_2 - a_1| + |a_2 - a_3| + |a_4 - a_1| + |a_4 - a_3| = 1 + 2 + 2 + 1 = 6$

Sample Input 1

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

Sample Output 1

```
2
```

Explanation 1

The following division of numbers is optimal for this input: $I = \{1\}$; $J = \{2, 3, 4\}$.

The Maximum Subarray

We define *subsequence* as any subset of an array. We define a *subarray* as a *contiguous subsequence* in an array.

Given an array, find the maximum possible sum among:

1. all nonempty subarrays.
2. all nonempty subsequences.

Print the two values as space-separated integers on one line.

Note that empty subarrays/subsequences should not be considered.

For example, given an array $arr = [-1, 2, 3, -4, 5, 10]$, the maximum subarray sum is comprised of element inidices $[1 - 5]$ and the sum is $2 + 3 + -4 + 5 + 10 = 16$. The maximum subsequence sum is comprised of element indices $[1, 2, 4, 5]$ and the sum is $2 + 3 + 5 + 10 = 20$.

Function Description

Complete the *maxSubarray* function in the editor below. It should return an array of two integers: the maximum subarray sum and the maximum subsequence sum of arr.

maxSubarray has the following parameter(s):

- *arr*: an array of integers

Input Format

The first line of input contains a single integer t, the number of test cases.

The first line of each test case contains a single integer n.
The second line contains n space-separated integers $arr[i]$ where $0 \leq i < n$.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n \leq 10^5$
- $-10^4 \leq a[i] \leq 10^4$

The subarray and subsequences you consider should have at least one element.

Output Format

Print two space-separated integers denoting the maximum sums of nonempty subarrays and nonempty subsequences, respectively.

Sample Input 0

```
2
4
1 2 3 4
6
2 -1 2 3 4 -5
```

Sample Output 0

```
10 10
10 11
```

Explanation 0

In the first case: The maximum sum for both types of subsequences is just the sum of all the elements since they are all positive.

In the second case: The subarray $[2, -1, 2, 3, 4]$ is the subarray with the maximum sum, and $[2, 2, 3, 4]$ is the subsequence with the maximum sum.

Sample Input 1

```
1
5
-2 -3 -1 -4 -6
```

Sample Output 1

```
-1 -1
```

Explanation 1

Since all of the numbers are negative, both the maximum subarray and maximum subsequence sums are made up of one element, -1.

Angry Children 2

Bill Gates is on one of his philanthropic journeys to a village in Utopia. He has **N** packets of candies and would like to distribute one packet to each of the **K** children in the village (each packet may contain different number of candies). To avoid a fight between the children, he would like to pick **K** out of **N** packets such that the unfairness is minimized.

Suppose the **K** packets have (x_1, x_2, x_3,....x_k) candies in them, where x_i denotes the number of candies in the i^{th} packet, then we define *unfairness* as

$$\sum_{1 \le i < j \le k} |X_i - X_j|$$

where |a| denotes the absolute value of a.

Input Format
The first line contains an integer N.
The second line contains an integer K.
N lines follow each integer containing the candy in the i^{th} packet.

Output Format
A single integer which will be minimum unfairness.

Constraints
2<=N<=10^5
2<=K<=N
0<= number of candies in each packet <=10^9

Sample Input #00

```
7
3
10
100
300
200
1000
20
30
```

Sample Output #00

```
40
```

Explanation #00

Bill Gates will choose packets having 10, 20 and 30 candies.So unfairness will be |10-20| + |20-30| + |10-30| = 40. We can verify that it will be minimum in this way.

Sample Input #01

```
10
4
1
2
3
```

```
4
10
20
30
40
100
200
```

Sample Output #01

```
10
```

Explanation #01

Bill Gates will choose 4 packets having 1,2,3 and 4 candies. So, unfairness will be |1-2| + |1-3| + |1-4| + |2-3| + |2-4| + |3-4| = 10

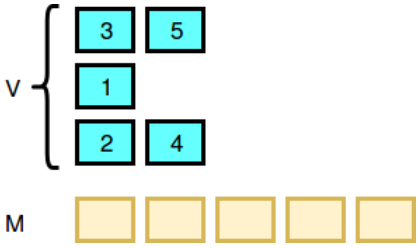
Sherlock's Array Merging Algorithm

Watson gave Sherlock a collection of arrays V. Here each V_i is an array of variable length. It is guaranteed that if you merge the arrays into one single array, you'll get an array, M, of n distinct integers in the range $[1, n]$.

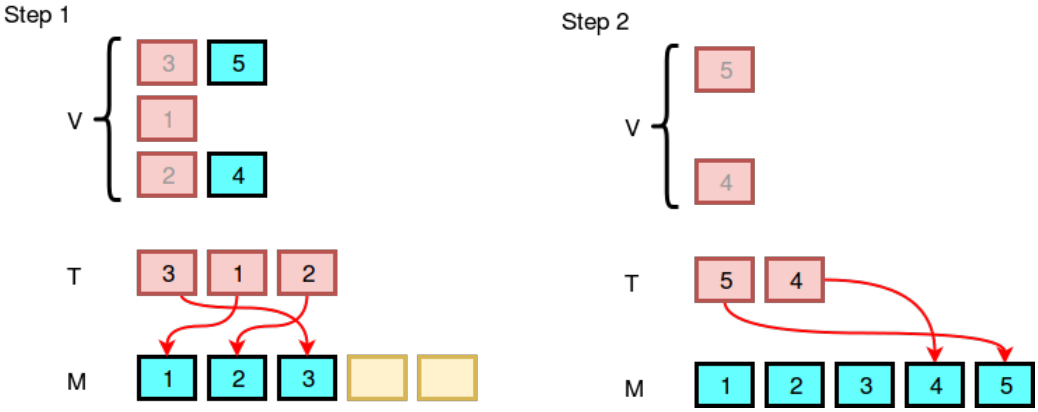
Watson asks Sherlock to merge V into a sorted array. Sherlock is new to coding, but he accepts the challenge and writes the following algorithm:

- $M \leftarrow [\,]$ (an empty array).
- $k \leftarrow$ size of the collection V.
- While there is at least one non-empty array in V:
  - $T \leftarrow [\,]$ (an empty array) and $i \leftarrow 1$.
  - While $i \leq k$:
    - If V_i is not empty:
      - Remove the first element of V_i and push it to T.
    - $i \leftarrow i + 1$.
  - While T is not empty:
    - Remove the minimum element of T and push it to M.
- Return M as the *output*.

Let's see an example. Let V be $\{[3, 5], [1], [2, 4]\}$.



The image below demonstrates how Sherlock will do the merging according to the algorithm:



Sherlock isn't sure if his algorithm is correct or not. He ran Watson's *input*, V, through his pseudocode algorithm to produce an *output*, M, that contains an array of n integers. However, Watson forgot the contents of V and only has Sherlock's M with him! Can you help Watson reverse-engineer M to get the

original contents of V?

Given m, find the number of different ways to create collection V such that it produces m when given to Sherlock's algorithm as *input*. As this number can be quite large, print it modulo $10^9 + 7$.

Notes:

- Two collections of arrays are *different* if one of the following is *true*:
  - Their sizes are different.
  - Their sizes are the same but at least one array is present in one collection but not in the other.
- Two arrays, A and B, are different if one of the following is *true*:
  - Their sizes are different.
  - Their sizes are the same, but there exists an index i such that $a_i \neq b_i$.

Input Format

The first line contains an integer, n, denoting the size of array M.
The second line contains n space-separated integers describing the respective values of $m_0, m_1, \ldots, m_{n-1}$.

Constraints

- $1 \leq n \leq 1200$
- $1 \leq m_i \leq n$

Output Format

Print the number of different ways to create collection V, modulo $10^9 + 7$.

Sample Input 0

```
3
1 2 3
```

Sample Output 0

```
4
```

Explanation 0

There are four distinct possible collections:

1. $V = \{[1, 2, 3]\}$
2. $V = \{[1], [2], [3]\}$
3. $V = \{[1, 3], [2]\}$
4. $V = \{[1], [2, 3]\}$.

Thus, we print the result of $4 \bmod (10^9 + 7) = 4$ as our answer.

Sample Input 1

```
2
2 1
```

Sample Output 1

```
1
```

Explanation 1

The only distinct possible collection is $V = \{[2, 1]\}$, so we print the result of $1 \bmod (10^9 + 7) = 1$ as our answer.

Prime Digit Sums

Chloe is fascinated by prime numbers. She came across the number 283002 on a sign and, though the number is not prime, found some primes hiding in it by using the following rules:

- Every three consecutive digits sum to a prime:

$$\underbrace{283}002 \quad 2\underbrace{830}02 \quad 28\underbrace{300}2 \quad 283\underbrace{002}$$

- Every four consecutive digits sum to a prime:

$$\underbrace{2830}02 \quad 2\underbrace{8300}2 \quad 28\underbrace{3002}$$

- Every five consecutive digits sum to a prime:

$$\underbrace{28300}2 \quad 2\underbrace{83002}$$

You must answer q queries, where each query consists of an integer, n. For each n, find and print the number of positive n-digit numbers, modulo $10^9 + 7$, that satisfy *all three* of Chloe's rules (i.e., every three, four, and five consecutive digits sum to a prime).

Input Format

The first line contains an integer, q, denoting the number of queries.
Each of the q subsequent lines contains an integer denoting the value of n for a query.

Constraints

- $1 \leq q \leq 2 \times 10^4$
- $1 \leq n \leq 4 \times 10^5$

Output Format

For each query, print the number of n-digit numbers satisfying Chloe's rules, modulo $10^9 + 7$, on a new line.

Sample Input 0

```
1
6
```

Sample Output 0

```
95
```

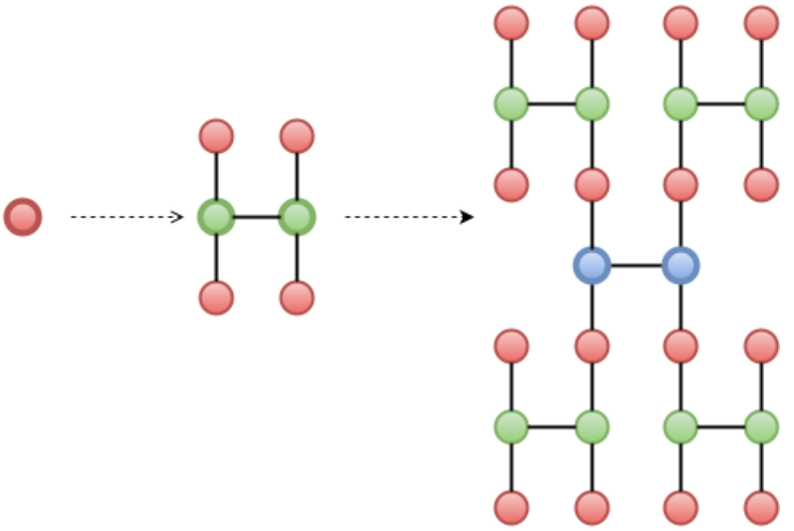
Explanation 0

There are 95 six-digit numbers satisfying the property above, where the respective first and last ones are 101101 and 902005.

HackerRank City

HackerRank-city is an acyclic connected graph (or tree). Its not an ordinary place, the construction of the whole tree takes place in N steps. The process is described below:

- It initially has 1 node.
- At each step, you must create 3 duplicates of the current tree, and create 2 new nodes to connect all 4 copies in the following `H` shape:



At each i^{th} step, the tree becomes 4 times bigger plus 2 new nodes, as well as 5 new edges connecting everything together. The length of the new edges being added at step i is denoted by input A_i.

Calculate the sum of distances between each pair of nodes; as these answers may run large, print your answer modulo 1000000007.

Input Format

The first line contains an integer, N (the number of steps). The second line contains N space-separated integers describing A_0, $A_1, \ldots, A_{N-2}, A_{N-1}$.

Constraints

$1 \leq N \leq 10^6$
$1 \leq A_i \leq 9$

Subtask

For 50% score $1 \leq N \leq 10$

Output Format

Print the sum of distances between each pair of nodes modulo 1000000007.

Sample Input 0

```
1
1
```

Sample Output 0

```
29
```

Sample Input 1

```
2
2 1
```

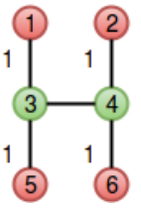
Sample Output 1

```
2641
```

Explanation

Sample 0

In this example, our tree looks like this:



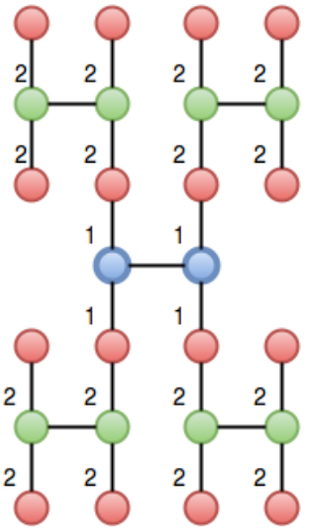
Let $d(u, v)$ denote the distance between nodes u and v.

$d(1,2) + d(1,3) + d(1,4) + d(1,5) + d(1,6)\ + d(2,3) + d(2,4) + d(2,5) + d(2,6) + d(3,4)$
$+ d(3,5) + d(3,6) + d(4,5) + d(4,6) + d(5,6) =$
$3 + 1 + 2 + 2 + 3 + 2 + 1 + 3 + 2 + 1 + 1 + 2 + 2 + 1 + 3 = 29$.

We print the result of $29\ \%\ 1000000007$ as our answer.

Sample 1

In this example, our tree looks like this:



We calculate and sum the distances between nodes in the same manner as *Sample 0* above, and print the result of our $answer\ \%\ 1000000007$, which is 2641.

Summing Pieces

Consider an array, A, of length n. We can split A into contiguous segments called *pieces* and store them as another array, B. For example, if $A = [1, 2, 3]$, we have the following arrays of pieces:

- $B = [(1), (2), (3)]$ contains three 1-element pieces.
- $B = [(1, 2), (3)]$ contains two pieces, one having 2 elements and the other having 1 element.
- $B = [(1), (2, 3)]$ contains two pieces, one having 1 element and the other having 2 elements.
- $B = [(1, 2, 3)]$ contains one 3-element piece.

We consider the *value* of a piece in some array B to be $(\textit{sum of all numbers in the piece}) \times (\textit{length of piece})$, and we consider the *total value* of some array B to be the sum of the values for all pieces in that B. For example, the total value of $B = [(1, 2, 4), (5, 1), (2)]$ is $(1 + 2 + 4) \times 3 + (5 + 1) \times 2 + (2) \times 1 = 35$.

Given A, find the total values for all possible B's, sum them together, and print this sum modulo $(10^9 + 7)$ on a new line.

Input Format

The first line contains a single integer, n, denoting the size of array A.
The second line contains n space-separated integers describing the respective values in A (i.e., $a_0, a_1, \ldots, a_{n-1}$).

Constraints

- $1 \leq n \leq 10^6$
- $1 \leq a_i \leq 10^9$

Output Format

Print a single integer denoting the sum of the total values for all piece arrays (B's) of A, modulo $(10^9 + 7)$.

Sample Input 0

```
3
1 3 6
```

Sample Output 0

```
73
```

Explanation 0
Given $A = [1, 3, 6]$, our piece arrays are:

- $B = [(1), (3), (6)]$, and $\textit{total value} = (1) \times 1 + (3) \times 1 + (6) \times 1 = 10$.
- $B = [(1, 3), (6)]$, and $\textit{total value} = (1 + 3) \times 2 + (6) \times 1 = 14$.
- $B = [(1), (3, 6)]$, and $\textit{total value} = (1) \times 1 + (3 + 6) \times 2 = 19$.
- $B = [(1, 3, 6)]$, and $\textit{total value} = (1 + 3 + 6) \times 3 = 30$.

When we sum all the total values, we get $10 + 14 + 19 + 30 = 73$. Thus, we print the result of $73 \bmod (10^9 + 7) = 73$ on a new line.

Sample Input 1

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

Sample Output 1

```
971
```

Mr K marsh

Mr K has a rectangular land of size $m \times n$. There are marshes in the land where the fence cannot hold. Mr K wants you to find the perimeter of the largest rectangular fence that can be built on this land.

Input format

The first line contains m and n. The next m lines contain n characters each describing the state of the land. 'x' (ascii value: 120) if it is a marsh and '.' (ascii value:46) otherwise.

Constraints

$2 \leq m, n \leq 500$

Output Format

Output contains a single integer - the largest perimeter. If the rectangular fence cannot be built, print "**impossible**" (without quotes).

Sample Input:1

```
4 5
.....
.x.x.
.....
.....
```

Output

```
14
```

Fence can be put up across the entire land owned by Mr K. The perimeter is
$2 * (4 - 1) + 2 * (5 - 1) = 14$.

Sample Input:2

```
2 2
.x
x.
```

Output

```
impossible
```

We need minimum of 4 points to place the 4 corners of the fence. Hence, impossible.

Sample Input:3

```
2 5
.....
xxxx.
```

Output

impossible

Substring Diff

Substring Diff

Given two strings of length n, $P = p_1p_2...p_n$ and $Q = q_1q_2...q_n$, we define $M(i, j, k)$ as the number of mismatches between $p(i), p(i + 1), ...p(i + k - 1)$ and $q(j), q(j + 1)..., q(j + k - 1)$. That is, in set notation, $M(i, j, k)$ refers to the size of the set

```
{0 <= x < k | p[i+x] ≠ q[j+x]}
```

Given an integer S, your task is to find the maximum length L, such that there exists a pair of indices (i, j) for which we have $M(i, j, L) \leq S$. Of course, we should also have $i + L - 1 \leq n$ and $j + L - 1 \leq n$.

Input Format

The first line of input contains a single integer, T, the number of test cases follow.
Each test case consists of an integer, S, and two strings, P and Q, separated by a single space.

Constraints

- $1 \leq T \leq 10$
- $0 \leq S \leq$ *length of* P
- P and Q has the same length, their length not exceeding 1500.
- All characters in P and Q are lower-case English letters.

Output Format

For each test case, output a single integer, L, which is the maximum value for which there exists a pair of indices (i, j) such that $M(i, j, L) \leq S$.

Sample Input

```
3
2 tabriz torino
0 abacba abcaba
3 helloworld yellomarin
```

Sample Output

```
4
3
8
```

Explanation

- First test case: If we take "briz" from the first string, and "orin" from the second string, then the number of mismatches between these two substrings is equal to 2, and the length of these substrings are 4. Hence we have chosen i=3, j=2, L=4, and we have M(3,2,4) = 2.
- Second test case: Since S=0, we should find the longest common substring for the given input strings. We can choose "aba" as the result, and we don't have longer common substring between two strings. So, the answer is 3 for this test-case. That's we have chosen i=1, j=4, and L=3, and we have M(1,4,3)=0.
- Third test case: We can choose "hellowor" from first string and "yellomar" from the second string. So,

we have chosen i=1, j=1, and L=8, and we have M(1,1,8)=3. Of course we can also choose i=2, j=2, and L=8 and we still have M(2,2,8)=3.

Xor and Sum

You are given two positive integers a and b in binary representation. You should find the following sum modulo $10^9 + 7$:

$$\sum_{i=0}^{314159} (a \; xor \; (b \; shl \; i))$$

where operation xor means exclusive OR operation, operation shl means binary shift to the left.

Please note, that we consider ideal model of binary integers. That is there is infinite number of bits in each number, and there are no disappearings (or cyclic shifts) of bits.

Input Format

The first line contains number a $(1 \leq a < 2^{10^5})$ in binary representation. The second line contains number b $(1 \leq b < 2^{10^5})$ in the same format. All the numbers do not contain leading zeros.

Output Format

Output a single integer — the required sum modulo $10^9 + 7$.

Sample Input

```
10
1010
```

Sample Output

```
489429555
```

Lego Blocks

You have 4 types of lego blocks, of sizes given as (1 x 1 x 1), (1 x 1 x 2), (1 x 1 x 3), and (1 x 1 x 4). Assume that you have an infinite number of blocks of each type.

Using these blocks, you want to make a wall of height N and width M. The wall should not have any holes in it. The wall you build should be one solid structure. A solid structure can be interpreted in one of the following ways:
(1)It should not be possible to separate the wall along any vertical line without cutting any lego block used to build the wall.
(2)You cannot make a vertical cut from top to bottom without cutting one or more lego blocks.

The blocks can only be placed horizontally. In how many ways can the wall be built?

Input Format

The first line contains the number of test cases T. T test cases follow. Each case contains two integers N and M.

Constraints

1 <= T <= 100
1 <= N,M <= 1000

Output Format

Output T lines, one for each test case containing the number of ways to build the wall. As the numbers can be very large, output the result modulo 1000000007.

Sample Input

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

Sample Output

```
3
7
9
3375
```

Explanation

For the first case, we can have

- two (1 * 1 * 2) lego blocks stacked one on top of another.
- one (1 * 1 * 2) block stacked on top of two (1 * 1 * 1) blocks.
- two (1 * 1 * 1) blocks stacked on top of one (1 * 1 * 2) block.

For the second case, each row of the wall can contain either two blocks of width 1, or one block of width 2. However, the wall where all rows contain two blocks of width 1 is not a solid one as it can be divided vertically. Thus, the number of ways is 2 * 2 * 2 - 1 = 7.

Brick Tiling

You are given a grid having N rows and M columns. A grid square can either be blocked or empty. Blocked squares are represented by a '#' and empty squares are represented by '.'. Find the number of ways to tile the grid using L shaped bricks. A L brick has one side of length three units while other of length 2 units. All empty squares in the grid should be covered by exactly one of the L shaped tiles, and blocked squares should not be covered by any tile. The bricks can be used in any orientation (they can be rotated or flipped).

Input Format

The first line contains the number of test cases T. T test cases follow. Each test case contains N and M on the first line, followed by N lines describing each row of the grid.

Constraints

1 <= T <= 50
1 <= N <= 20
1 <= M <= 8
Each grid square will be either '.' or '#'.

Output Format

Output the number of ways to tile the grid. Output each answer modulo 1000000007.

Sample Input

```
3
2 4
....
....
3 3
...
.#.
...
2 2
##
##
```

Sample Output

```
2
4
1
```

Explanation

NOTE:
If all points in the grid are blocked the number of ways is 1, as in the last sample testcase.

Alien Languages

Sophia has discovered several alien languages. Suprisingly, all of these languages have an alphabet, and each of them may contain thousands of characters! Also, all the words in a language have the same number of characters in it.

However, the aliens like their words to be aesthetically pleasing, which for them means that for the i^{th} letter of an n letter alphabet (letters are indexed at 1):

if $2 * i > n$

the i^{th} letter may be the last letter of a word, and it may be immediately followed by any letter including itself.

if $2 * i \leq n$

the i^{th} letter can not be the last letter of a word and also can only be immediately followed by j^{th} letter iff(if and only if) $j \geq 2 * i$.

Sophia wants to know how many different words exist in this language. Since the result may be large, she wants to know this number, modulo 100000007.

Input Format

The first line contains t, the number of test cases. The first line is followed by t lines, each line denoting a test case. Each test case will have two space separated integers n,*m* which denote the number of letters in the language and the length of words in this language respectively.

Constraints

$1 \leq t \leq 5$
$1 \leq n \leq 10^5$
$1 \leq m \leq 5*10^5$

Output Format

For each testcase output the number of possible words modulo 100000007.

Sample Input

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

Sample Output

```
1
3
6
```

Explanation

For the first test-case, there's one letter and all the words consist of 3 letters. There's only one possibility which is "aaa"

For the second test-case, there are two letters (a & b) and all the words are of 3 letters. The possible ones are "abb", "bab", & "bbb". The words can end only with 'b' because 2 * index(b) = 2 * 2 > 2 and for 'a', it's 2 * index(a) = 2 * 1 <= 2. "aab" is not allowed because 'a' can not be followed immediately by 'a'. For a

word of length 4 and alphabet of size 2, "abab" would be allowed.

For the third test-case, there are three letters (a, b & c) and all of the words are 2 letters. The words can end only with 'b' or 'c'. The possible words are "ab", "ac", "bb", "cc", "bc", "cb"

Stock Maximize

Your algorithms have become so good at predicting the market that you now know what the share price of Wooden Orange Toothpicks Inc. (WOT) will be for the next number of days.

Each day, you can either buy one share of WOT, sell any number of shares of WOT that you own, or not make any transaction at all. What is the maximum profit you can obtain with an optimum trading strategy?

For example, if you know that prices for the next two days are $prices = [1, 2]$, you should buy one share day one, and sell it day two for a profit of 1. If they are instead $prices = [2, 1]$, no profit can be made so you don't buy or sell stock those days.

Function Description

Complete the *stockmax* function in the editor below. It must return an integer that represents the maximum profit achievable.

stockmax has the following parameter(s):

- *prices*: an array of integers that represent predicted daily stock prices

Input Format

The first line contains the number of test cases t.

Each of the next t pairs of lines contain:
- The first line contains an integer n, the number of predicted prices for WOT.
- The next line contains n space-separated integers $prices[i]$, each a predicted stock price for day i.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n \leq 50000$
- $1 \leq prices[i] \leq 100000$

Output Format

Output t lines, each containing the maximum profit which can be obtained for the corresponding test case.

Sample Input

```
3
3
5 3 2
3
1 2 100
4
1 3 1 2
```

Sample Output

```
0
197
3
```

Explanation

For the first case, you cannot obtain any profit because the share price never rises.
For the second case, you can buy one share on the first two days and sell both of them on the third day.

For the third case, you can buy one share on day 1, sell one on day 2, buy one share on day 3, and sell one share on day 4.

Two Robots

You have a warehouse with M containers filled with an infinite number of candies. The containers are arranged in a single row, equally spaced to be 1 meter apart. You also have 2 robots that can pick up 1 piece of candy and transport it between any two containers.

The robots take instructions in the form of *queries* consisting of two integers, M_a and M_b, respectively. To execute a query, a robot travels to container M_a, picks up 1 candy, transports it to container M_b, and then stops at M_b until it receives another query.

Calculate the *minimum total distance* the robots must travel to execute N queries *in order*.

Note: You choose which robot executes each query.

Input Format

The first line contains a single integer, T (the number of test cases); each of the T test cases is described over $N+1$ lines.

The first line of a test case has two space-separated integers, M (the number of containers) and N (the number of queries).
The N subsequent lines each contain two space-separated integers, M_a and M_b, respectively; each line N_i describes the i^{th} query.

Constraints

- $1 \leq T \leq 50$
- $1 < M \leq 1000$
- $1 \leq N \leq 1000$
- $1 \leq a, b \leq M$
- $M_a \neq M_b$

Output Format

On a new line for each test case, print an integer denoting the *minimum total distance* that the robots must travel to execute the queries in order.

Sample Input

```
3
5 4
1 5
3 2
4 1
2 4
4 2
1 2
4 3
10 3
2 4
5 4
9 8
```

Sample Output

```
11
2
5
```

Explanation

In this explanation, we refer to the two robots as R_1 and R_2, each container i as M_i, and the total distance traveled for each query j as D_j.

Note: For the first query a robot executes, there is no travel distance. For each subsequent query that robot executes, it must travel from the location where it completed its last query.

Test Case 0:
The minimum distance traveled is 11:

- Robot: R_1
  $M_1 \rightarrow M_5$
  $D_0 = |1 - 5| = 4$ meters.
- Robot: R_2
  $M_3 \rightarrow M_2$
  $D_1 = |3 - 2| = 1$ meter.
- Robot: R_1
  $M_5 \rightarrow M_4 \rightarrow M_1$
  $D_2 = |5 - 4| + |4 - 1| = 1 + 3 = 4$ meters.
- Robot: R_2
  $M_2 \rightarrow M_2 \rightarrow M_4$
  $D_3 = |2 - 2| + |2 - 4| = 0 + 2 = 2$ meters.

Sum the distances traveled ($D_0 + D_1 + D_2 + D_3 = 4 + 1 + 4 + 2 = 11$) and print the result on a new line.

Test Case 1:

- Robot: R_1
  $M_1 \rightarrow M_2$
  $D_0 = |1 - 2| = 1$ meters.
- Robot: R_2
  $M_4 \rightarrow M_3$
  $D_1 = |4 - 3| = 1$ meters.

Sum the distances traveled ($D_0 + D_1 = 1 + 1 = 2$) and print the result on a new line.

Test Case 2:

- Robot: R_1
  $M_2 \rightarrow M_4$
  $D_0 = |2 - 4| = 2$ meters.
- Robot: R_1
  $M_4 \rightarrow M_5 \rightarrow M_4$
  $D_1 = |4 - 5| + |5 - 4| = 1 + 1 = 2$ meters.
- Robot: R_2
  $M_9 \rightarrow M_8$

$D_2 = |9 - 8| = 1$ meters.

Sum the distances traveled ($D_0 + D_1 + D_2 = 2 + 2 + 1 = 5$) and print the result on a new line.

Cut Tree

Given a tree *T* with *n* nodes, how many subtrees (*T'*) of *T* have at most *K* edges connected to (T - T')?

Input Format

The first line contains two integers *n* and *K* followed by *n-1* lines each containing two integers a & b denoting that there's an edge between a & b.

Constraints

1 <= K <= n <= 50
Every node is indicated by a distinct number from 1 to n.

Output Format

A single integer which denotes the number of possible subtrees.

Sample Input

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

Sample Output

```
6
```

Explanation

There are 2^3 possible sub-trees:

{} {1} {2} {3} {1, 2} {1, 3} {2, 3} {1, 2, 3}

But:
the sub-trees {2} and {1,3} are not valid. {2} isn't valid because it has 2 edges connecting to it's complement {1,3} whereas K = 1 in the sample test-case {1,3} isn't valid because, well, it's not a sub-tree. The nodes aren't connected.

Tara's Beautiful Permutations

Tara has an array, A, consisting of n integers where each integer occurs *at most* 2 times in the array.

Let's define P to be a permutation of A where p_i is the i^{th} element of permutation P. Tara thinks a permutation is *beautiful* if there is no index i such that $p_i - p_{i+1} = 0$ where $i \in [0, n-1)$.

You are given q queries where each query consists of some array A. For each A, help Tara count the number of possible beautiful permutations of the n integers in A and print the count, modulo $10^9 + 7$, on a new line.

Note: Two permutations, P and Q, are considered to be *different* if and only if there exists an index i such that $p_i \neq q_i$ and $i \in [0, n)$.

Input Format

The first line contains a single integer, q, denoting the number of queries. The $2 \cdot q$ subsequent lines describe each query in the following form:

1. The first line contains an integer, n, denoting the number of elements in array A.
2. The second line contains n space-separated integers describing the respective values of $a_0, a_1, \ldots, a_{n-1}$ in array A.

Constraints

- $1 \leq a_i \leq 10^9$
- Each integer in A can occur at most 2 times.

For 40% of the maximum score:

- $1 \leq q \leq 100$
- $1 \leq n \leq 1000$
- The sum of n over all queries does not exceed 10^4.

For 100% of the maximum score:

- $1 \leq q \leq 100$
- $1 \leq n \leq 2000$

Output Format

For each query, print the the number of possible beautiful permutations, modulo $10^9 + 7$, on a new line.

Sample Input 0

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

Sample Output 0

```
1
2
2
```

Explanation 0

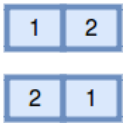
We perform the following $q = 3$ queries:

1. Array $A = [1, 2, 1]$ and there is only one good permutation:

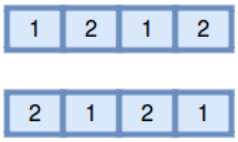
   Thus, we print the result of $1 \bmod (10^9 + 7) = 1$ on a new line.

2. Array $A = [1, 2]$ and there are two good permutations:

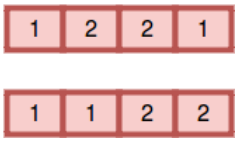
   

   Thus, we print the result of $2 \bmod (10^9 + 7) = 2$ on a new line.

3. Array $A = [1, 2, 2, 1]$ and there are two good permutations:

   For demonstration purposes, the following two permutations are invalid (i.e., not good):

   Because we only want the number of good permutations, we print the result of $2 \bmod (10^9 + 7) = 2$ on a new line.

Wet Shark and Two Subsequences

One day, Wet Shark was given an array $X = \{x_1, x_2, \ldots, x_m\}$. As always, he started playing with its subsequences.

When you came to know about this habit, you presented him a task of finding all pairs of subsequences, (A, B), which satisfies all of the following constraints. We will represent a pair of subsequence as $A = \{x_{a_1}, x_{a_2}, \ldots, x_{a_n}\}$ and $B = \{x_{b_1}, x_{b_2}, \ldots, x_{b_n}\}$

- A and B must be of same length, i.e., $|A| = |B|$.
- $\sum_{i=1}^{n}(x_{a_i} + x_{b_i}) = r$
- $\sum_{i=1}^{n}(x_{a_i} - x_{b_i}) = s$

Please help Wet Shark determine how many possible subsequences A and B can exist. Because the number of choices may be big, output your answer modulo $10^9 + 7 = 1000000007$.

Note:

- Two segments are different if there's exists at least one index i such that element x_i is present in exactly one of them.
- Both subsequences can overlap each other.
- Subsequences do not necessarily have to be distinct

Input Format

The first line consists of 3 space-separated integers m, r, s, where m denotes the length of the original array, X, and r and s are as defined above.
The next line contains m space-separated integers, $x_1, x_2, \ldots, x_m$, representing the elements of X.

Constraints

- $1 \leq m \leq 100$
- $0 \leq r,\ s \leq 2000$
- $1 \leq x_i \leq 2000$

Output Format

Output total number of pairs of subsequences, (A, B), satisfying the above conditions. As the number can be large, output it's modulo $10^9 + 7 = 1000000007$

Sample Input 0

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

Sample Output 0

```
3
```

Explanation 0

For array $X = \{x_1, x_2, x_3, x_4\} = \{1, 1, 1, 4\}$ there are three pairs of subsequences:

1. $A = \{x_4\} = \{4\}; B = \{x_1\} = \{1\}$
2. $A = \{x_4\} = \{4\}; B = \{x_2\} = \{1\}$
3. $A = \{x_4\} = \{4\}; B = \{x_3\} = \{1\}$

Nikita and the Game

Nikita just came up with a new array game. The rules are as follows:

- Initially, there is an array, A, containing N integers.
- In each move, Nikita must partition the array into 2 non-empty contiguous parts such that the sum of the elements in the left partition is equal to the sum of the elements in the right partition. If Nikita can make such a move, she gets 1 point; otherwise, the game ends.
- After each successful move, Nikita discards either the left partition or the right partition and continues playing by using the remaining partition as array A.

Nikita loves this game and wants your help getting the best score possible. Given A, can you find and print the maximum number of points she can score?

Input Format

The first line contains an integer, T, denoting the number of test cases. Each test case is described over 2 lines in the following format:

1. A line containing a single integer, N, denoting the size of array A.
2. A line of N space-separated integers describing the elements in array A.

Constraints

- $1 \leq T \leq 10$
- $1 \leq N \leq 2^{14}$
- $0 \leq A_i \leq 10^9$

Scoring

- $1 \leq N \leq 2^8$ for 30% of the test data
- $1 \leq N \leq 2^{11}$ for 60% of the test data
- $1 \leq N \leq 2^{14}$ for 100% of the test data

Output Format

For each test case, print Nikita's maximum possible score on a new line.

Sample Input

```
3
3
3 3 3
4
2 2 2 2
7
4 1 0 1 1 0 1
```

Sample Output

```
0
2
3
```

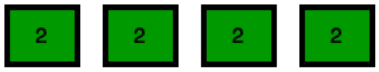
Explanation

Test Case 0:

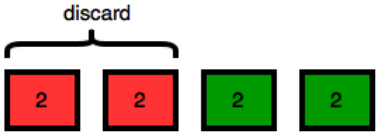
Nikita cannot partition A into 2 parts having equal sums. Therefore, her maximum possible score is 0 and we print 0 on a new line.

Test Case 1:

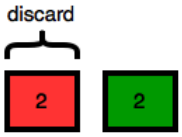
Initially, A looks like this:



She splits the array into 2 partitions having equal sums, and then discards the left partition:



She then splits the new array into 2 partitions having equal sums, and then discards the left partition:



At this point the array only has 1 element and can no longer be partitioned, so the game ends. Because Nikita successfully split the array twice, she gets 2 points and we print 2 on a new line.

Choosing White Balls

There are n balls in a row, and each ball is either *black* (B) or *white* (W). Perform k removal operations with the goal of *maximizing the number of white balls* picked. For each operation i (where $1 \leq i \leq k$):

1. Choose an integer, x_i, uniformly and independently from 1 to $n - i + 1$ (inclusive).
2. Remove the x_i^{th} ball from either the left end or right end of the row, which decrements the number of available balls in the row by 1. You can choose to remove the ball from whichever end in each step maximizing the expected total number of white balls picked at the end.

Given a string describing the initial row of balls as a sequence of n W's and B's, find and print the expected number of *white* balls providing that you make all choices optimally. A correct answer has an *absolute* error of *at most* 10^{-6}.

Input Format

The first line contains two space-separated integers describing the respective values of n (the number of balls) and k (the number of operations).
The second line describes the initial sequence balls as a single string of n characters; each character is either B or W and describes a *black* or *white* ball, respectively.

Constraints

- $1 \leq k \leq n < 30$

Output Format

Print a single floating-point number denoting the expected number of *white* balls picked. Your answer is considered to be correct if it has an *absolute* error of *at most* 10^{-6}.

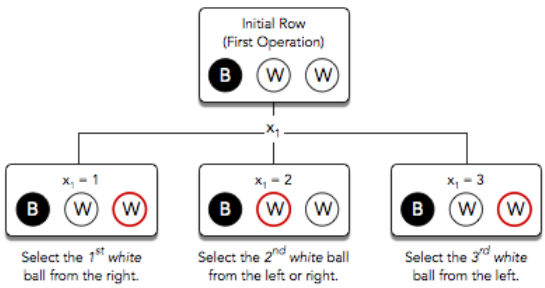
Sample Input 0

```
3 1
BWW
```

Sample Output 0

```
1.0000000000
```

Explanation 0



Independent of your choice of x, one *white* ball will always be picked so the expected number of *white* balls chosen after $k = 1$ operation is 1. Thus, we print 1 as our answer.

Sample Input 1

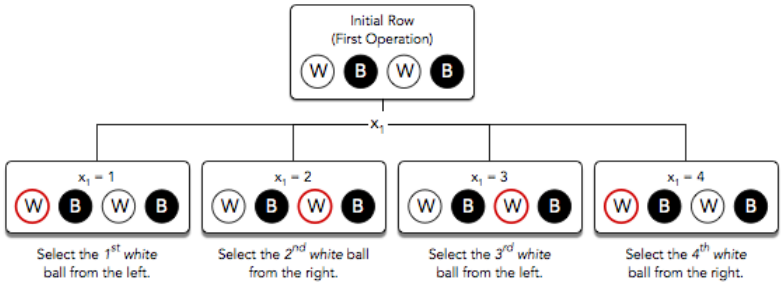
```
4 2
WBWB
```

Sample Output 1

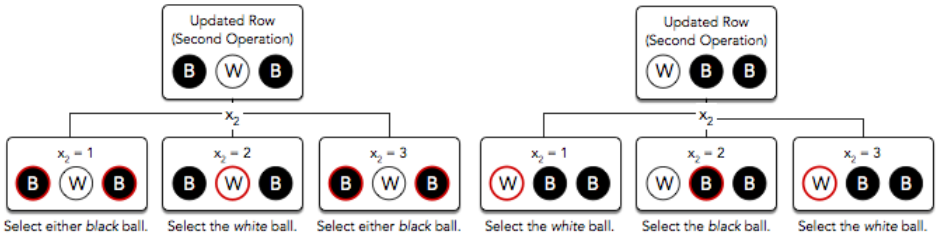
```
1.5000000000
```

Explanation 1

We perform the following $k = 2$ operations:

1. 

   Independent of your choice of x, a *white* ball will always be chosen during the first operation (meaning the expected number of *white* balls in the first operation is 1).

2. 

   For the second operation, there are 2 possible row orderings (depending on which ball was picked during the first operation). In the first possible row ordering, the probability of picking a *white* ball is $\frac{1}{3}$. In the second possible row ordering, the probability of picking a *white* ball is $\frac{2}{3}$. This means the expected number of *white* balls chosen in the second operation is $\frac{1}{2} \cdot \frac{1}{3} + \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{2}$.

After performing all $k = 2$ operations, we print the total expected number of *white* balls chosen, which is $1 + \frac{1}{2} = 1.5$.

Mandragora Forest

The evil forest is guarded by N vicious mandragoras. Each i^{th} mandragora has H_i health points ($1 \leq i \leq N$).

Garnet and her pet begin their journey through the evil forest with $S = 1$ strength points and $P = 0$ experience points. For each undefeated mandragora i, she can perform *either* of the following actions:

1. Garnet's pet *eats* mandragora i. This increments S by 1 and defeats mandragora i.
2. Garnet's pet *battles* mandragora i. This increases P by $S \times H_i$ experience points and defeats mandragora i.

Each mandragora can only be defeated once, and Garnet can defeat the mandragoras in any order. Given the respective health points for each mandragora, can you find the maximum number of experience points she can earn from defeating all N mandragoras?

Input Format

The first line contains an integer, T, denoting the number of test cases. Each test case is described over two lines:

1. The first line contains a single integer, N, denoting the number of mandragoras in the forest.
2. The second line contains N space-separated integers describing the respective health points for the mandragoras (i.e., $H_1, H_2, \ldots, H_N$).

Constraints

- $1 \leq T \leq 10^5$
- $1 \leq N \leq 10^5$
- $1 \leq H_i \leq 10^7$, where $1 \leq i \leq N$
- The sum of all Ns in a single test case is $\leq 10^6$

Output Format

For each test case, print a single line with an integer denoting the maximum number of experience points that Garnet can earn.

Sample Input

```
1
3
3 2 2
```

Sample Output

```
10
```

Explanation

There are $N = 3$ mandragoras having the following health points: $H = [3, 2, 2]$. Initially, $S = 1$ and $P = 0$. The following is an optimal sequence of actions for achieving the maximum number of experience

points possible:

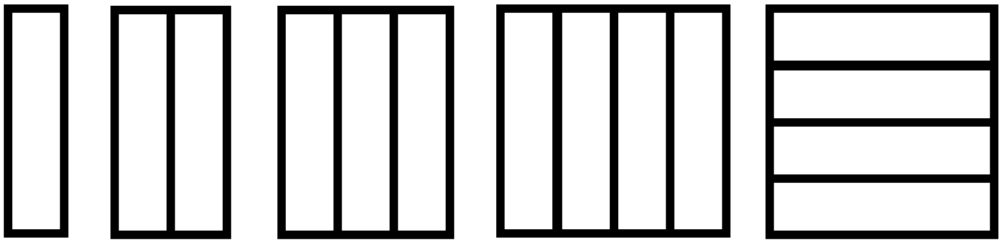
1. *Eat* the second mandragora ($H_1 = 2$). S is increased from 1 to 2, and P is still 0.
2. *Battle* the first mandragora ($H_0 = 3$). S remains the same, but P increases by $S \times H_0 = 2 \times 3 = 6$ experience points.
3. *Battle* the third mandragora ($H_2 = 2$). S remains the same, but P increases by $S \times H_2 = 2 \times 2 = 4$ experience points.

Garnet earns $P = 6 + 4 = 10$ experience points, so we print 10 on a new line.

Red John is Back

Red John has committed another murder. This time, he doesn't leave a red smiley behind. Instead he leaves a puzzle for Patrick Jane to solve. He also texts Teresa Lisbon that if Patrick is successful, he will turn himself in. The puzzle begins as follows.

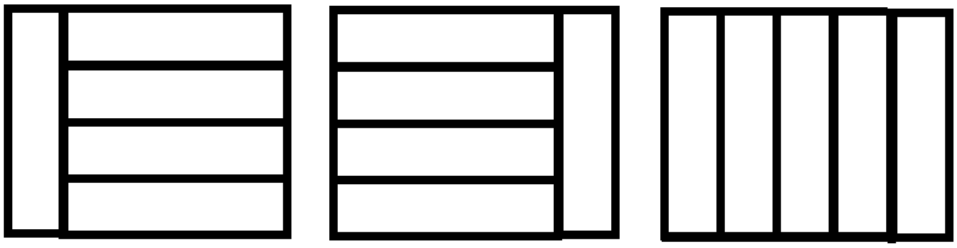
There is a wall of size *4xn* in the victim's house. The victim has an infinite supply of bricks of size *4x1* and *1x4* in her house. There is a hidden safe which can only be opened by a particular configuration of bricks. First we must calculate the total number of ways in which the bricks can be arranged so that the entire wall is covered. The following diagram shows how bricks might be arranged to cover walls where $1 \leq n \leq 4$:



There is one more step to the puzzle. Call the number of possible arrangements M. Patrick must calculate the number of prime numbers P in the inclusive range $0 - M$.

As an example, assume $n = 3$. From the diagram above, we determine that there is only one configuration that will cover the wall properly. 1 is not a prime number, so $P = 0$.

A more complex example is $n = 5$. The bricks can be oriented in 3 total configurations that cover the wall. The two primes 2 and 3 are less than or equal to 3, so $P = 2$.



Function Description

Complete the *redJohn* function in the editor below. It should return the number of primes determined, as an integer.

redJohn has the following parameter(s):

- *n*: an integer that denotes the length of the wall

Input Format

The first line contains the integer t, the number of test cases.
Each of the next t lines contains an integer n, the length of the $4 \times n$ wall.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 40$

Output Format

Print the integer P on a separate line for each test case.

Sample Input

```
2
1
7
```

Sample Output

```
0
3
```

Explanation

For $n = 1$, the brick can be laid in 1 format only: vertically.

The number of primes ≤ 1 is 0.

For $n = 7$, one of the ways in which we can lay the bricks is

There are 5 ways of arranging the bricks for $n = 7$ and there are 3 primes ≤ 5.

LCS Returns

Given two strings, a and b, find and print the total number of ways to insert a character at any position in string a such that the length of the Longest Common Subsequence of characters in the two strings increases by one.

Input Format

The first line contains a single string denoting a.
The second line contains a single string denoting b.

Constraints

Scoring

- $1 \leq |a|, |b| \leq 5000$
- Strings a and b are alphanumeric (i.e., consisting of arabic digits and/or upper and lower case English letters).
- The new character being inserted must also be alphanumeric (i.e., a digit or upper/lower case English letter).

Subtask

- $1 \leq |a|, |b| \leq 1000$ for 66.67% of the maximum score.

Output Format

Print a single integer denoting the total number of ways to insert a character into string a in such a way that the length of the longest common subsequence of a and b increases by one.

Sample Input

```
aa
baaa
```

Sample Output

```
4
```

Explanation

The longest common subsequence shared by $a =$ **"aa"** and $b =$ **"baaa"** is `aa`, which has a length of 2. There are two ways that the length of the longest common subsequence can be increased to 3 by adding a single character to a:

1. There are 3 different positions in string a where we could insert an additional `a` to create longest common subsequence `aaa` (i.e., at the beginning, middle, and end of the string).
2. We can insert a `b` at the beginning of the string for a new longest common subsequence of `baa`.

As we have $3 + 1 = 4$ ways to insert an alphanumeric character into a and increase the length of the longest common subsequence by one, we print 4 on a new line.

Grid Walking

You are situated in an N dimensional grid at position $(x_1, x_2, \ldots, x_N)$. The dimensions of the grid are $(D_1, D_2, \ldots D_N)$. In one step, you can walk one step ahead or behind in any one of the N dimensions. (So there are always $2 \times N$ possible different moves). In how many ways can you take M steps such that you do not leave the grid at any point? You leave the grid if at any point x_i, either $x_i \leq 0$ or $x_i > D_i$.

Input Format

The first line contains the number of test cases T. T test cases follow. For each test case, the first line contains N and M, the second line contains $x_1, x_2, \ldots, x_N$ and the 3rd line contains $D_1, D_2, \ldots, D_N$.

Constraints

- $1 \leq T \leq 10$
- $1 \leq N \leq 10$
- $1 \leq M \leq 300$
- $1 \leq D_i \leq 100$
- $1 \leq x_i \leq D_i$

Output Format

Output T lines, one corresponding to each test case. Since the answer can be really huge, output it modulo 1000000007.

Sample Input

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

Sample Output

```
12
```

Explanation

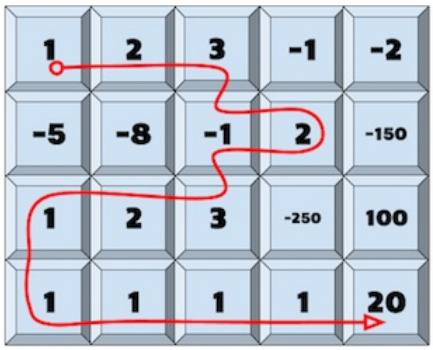
Starting from (1, 1) in a 2×3 2-D grid, and need to count the number of possible paths with length equal to 3. Here are the 12 paths:

```
(1, 1) -> (1, 2) -> (1, 1) -> (1, 2)
(1, 1) -> (1, 2) -> (1, 1) -> (2, 1)
(1, 1) -> (1, 2) -> (1, 3) -> (1, 2)
(1, 1) -> (1, 2) -> (1, 3) -> (2, 3)
(1, 1) -> (1, 2) -> (2, 2) -> (1, 2)
(1, 1) -> (1, 2) -> (2, 2) -> (2, 1)
(1, 1) -> (1, 2) -> (2, 2) -> (2, 3)
(1, 1) -> (2, 1) -> (1, 1) -> (1, 2)
(1, 1) -> (2, 1) -> (1, 1) -> (2, 1)
(1, 1) -> (2, 1) -> (2, 2) -> (2, 1)
(1, 1) -> (2, 1) -> (2, 2) -> (1, 2)
(1, 1) -> (2, 1) -> (2, 2) -> (2, 3)
```

Matrix Land

You are playing a matrix-based game with the following setup and rules:

- You are given a matrix A with n rows and m columns. Each cell contains some points. When a player passes a cell her score increases by the number written in that cell and the number in the cell becomes 0. (If the cell number is positive her score increases, otherwise it decreases.)
- The player starts from any cell in the *first* row and can move *left*, *right* or *down*.
- The game is over when the player reaches the *last* row and stops moving.



Input Format

The first line contains n and m. The next n lines contain m numbers each, j^{th} number in i^{th} line denotes the number that is written on cell $A_{i,j}$.

Constraints

- $1 \leq n \times m \leq 4 \times 10^6$
- $-250 \leq A_{i,j} \leq 250$

Subtasks

- for 20% tests $1 \leq n, m \leq 40$.
- for 20% tests $40 < n, m \leq 500$.

Output Format

Print the maximum score that the player can get.

Sample Input 0

```
4 5
1 2 3 -1 -2
-5 -8 -1 2 -150
1 2 3 -250 100
1 1 1 1 20
```

Sample Output 0

```
37
```

Explanation 0

Refer the image given in statement, the path followed is $1, 2, 3, -1, 2, -1, 3, 2, 1, 1, 1, 1, 1, 20$ summing upto 37.
Note that, -1 is traversed 2 times, but the second time it only contributes 0 to the sum.

Knapsack

Given an array of integers and a target sum, determine the sum nearest to but not exceeding the target that can be created. To create the sum, use any element of your array zero or more times.

For example, if $arr = [2, 3, 4]$ and your target sum is 10, you might select $[2, 2, 2, 2, 2]$, $[2, 2, 3, 3]$ or $[3, 3, 3, 1]$. In this case, you can arrive at exactly the target.

Function Description

Complete the *unboundedKnapsack* function in the editor below. It must return an integer that represents the sum nearest to without exceeding the target value.

unboundedKnapsack has the following parameter(s):

- *k*: an integer
- *arr*: an array of integers

Input Format

The first line contains an integer t, the number of test cases.

Each of the next t pairs of lines are as follows:
- The first line contains two integers n and k, the length of arr and the target sum.
- The second line contains n space separated integers $arr[i]$.

Constraints

$1 \leq t \leq 10$
$1 \leq n, k, arr[i] \leq 2000$

Output Format

Print the maximum sum for each test case which is as near as possible, but not exceeding, to the target sum on a separate line.

Sample Input

```
2
3 12
1 6 9
5 9
3 4 4 4 8
```

Sample Output

```
12
9
```

Explanation

In the first test case, one can pick {6, 6}. In the second, we can pick {3,3,3}.

Bricks Game

Русский \| □□

You and your friend decide to play a game using a stack consisting of N bricks. In this game, you can alternatively remove 1, 2 or 3 bricks from the top, and the numbers etched on the removed bricks are added to your score. You have to play so that you obtain the maximum possible score. It is given that your friend will also play optimally and you make the first move.

Input Format

First line will contain an integer T i.e. number of test cases. There will be two lines corresponding to each test case: first line will contain a number N i.e. number of elements in the stack and next line will contain N numbers i.e. numbers etched on bricks from top to bottom.

Constraints

$1 \leq T \leq 5$
$1 \leq N \leq 10^5$
$0 \leq$ each number on brick $\leq 10^9$

Output Format

For each test case, print a single line containing your maximum score.

Sample Input

```
2
5
999 1 1 1 0
5
0 1 1 1 999
```

Sample Output

```
1001
999
```

Explanation

In first test case, you will pick 999,1,1. If you play in any other way, you will not get a score of 1001.
In second case, best option will be to pick up the first brick (with 0 score) at first. Then your friend will choose the next three blocks, and you will get the last brick.

The Longest Increasing Subsequence

An Introduction to the Longest Increasing Subsequence Problem

The task is to find the length of the longest subsequence in a given array of integers such that all elements of the subsequence are sorted in strictly ascending order. This is called the Longest Increasing Subsequence (LIS) problem.

For example, the length of the LIS for $[15, 27, 14, 38, 26, 55, 46, 65, 85]$ is 6 since the longest increasing subsequence is $[15, 27, 38, 55, 65, 85]$.

Here's a great YouTube video of a lecture from MIT's Open-CourseWare covering the topic.

This is one approach which solves this in quadratic time using dynamic programming. A more efficient algorithm which solves the problem in $O(n \log n)$ time is available here.

Given a sequence of integers, find the length of its longest strictly increasing subsequence.

Function Description

Complete the *longestIncreasingSubsequence* function in the editor below. It should return an integer that denotes the array's LIS.

longestIncreasingSubsequence has the following parameter(s):

- *arr*: an unordered array of integers

Input Format

The first line contains a single integer n, the number of elements in arr.
Each of the next n lines contains an integer, $arr[i]$

Constraints

- $1 \leq n \leq 10^6$
- $1 \leq arr[i] \leq 10^5$

Output Format

Print a single line containing a single integer denoting the length of the longest increasing subsequence.

Sample Input 0

```
5
2
```

```
7
4
3
8
```

Sample Output 0

```
3
```

Explanation 0

In the array $arr = [2, 7, 4, 3, 8]$, the longest increasing subsequence is $[2, 7, 8]$. It has a length of 3.

Sample Input 1

```
6
2
4
3
7
4
5
```

Sample Output 1

```
4
```

Explanation 1

The LIS of $arr = [2, 4, 3, 7, 4, 5]$ is $[2, 3, 4, 5]$.

Coin on the Table

You have a rectangular board consisting of N rows, numbered from 1 to N, and M columns, numbered from 1 to M. The top left is $(1,1)$ and the bottom right is (N,M). Initially - at time 0 - there is a coin on the top-left cell of your board. Each cell of your board contains one of these letters:

- *: Exactly one of your cells has letter '*'.
- **U**: If at time t the coin is on cell (i,j) and cell (i,j) has letter 'U', the coin will be on cell $(i-1,j)$ at time $t+1$, if $i>1$. Otherwise, there is no coin on your board at time $t+1$.
- **L**: If at time t the coin is on cell (i,j) and cell (i,j) has letter 'L', the coin will be on cell $(i,j-1)$ at time $t+1$, if $j>1$. Otherwise, there is no coin on your board at time $t+1$.
- **D**: If at time t the coin is on cell (i,j) and cell (i,j) has letter 'D', the coin will be on cell $(i+1,j)$ at time $t+1$, if $i<N$. Otherwise, there is no coin on your board at time $t+1$.
- **R**: If at time t the coin is on cell (i,j) and cell (i,j) has letter 'R', the coin will be on cell $(i,j+1)$ at time $t+1$, if $j<M$. Otherwise, there is no coin on your board at time $t+1$.

When the coin reaches a cell that has letter '*', it will stay there permanently. When you punch on your board, your timer starts and the coin moves between cells. Before starting the game, you can make operations to change the board, such that you are sure that at or before time K the coin will reach the cell having letter '*'. In each operation you can select a cell with some letter other than '*' and change the letter to '*U*', '*L*', '*R*' or '*D*'. You need to carry out as few operations as possible in order to achieve your goal. Your task is to find the minimum number of operations.

Input:
The first line of input contains three integers, N, M, and K, respectively. The next N lines contain M letters each, describing your board.

Output:
Print an integer which represents the minimum number of operations required to achieve your goal. If you cannot achieve your goal, print -1.

Constraints
$N, M \leq 51$
$K \leq 1000$

Sample input :

```
2 2 3
RD
*L
```

Sample output :

```
0
```

Sample input :

```
2 2 1
RD
```

```
*L
```

Sample output :

```
1
```

Explanation :

In the first example, you don't have to change any letter; but in the second example, you should change the letter of cell (1,1) to 'D'.

The Longest Common Subsequence

A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements. Longest common subsequence (*LCS*) of 2 sequences is a subsequence, with maximal length, which is common to both the sequences.

Given two sequence of integers, $A = [a_1, a_2, \ldots, a_n]$ and $B = [b_1, b_2, \ldots, b_m]$, find **any one** longest common subsequence.

In case multiple solutions exist, print any of them. It is guaranteed that at least one non-empty common subsequence will exist.

Recommended References

This Youtube video tutorial explains the problem and its solution quite well.

Input Format

First line contains two space separated integers, n and m, where n is the size of sequence A, while m is size of sequence B. In next line there are n space separated integers representing sequence A, and in third line there are m space separated integers representing sequence B.

```
n m
A1 A2 ... An
B1 B2 ... Bm
```

Constraints

$1 \leq n \leq 100$
$1 \leq m \leq 100$
$0 \leq a_i < 1000, where\ i \in [1, n]$
$0 \leq b_j < 1000, where\ j \in [1, m]$

Output Format

Print the longest common subsequence and each element should be separated by at least one white-space. In case of multiple answers, print any one of them.

Sample Input

```
5 6
1 2 3 4 1
3 4 1 2 1 3
```

Sample Output

```
1 2 3
```

Explanation

There is no common subsequence with length larger than 3. And "1 2 3", "1 2 1", "3 4 1" are all correct answers.

Tested by Khongor

Play with words

Shaka and his brother have created a boring game which is played like this:

They take a word composed of lowercase English letters and try to get the maximum possible score by building exactly 2 **palindromic subsequences**. The score obtained is the product of the length of these 2 subsequences.

Let's say A and B are two subsequences from the initial string. If A_i & A_j are the smallest and the largest positions (from the initial word) respectively in A ; and B_i & B_j are the smallest and the largest positions (from the initial word) respectively in B, then the following statements hold true:
$A_i \leq A_j$,
$B_i \leq B_j$, &
$A_j < B_i$.
i.e., the positions of the subsequences should not cross over each other.

Hence the score obtained is the product of lengths of subsequences A & B. Such subsequences can be numerous for a larger initial word, and hence it becomes harder to find out the maximum possible score. Can you help Shaka and his brother find this out?

Input Format

Input contains a word S composed of lowercase English letters in a single line.

Constraints

$1 < |S| \leq 3000$
each character will be a lower case english alphabet.

Output Format

Output the maximum score the boys can get from S.

Sample Input

```
eeegeeksforskeeggeeks
```

Sample Output

```
50
```

Explanation

A possible optimal solution is **eee**-g-**ee**-ksfor-**skeeggeeks** being **eeeee** the one subsequence and **skeeggeeks** the other one. We can also select **eegee** in place of **eeeee**, as both have the same length.

Black and White Tree

Nikita is making a graph as a birthday gift for her boyfriend, a fellow programmer! She drew an undirected connected graph with N nodes numbered from 1 to N in her notebook.

Each node is shaded in either *white* or *black*. We define n_W to be the number of white nodes, and n_B to be the number of black nodes. The graph is drawn in such a way that:

- No 2 adjacent nodes have same coloring.
- The value of $|n_W - n_B|$, which we'll call D, is minimal.

Nikita's mischievous little brother erased some of the edges and all of the coloring from her graph! As a result, the graph is now decomposed into one or more components. Because you're her best friend, you've decided to help her reconstruct the graph by adding K edges such that the aforementioned graph properties hold true.

Given the decomposed graph, construct and shade a valid connected graph such that the difference $|n_W - n_B|$ between its shaded nodes is minimal.

Input Format

The first line contains 2 space-separated integers, N (the number of nodes in the original graph) and M (the number of edges in the decomposed graph), respectively.
The M subsequent lines each contain 2 space-separated integers, u and v, describing a bidirectional edge between nodes u and v in the decomposed graph.

Constraints

- $1 \leq N \leq 2 \times 10^5$
- $0 \leq M \leq min(5 \times 10^5, \frac{N \times (N-1)}{2})$
- It is guaranteed that every edge will be between 2 distinct nodes, and there will never be more than 1 edge between any 2 nodes.
- Your answer *must* meet the following criteria:
  - The graph is connected and no 2 adjacent nodes have the same coloring.
  - The value of $|n_B - n_W|$ is minimal.
  - $K \leq 2 \times 10^5$

Output Format

You must have $K + 1$ lines of output. The first line contains 2 space-separated integers: D (the minimum possible value of $|n_B - n_W|$) and K (the number of edges you've added to the graph), respectively.
Each of the K subsequent lines contains 2 space-separated integers, u and v, describing a newly-added bidirectional edge in your final graph (i.e.: new edge $u \leftrightarrow v$).

You may print *any* 1 of the possible reconstructions of Nikita's graph such that the value of D in the reconstructed shaded graph is minimal.

Sample Input 0

```
8 8
1 2
2 3
3 4
4 1
1 5
2 6
3 7
4 8
```

Sample output 0

```
0 0
```

Sample Input 1

```
8 6
1 2
3 4
3 5
3 6
3 7
3 8
```

Sample Output 1

```
4 1
1 5
```

Sample Input 2

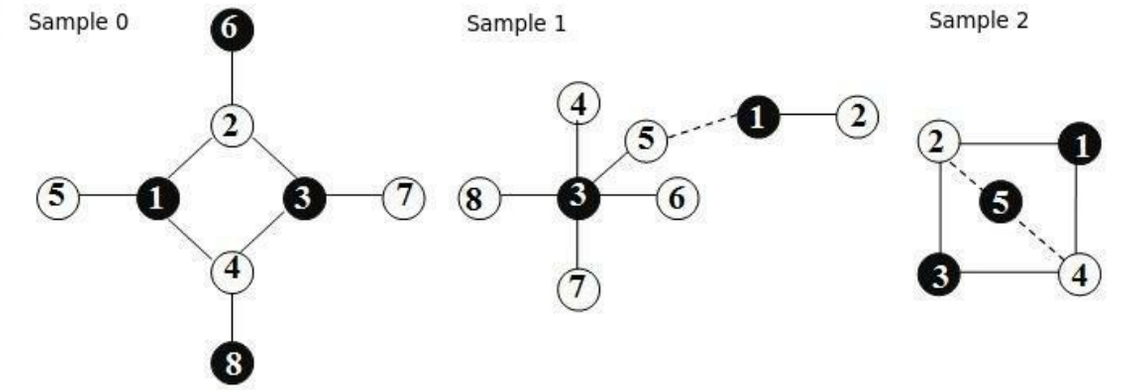
```
5 4
1 2
2 3
3 4
4 1
```

Sample Output 2

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

Explanation

In the figure below, the solid lines show the decomposed graph after Nikita's brother erased the edges, and the dotted lines show one possible correct answer:



In *Sample* 0, no additional edges are added and $K = 0$. Because $n_W = 4$ and $n_B = 4$, we get $|n_W - n_B| = 0$. Thus, we print `0 0` on a new line (there is only 1 line of output, as $K = 0$).

In *Sample* 1, the only edge added is $(5, 1)$, so $K = 1$. Here, $n_W = 6$ and $n_B = 2$, so $|n_W - n_B| = 4$. Thus, we print `4 1` on the first line. Next, we must print K lines describing each edge added; because $K = 1$, we print a single line describing the 2 space-separated nodes connected by our new edge: `1 5`.

Counting Special Sub-Cubes

Given an $n \times n \times n$ *cube*, let $f(x, y, z)$ (where $1 \leq x, y, z \leq n$) denote the value stored in cell (x, y, z).

A $k \times k \times k$ *sub-cube* (where $1 \leq k \leq n$) of an $n \times n \times n$ cube is considered to be *special* if the maximum value stored in any cell in the sub-cube is equal to k.

For each k in the inclusive range $[1, n]$, calculate the number of special sub-cubes. Then print each $count_k$ as a single line of space-separated integers (i.e., $count_1$ $count_2$ $\ldots$ $count_n$).

Input Format

The first line contains an integer, q, denoting the number of queries. The $2 \cdot q$ subsequent lines describe each query over two lines:

1. The first line contains an integer, n, denoting the side length of the initial cube.
2. The second line contains n^3 space-separated integers describing an array of n^3 integers in the form $a_0, a_1, \ldots, a_{n^3-1}$. The integer in some cell (x, y, z) is calculated using the formula $a[(x-1) \cdot n^2 + (y-1) \cdot n + z]$.

Constraints

- $1 \leq q \leq 5$
- $1 \leq n \leq 50$
- $1 \leq f(x, y, z) \leq n$ where $1 \leq x, y, z \leq n$

Output Format

For each query, print n space-separated integers where the i^{th} integer denotes the number of special sub-cubes for $k = i$.

Sample Input

```
2
2
2 1 1 1 1 1 1 1
2
1 1 1 1 2 1 1 2
```

Sample Output

```
7 1
6 1
```

Explanation

We must perform the following $q = 2$ queries:

1. We have a cube of size $n = 2$ and must calculate the number of special sub-cubes for the following values of k:
    - $k = 1$: There are $2^3 = 8$ sub-cubes of size 1 and seven of them have a maximum value of 1 written inside them. So, for $k = 1$, the answer is 7.
    - $k = 2$: There is only one sub-cube of size 2 and the maximum number written inside it is 2. So, for $k = 2$, the answer is 1.

We then print the respective values for each k as a single line of space-separated integers (i.e., `7 1`).

2. We have a cube of size $n = 2$ and must calculate the number of special sub-cubes for the following values of k:

   - $k = 1$: There are $2^3 = 8$ sub-cubes of size 1 and six of them have a maximum value of 1 written inside them. So, for $k = 1$, the answer is 6.
   - $k = 2$: There is only one sub-cube of size 2 and the maximum number written inside it is 2. So, for $k = 2$, the answer is 1.

   We then print the respective values for each k as a single line of space-separated integers (i.e., `6 1`).

Interval Selection

Given a set of n intervals, find the size of its largest possible subset of intervals such that no three intervals in the subset share a common point.

Input Format

The first line contains an integer, s, denoting the number of interval sets you must find answers for. The $s \cdot (n+1)$ subsequent lines describe each of the s interval sets as follows:

1. The first line contains an integer, n, denoting the number of intervals in the list.
2. Each line i of the n subsequent lines contains two space-separated integers describing the respective starting (a_i) and ending (b_i) boundaries of an interval.

Constraints

- $1 \leq s \leq 100$
- $2 \leq n \leq 1000$
- $1 \leq a_i \leq b_i \leq 10^9$

Output Format

For each of the s interval sets, print an integer denoting the size of the largest possible subset of intervals in the given set such that no three points in the subset overlap.

Sample Input

```
4
3
1 2
2 3
2 4
3
1 5
1 5
1 5
4
1 10
1 3
4 6
7 10
4
1 10
1 3
3 6
7 10
```

Sample Output

```
2
2
4
3
```

Explanation

For set s_0, all three intervals fall on point 2 so we can only choose any 2 of the intervals. Thus, we print 2 on a new line.

For set s_1, all three intervals span the range from 1 to 5 so we can only choose any 2 of them. Thus, we print 2 on a new line.

For set s_2, we can choose all 4 intervals without having more than two of them overlap at any given point. Thus, we print 4 on a new line.

For set s_3, the intervals $[1, 10]$, $[1, 3]$, and $[3, 6]$ all overlap at point 3, so we must only choose 2 of these intervals to combine with the last interval, $[7, 10]$, for a total of 3 qualifying intervals. Thus, we print 3 on a new line.

String Reduction

Given a string consisting of letters, 'a', 'b' and 'c', we can perform the following operation:

- Take any two adjacent distinct characters and replace them with the third character.

For example, if 'a' and 'c' are adjacent, they can replaced by 'b'.

Find the smallest string which we can obtain by applying this operation repeatedly.

Input Format

The first line contains the number of test cases T. T test cases follow. Each test case contains the string you start with.

Constraints

- $1 \leq T \leq 100$
- The string will have at most 100 characters.

Output Format

Output T lines, one for each test case, containing the smallest length of the resultant string after applying the operations optimally.

Sample Input

```
3
cab
bcab
ccccc
```

Sample Output

```
2
1
5
```

Explanation

For the first case, you can either get $cab -> cc$ or $cab -> bb$, resulting in a string of length 2.
For the second case, one optimal solution is: $bcab -> aab -> ac -> b$. No more operations can be applied and the resultant string has length 1.
For the third case, no operations can be performed. So the answer is 5.

Far Vertices

You are given a tree that has N vertices and N-1 edges. Your task is to mark as small number of vertices as possible, such that, the maximum distance between two unmarked vertices is less than or equal to K. Output this value. Distance between two vertices i and j is defined as the minimum number of edges you have to pass in order to reach vertex i from vertex j.

Input Format

The first line of input contains two integers N and K. The next N-1 lines contain two integers (ui,vi) each, where 1 <= ui,vi <= N. Each of these lines specifies an edge.
N is no more than 100. K is less than N.

Output Format

Print an integer that denotes the result of the test.

Sample Input:

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

Sample Output:

```
3
```

Sample Input:

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

Sample Output:

```
0
```

Explanation:

In the first case you have to mark at least 3 vertices, and in the second case you don't need to mark any vertices.

Counting Road Networks

Lukas is a Civil Engineer who loves designing road networks to connect n cities numbered from 1 to n. He can build any number of bidirectional roads as long as the resultant network satisfies these constraints:

1. It must be possible to reach any city from any other city by traveling along the network of roads.
2. No two roads can directly connect the same two cities.
3. A road cannot directly connect a city to itself.

In other words, the roads and cities must form a simple connected labeled graph.

You must answer q queries, where each query consists of some n denoting the number of cities Lukas wants to design a bidirectional network of roads for. For each query, find and print the number of ways he can build roads connecting n cities on a new line; as the number of ways can be quite large, print it modulo 663224321.

Input Format

The first line contains an integer, q, denoting the number of queries.
Each of the q subsequent lines contains an integer denoting the value of n for a query.

Constraints

- $1 \leq q, n \leq 10^5$

Output Format

For each of the q queries, print the number of ways Lukas can build a network of bidirectional roads connecting n cities, modulo 663224321, on a new line.

Sample Input 0

```
3
1
3
10
```

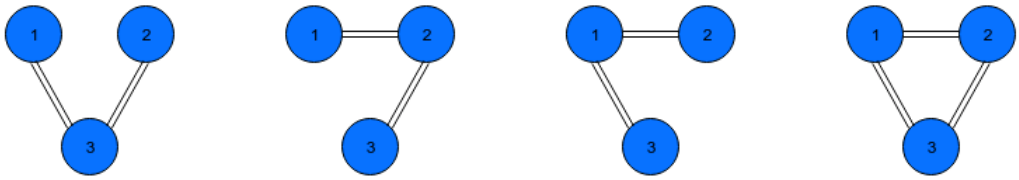
Sample Output 0

```
1
4
201986643
```

Explanation 0

We answer the first two queries like this:

1. When $n = 1$, the only option satisfying Lukas' three constraints is to not build any roads at all. Thus, we print the result of $1 \bmod 663224321 = 1$ on a new line.
2. When $n = 3$, there are four ways for Lukas to build roads that satisfy his three constraints:



Thus, we print the result of $4 \bmod 663224321 = 4$ on a new line.

Superman Celebrates Diwali

Superman has been invited to India to celebrate Diwali. Unfortunately, on his arrival he learns that he has been invited mainly to help rescue people from a fire accident that has happened in a posh residential locale of New Delhi, where rescue is proving to be especially difficult. As he reaches the place of the fire, before him there are N buildings, each of the same height H, which are on fire. Since it is Diwali, some floors of the buildings are empty as the occupants have gone elsewhere for celebrations. In his hurry to start the rescue Superman reaches the top of the building, but realizes that his jumping power is depleted and restricted due to change in his geographical setting. He soon understands the restrictions of his jumping power, and they are as follows:

- He can use the jumping power any number of times until he reaches the bottom floor, which means he can use the jumping power only until before he reaches the bottom (Ground floor), which means, once he reaches the bottom floor, he cannot move to the top floor again and try to save people. (In one single drop from the top to bottom)
- While switching buildings, he loses height I while jumping.

The second restriction is explained below with an example.

Assume $I = 2$. Now Superman is in the 2nd building 5th floor ($B = 2$, $F = 5$). If he wants to switch to the fifth building ($B = 5$), he will lose height ($I = 2$), which means he will be at floor *3* at building *5* ($B = 5$, $F = 3$). He can jump freely from the current floor to the floor below on the same building . That is, suppose if he is at $(B = 5, F = 4)$, he can go to $(B = 5, F = 3)$ without any restrictions. He cannot skip a floor while jumping in the same building. He can go to the floor below the current floor of the same building or use his jumping power, switch building, and lose height I.

Given the information about the occupied floors in each of the N buildings, help Superman to determine the maximum number of people he can save in one single drop from the top to the bottom floor with the given restrictions.

Input Format

Input starts with three values: the number of buildings N, the height of the buildings H, and the height Superman will lose when he switches buildings I.

These are followed by N lines. Each i^{th} line starts with a non negative integer u indicating how many people are in the i^{th} building. Each of the following u integers indicates that a person is at height u_i in the i^{th} buiding. Each of the following u integers are given and repetitions are allowed which means there can be more than one person in a floor.

i indicates building number and j indicates floor number. Building number will not be given; since N lines follow the first line, you can assume that the i^{th} line indicates the i^{th} building's specifications.

Constraints
$1 \leq H, N \leq 1900$
$1 \leq I \leq 450$
$0 \leq u \leq 1900$ (for each i, which means the maximum number of people in a particular building will not exceed 1900)
$1 \leq u_{ij} \leq H$

Output Format

Output the maximum number of people Superman can save.

Sample Input

```
4 15 2
5 1 1 1 4 10
8 9 5 7 7 3 9 8 8
5 9 5 6 4 3
0
```

Sample Output

```
12
```

Explanation

Input starts with $N = 4$, $H = 15$, $I = 2$.

N lines follow. Each line describes building i.

Each line begins with u, which denotes the number of persons in a particular building, followed by floor number, where each person resides. Floor number can repeat as any number of people can reside on a particular floor.

I've attached a figure here to explain the sample test case.

You can verify the first building's specifications with the figure.

$u = 5$ (Total number of persons in the first building), followed by 1 1 1 4 10(Floor numbers).

$1st$ floor = 3 persons.

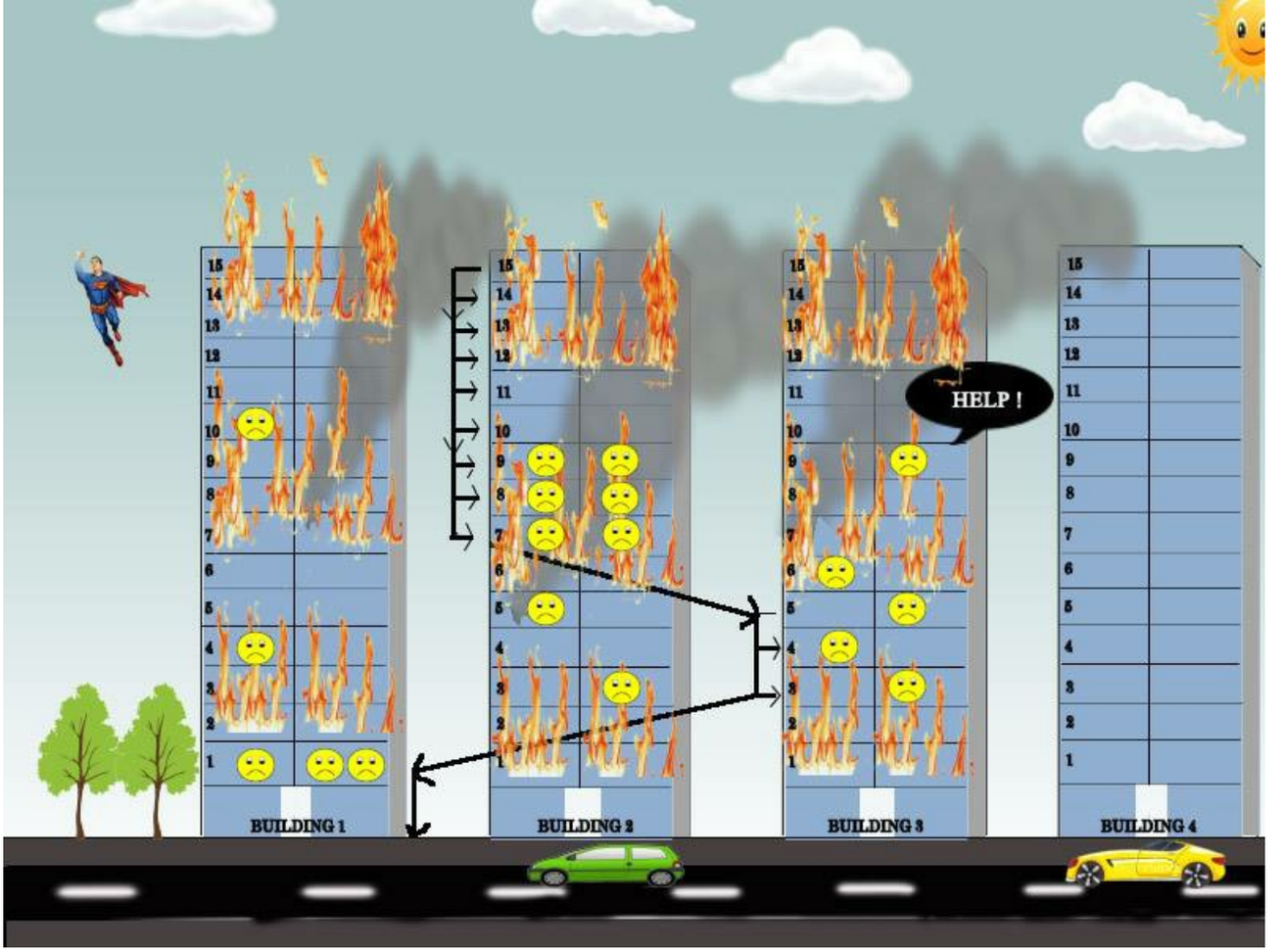
$4th$ floor = 1 person.

$10th$ floor = 1 person.

Similarly, the specifications for the other three buildings follow.

The connected line shows the path which Superman can use to save the maximum number of people. In this case, that number is 12.

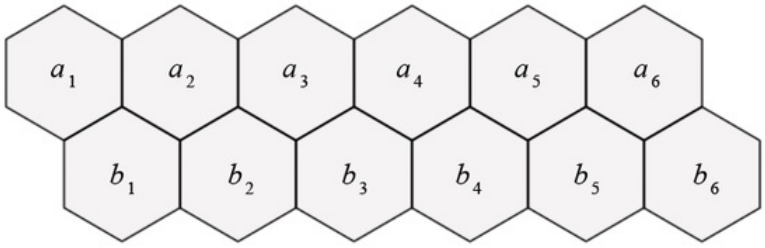
You can also note in the figure that when he switches from Building *2* to Building *3*, he loses height I ($I = 2$). Similarly, when he switches from Building *3* to Building *1* ,the same height loss happens as mentioned in the problem statement.


HELP !
BUILDING 1
BUILDING 2
BUILDING 3
BUILDING 4

Hexagonal Grid

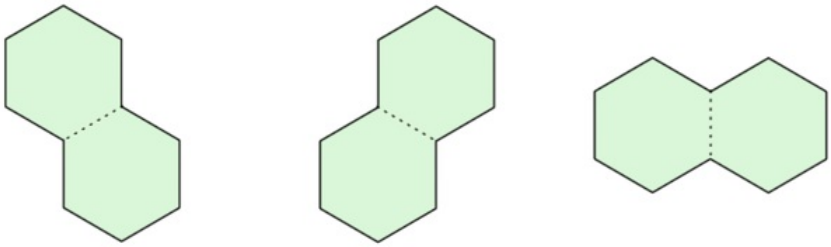
You are given a hexagonal grid consisting of two rows, each row consisting of n cells. The cells of the first row are labelled $a_1, a_2, \ldots a_n$ and the cells of the second row are labelled $b_1, b_2, \ldots, b_n$.

For example, for $n = 6$:



(Note that the b_i is connected with a_{i+1}.)

Your task is to tile this grid with 2×1 *tiles* that look like the following:

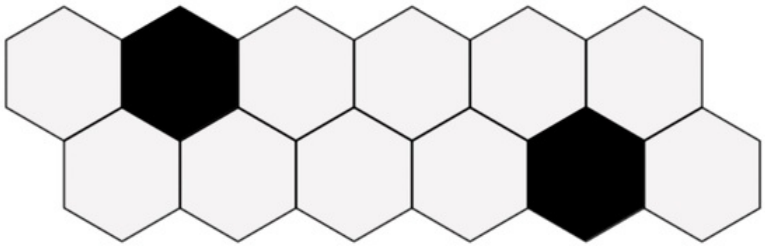


As you can see above, there are three possible orientations in which a tile can be placed.

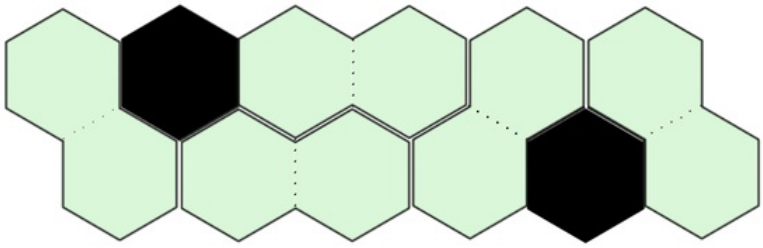
Your goal is to tile the whole grid such that every cell is covered by a tile, and no two tiles occupy the same cell. To add to the woes, certain cells of the hexagonal grid are *blackened*. No tile must occupy a blackened cell.

Is it possible to tile the grid?

Here's an example. Suppose we want to tile this grid:



Then we can do the tiling as follows:



Input Format

The first line contains a single integer t, the number of test cases.

The first line of each test case contains a single integer n denoting the length of the grid.

The second line contains a binary string of length n. The i^{th} character describes whether cell a_i is blackened.
The third line contains a binary string of length n. The i^{th} character describes whether cell b_i is blackened.
A `0` corresponds to an empty cell and a `1` corresponds to blackened cell.

Constraints

- $1 \leq t \leq 100$
- $1 \leq n \leq 10$

Output Format

For each test case, print `YES` if there exists at least one way to tile the grid, and `NO` otherwise.

Sample Input 0

```
6
6
010000
000010
2
00
00
2
00
10
2
00
01
2
00
11
2
10
00
```

Sample Output 0

```
YES
YES
NO
NO
YES
NO
```

Explanation 0

The first test case in the sample input describes the example given in the problem statement above.
For the second test case, there are two ways to fill it: either place two diagonal tiles side-by-side or place two horizontal tiles.

Queens on Board

Queens on Board

You have an $N * M$ chessboard on which some squares are blocked out. In how many ways can you place one or more queens on the board, such that, no two queens attack each other? Two queens attack each other, if one can reach the other by moving horizontally, vertically, or diagonally without passing over any blocked square. At most one queen can be placed on a square. A queen cannot be placed on a blocked square.

Input Format

The first line contains the number of test cases T. T test cases follow. Each test case contains integers N and M on the first line. The following N lines contain M characters each, and represent a board. A '#' represents a blocked square and a '.' represents an unblocked square.

Constraints

$1 <= T <= 100$
$1 <= N <= 50$
$1 <= M <= 5$

Output Format

Output T lines containing the required answer for each test case. As the answers can be really large, output them modulo 10^9+7.

Sample Input

```
4
3 3
...
...
...
3 3
.#.
.#.
...
2 4
.#..
....
1 1
#
```

Sample Output

```
17
18
14
0
```

Shashank and the Palindromic Strings

Shashank loves strings, but he loves palindromic strings the most. He has a list of n strings, $A = [a_0, a_1, \ldots, a_{n-1}]$, where each string, a_i, consists of lowercase English alphabetic letters. Shashank wants to count the number of ways of choosing non-empty subsequences $s_0, s_1, s_2, \ldots, s_{n-1}$ such that the following conditions are satisfied:

1. s_0 is a subsequence of string a_0, s_1 is a subsequence of string a_1, s_2 is a subsequence of string a_2, $\ldots$, and s_{n-1} is a subsequence of string a_{n-1}.
2. $s_0 + s_1 + s_2 + \ldots + s_{n-1}$ is a palindromic string, where `+` denotes the string concatenation operator.

You are given q queries where each query consists of some list, A. For each query, find and print the number of ways Shashank can choose n non-empty subsequences satisfying the criteria above, modulo $10^9 + 7$, on a new line.

Note: Two subsequences consisting of the same characters are considered to be different if their characters came from different indices in the original string.

Input Format

The first line contains a single integer, q, denoting the number of queries. The subsequent lines describe each query in the following format:

- The first line contains an integer, n, denoting the size of the list.
- Each line i of the n subsequent lines contains a non-empty string describing a_i.

Constraints

- $1 \leq q \leq 50$
- $1 \leq n \leq 50$
- $\sum_{i=0}^{n-1} |a_i| \leq 1000$ over a test case.

For 40% of the maximum score:

- $1 \leq n \leq 5$
- $\sum_{i=0}^{n-1} |a_i| \leq 250$ over a test case.

Output Format

For each query, print the number of ways of choosing non-empty subsequences, modulo $10^9 + 7$, on a new line.

Sample Input 0

```
3
3
aa
b
aa
3
a
b
c
2
abc
```

```
abc
```

Sample Output 0

```
5
0
9
```

Explanation 0

The first two queries are explained below:

1. We can choose the following five subsequences:
    1. $s_0 = \texttt{"a"}$, $s_1 = \texttt{"b"}$, $s_2 = \texttt{"a"}$, where s_0 is the first character of a_0 and s_2 is the first character of a_2.
    2. $s_0 = \texttt{"a"}$, $s_1 = \texttt{"b"}$, $s_2 = \texttt{"a"}$, where s_0 is the second character of a_0 and s_2 is the second character of a_2.
    3. $s_0 = \texttt{"a"}$, $s_1 = \texttt{"b"}$, $s_2 = \texttt{"a"}$, where s_0 is the first character of a_0 and s_2 is the second character of a_2.
    4. $s_0 = \texttt{"a"}$, $s_1 = \texttt{"b"}$, $s_2 = \texttt{"a"}$, where s_0 is the second character of a_0 and s_2 is the first character of a_2.
    5. $s_0 = \texttt{"aa"}$, $s_1 = \texttt{"b"}$, $s_2 = \texttt{"aa"}$

    Thus, we print the result of $5 \bmod (10^9 + 7) = 5$ on a new line.
2. There is no way to choose non-empty subsequences such that their concatenation results in a palindrome, as each string contains unique characters. Thus, we print 0 on a new line.

Points in a Plane

There are N points on an XY plane. In one turn, you can select a set of collinear points on the plane and remove them. Your goal is to remove all the points in the least number of turns. Given the coordinates of the points, calculate two things:

- The minimum number of turns (T) needed to remove all the points.
- The number of ways to to remove them in T turns. Two ways are considered different if any point is removed in a different turn.

Input Format

The first line contains the number of test cases T. T test cases follow. Each test case contains N on the first line, followed by N lines giving the coordinates of the points.

Constraints

1 <= T <= 50
1 <= N <= 16
0 <= xi,yi <= 100
No two points will have the same coordinates.

Output Format

Output T lines, one for each test case, containing the least number of turns needed to remove all points and the number of ways to do so. As the answers can be large, output them modulo 1000000007.

Sample Input

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

Sample Output

```
2 6
2 8
```

Explanation

For the 1st input, Let the points be labelled p1,p2,p3. These are the ways to remove them (first turn's points, followed by second turn's points):

a. 1) p1,p2 2) p3
b. 1) p1,p3 2) p2
c. 1) p2,p3 2) p1
d. 1) p3 2) p1,p2
e. 1) p2 2) p1,p3
f. 1) p1 2) p3,p2

Turn Off the Lights

There are n bulbs in a straight line, numbered from 0 to $n-1$. Each bulb i has a button associated with it, and there is a *cost*, c_i, for pressing this button. When some button i is pressed, all the bulbs at a distance $\leq k$ from bulb i will be toggled(off->on, on->off).

Given n, k, and the costs for each button, find and print the minimum cost of turning off all n bulbs if they're all on initially.

Input Format

The first line contains two space-separated integers describing the respective values of n and k.
The second line contains n space-separated integers describing the respective costs of each bulb (i.e., $c_0, c_1, \ldots, c_{n-1}$).

Constraints

- $3 \leq n \leq 10^4$
- $0 \leq k \leq 1000$
- $0 \leq c_i \leq 10^9$

Output Format

Print a long integer denoting the minimum cost of turning off all n bulbs.

Sample Input

```
3 1
1 1 1
```

Sample Output

```
1
```

Explanation

If we press the middle switch, the middle bulb and the $k = 1$ closest adjacent bulbs (i.e., the first and third) will turn off. Because all bulbs will be off in one button press, this cost is minimal. Thus, we print 1 as our answer.

Animal Transport

Capeta is working part-time for an animal shipping company. He needs to pick up animals from various zoos and drop them to other zoos. The company ships four kinds of animals: elephants, dogs, cats, and mice.

There are m zoos, numbered 1 to m. Also, there are n animals. For each animal i, Capeta knows its type t_i (`E` for elephant, `D` for dog, `C` for cat and `M` for mouse), source zoo s_i where Capeta has to pick it up from, and destination zoo d_i where Capeta needs to deliver it to.

Capeta is given a truck with a huge capacity where n animals can easily fit. He is also given additional instructions:

1. He must visit the zoos in **increasing** order.
2. **Dogs** are scared of **elephants**, so he is not allowed to bring them together at the same time.
3. **Cats** are scared of **dogs**, so he is not allowed to bring them together at the same time.
4. **Mice** are scared of **cats**, so he is not allowed to bring them together at the same time.
5. **Elephants** are scared of **mice**, so he is not allowed to bring them together at the same time.

Also, loading and unloading animals are complicated, so once an animal is loaded onto the truck, that animal will only be unloaded at its destination.

Because of these reasons, Capeta might not be able to transport all animals. He will need to ignore some animals. Which ones? The company decided to leave that decision for Capeta. He is asked to prepare a report and present it at a board meeting of the company.

Capeta needs to report the minimum zoo number that must be reached in order to transport x animals, for each x from 1 to n.

He is good at racing and driving, but not in planning. So he asked for your help.

Input Format

The first line contains a single integer t, the number of test cases.

Each test case consists of four lines. The first line contains two space-separated integers m and n. The second line contains n space-separated characters $t_1, t_2, \ldots, t_n$. The third line contains n space-separated integers $s_1, s_2, \ldots, s_n$. The fourth line contains n space-separated integers $d_1, d_2, \ldots, d_n$.

t_i, s_i and d_i are the details for the ith animal, as described in the problem statement.

Constraints

- $1 \leq t \leq 10$
- $1 \leq m, n \leq 5 \cdot 10^4$
- $1 \leq s_i, d_i \leq m$
- $s_i \neq d_i$
- t_i is either `E`, `D`, `C` or `M`

Subtasks

- For 30% of the total score, $m, n \leq 10^3$

Output Format

For each case, print a single line containing n space-separated integers, where the x^{th} integer is the minimum zoo number that Capeta needs to reach in order to transport x animals. If it is not possible to transport x animals at all, then put -1 instead.

Sample Input 0

```
2
10 3
E D C
4 1 4
7 5 8
10 6
E D C M E D
1 1 1 2 9 7
2 2 2 4 10 10
```

Sample Output 0

```
5 8 -1
2 2 4 10 -1 -1
```

Explanation 0

First Test Case

Capeta can transport one animal by traveling up to zoo number 5. Just drop the dog there. Next, in order to transport 2 animals (elephant and cat), Capeta has to go up to zoo number 8.

Second Test Case

- 1 Animal: Drop the elephant to zoo 2.
- 2 Animal: Drop the elephant and cat to zoo 2.
- 3 Animal: Drop the elephant and cat to zoo 2. Then drop the mouse to zoo 4.
- 4 Animal: Drop the elephant and cat to zoo 2. Then drop the mouse to zoo 4. Finally, drop either the elephant or the dog to 10.
- It is impossible to transport 5 or 6 animals.

The Indian Job

It is the Indian version of the famous heist “The Italian Job”. *N* robbers have already broken into the National Museum and are just about to get inside the main vault which is full of jewels. They were lucky that just as they broke into the museum, the guard was leaving the museum for exactly *G* minutes. But there are other problems too. The main vault has heat sensors that if at any moment of time there are more than two people present in the vault, the alarm goes off.

To collect the jewels, the i^{th} robber needs to be inside the vault for exactly *A[i]* minutes, *0 <= i < N*, in one continuous stretch. As guard will return after *G* minutes, they have to finish their tasks within *G* minutes. The robbers want to know if there exists any arrangement such that demands of each robber is satisfied and also they are not caught?

Gotchas

If a robber goes inside the vault at a time "X" and at the same time another robber comes out, it's equivalent to saying they were never in the vault at the same time.
Similarly, when the guard gets inside vault at time *G* and a robber comes out exactly at time *G*, the guard will not be able see the robber.

Input Format

The first line contains an integer *T* denoting the number of testcases. *T* testcases follow. Each testcase consists of two lines. First line contains two space separated integers denoting *N* and *G* denoting the number of thieves and duration for which guard leaves the museum.
The next line contains N space separated numbers where the ith integer, A[i] represents the time the i^{th} robber needs to be in the vault.

Constraints

1 <= T <= 20
1 <= N <= 100
0 <= G <= 1000000 (10^6)
0 <= A[i] <= 100

Output Format

For each testcase print `YES` if there exists such an arrangement or `NO` otherwise in a newline.

Sample Input

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

Sample Output

```
YES
NO
```

Explanation

Test case #00: In first testcase, one possible arrangement is:
at t=0, robber1 goes inside and comes out at t=2
at t=0, robber2 goes inside and comes out at t=4
at t=2, robber3 goes inside and comes out at t=4

Test case #01: No possible arrangement is possible in second testcase.

Requirement

There are n variables and m requirements. Requirements are represented as $(x \leq y)$, meaning that the x^{th} variable must be less than or equal to the y^{th} variable.

Your task is to assign non-negative numbers smaller than 10 to each variable and then calculate the number of different assignments satisfying all requirements. Two assignments are different if and only if at least one variable is assigned to a different number in both assignments. Print your answer modulo $10^3 + 7$.

Input Format

The first line contains 2 space-separated integers, n and m, respectively. Each of the m subsequent lines contains 2 space-seperated integers describing the respective x and y values for an $(x \leq y)$ requirement.

Constraints

- $0 < n < 14$
- $0 < m < 200$
- $0 \leq x, y < n$

Output Format

Print your answer modulo $10^3 + 7$.

Sample Input 0

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

Sample Output 0

```
1000
```

Explanation 0

There are 6 variables and 7 requirements.

Let the variables be in the array $a[6]$.

Requirements are -
$a[1] <= a[3], a[0] <= a[1], a[2] <= a[4], a[0] <= a[4], a[2] <= a[5], a[3] <= a[4], a[0] <= a[2]$

One of the assignments is - $\{1, 2, 3, 4, 5, 6\}$

Similarly there are 25168 assignments possible.

Result = $25168 \bmod 1007 = 1000$.

A Super Hero

Ma5termind is crazy about Action Games. He just bought a new one and got down to play it. Ma5termind usually finishes all the levels of a game very fast. But, This time however he got stuck at the very first level of this new game. Can you help him play this game.

To finish the game, Ma5termind has to cross N levels. At each level of the game, Ma5termind has to face M enemies. Each enemy has its associated power P and some number of bullets B. To knock down an enemy, Ma5termind needs to shoot him with one or multiple bullets whose collective count is equal to the power of the enemy. If Ma5termind manages to knock down any one enemy at a level, the rest of them run away and the level is cleared.

Here comes the challenging part of the game.
Ma5termind acquires all the bullets of an enemy once he has knocked him down. Ma5termind can use the bullets acquired after killing an enemy at i^{th} level only till the $(i+1)^{th}$ level.

However, the bullets Ma5termind carried before the start of the game can be taken forward and can be used to kill more enemies.

Now, Ma5termind has to guess the minimum number of bullets he must have before the start of the game so that he clears all the N levels successfully.

NOTE

1. Bullets carried before the start of the game can be used to kill an enemy at any level.
2. One bullet decreases the power of an enemy by 1 Unit.
3. For better understanding of the problem look at the sample testcases.

Input Format

First line of input contains a single integer T denoting the number of test cases.
First line of each test case contains two space separated integers N and M denoting the number of levels and number of enemies at each level respectively.
Each of next N lines of a test case contain M space separated integers, where j^{th} integer in the i^{th} line denotes the power P of j^{th} enemy on the i^{th} level.
Each of the next N lines of a test case contains M space separated integers, where j^{th} integer in the i^{th} line denotes the number of bullets B j^{th} enemy of i^{th} level has.

Constraints
$1 \leq T \leq 100$
$1 \leq N \leq 100$
$1 \leq M \leq 5 \times 10^5$
$1 \leq P, B \leq 1000$
For each test file, sum of $N \times M$ over all the test cases does not exceed 5×10^5.

Output Format

For each test case, print the required answer.

Sample Input

```
2
```

```
3 3
3 2 1
1 2 3
3 2 1
1 2 3
3 2 1
1 2 3
3 3
3 2 5
8 9 1
4 7 6
1 1 1
1 1 1
1 1 1
```

Sample Output

```
1
5
```

Explanation

For the First test case , Ma5termind kills the enemy in the following order:

1. Ma5termind kills the 3^{rd} enemy at the 1^{st} level, takes all his bullets and moves to the next level.
2. Ma5termind kills the 1^{st} enemy at the 2^{nd} level, takes all his bullets and moves to the next level.
3. Ma5termind kills the 1^{st} enemy at the 3^{rd} level, takes all his bullets and moves to the next level.

So this way Ma5termind can successfully finish this game with only 1 bullet in hand before the start of the game.

For the second test case , Ma5termind kills the enemy in the following order:

1. Ma5termind kills the 2^{nd} enemy at the 1^{st} level, takes all his bullets and moves to the next level.
2. Ma5termind kills the 3^{rd} enemy at the 2^{nd} level, takes all his bullets and moves to the next level.
3. Ma5termind kills the 1^{st} enemy at the 3^{rd} level, takes all his bullets and moves to the next level.

So this way Ma5termind can successfully finish this game with only 5 bullet in hand before the start of the game.

NOTE:
There can be more than one way of getting the optimal answer but that does not matter in our case, because we need to answer the minimum number of bullets required.

Clues on a Binary Path

Logan and Veronica live in Neptune, which has n houses and m bidirectional roads connecting them. Each road has an assigned value, c_i, where $c_i \in \{0, 1\}$, and each house is numbered with a distinct integer from 1 to n.

Logan and Veronica are looking for clues and need to find the number of different paths of length d from house number 1. Each path is characterized by a binary sequence of length d, where each integer j in the path is the value of c_j for the j^{th} edge in the path. Two paths are different if the binary sequences characterizing these paths are distinct. Note that they may need to visit the same house several times or use the same road several times to find all possible paths.

Given a map of Neptune, help Logan and Veronica find and print the number of different paths of length d from house number 1 to the other houses in Neptune.

Input Format

The first line contains three space-separated integers describing the respective values of n (the number of houses), m (the number of bidirectional roads), and d (the distance they want to travel).
Each of the m subsequent lines contains three space-separated integers describing the respective values of u, v, and c that define a bidirectional road between houses u and v having assigned value c.

Constraints

- $1 \leq n \leq 90$
- $0 \leq m \leq n \cdot (n-1)$
- $1 \leq d \leq 20$
- $c \in \{0, 1\}$
- There may be roads connecting house to itself.
- There may be more than one road between two houses.

Output Format

Print an integer denoting the total number of paths.

Sample Input

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

Sample Output

```
4
```

Explanation

There are four possible paths:

1. $1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \Rightarrow 000$
2. $1 \rightarrow 2 \rightarrow 1 \rightarrow 3 \Rightarrow 001$

3. $1 \to 3 \to 1 \to 3 \Rightarrow 111$
4. $1 \to 3 \to 1 \to 2 \Rightarrow 110$

Thus, we print 4 as our answer.

Road Maintenance

Byteland has N cities (numbered from 1 to N) and $N-1$ bidirectional roads. A *path* is comprised of 1 or more connected roads. It is guaranteed that there is a path from any city to any other city.

Steven is a road maintenance worker in Byteland. He is required to maintain *exactly* M paths on any given workday. He *cannot* work on the same road twice in one day (so no 2 paths can contain the same 2 roads). Steven can start his workday in any city and, once he has finished maintaining a path, teleport to his next starting city.

Given M, help Steven determine how many different possible M−path sets will allow him to perform his maintenance duties. Then print the answer modulo 10^9+7.

Input Format

The first line contains 2 space-separated integers, N (the number of cities) and M (the number of roads to maintain).
Each line i of the $N-1$ subsequent lines contains 2 space-separated integers, A_i B_i, describing a bidirectional road between cities A_i and B_i.

Constraints

- $1 \leq N \leq 10^5$
- $1 \leq M \leq 5$
- $A_i \neq B_i$
- $1 \leq A_i, B_i \leq N$

Output Format

Find the number of different M−path sets that will allow Steven to complete M orders, and print the answer $\% \ (10^9+7)$.

Sample Input

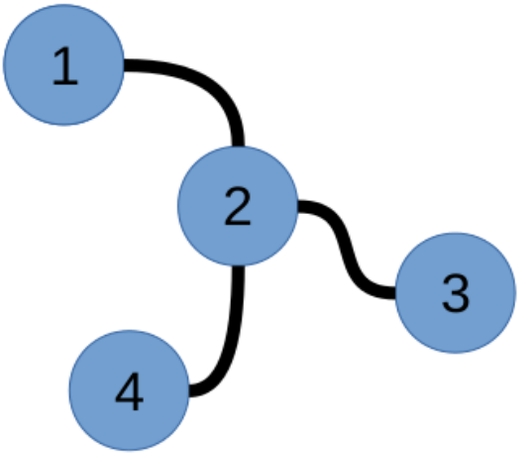
```
4 2
1 2
2 3
2 4
```

Sample Output

```
6
```

Explanation

For the following Byteland map:



Steven can maintain $M = 2$ roads using any of the following 6 routes:

1. $[1, 2]$ and $[2, 3]$
2. $[1, 2]$ and $[2, 4]$
3. $[1, 2]$ and $[3, 4]$
4. $[1, 3]$ and $[2, 4]$
5. $[1, 4]$ and $[2, 3]$
6. $[2, 3]$ and $[2, 4]$

Thus, we print the result of $6 \% (10^9 + 7)$ on a new line, which is 6.

Billboards

ADZEN is a popular advertising firm in your city that owns all n billboard locations on Main street. The city council passed a new zoning ordinance mandating that no more than k consecutive billboards be up at any given time. For example, if there are $n = 3$ billboards on Main street and $k = 1$, ADZEN must remove either the middle billboard, the first two billboards, or the last two billboards.

Being a for-profit company, ADZEN wants to lose as little advertising revenue as possible when removing the billboards. They want to comply with the new ordinance in such a way that the remaining billboards maximize their total profits (i.e., the total sum of all the billboards left standing on Main street).

Given n, k, and the revenue value of each of the n billboards, find and print the maximum profit that ADZEN can earn while complying with the zoning ordinance. Assume that Main street is a straight, contiguous block of n billboards that can be removed but *cannot* be reordered in any way.

Input Format

The first line contains two space-separated integers, n (the number of billboards) and k (the maximum number of billboards that can stand together on any part of the road).
Each line i of the n subsequent lines contains an integer denoting the revenue value of billboard i (where $0 \leq i < n$).

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq k \leq n$
- $0 \leq$ $\text{revenue value of any billboard}$ $\leq 2 \cdot 10^9$

Output Format

Print a single integer denoting the maximum profit ADZEN can earn from Main street after complying with the city's ordinance.

Sample Input 0

```
6 2
1
2
3
1
6
10
```

Sample Output 0

```
21
```

Explanation 0
There are $n = 6$ billboards, and we must remove some of them so that no more than $k = 2$ billboards are immediately next to one another.

We remove the first and fourth billboards, which gives us the configuration `_ 2 3 _ 6 10` and a profit of $2 + 3 + 6 + 10 = 21$. As no other configuration has a profit greater than 21, we print 21 as our answer.

Sample Input 1

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

Sample Output 1

```
14
```

Explanation 1

There are $n = 5$ billboards, and we must remove some of them so that no more than $k = 4$ billboards are immediately next to one another.

We remove the first billboard, which gives us the configuration `_ 2 3 4 5` and a profit of $2 + 3 + 4 + 5 = 14$. As no other configuration has a profit greater than 14, we print 14 as our answer.

Beautiful Strings

You are given a string, S, consisting of lowercase English letters.

A string is *beautiful* with respect to S if it can be derived from S by removing *exactly* 2 characters.

Find and print the number of different strings that are *beautiful* with respect to S.

Input Format

A single string of lowercase English letters denoting S.

Constraints

- $3 \leq |S| \leq 10^6$
- $3 \leq |S| \leq 20$ holds for test cases worth at least 15% of the problem's score.
- $3 \leq |S| \leq 2000$ holds for test cases worth at least 30% of the problem's score.

Output Format

Print the number of different strings that are *beautiful* with respect to S.

Sample Input

```
abba
```

Sample Output

```
4
```

Explanation

$S = \{abba\}$

The following strings can be derived by removing 2 characters from S: $ab, bb, ba, ab, ba, aa,$ and bb.

This gives us our set of *unique* beautiful strings, $B = \{ab, ba, aa, bb\}$. As $|B| = 4$, we print 4.

Covering the stains

There is a huge blanket on your bed but unfortunately it has **N** stains. You cover them using a single, rectangular silk cloth. The silk is expensive, which is why the rectangular piece needs to have the least area as possible. You love this blanket and decide to minimize the area covering the stains. You buy some cleaning liquid to remove the stains but sadly it isn't enough to clean all of them. You can just remove exactly **K** stains. The rest of the stains need to be covered using a single, rectangular fragment of silk cloth.

Let **X** denote the area of the smallest possible silk cloth that may cover all the stains originally. You need to find the number of different ways in which you may remove **K** stains so that the remaining **N-K** stains can be covered with silk of **area strictly less than X** (We are looking for any configuration that will reduce the cost).

Assume that each stain is a point and that the rectangle is aligned parallel to the axes.

Input Format

The first line contains two integers **N** (1<=N<=1000) and **K** (0<=K<=N). Next follow **N** lines, one for each stain. Each line contains two integers in the form 'X Y', (0<=X,Y<100000), the coordinates of each stain into the blanket. Each pair of coordinates is unique.

Output Format

Output a single integer. The remainder of the division by 1000000007 of the answer.

Sample Input

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

Sample Output

```
8
```

Explanation

We can clean two spots. So removing any of the following set of stains will lead us to a conbination that will need less amount of silk.(The numbers refer to the **indices of the stains** in the input and they begin from 1).

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

So there are 8 ways.

GCD Matrix

Alex has two arrays defined as $A = [a_0, a_1, \ldots, a_{n-1}]$ and $B = [b_0, b_1, \ldots, b_{m-1}]$. He created an $n \times m$ matrix, M, where $M_{i,j} = \gcd(a_i, b_j)$ for each i, j in M. Recall that $\gcd(a, b)$ is the greatest common divisor of a and b.

For example, if $A = [2, 3]$ and $B = [5, 6]$, he builds $M = [[1, 2], [1, 3]]$ like so:

(i, j)	0	1
0	$\gcd(2,5) = 1$	$\gcd(2,6) = 2$
1	$\gcd(3,5) = 1$	$\gcd(3,6) = 3$

Alex's friend Kiara loves matrices, so he gives her q questions about matrix M where each question is in the form of some submatrix of M with its upper-left corner at M_{r_1,c_1} and its bottom-right corner at M_{r_2,c_2}. For each question, find and print the number of *distinct* integers in the given submatrix on a new line.

Input Format

The first line contains three space-separated integers describing the respective values of n (the size of array A), m (the size of array B), and q (Alex's number of questions).
The second line contains n space-separated integers describing $a_0, a_1, \ldots, a_{n-1}$.
The third line contains m space-separated integers describing $b_0, b_1, \ldots, b_{m-1}$.
Each line i of the q subsequent lines contains four space-separated integers describing the respective values of r_1, c_1, r_2, and c_2 for the i^{th} question (i.e., defining a submatrix with upper-left corner (r_1, c_1) and bottom-right corner (r_2, c_2)).

Constraints

- $1 \leq n, m \leq 10^5$
- $1 \leq a_i, b_i \leq 10^5$
- $1 \leq q \leq 10$
- $0 \leq r_1, r_2 < n$
- $0 \leq c_1, c_2 < m$

Scoring

- $1 \leq n, m \leq 1000$ for 25% of score.
- $1 \leq n, m \leq 10^5$ for 100% of score.

Output Format

For each of Alex's questions, print the number of *distinct* integers in the given submatrix on a new line.

Sample Input 0

```
3 3 3
1 2 3
2 4 6
0 0 1 1
0 0 2 2
1 1 2 2
```

Sample Output 0

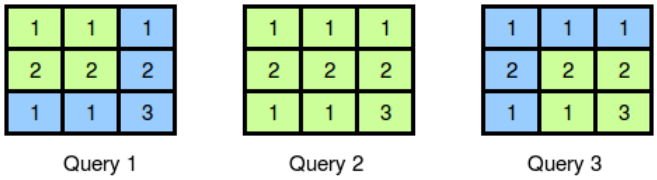
```
2
3
3
```

Explanation 0

Given $A = [1, 2, 3]$ and $B = [2, 4, 6]$, we build the following M:

(i, j)	0	1	2
0	$\gcd(1, 2) = 1$	$\gcd(1, 4) = 1$	$\gcd(1, 6) = 1$
1	$\gcd(2, 2) = 2$	$\gcd(2, 4) = 2$	$\gcd(2, 6) = 2$
2	$\gcd(3, 2) = 1$	$\gcd(3, 4) = 1$	$\gcd(3, 6) = 3$

The diagram below depicts the submatrices for each of the $q = 3$ questions in *green*:



1. For the submatrix between $M_{0,0}$ and $M_{1,1}$, the set of integers is $\{1, 2\}$. The number of distinct integers is 2.
2. For the submatrix between $M_{0,0}$ and $M_{2,2}$, the set of integers is $\{1, 2, 3\}$. The number of distinct integers is 3.
3. For the submatrix between $M_{1,1}$ and $M_{2,2}$, the set of integers is $\{1, 2, 3\}$. The number of distinct integers is 3.

Fairy Chess

Let's play *Fairy Chess*!

You have an $n \times n$ chessboard. An s-leaper is a chess piece which can move from some square (x_0, y_0) to some square (x_1, y_1) if $abs(x_0 - x_1) + abs(y_0 - y_1) \leq s$; however, its movements are restricted to *up* (↑), *down* (↓), *left* (←), and *right* (→) within the confines of the chessboard, meaning that diagonal moves are not allowed. In addition, the leaper cannot leap to any square that is occupied by a *pawn*.

Given the layout of the chessboard, can you determine the number of ways a leaper can move m times within the chessboard?

Note: $abs(x)$ refers to the absolute value of some integer, x.

Input Format

The first line contains an integer, q, denoting the number of queries. Each query is described as follows:

1. The first line contains three space-separated integers denoting n, m, and s, respectively.
2. Each line i of the n subsequent lines contains n characters. The j^{th} character in the i^{th} line describes the contents of square (i, j) according to the following key:
   - `.` indicates the location is *empty*.
   - `P` indicates the location is occupied by a *pawn*.
   - `L` indicates the location of the *leaper*.

Constraints

- $1 \leq q \leq 10$
- $1 \leq m \leq 200$
- There will be exactly one `L` character on the chessboard.
- The s-leaper can move *up* (↑), *down* (↓), *left* (←), and *right* (→) within the confines of the chessboard. It *cannot* move diagonally.

Output Format

For each query, print the number of ways the leaper can make m moves on a new line. Because this value can be quite large, your answer must be modulo $10^9 + 7$.

Sample Input 0

```
3
4 1 1
....
.L..
.P..
....
3 2 1
...
...
..L
4 3 2
....
...L
..P.
P...
```

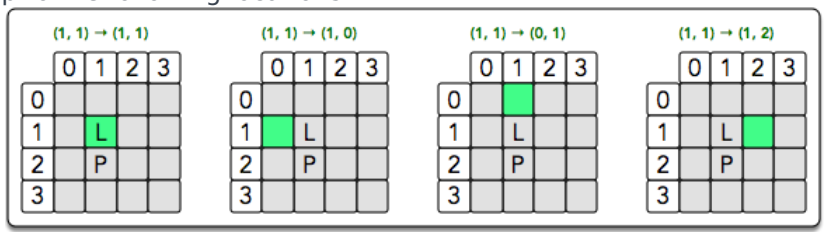
Sample Output 0

```
4
11
385
```

Explanation 0

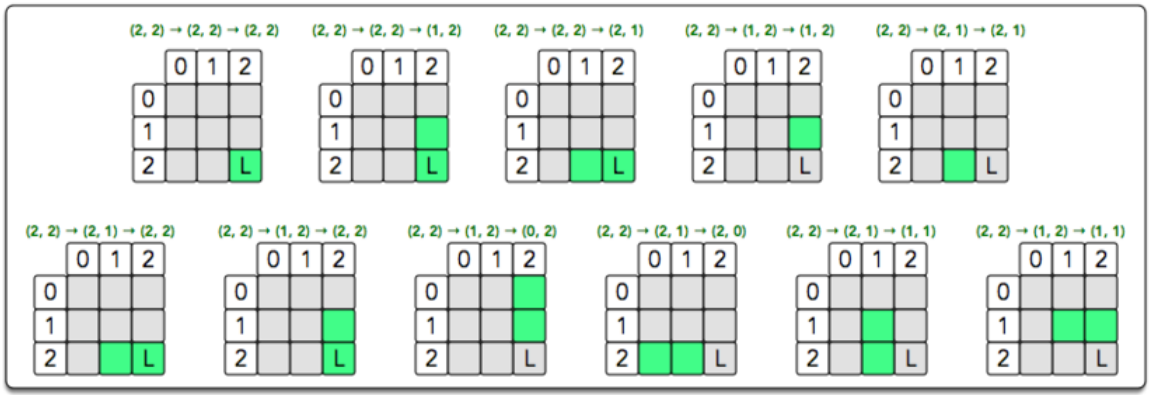
You must perform two queries, outlined below. The *green* cells denote a cell that was leaped to by the leaper, and coordinates are defined as $(row, column)$.

1. The leaper can leap to the following locations:



   Observe that the leaper cannot leap to the square directly underneath it because it's occupied by a pawn. Thus, there are 4 ways to make 1 move and we print 4 on a new line.

2. The leaper can leap to the following locations:



   Thus, we print 11 on a new line.

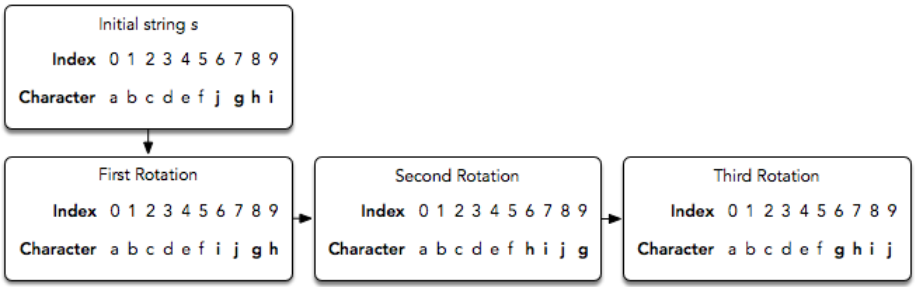
Note: Don't forget that your answer must be modulo $10^9 + 7$.

Suffix Rotation

Megan is playing a string game with the following rules:

- It starts with a string, s.
- During each turn, she performs the following move:
  - Choose an index in s. The chosen index must be strictly greater than any index chosen in a prior move.
  - Perform one or more circular rotations (in either direction) of the suffix starting at the chosen index.

  For example, let's say $s =$ `abcdefjghi`. During our move, we choose to do three right rotations of the suffix starting at index 6:



  Note that this counts as *one* move.
- The goal of the game is to convert s into the lexicographically smallest possible string *in as few moves as possible*. In other words, we want the characters to be in alphabetical order.

Megan plays this game g times, starting with a new string s each time. For each game, find the minimum number of moves necessary to convert s into the lexicographically smallest string and print that number on a new line.

Input Format

The first line contains an integer, g, denoting the number of games.
Each of the g subsequent lines contains a single string denoting the initial value of string s for a game.

Constraints

- $1 \leq g \leq 100$
- $1 \leq |s| \leq 1000$
- s consists of lowercase English alphabetic letters only.

Output Format

For each game, print an integer on a new line denoting the minimum number of moves required to convert s into the lexicographically smallest string possible.

Sample Input 0

```
3
abcdefghij
acab
baba
```

Sample Output 0

```
0
1
2
```

Explanation 0

We play the following $g = 3$ games:

1. In the first game, `abcdefghij` is already as lexicographically small as possible (each sequential letter is in alphabetical order). Because we don't need to perform any moves, we print 0 on a new line.
2. In the second game, we rotate the suffix starting at index 1, so `a**cab**` becomes `a**abc**`. Because the string is lexicographically smallest after one move, we print 1 on a new line.
3. In the third game, we perform the following moves:
   - Rotate the suffix starting at index 0 (i.e., the entire string), so `**baba**` becomes `**abab**`.
   - Rotate the suffix starting at index 1, so `a**bab**` becomes `a**abb**`.

   Because the string is lexicographically smallest after two moves, we print 2 on a new line.

New Year Present

Nina received an odd New Year's present from a student: a set of n unbreakable sticks. Each stick has a length, l, and the length of the i^{th} stick is l_{i-1}. Deciding to turn the gift into a lesson, Nina asks her students the following:

How many ways can you build a square using *exactly* 6 of these unbreakable sticks?

Note: Two ways are distinct if they use at least one different stick. As there are $\binom{n}{6}$ choices of sticks, we must determine which combinations of sticks can build a square.

Input Format

The first line contains an integer, n, denoting the number of sticks. The second line contains n space-separated integers $l_0, l_1, \ldots, l_{n-2}, l_{n-1}$ describing the length of each stick in the set.

Constraints

- $6 \leq n \leq 3000$
- $1 \leq l_i \leq 10^7$

Output Format

On a single line, print an integer representing the number of ways that 6 unbreakable sticks can be used to make a square.

Sample Input 0

```
8
4 5 1 5 1 9 4 5
```

Sample Output 0

```
3
```

Sample Input 1

```
6
1 2 3 4 5 6
```

Sample Output 1

```
0
```

Explanation

Sample 0
Given 8 sticks ($l = 4, 5, 1, 5, 1, 9, 4, 5$), the only possible side length for our square is 5. We can build square S in 3 different ways:

1. $S = \{l_0, l_1, l_2, l_3, l_4, l_6\} = \{4, 5, 1, 5, 1, 4\}$
2. $S = \{l_0, l_1, l_2, l_4, l_6, l_7\} = \{4, 5, 1, 1, 4, 5\}$
3. $S = \{l_0, l_2, l_3, l_4, l_6, l_7\} = \{4, 1, 5, 1, 4, 5\}$

In order to build a square with side length 5 using *exactly* 6 sticks, $l_0, l_2, l_4,$ and l_6 must always build two of the sides. For the remaining two sides, you must choose 2 of the remaining 3 sticks of length 5 ($l_1, l_3,$ and l_7).

Sample 1

We have to use all 6 sticks, making the largest stick length (6) the minimum side length for our square. No combination of the remaining sticks can build 3 more sides of length 6 (total length of all other sticks is $1+2+3+4+5=15$ and we need at least length $3*6=18$), so we print 0.

Travel around the world

There are N cities and N directed roads in Steven's world. The cities are numbered from 0 to N - 1. Steven can travel from city *i* to city *(i + 1) % N*, (0-> 1 -> 2 -> .... -> N - 1 -> 0).

Steven wants to travel around the world by car. The capacity of his car's fuel tank is C gallons. There are a[i] gallons he can use at the beginning of city *i* and the car takes b[i] gallons to travel from city *i* to *(i + 1) % N*.

How many cities can Steven start his car from so that he can travel around the world and reach the same city he started?

Note

The fuel tank is initially empty.

Input Format

The first line contains two integers (separated by a space): city number *N* and capacity *C*.
The second line contains *N* space-separated integers: a[0], a[1], ... , a[N - 1].
The third line contains *N* space-separated integers: b[0], b[1], ... , b[N - 1].

Constraints

$2 \leq N \leq 10^5$
$1 \leq C \leq 10^{18}$
$0 \leq a[i], b[i] \leq 10^9$

Output Format

The number of cities which can be chosen as the start city.

Sample Input

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

Sample Output

```
2
```

Explanation

Steven starts from city 0, fills his car with 3 gallons of fuel, and use 2 gallons of fuel to travel to city 1. His fuel tank now has 1 gallon of fuel.
On refueling 1 gallon of fuel at city 1, he then travels to city 2 by using 2 gallons of fuel. His fuel tank is now empty.
On refueling 2 gallon of fuel at city 2, he then travels back to city 0 by using 2 gallons of fuel.

Here is the second possible solution.
Steven starts from city 2, fill his car with 2 gallons, and travels to city 0.
On refueling 3 gallons of fuel from city 0, he then travels to city 1, and exhausts 2 gallons of fuel. His fuel tank contains 1 gallon of fuel now. He can then refuel 1 gallon of fuel at City 1, and increase his car's fuel to 2 gallons and travel to city 2.

However, Steven cannot start from city 1, because he is given only 1 gallon of fuel, but travelling to city 2 requires 2 gallons.

Hence the answer 2.

Longest Palindromic Subsequence

Steve loves playing with palindromes. He has a string, s, consisting of n lowercase English alphabetic characters (i.e., `a` through `z`). He wants to calculate the number of ways to insert exactly 1 lowercase character into string s such that the length of the longest palindromic subsequence of s increases by *at least* k. Two ways are considered to be *different* if either of the following conditions are satisfied:

- The positions of insertion are different.
- The inserted characters are different.

This means there are *at most* $26 \times (n+1)$ different ways to insert exactly 1 character into a string of length n.

Given q queries consisting of n, k, and s, print the number of different ways of inserting exactly 1 new lowercase letter into string s such that the length of the longest palindromic subsequence of s increases by *at least* k.

Input Format

The first line contains a single integer, q, denoting the number of queries. The $2q$ subsequent lines describe each query over two lines:

1. The first line of a query contains two space-separated integers denoting the respective values of n and k.
2. The second line contains a single string denoting s.

Constraints

- $1 \leq q \leq 10$
- $1 \leq n \leq 3000$
- $0 \leq k \leq 50$
- It is guaranteed that s consists of lowercase English alphabetic letters (i.e., `a` to `z`) only.

Subtasks

- $1 \leq n \leq 100$ for 25% of the maximum score.
- $1 \leq n \leq 1000$ for 70% of the maximum score.

Output Format

On a new line for each query, print the number of ways to insert exactly 1 new lowercase letter into string s such that the length of the longest palindromic subsequence of s increases by *at least* k.

Sample Input

```
3
1 1
a
3 2
aab
3 0
aba
```

Sample Output

```
2
1
104
```

Explanation

We perform the following $q = 2$ queries:

1. The length of the longest palindromic subsequence of $s =$ `a` is 1. There are two ways to increase this string's length by *at least* $k = 1$:

   1. Insert an `a` at the start of string s, making it `aa`.
   2. Insert an `a` at the end of string s, making it `aa`.

   Both methods result in `aa`, which has a longest palindromic subsequence of length 2 (which is longer than the original longest palindromic subsequence's length by $k = 1$). Because there are two such ways, we print 2 on a new line.

2. The length of the longest palindromic subsequence of $s =$ `aab` is 2. There is one way to increase the length by *at least* $k = 2$:

   1. Insert a `b` at the start of string s, making it `baab`.

   We only have one possible string, `baab`, and the length of its longest palindromic subsequence is 4 (which is longer than the original longest palindromic subsequence's length by $k = 2$). Because there is one such way, we print 1 on a new line.

Candles Counting

Tim is visiting his grandma for two days and is bored due to the lack of the electricity over there. That's why he starts to play with grandma's colorful candle collection.

He aligned the N candles from left to right. The ith candle from the left has the height H_i and the color C_i, an integer ranged from 1 to a given K, the number of colors.

Now he stares at the sequence of candles and wonders, how many strictly increasing (in height) colorful subsequences are there? A subsequence is considered as colorful if every of the K colors appears at least one times in the subsequence.

As the number of subsequences fulfilling the requirement can be large, print the result modulo $10^9 + 7$.

Input Format

On the first line you will be given N and K, then N lines will follow. On the ith line you will be given two integers H_i and C_i.

Constraints

- $1 \leq N \leq 5 \cdot 10^4$
- $1 \leq C_i \leq K \leq 7$
- $1 \leq H_i \leq 5 \cdot 10^4$

Output Format

Print the number of strictly increasing colorful subsequences modulo $10^9 + 7$.

Sample Input

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

Sample Output

```
2
```

Explanation

In the first sample the only two valid subsequences are (1, 2, 4) and (1, 3, 4).

Hyper Strings

String A is called a *Super String* if and only if:

- A contains only letters $a, b, c, d, e, f, g, h, i, j$;
- For any i and j, $A[i]$ has lower ascii code than $A[j]$, where $0 < i < j < length(A)$

Given a set of Super Strings H, a *Hyper String* is a string that can be constructed by concatenation of some Super Strings of the set H. We can use each Super String as many times as we want.

Given set H, you have to compute the number of Hyper Strings with length no greater than M.

Input Format

The first line of input contains two integers, N (the number of Super Strings in H) and M. The next N lines describe the Super Strings in set H.

Constraints

N and M are not greater than 100.

Output Format

Output an integer which is the number of possible Hyper Strings that can be derived. Since it may not fit in 32 bit integer, print the output module 1000000007. (i.e. answer = answer % 1000000007)

Sample Input

```
2 3
a
ab
```

Sample Output

```
7
```

Explanation

In the example all the Hyper Strings are : "" (empty string), "a", "ab", "aa", "aaa", "aba", and "aab".

Swap Permutation

You are given an array *A = [1, 2, 3, ..., n]*:

1. How many sequences *(S_1)* can you get after exact *k* adjacent swaps on *A*?
2. How many sequences *(S_2)* can you get after at most *k* swaps on *A*?

An adjacent swap can be made between two elements of the Array A, A[i] and A[i+1] or A[i] and A[i-1]. A swap otherwise can be between any two elements of the array A[i] and A[j] ∀ 1 ≤ i, j ≤ N, i ≠ j.

Input Format

First and only line contains *n* and *k* separated by space.

Constraints

1 ≤ n ≤ 2500
1 ≤ k ≤ 2500

Output Format

Output *S_1 % MOD* and *S_2 % MOD* in one line, where `MOD = 1000000007`.

Sample Input

```
3 2
```

Sample Output

```
3 6
```

Explanation

```
Original array: [1, 2, 3]
1. After 2 adjacent swaps:
We can get [1, 2, 3], [2, 3, 1], [3, 1, 2] ==> S1 == 3

2. After at most 2 swaps:
1) After 0 swap: [1, 2, 3]
2) After 1 swap: [2, 1, 3], [3, 2, 1], [1, 3, 2].
3) After 2 swaps: [1, 2, 3], [2, 3, 1], [3, 1, 2]
==> S2 == 6
```

Extremum Permutations

Let's consider a permutation $P = \{p_1, p_2, ..., p_N\}$ of the set of $N = \{1, 2, 3, ..., N\}$ elements .

P is called a magic set if it satisfies both of the following constraints:

- Given a set of K integers, the elements in positions $a_1, a_2, ..., a_K$ are less than their adjacent elements, i.e., $p_{a_i-1} > p_{a_i} < p_{a_i+1}$
- Given a set of L integers, elements in positions $b_1, b_2, ..., b_L$ are greater than their adjacent elements, i.e., $p_{b_i-1} < p_{b_i} > p_{b_i+1}$

How many such magic sets are there?

Input Format
The first line of input contains three integers N, K, L separated by a single space.
The second line contains K integers, $a_1, a_2, ... a_K$ each separated by single space.
the third line contains L integers, $b_1, b_2, ... b_L$ each separated by single space.

Output Format
Output the answer modulo 1000000007 (10^9+7).

Constraints
$3 <= N <= 5000$
$1 <= K, L <= 5000$
$2 <= a_i, b_j <= N-1$, where $i \in [1, K]$ AND $j \in [1, L]$

Sample Input #00

```
4 1 1
2
3
```

Sample Output #00

```
5
```

Explanation #00

Here, $N = 4$ $a_1 = 2$ and $b_1 = 3$. The 5 permutations of {1,2,3,4} that satisfy the condition are

- 2 1 4 3
- 3 2 4 1
- 4 2 3 1
- 3 1 4 2
- 4 1 3 2

Sample Input #01

```
10 2 2
2 4
```

```
3 9
```

Sample Output #01

```
161280
```

Square Subsequences

HackerRank

Square Subsequences

A string is called a square string if it can be obtained by concatenating two copies of the same string. For example, "abab", "aa" are square strings, while "aaa", "abba" are not. Given a string, how many (non-empty) subsequences of the string are square strings? A subsequence of a string can be obtained by deleting zero or more characters from it, and maintaining the relative order of the remaining characters.

Input Format

The first line contains the number of test cases, T.
T test cases follow. Each case contains a string, S.

Output Format

Output T lines, one for each test case, containing the required answer modulo 1000000007.

Constraints:

$1 \leq T \leq 20$
S will have at most 200 lowercase characters ('a' - 'z').

Sample Input

3
aaa
abab
baaba

Sample Output

3
3
6

Explanation

For the first case, there are 3 subsequences of length 2, all of which are square strings.
For the second case, the subsequences "abab", "aa", "bb" are square strings.
Similarly, for the third case, "bb", "baba" (twice), and "aa" (3 of them) are the square subsequences.

Dorsey Thief

Mr. Dorsey Dawson recently stole X grams of gold from ACME Jewellers. He is now on a train back home. To avoid getting caught by the police, he has to convert all the gold he has into paper money. He turns into a salesman and starts selling the gold in the train.

There are N passengers who have shown interest in buying the gold. The i^{th} passenger agrees to buy a_i grams of gold by paying v_i dollars. Dawson wants to escape from the police and also maximize the profit. Can you help him maximize the profit?

Note: The i^{th} passenger would buy **exactly** a_i grams if the transaction is successful.

Input Format

The first line contains two space separated integers, N and X, where N is the number of passengers who agreed to buy and X is the stolen amount of gold (in grams).
N lines follow. Each line contains two space separated integers - v_i and a_i, where v_i is the the value which the i^{th} passenger has agreed to pay in exchange for a_i grams of gold.

Constraints

- $1 \leq X \leq 5000$
- $1 \leq N \leq 10^6$
- all v_i's and a_i's are less than or equal to 10^6 and greater than 0.

Output Format

If it's possible for Dorsey to escape, print the maximum profit he can enjoy, otherwise print `Got caught!`.

Sample Input 0

```
4 10
460 4
590 6
550 5
590 5
```

Sample Output 0

```
1140
```

Explanation 0

Selling it to passengers buying 4 grams and 6 grams would lead to 1050 dollars whereas selling it to passengers buying 5 grams gold would lead to 1140 dollars. Hence the answer.

Sample Input 1

```
4 9
100 5
120 10
300 2
500 3
```

Sample Output 1

```
Got caught!
```

Explanation 1

There is no way to sell all 9 grams of gold.

Mining

There are n gold mines along a river, and each mine i produces w_i tons of gold. In order to collect the mined gold, we want to redistribute and consolidate it amongst exactly k mines where it can be picked up by trucks. We do this according to the following rules:

- You can move gold between any pair of mines (i.e., i and j, where $1 \leq i < j \leq n$).
- All the gold at some pickup mine i must either stay at mine i or be completely moved to some other mine, j.
- Move w tons of gold between the mine at location x_i and the mine at location x_j at a cost of $|x_i - x_j| \times w$.

Given n, k, and the amount of gold produced at each mine, find and print the minimum cost of consolidating the gold into k pickup locations according to the above conditions.

Input Format

The first line contains two space-separated integers describing the respective values of n (the number of mines) and k (the number of pickup locations).
Each line i of the n subsequent lines contains two space-separated integers describing the respective values of x_i (the mine's distance from the mouth of the river) and w_i (the amount of gold produced in tons) for mine i.

Note: It is guaranteed that the mines are will be given in order of ascending location.

Constraints

- $1 \leq k < n \leq 5000$
- $1 \leq w_i, x_i \leq 10^6$

Output Format

Print a single line with the minimum cost of consolidating the mined gold amongst k different pickup sites according to the rules stated above.

Sample Input 0

```
3 1
20 1
30 1
40 1
```

Sample Output 0

```
20
```

Explanation 0
We need to consolidate the gold from $n = 3$ mines into a single pickup location (because $k = 1$). The mines are all equidistant and they all produce the same amount of gold, so we just move the gold from the mines at locations $x = 20$ and $x = 40$ to the mine at $x = 30$ for a minimal cost of 20.

Sample Input 1

```
3 1
11 3
12 2
```

```
13 1
```

Sample Input 1

```
4
```

Explanation 1

We need to consolidate the gold from $n = 3$ mines into a single pickup location (because $k = 1$). We can achieve a minimum cost of 4 by moving the gold from mines $x = 12$ and $x = 13$ to the mine at $x = 11$.

Sample Input 2

```
6 2
10 15
12 17
16 18
18 13
30 10
32 1
```

Sample Output 2

```
182
```

Explanation 2

We need to consolidate the gold from $n = 6$ mines into $k = 2$ pickup locations. We can minimize the cost of doing this by doing the following:

1. Move the gold from the mines at locations $x = 10$, $x = 16$, and $x = 18$ to the mine at $x = 12$.
2. Move the gold from the mine at location $x = 32$ to the mine at $x = 30$.

Police Operation

Roy is helping the police department of his city in crime fighting. Today, they informed him about a new planned operation.

Think of the city as a $2D$ plane. The road along the X-axis is very crime prone, because n criminals live there. No two criminals live at the same position.

To catch these criminals, the police department has to recruit some police officers and give each of them USD h as wages. A police officer can start his operation from any point a, drive his car to point b in a straight line, and catch all the criminals who live on this way. The cost of fuel used by the officer's car is equal to the square of the euclidean distance between points a and b (Euclidean distance between points (x_1, y_1) and (x_2, y_2) equals to $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$).

The police department asks Roy to plan this operation. So Roy has to tell them the number of officers to recruit and the routes these officers should take in order to catch all the criminals. Roy has to provide this information while minimizing the total expenses of this operation.

Find out the minimum amount of money required to complete the operation.

Input Format

The first line contains two integers n $(0 \leq n \leq 2 \times 10^6)$, number of criminals, and h $(0 \leq h \leq 10^9)$, the cost of recruiting a police officer. The next line contains n space separated integers. The i^{th} integer indicates the position of the i^{th} criminal on X-axis (in other words, if the i^{th} integer is x, then location of the i^{th} criminal is $(x, 0)$). The value of the positions are between 1 and 10^9 and are given in increasing order in the input.

Output Format

Print the minimum amount of money required to complete the operation.

Sample Input

```
5 10
1 4 5 6 9
```

Sample Output

```
34
```

Explanation

For the sample test case, police department recruits 3 officers who get paid $3 \times 10 = 30$. The first officer starts at point $(1, 0)$ and catches the criminal there. So he does not use any fuel. The second officer catches criminals at points $(4, 0)$, $(5, 0)$ and $(6, 0)$. He burns fuel worth USD 4. The third officer catches the criminal at point $(9, 0)$. He also does not burn any fuel. The total money spent by the department is, $30 + 4 = 34$.

Timelimits

Timelimits for this challenge are given here

Zurikela's Graph

Zurikela is creating a graph with a special graph maker. At the begining, it is empty and has no nodes or edges. He can perform 3 types of operations:

1. $A\ x$: Create a set of x new nodes and name it $set\text{-}K$.
2. $B\ x\ y$: Create edges between nodes of $set\text{-}x$ and $set\text{-}y$.
3. $C\ x$: Create a set composed of nodes from $set\text{-}x$ and its directly and indirectly connected nodes, called $set\text{-}K$. Note that each node can only exist in one set, so other sets become empty.

The first set's name will be $set\text{-}1$. In first and third operation K is referring to the index of new set:

```
K = [index of last created set] + 1
```

Create the graph by completing the Q operations specified during input. Then calculate the maximum number of independent nodes (i.e.:how many nodes in the final graph which don't have direct edge between them).

Input Format

The first line contains Q.
The Q subsequent lines each contain an operation to be performed.

Constraints
$1 \leq Q \leq 10^5$.
For the first operation, $1 \leq x \leq 10^4$.
For the second operation, $x < y$ and all ys are *distinct*.
For the second and third operation, it's guaranteed that $set\text{-}x$ and $set\text{-}y$ exist.

Output Format

Print maximum number of *independent nodes* in the final graph (i.e.: nodes which have no direct connection to one another).

Sample Input

```
8
A 1
A 2
B 1 2
C 1
A 2
A 3
B 3 4
B 4 5
```

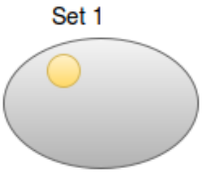
Sample Output

```
5
```

Explanation

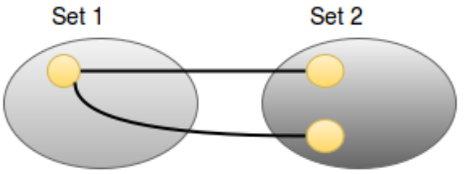
There are 8 operations.

After first operation$(A\ 1)$:

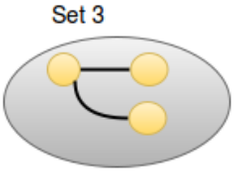


After second operation$(A\ 2)$:

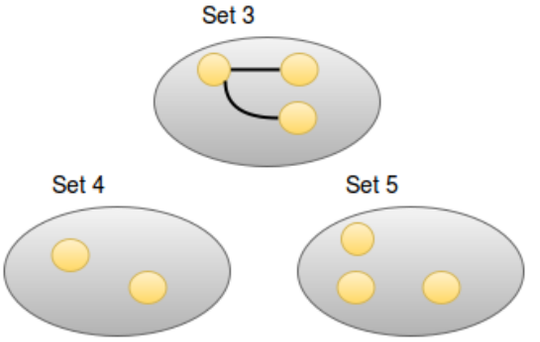
After third operation$(B\ 1\ 2)$:



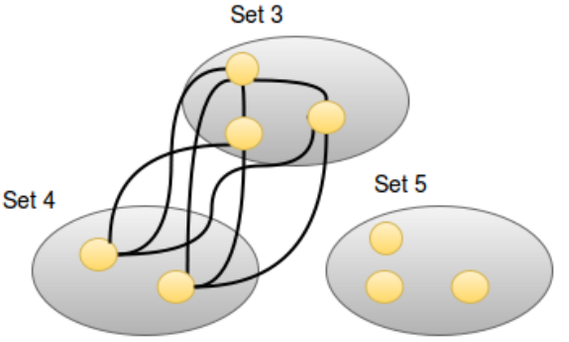
After fourth operation$(C\ 1)$:



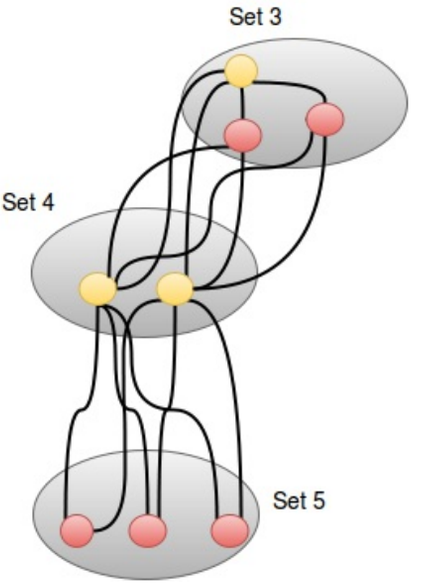
After fifth and sixth operation $(A\ 2)$ and $(A\ 3)$:



After seventh operation$(B\ 3\ 4)$:



After eigth operation$(B\ 4\ 5)$:



There are 2 independent nodes in set-3 and 3 independent nodes in set-5, so we print their sum (5) as our answer.

Modify The Sequence

You are given a sequence of integers a_1,a_2,a_3.....a_n. You are free to replace any integer with any other positive integer. How many integers must be replaced to make the resulting sequence strictly increasing?

Input Format

The first line of the test case contains an integer N - the number of entries in the sequence.
The next line contains N space separated integers where the i^{th} integer is a_i.

Output Format

Output the minimal number of integers that should be replaced to make the sequence strictly increasing.

Constraints

$0 < N \leq 10^6$
$0 < a_i \leq 10^9$

Sample Input #00

```
3
4 10 20
```

Sample Output #00

```
0
```

Sample Input #01

```
6
1 7 10 2 20 22
```

Sample Output #01

```
1
```

Sample Input #02

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

Sample Output #02

```
3
```

Explanation

In the first sample input, we need not replace anything, hence the output is 0.
In the second sample input, we can replace 2 with any integer between 11 and 19 to make the sequence strictly increasing, hence the output is 1.
In the third sample input, we can obtain 1, 2, 3, 4, 5 by changing the last three elements of the sequence.

Longest Mod Path

In the middle of a nightmare, Maxine suddenly finds herself in a mysterious room with the following items:

1. A piece of paper with the word *score* and the integer 0 written on it.
2. A map of the castle where the room is located.
   - There are N rooms uniquely labeled from 1 to N.
   - There are N bidirectional corridors connecting pairs of rooms. The value of *score* changes every time she travels up or down a corridor, and this value differs depending on her direction of travel along the corridor. Each corridor can be traveled any number of times in either direction.
   - Every room is reachable from every other room.
   - Maxine is located in the room labeled S.
   - The exit is located in the room labeled E. Once this room is reached, *score* is reduced *modulo* M and Maxine can (but is not required to) exit that level!

Assume some corridor i (where $1 \leq i \leq N$) is associated with an integer, x_i, and connects rooms a_i and b_i. Then:

- Traveling corridor i from room a_i to room b_i *increases score* by x_i.
- Traveling corridor i from room b_i to room a_i *decreases score* by x_i.

There are Q levels to Maxine's nightmare castle, and each one has a different set of values for S, E, and M. Given the above information, help Maxine by finding and printing her maximum possible score for each level. Only you can help her wake up from this nightmare!

Note: Recall that the result of a modulo operation is *always non-negative*. For example, $(-8) \bmod 5 = 2$.

Input Format

The first line contains a single integer, N, denoting the number of rooms.
Each of the N subsequent lines describes a corridor in the form of three space-separated integers denoting the respective values for a_i, b_i, and x_i.
The next line contains a single integer, Q, denoting the number of queries.
Each of the Q subsequent lines describes a level in the form of three space-separated integers denoting its respective S, E, and M values.

Constraints

- $1 \leq N \leq 10^5$
- $1 \leq a_i, b_i \leq N$, $a_i \neq b_i$
- $1 \leq x_i \leq 10^9$
- $1 \leq Q \leq 10^5$

For each level:

- The room layout is the same
- $1 \leq S, E \leq N$

- $1 \leq M \leq 10^9$

Subtask

- $1 \leq N, Q, M \leq 300$ for 30% of max score.

Output Format

For each of the Q levels, print the maximum possible score for that level on a new line.

Sample Input

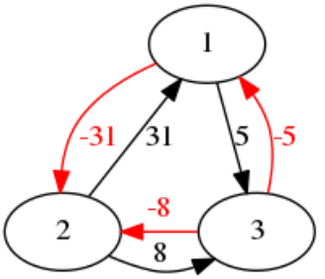
```
3
1 3 5
2 3 8
2 1 31
1
1 2 13
```

Sample Output

```
12
```

Explanation

The *Sample Input* represents the following setup:



We want to travel from room 1 to room 2 while maximizing the value of *score*. There are at least two ways to achieve the maximum *score* value of 12:

1. Travel through corridors 5 times: $1 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 2$

   $score = (5 - 8 + 31 + 5 - 8) \bmod 13 = 25 \bmod 13 = 12$.

2. Travel through corridors 34 times:
   $1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow \ldots \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2$
   $score = -339 \bmod 13 = 12$, because 12 is the smallest non-negative integer x such that 13 divides $(-339 - x)$.

P-sequences

We call a sequence of N natural numbers (a_1, a_2, ..., a_N) a *P-sequence*, if the product of any two adjacent numbers in it is not greater than *P*. In other words, if a sequence (a_1, a_2, ..., a_N) is a *P-sequence*, then $a_i * a_{i+1} \leq P \ \forall \ 1 \leq i < N$

You are given N and P. Your task is to find the number of such *P-sequences* of N integers modulo 10^9+7.

Input Format

The first line of input consists of N
The second line of the input consists of P.

Constraints

$2 \leq N \leq 10^3$
$1 \leq P \leq 10^9$
$1 \leq a_i$

Output Format

Output the number of *P-sequences* of N integers modulo 10^9+7.

Sample Input 0

```
2
2
```

Sample Output 0

```
3
```

Explanation 0

3 such sequences are {1,1},{1,2} and {2,1}

Robot

You have two arrays of integers, $V = \{V_1, V_2, \ldots, V_N\}$ and $P = \{P_1, P_2, \ldots, P_N\}$, where both have N number of elements. Consider the following function:

```
score = 0

int Go(step, energy) {
   if (step == N) {
      score += V[step];
      return (score);
   }
   else {
      int way = random(1, 2);
      if (way == 1) {
         score += V[step];
      }
      else {
         energy = P[step];
      }
      if (energy > 0) {
         Go(step + 1, energy - 1);
      }
      else {
         KillTheWorld();
      }
   }
}
```

What is the maximum possible value of score that we can get in the end, if we call $Go(1, 0)$?.
Note that the function should never invoke *KillTheWorld* function. And $random(1, 2)$ generates a random integer from set *[1, 2]*.
It is guaranteed there will be a solution that *wont* kill the world.

Input Format

The first line contains an integer N. Each of the following N lines contains a pair of integers. The i-th line contains a pair of numbers, V_i, P_i, separated by space.

Constraints

$1 \leq N \leq 5 \times 10^5$
$0 \leq V_i \leq 10^9$
$0 \leq P_i \leq 10^5$

Output Format

Derive the maximum score given by `return (score);`.

Sample Input

```
4
4 2
0 2
4 0
3 4
```

Sample Output

```
7
```

Explanation

In the best case, the first and second function call in Go variable way will take value 2, while in the other calls it will be equal to 1 then the final score will be equal to the value of 7.

Lucky Numbers

A number is called *lucky* if the sum of its digits, as well as the sum of the squares of its digits is a prime number. How many numbers between a and b inclusive, are lucky?

For example, $a = 20$ and $b = 25$. Each number is tested below:

```
      digit   digit   squares
value   sum     squares sum
20      2       4,0     4
21      3       4,1     5
22      4       4,4     8
23      5       4,9     13
24      6       4,16    20
25      7       4,25    29
```

We see that two numbers, 21 and 25 are *lucky*.

Note: These lucky numbers are not to be confused with Lucky Numbers

Input Format

The first line contains the number of test cases T.
Each of the next T lines contains two space-separated integers, a and b.

Constraints

- $1 \leq T \leq 10^4$
- $1 \leq a \leq b \leq 10^{18}$

Output Format

Output T lines, one for each test case in the order given.

Sample Input

```
2
1 20
120 130
```

Sample Output

```
4
1
```

Explanation

For the first case, the lucky numbers are $11, 12, 14$, and 16.
For the second case, the only lucky number is 120.

Unfair Game

You are playing a game of Nim with a friend. The rules are are follows:

1) Initially, there are N piles of stones. Two players play alternately.

2) In each turn, a player can choose one non empty pile and remove any number of stones from it. At least one stone must be removed.

3) The player who picks the last stone from the last non empty pile wins the game.

It is currently your friend's turn. You suddenly realize that if your friend was to play optimally in that position, you would lose the game. So while he is not looking, you decide to cheat and add some (possibly 0) stones to each pile. You want the resultant position to be such that your friend has no guaranteed winning strategy, even if plays optimally. You cannot create a new pile of stones. *You can only add stones, and not remove stones from a pile*. What is the least number of stones you need to add?

Input Format

The first line contains the number of cases T. T cases follow. Each case contains the number N on the first line followed by N numbers on the second line. The ith number denotes si, the number of stones in the ith pile currently.

Constraints

1 <= T <= 20

2 <= N <= 15

1 <= si < 1000000000 (10^9)

Output Format

Output T lines, containing the answer for each case. If the current position is already losing for your friend, output 0.

Sample Input

3

2

1 3

3

1 1 1

4

10 4 5 1

Sample Output

2

3

6

Explanation

For the first case, add 2 stones to the first pile. Then, both piles will have 3 stones each. It is easy to verify that your friend cannot win the game unless you make a mistake.

For the second case, add 1 stone to the first pile, and 2 stones to the second pile.

Oil Well

Mr. Road Runner bought a piece of land in the middle of a desert for a nominal amount. It turns out that the piece of land is now worth millions of dollars as it has an oil reserve under it. Mr. Road Runner contacts the ACME corp to set up the oil wells on his land. Setting up oil wells is a costly affair and the charges of setting up oil wells are as follows.

The rectangular plot bought by Mr. Road Runner is divided into *r* * *c* blocks. Only some blocks are suitable for setting up the oil well and these blocks have been marked. ACME charges nothing for building the first oil well. For every subsequent oil well built, the cost would be the maximum **ACME distance** between the new oil well and the existing oil wells.

If *(x,y)* is the position of the block where a new oil well is setup and *(x1, y1)* is the position of the block of an existing oil well, the *ACME distance* is given by

```
max(|x-x1|, |y-y1|)
```

the maximum *ACME distance* is the maximum among all the *ACME distance* between existing oil wells and new wells.

If the distance of any two adjacent blocks (horizontal or vertical) is considered 1 unit, what is the minimum cost *(E)* in units it takes to set up oil wells across all the marked blocks?

Input Format

The first line of the input contains two space separated integers *r**c*.
r lines follow each containing *c* space separated integers.
1 indicates that the block is suitable for setting up an oil well, whereas 0 isn't.

```
r c
M11 M12 ... M1c
M21 M22 ... M2c
...
Mr1 Mr2 ... Mrc
```

Constraints

```
1 <= r, c <= 50
```

Output Format

Print the minimum value E as the answer.

Sample Input

```
3 4
1 0 0 0
1 0 0 0
0 0 1 0
```

Sample Output

```
3
```

Explanation

(1, 1) (2, 1) (3, 3) are the places where are to be setup.

There are 3! = 6 ways to do it.

(1, 1) (2, 1) (3, 3) ==> cost = 0 + 1 + 2 = 3
(1, 1) (3, 3) (2, 1) ==> cost = 0 + 2 + 2 = 4
(2, 1) (1, 1) (3, 3) ==> cost = 0 + 1 + 2 = 3
(2, 1) (3, 3) (1, 1) ==> cost = 0 + 2 + 2 = 4
(3, 3) (1, 1) (2, 1) ==> cost = 0 + 2 + 2 = 4
(3, 3) (2, 1) (1, 1) ==> cost = 0 + 2 + 2 = 4

So E = 3

Find the Seed

A company needs random numbers for its operation. N random numbers have been generated using N numbers as seeds and the following recurrence formula:

$$F(K) = (C(1) \times F(K-1) + C(2) \times F(K-2) + \cdots + C(N-1) \times F(K-N+1) + C(N) \times F(K-N)) \ \% \ (10^9 + 7)$$

The numbers used as seeds are $F(N-1), F(N-2), \ldots, F(1), F(0)$. $F(K)$ is the K^{th} term of the recurrence.

Due to a failure on the servers, the company lost its seed numbers. Now they just have the recurrence formula and the previously generated N random numbers.

The company wants to recover the numbers used as seeds, so they have hired you for doing this task.

Input Format

The first line contains two space-separated integers, N and K, respectively.
The second line contains the space-separated integers describing
$F(K), F(K-1), \ldots, F(K-N+2), F(K-N+1)$ (all these numbers are non-negative integers $< 10^9$).
The third line contains the space-separated coefficients of the recurrence formula,
$C(1), C(2), \ldots, C(N-1), C(N)$. All of these coefficients are positive integers $< 10^9$.

Constraints

- $1 \leq N \leq 50$
- $1 \leq K \leq 10^9$
- $0 \leq K - N + 1$

Output Format

The output must be one line containing the space-separated seeds of the random numbers -
$F(N-1), F(N-2), \ldots, F(1), F(0)$.

Sample Input

```
2 6
13 8
1 1
```

Sample Output

```
1 1
```

Explanation

This is the classic Fibonacci recurrence. We have the 6^{th} and 5^{th} terms, and, of course, the seeds are the numbers 1 and 1.

The Blacklist

A new gangster is trying to take control of the city. He makes a list of his N adversaries (e.g. *gangster* 1, *gangster* 2, ... *gangster* $N-1$, *gangster* N) and plans to get rid of them.

K mercenaries are willing to do the job. The gangster can use any number of these mercenaries. But he has to honor one condition set by them: they have to be assigned in such a way that they eliminate a consecutive group of gangsters in the list, e.g. *gangster* i, *gangster* $i+1$, ..., *gangster* $j-1$, *gangster* j, where the following is true: $1 \leq i \leq j \leq N$.

While our new gangster wants to kill all of them, he also wants to pay the least amount of money. All mercenaries charge a different amount to kill different people. So he asks you to help him minimize his expenses.

Input Format

The first line contains two space-separated integers, N and K. Then K lines follow, each containing N integers as follows:
The j^{th} number on the i^{th} line is the amount charged by the i^{th} mercenary for killing the j^{th} gangster on the list.

Constraints

- $1 \leq N \leq 20$
- $1 \leq K \leq 10$
- $0 \leq amount\ charged \leq 10000$

Output Format

Just one line, the minimum cost for killing the N gangsters on the list.

Sample Input

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

Sample Output

```
5
```

Explanation

The new gangster can assign *mercenary 1* to kill *gangster 1*, and *mercenary 2* to kill *gangster 2* and *gangster 3*.

Tree Pruning

A tree, t, has n vertices numbered from 1 to n and is rooted at vertex 1. Each vertex i has an integer weight, w_i, associated with it, and t's *total weight* is the sum of the weights of its nodes. A single *remove operation* removes the subtree rooted at some arbitrary vertex u from tree t.

Given t, perform up to k remove operations so that the total weight of the remaining vertices in t is maximal. Then print t's maximal total weight on a new line.

Note: If t's total weight is already maximal, you may opt to remove 0 nodes.

Input Format

The first line contains two space-separated integers, n and k, respectively.
The second line contains n space-separated integers describing the respective weights for each node in the tree, where the i^{th} integer is the weight of the i^{th} vertex.
Each of the $n-1$ subsequent lines contains a pair of space-separated integers, u and v, describing an edge connecting vertex u to vertex v.

Constraints

- $2 \leq n \leq 10^5$
- $1 \leq k \leq 200$
- $1 \leq i \leq n$
- $-10^9 \leq w_i \leq 10^9$

Output Format

Print a single integer denoting the largest total weight of t's remaining vertices.

Sample Input

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

Sample Output

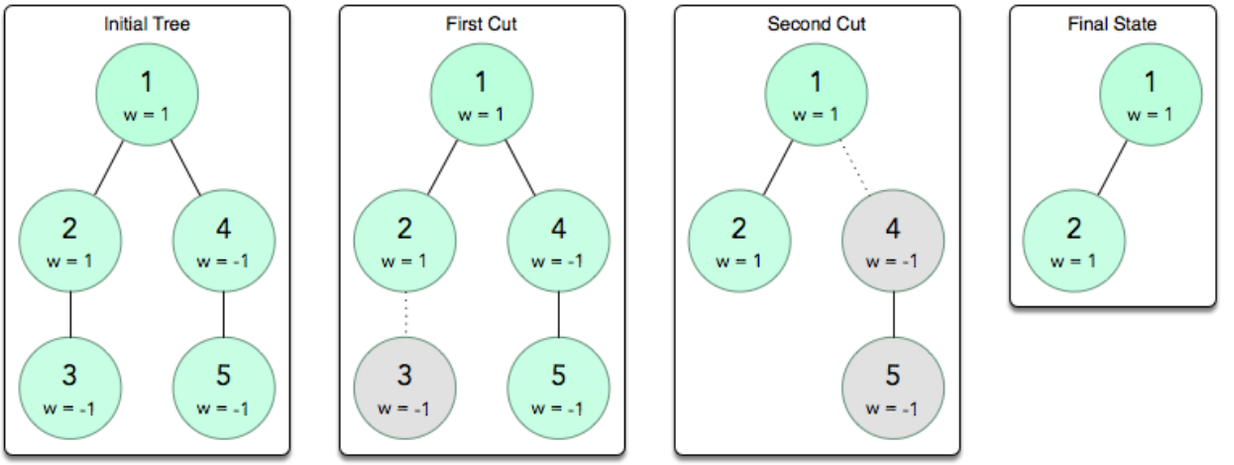
```
2
```

Explanation

We perform 2 remove operations:

1. Remove the subtree rooted at node 3. Losing this subtree's -1 weight increases the tree's total weight by 1.
2. Remove the subtree rooted at node 4. Losing this subtree's -2 weight increases the tree's total weight by 2.

The sum of our remaining positively-weighted nodes is $1 + 1 = 2$, so we print 2 on a new line.


Initial Tree
1
w = 1
2
w = 1
4
w = -1
3
w = -1
5
w = -1
First Cut
1
w = 1
2
w = 1
4
w = -1
3
w = -1
5
w = -1
Second Cut
1
w = 1
2
w = 1
4
w = -1
5
w = -1
Final State
1
w = 1
2
w = 1

Ones and Twos

You are using at most **A** number of 1s and at most **B** number of 2s. How many different evaluation results are possible when they are formed in an expression containing only addition `+` sign and multiplication `*` sign are allowed?

Note that, multiplication takes precedence over addition.

For example, if **A=2** and **B=2**, then we have the following expressions:

- `1`, `1*1` = 1
- `2`, `1*2`, `1*1*2`, `1+1` = 2
- `1+2`, `1+1*2` = 3
- `2+2`, `2*2`, `1+1+2`, `1*2*2`, `1*1*2*2`, `1*2+1*2`, `1*1*2+2`, `1*2+2` = 4
- `1+2+2`, `1+1*2+2` = 5
- `1+1+2+2`, `1+1+2*2` = 6

So there are 6 unique results that can be formed if A = 2 and B = 2.

Input Format

The first line contains the number of test cases T, T testcases follow each in a newline.
Each testcase contains 2 integers A and B separated by a single space.

Constraints

$1 <= T <= 10^5$
0<=A<=1000000000
0<=B<=1000

Output Format

Print the number of different evaluations modulo (%) (10^9+7.)

Sample Input

```
4
0 0
2 2
0 2
2 0
```

Sample Output

```
0
6
2
2
```

Explanation

- When A = 0, B = 0, there are no expressions, hence 0.
- When A = 2, B = 2, as explained in the problem statement above, expressions leads to 6 possible solutions.
- When A = 0, B = 2, we have `2`, `2+2` or `2*2`, hence 2.
- When A = 2, B = 0, we have `1` or `1*1`, `1+1` hence 2.

Count Scorecards

In a tournament, N players play against each other exactly once. Each game results in exactly one player winning. There are no ties. You have been given a scorecard containing the scores of each player at the end of the tournament. The score of a player is the total number of games the player won in the tournament. However, the scores of some players might have been erased from the scorecard. How many possible scorecards are consistent with the input scorecard?

Input Format

The first line contains the number of cases T. T cases follow. Each case contains the number N on the first line followed by N numbers on the second line. The ith number denotes s_i, the score of the ith player. If the score of the ith player has been erased, it is represented by -1.

Constraints

$1 <= T <= 20$
$1 <= N <= 40$
$-1 <= s\_i < N$

Output Format

Output T lines, containing the answer for each case. Output each result modulo 1000000007.

Sample Input 0

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

Sample Output 0

```
2
7
1
0
12
```

Explanation 0

For the first case, there are 2 scorecards possible: (0,1,2) or (1,0,2).
For the second case, the valid scorecards are (1,1,1), (0,1,2), (0,2,1), (1,0,2), (1,2,0), (2,0,1), (2,1,0).
For the third case, the only valid scorecard is (0,1,2,3).
For the fourth case, there is no valid scorecard. It is not possible for both players to have score of 1.
For the fifth case, 6-variations of {(3,1,0)[2]}, and 3 variations each of {(2,2,0)[2]} and {(2,1,1)[2]}.

Vim War

A war has broken down between Vim and Emacs. Gedit, being Vim's ally, is captured by Emacs as a prisoner of war and it is up to Vim to rescue him by defeating Emacs.

For this task, Vim has to assemble an army of appropriate skills. He can choose a **non-empty** subset of soldiers from a set of N soldiers (numbered from 1 to N). Each soldier has some subset of skills out of M different skills (numbered from 1 to M). The skill-set of an army is the union of skill-sets of its constituent soldiers. To win the war, Vim needs to know how many different subsets of soldiers satisfy his skill-set requirement. Since the answer can be huge, print it modulo $10^9 + 7$.

Note : The chosen army's skill-set must **exactly** match the skill-set requirement of Vim (i.e no extra skills must be present in the army's skill-set than what is required).

Input Format

The first line contains N and M, the number of soldiers to choose from and the number of different skills possible respectively.
The next N lines contain M boolean characters each. If the j^{th} character of the i^{th} line is 1, then the i^{th} soldier possess the j^{th} skill and if it is 0, then not.
The last line contains M boolean characters denoting the requirement skill-set of Vim where the j^{th} character being 1 signifies that Vim wants the j^{th} skill to be present in his final army and not, otherwise.

Constraints

$1 \leq N \leq 10^5$
$1 \leq M \leq 20$

Output Format

Output in a single line the required answer, as explained above.

Sample Input

```
4 2
00
10
01
11
11
```

Sample Output

```
10
```

Explanation

Vim wants both the skills to be present in his selected army. Hence, he can choose the following subsets of soldiers:

1. $1, 2, 3, 4$
2. $1, 2, 4$
3. $1, 3, 4$
4. $2, 3, 4$
5. $1, 4$

6. $\mathbf{2, 4}$
7. $\mathbf{3, 4}$
8. $\mathbf{4}$
9. $\mathbf{1, 2, 3}$
10. $\mathbf{2, 3}$

Best spot

In Chile, land are partitioned into a one large grid, where each element represents a land of size *1x1*.

Shaka is a newcomer in Chile and is trying to start his own business. He is planning to build a store. He has his own ideas for the "perfect store" which can be represented by a *HxW* grid. Element at position *(i, j)* represents height of land at index *(i, j)* in the grid.

Shaka has purchased a land area which can be represented *RxC* grid (*H <= R, W <= C*). Shaka is interested in finding best *HxW* sub-grid in the acquired land. In order to compare the possible sub-grids, Shaka will be using the sum of squared difference between each cell of his "perfect store" and it's corresponding cell in the subgrid. Amongst all possible sub-grids, he will choose the one with smallest such sum.

Note

- The grids are 1-indexed and rows increase from top to bottom and columns increase from left to right.
- If *x* is the height of a cell in the "perfect store" and *y* is the height of the corresponding cell in a sub-grid of the acquired land, then the squared difference is defined as $(x-y)^2$

Input Format

The first line of the input consists of two integers, *R C*, separated by single space.
Then *R* lines follow, each one containing *C* space separated integers, which describe the height of each land spot of the purchased land.
The next line contains two integers, *H W*, separated by a single space, followed by *H* lines with *W* space separated integers, which describes the "perfect store".

Constraints

1 <= *R, C* <= 500
1 <= *H* <= *R*
1 <= *W* <= *C*
No height will have an absolute value greater than 20.

Output Format

In the first line, output the smallest possible sum (as defined above) Shaka can find on exploring all the sub-grids (of size *HxW*) in the purchased land.
In second line, output two space separated integers, *i j*, which represents the index of top left corner of sub-grid (on the acquired land) with the minimal such sum. If there are multiple sub-grids with minimal sum, output the one with the smaller row index. If there are still multiple sub-grids with minimal sum, output the one with smaller column index.

Sample Input

```
3 3
19 19 -12
5 8 -14
-12 -11 9
2 2
-18 -12
-10 -7
```

Sample Output

```
937
```

```
2 2
```

Explanation

The result is computed as follows: $(8 - (-18))^2 + (-14 - (-12))^2 + (-11 - (-10))^2 + (9 - (-7))^2 = 937$

Divisible Numbers

Given an integer, n, find the smallest integer m such that m is divisible by n (i.e., n is a factor of m) and satisfies the following properties:

- m must not contain zeroes in its decimal representation.
- The sum of m's digits must be *greater than or equal to* the product of m's digits.

Given n, find m and print *the number of digits* in m's decimal representation.

Input Format

A single integer denoting n.

Constraints

- $1 \leq n \leq 3 \times 10^4$
- n is not divisible by 10.

Time Limits

- The time limits for this challenge are available here.

Output Format

Print the *number of digits* in the decimal representation of the smallest possible m.

Sample Input 0

```
1
```

Sample Output 0

```
1
```

Explanation 0

$m = 1$ is evenly divided by $n = 1$, doesn't contain any zeroes in its decimal representation, and the sum of its digits is not less than the product of its digits. Thus, we print the number of digits in $m = 1$ (which also happens to be 1) as our answer.

Sample Input 1

```
9
```

Sample Output 1

```
1
```

Explanation 1

$m = 9$ is evenly divided by $n = 9$, doesn't contain any zeroes in its decimal representation, and the sum of its digits is not less than the product of its digits. Thus, we print the number of digits in $m = 9$, which is 1, as our answer.

Unique Divide And Conquer

Divide-and-Conquer on a tree is a powerful approach to solving tree problems.

Imagine a tree, t, with n vertices. Let's remove some vertex v from tree t, splitting t into zero or more connected components, $t_1, t_2, \ldots, t_k$, with vertices $n_1, n_2, \ldots, n_k$. We can prove that there is a vertex, v, such that the size of each formed components is *at most* $\lfloor \frac{n}{2} \rfloor$.

The Divide-and-Conquer approach can be described as follows:

- Initially, there is a tree, t, with n vertices.
- Find vertex v such that, if v is removed from the tree, the size of each formed component after removing v is *at most* $\lfloor \frac{n}{2} \rfloor$.
- Remove v from tree t.
- Perform this approach recursively for each of the connected components.

We can prove that if we find such a vertex v in linear time (e.g., using *DFS*), the entire approach works in $\mathcal{O}(n \cdot \log n)$. Of course, sometimes there are several such vertices v that we can choose on some step, we can take and remove any of them. However, right now we are interested in trees such that *at each step* there is a unique vertex v that we can choose.

Given n, count the number of tree t's such that the Divide-and-Conquer approach works determinately on them. As this number can be quite large, your answer must be modulo m.

Input Format

A single line of two space-separated positive integers describing the respective values of n (the number of vertices in tree t) and m (the modulo value).

Constraints

- $1 \leq n \leq 3000$
- $n < m \leq 10^9$
- m is a prime number.

Subtasks

- $n \leq 9$ for 40% of the maximum score.
- $n \leq 500$ for 70% of the maximum score.

Output Format

Print a single integer denoting the number of tree t's such that vertex v is unique at each step when applying the Divide-and-Conquer approach, modulo m.

Sample Input 0

```
1 103
```

Sample Output 0

```
1
```

Explanation 0

For $n = 1$, there is only one way to build a tree so we print the value of $1 \bmod 103 = 1$ as our answer.

Sample Input 1

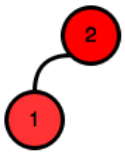
```
2 103
```

Sample Output 1

```
0
```

Explanation 1

For $n = 2$, there is only one way to build a tree:



This tree is *not valid* because we can choose to remove either node 1 or node 2 in the first step. Thus, we print 0 as no valid tree exists.

Sample Input 2

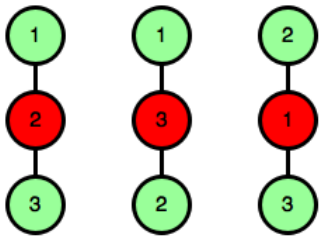
```
3 103
```

Sample Output 2

```
3
```

Explanation 2

For $n = 3$, there are 3 valid trees depicted in the diagram below (the unique vertex removed in the first step is shown in red):



Thus, we print the value of $3 \bmod 103 = 3$ as our answer.

Sample Input 3

```
4 103
```

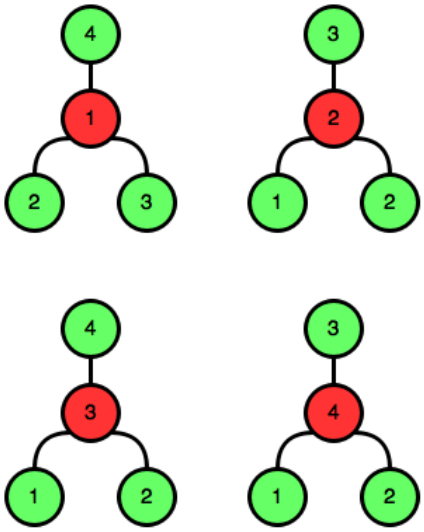
Sample Output 3

```
4
```

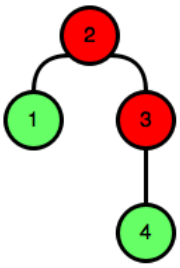
Explanation 3

For $n = 4$, there are 4 valid trees depicted in the diagram below (the unique vertex removed in the first

step is shown in red):



The figure below shows an invalid tree with $n = 4$:



This tree is *not valid* because we can choose to remove node 2 or node 3 in the first step. Because we had four valid trees, we print the value of $4 \bmod 103 = 4$ as our answer.

King and Four Sons

The King of Byteland wants to grow his territory by conquering K other countries. To prepare his 4 heirs for the future, he decides they must work together to capture each country.

The King has an army, A, of N battalions; the i^{th} battalion has A_i soldiers. For each battle, the heirs get a detachment of soldiers to share but will fight amongst themselves and lose the battle if they don't each command the same number of soldiers (i.e.: the detachment must be divisible by 4). If given a detachment of size 0, the heirs will fight alone without any help.

The battalions chosen for battle must be selected in the following way:

1. A subsequence of K battalions must be selected (from the N battalions in army A).
2. The j^{th} battle will have a squad of soldiers from the j^{th} selected battalion such that its size is divisible by 4.

The soldiers within a battalion have unique strengths. For a battalion of size 5, the detachment of soldiers $\{0, 1, 2, 3\}$ is *different* from the detachment of soldiers $\{0, 1, 2, 4\}$

The King tasks you with finding the number of ways of selecting K detachments of battalions to capture K countries using the criterion above. As this number may be quite large, print the answer modulo $10^9 + 7$.

Input Format

The first line contains two space-separated integers, N (the number of battalions in the King's army) and K (the number of countries to conquer), respectively.

The second line contains N space-separated integers describing the King's army, A, where the i^{th} integer denotes the number of soldiers in the i^{th} battalion (A_i).

Constraints

- $1 \leq N \leq 10^4$
- $1 \leq K \leq min(100, N)$
- $1 \leq A_i \leq 10^9$
- $1 \leq A_i \leq 10^3$ holds for test cases worth at least 30% of the problem's score.

Output Format

Print the number of ways of selecting the K detachments of battalions modulo $10^9 + 7$.

Sample Input

```
3 2
3 4 5
```

Sample Output

```
20
```

Explanation

First, we must find the ways of selecting 2 of the army's 3 battalions; then we must find all the ways of selecting detachments for each choice of battalion.

Battalions $\{A_0, A_1\}$*:*
A_0 has 3 soldiers, so the only option is an empty detachment ($\{\}$).
A_1 has 4 soldiers, giving us 2 detachment options ($\{\}$ and $\{0, 1, 2, 3\}$).
So for this subset of battalions, we get $1 \times 2 = 2$ possible detachments.

Battalions $\{A_0, A_2\}$*:*
A_0 has 3 soldiers, so the only option is an empty detachment ($\{\}$).
A_2 has 5 soldiers, giving us 6 detachment options ($\{\}$, $\{0, 1, 2, 3\}$, $\{0, 1, 2, 4\}$, $\{1, 2, 3, 4\}$, $\{0, 1, 3, 4\}$, $\{0, 2, 3, 4\}$). So for this subset of battalions, we get $1 \times 6 = 6$ possible detachments.

Battalions $\{A_1, A_2\}$*:*
A_1 has 4 soldiers, giving us 2 detachment options ($\{\}$ and $\{0, 1, 2, 3\}$).
A_2 has 5 soldiers, giving us 6 detachment options ($\{\}$, $\{0, 1, 2, 3\}$, $\{0, 1, 2, 4\}$, $\{1, 2, 3, 4\}$, $\{0, 1, 3, 4\}$, $\{0, 2, 3, 4\}$).
So for this subset of battalions, we get $2 \times 6 = 12$ possible detachments.

In total, we have $2 + 6 + 12 = 20$ ways to choose detachments, so we print $20 \ \% \ (10^9 + 7)$, which is 20.

Dortmund Dilemma

Borussia Dortmund are a famous football (soccer) club from Germany. Apart from their fast-paced style of playing, the thing that makes them unique is the hard to pronounce names of their players (błaszczykowski , papastathopoulos , großkreutz etc.).

The team's coach is your friend. He is in a dilemma as he can't decide how to make it easier to call the players by name, during practice sessions. So, you advise him to assign easy names to his players. A name is easy to him if

1. It consists of only one word.
2. It consists of only lowercase english letters.
3. Its length is **exactly** N.
4. It contains **exactly** K different letters from the 26 letters of English alphabet.
5. At least one of its **proper** prefixes matches with its **proper** suffix of same length.

Given, N and K you have to tell him the number of easy names he can choose from modulo $(10^9 + 9)$.

Note : A prefix P of a name W is proper if, $P \neq W$. Similarly, a suffix S of a name W is proper if, $S \neq W$.

Input Format
The first line of the input will contain T (the number of testcases). Each of the next T lines will contain 2 space separated integers N and K.

Output Format
For each testcase, output the number of ways the coach can assign names to his players modulo $(10^9 + 9)$.

Constraints
$1 \leq T \leq 10^5$
$1 \leq N \leq 10^5$
$1 \leq K \leq 26$

Sample Input #1

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

Sample Output #1

```
0
26
2600
```

Sample Input #2

```
5
2 2
5 1
3 2
6 2
1 3
```

Sample Output #2

```
0
26
650
13650
0
```

Super Kth LIS

Given an array of N integers ($a_0, a_1, \ldots, a_{N-1}$), find all possible increasing subsequences of maximum length, L. Then print the lexicographically K^{th} longest increasing subsequence as a single line of space-separated integers; if there are less than K subsequences of length L, print -1.

Two subsequences $[a_{p_0}, a_{p_1}, \ldots, a_{p_{L-2}}, a_{p_{L-1}}]$ and $[a_{q_0}, a_{q_1}, a_{q_2}, \ldots, a_{q_{L-2}}, a_{q_{L-1}}]$ are considered to be *different* if there exists at least one i such that $p_i \neq q_i$.

Input Format

The first line contains 2 space-separated integers, N and K, respectively.
The second line consists of N space-separated integers denoting $a_0, a_1, \ldots, a_{N-1}$ respectively.

Constraints

- $1 \leq N \leq 10^5$
- $1 \leq K \leq 10^{18}$
- $1 \leq a_i \leq N$

Scoring

- $1 \leq N \leq 10^3$ for 30% of the test data.
- $1 \leq N \leq 10^5$ for 100% of the test data.

Output Format

Print a single line of L space-separated integers denoting the lexicographically K^{th} longest increasing subsequence; if there are less than K subsequences of length L, print -1.

Note: L is the length of longest increasing subsequence in the array.

Sample Input 0

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

Sample Output 0

```
1 3 5
```

Sample Input 1

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

Sample Output 1

```
1 3 4 5
```

Explanation

Sample Case 0:
The longest possible increasing subsequences in lexicographical order are:

1. $[1, 2, 5]$
2. $[1, 2, 5]$
3. $[1, 3, 5]$

Notice that the first and second subsequences appear the same; they are actually both *different* because the 1 in the first subsequence comes from array element a_0, and the 1 in the second subsequence comes from array element a_2. Because $K = 3$, we print the 3^{rd} one ($[1, 3, 5]$) as a single line of space-separated integers.

Sample Case 1:
The longest possible increasing subsequences in lexicographical order are:

1. $[1, 2, 4, 5]$
2. $[1, 3, 4, 5]$

Because $K = 2$, we print the 2^{nd} one ($[1, 3, 4, 5]$) as a single line of space-separated integers.

Counting the Ways

Little Walter likes playing with his toy scales. He has N types of weights. The i^{th} weight type has weight a_i . There are infinitely many weights of each type.

Recently, Walter defined a function, $F(X)$, denoting the number of different ways to combine several weights so their total weight is equal to X. Ways are considered to be different if there is a type which has a different number of weights used in these two ways.

For example, if there are 3 types of weights with corresonding weights 1, 1, and 2, then there are 4 ways to get a total weight of 2:

1. Use 2 weights of type 1.
2. Use 2 weights of type 2.
3. Use 1 weight of type 1 and 1 weight of type 2.
4. Use 1 weight of type 3.

Given N, L, R, and $a_1, a_2, \ldots, a_N$, can you find the value of $F(L) + F(L+1) + \ldots + F(R)$?

Input Format

The first line contains a single integer, N, denoting the number of types of weights.
The second line contains N space-separated integers describing the values of $a_1, a_2, \ldots, a_N$, respectively
The third line contains two space-separated integers denoting the respective values of L and R.

Constraints

- $1 \leq N \leq 10$
- $0 < a_i \leq 10^5$
- $a_1 \times a_2 \times \ldots \times a_N \leq 10^5$
- $1 \leq L \leq R \leq 10^{17}$

Note: The time limit for C/C++ is 1 second, and for Java it's 2 seconds.

Output Format

Print a single integer denoting the answer to the question. As this value can be very large, your answer must be modulo $10^9 + 7$.

Sample Input

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

Sample Output

```
22
```

Explanation

$F(1) = 1$
$F(2) = 2$
$F(3) = 3$
$F(4) = 4$
$F(5) = 5$
$F(6) = 7$

Hard Disk Drives

There are n *pairs* of hard disk drives (HDDs) in a cluster. Each HDD is located at an integer coordinate on an infinite straight line, and each pair consists of one *primary* HDD and one *backup* HDD.

Next, you want to place k computers at integer coordinates on the same infinite straight line. Each *pair* of HDDs must then be connected to a single computer via *wires*, but a computer can have any number (even zero) of HDDs connected to it. The *length* of a wire connecting a single HDD to a computer is the absolute value of the distance between their respective coordinates on the infinite line. We consider the *total length* of wire used to connect all the HDDs to computers to be the sum of the lengths of all the wires used to connect HDDs to computers. Note that both the primary and secondary HDDs in a pair *must* connect to the same computer.

Given the locations of n pairs (i.e., primary and backup) of HDDs and the value of k, place all k computers in such a way that the total length of wire needed to connect each pair of HDDs to computers is *minimal*. Then print the total length on a new line.

Input Format

The first line contains two space-separated integers denoting the respective values of n (the number of *pairs* of HDDs) and k (the number of computers).
Each line i of the n subsequent lines contains two space-separated integers describing the respective values of a_i (coordinate of the primary HDD) and b_i (coordinate of the backup HDD) for a pair of HDDs.

Constraints

- $2 \leq k \leq n \leq 10^5$
- $4 \leq k \times n \leq 10^5$
- $-10^9 \leq a_i, b_i \leq 10^9$

Output Format

Print a single integer denoting the minimum total length of wire needed to connect all the pairs of HDDs to computers.

Sample Input

```
5 2
6 7
-1 1
0 1
5 2
7 3
```

Sample Output

```
13
```

Explanation

For the given *Sample Case*, it's optimal to place computers at positions 0 and 6 on our infinite line. We then connect the second ($a = -1, b = 1$) and the third ($a = 0, b = 1$) pairs of HDDs to the first computer (at position 0) and then connect the remaining pairs to the second computer (at position 6).

We calculate the wire lengths needed to connect the drives to each computer. The amount of wire needed to connect the second and third drives to the first computer is $(1 + 1) + (0 + 1) = 3$, and the amount of

wire needed to connect the rest of the drives to the second computer is $(0+1)+(1+4)+(1+3)=10$. When we sum the lengths of wire needed to connect all pairs of drives to the two computers, we get a total length of $3+10=13$. Thus, we print 13 as our answer.

Separate the chocolate

Chinese Version
Russian Version

Tom and Derpina have a rectangular shaped chocolate bar with chocolates labeled T, D and U. They want to split the bar into exactly two pieces such that:

- Tom's piece can not contain any chocolate labeled D and similarly, Derpina's piece can not contain any chocolate labeled T and U can be used by either of the two.
- All chocolates in each piece must be connected (two chocolates are connected if they share an edge), i.e. the chocolates should form one connected component
- The absolute difference between the number of chocolates in pieces should be at most K
- After dividing it into exactly two pieces, in any piece, there should not be 4 adjacent chocolates that form a square, i.e. there should not be a fragment like this:
  XX
  XX

Input Format

The first line of the input contains 3 integers M, N and K separated by a single space.
M lines follow, each of which contains N characters. Each character is 'T','D' or 'U'.

Constraints

$0 \leq M, N \leq 8$
$0 \leq K \leq M * N$

Output Format

A single line containing the number of ways to divide the chocolate bar.

Sample Input

```
2 2 4
UU
UU
```

Sample Output

```
12
```

Explanation

Note: In the explanation T and D are used to represent, which parts belong to Tom and Derpina respectively. There are $2^4 = 16$ possible separations. The 4 invalid are:

```
TT
TT

DD
DD

DT
TD

TD
DT
```

Some of the valid ones are:

```
TD
TD

TT
DD

DD
TT

DT
DT
```