Codeforces Round #302 (Div. 2)

A. Set of Strings

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a string q. A sequence of k strings $s_1, s_2, ..., s_k$ is called *beautiful*, if the concatenation of these strings is string q (formally, $s_1 + s_2 + ... + s_k = q$) and the first characters of these strings are distinct.

Find any *beautiful* sequence of strings or determine that the *beautiful* sequence doesn't exist.

Input

The first line contains a positive integer k $(1 \leq k \leq 26)$ — the number of strings that should be in a *beautiful* sequence.

The second line contains string q, consisting of lowercase Latin letters. The length of the string is within range from 1 to 100, inclusive.

Output

If such sequence doesn't exist, then print in a single line "NO" (without the quotes). Otherwise, print in the first line "YES" (without the quotes) and in the next k lines print the *beautiful* sequence of strings $s_1, s_2, ..., s_k$.

If there are multiple possible answers, print any of them.

Sample test(s)

input
1 abca
output
YES abca

input
2 aaacas
output
YES aaa cas

input
4 abc
output
NO

Note

In the second sample there are two possible answers: $\{"aaaca", "s"\}$ and $\{"aaa", "cas"\}$.

B. Sea and Islands

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A map of some object is a rectangular field consisting of n rows and n columns. Each cell is initially occupied by the sea but you can cover some some cells of the map with sand so that exactly k islands appear on the map. We will call a set of sand cells to be **island** if it is possible to get from each of them to each of them by moving only through sand cells and by moving from a cell only to a side-adjacent cell. The cells are called to be side-adjacent if they share a vertical or horizontal side. It is easy to see that islands do not share cells (otherwise they together form a bigger island).

Find a way to cover some cells with sand so that exactly k islands appear on the $n \times n$ map, or determine that no such way exists.

Input

The single line contains two positive integers n, k ($1 \le n \le 100$, $0 \le k \le n^2$) — the size of the map and the number of islands you should form.

Output

If the answer doesn't exist, print "NO" (without the quotes) in a single line.

Otherwise, print "YES" in the first line. In the next n lines print the description of the map. Each of the lines of the description must consist only of characters `'S'` and `'L'`, where `'S'` is a cell that is occupied by the sea and `'L'` is the cell covered with sand. The length of each line of the description must equal n.

If there are multiple answers, you may print any of them.

You **should not** maximize the sizes of islands.

Sample test(s)

input
5 2

output
YES SSSSS LLLLL SSSSS LLLLL SSSSS

input
5 25

output
NO

C. Writing Code

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Programmers working on a large project have just received a task to write exactly m lines of code. There are n programmers working on a project, the i-th of them makes exactly a_i bugs in every line of code that he writes.

Let's call a sequence of non-negative integers $v_1, v_2, ..., v_n$ a *plan*, if $v_1 + v_2 + ... + v_n = m$. The programmers follow the plan like that: in the beginning the first programmer writes the first v_1 lines of the given task, then the second programmer writes v_2 more lines of the given task, and so on. In the end, the last programmer writes the remaining lines of the code. Let's call a plan *good*, if all the written lines of the task contain at most b bugs in total.

Your task is to determine how many distinct *good* plans are there. As the number of plans can be large, print the remainder of this number modulo given positive integer mod.

Input

The first line contains four integers n, m, b, mod ($1 \le n, m \le 500, 0 \le b \le 500; 1 \le mod \le 10^9 + 7$) — the number of programmers, the number of lines of code in the task, the maximum total number of bugs respectively and the modulo you should use when printing the answer.

The next line contains n space-separated integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 500$) — the number of bugs per line for each programmer.

Output

Print a single integer — the answer to the problem modulo mod.

Sample test(s)

input
3 3 3 100 1 1 1
output
10

input
3 6 5 1000000007 1 2 3
output
0

input
3 5 6 11 1 2 1
output
0

D. Destroying Roads

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In some country there are exactly n cities and m bidirectional roads connecting the cities. Cities are numbered with integers from 1 to n. If cities a and b are connected by a road, then in an hour you can go along this road either from city a to city b, or from city b to city a. The road network is such that from any city you can get to any other one by moving along the roads.

You want to destroy the largest possible number of roads in the country so that the remaining roads would allow you to get from city s_1 to city t_1 in at most l_1 hours and get from city s_2 to city t_2 in at most l_2 hours.

Determine what maximum number of roads you need to destroy in order to meet the condition of your plan. If it is impossible to reach the desired result, print -1.

Input

The first line contains two integers n, m ($1 \le n \le 3000$, $n - 1 \le m \le \min\{3000, \frac{n(n-1)}{2}\}$) — the number of cities and roads in the country, respectively.

Next m lines contain the descriptions of the roads as pairs of integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$). It is guaranteed that the roads that are given in the description can transport you from any city to any other one. It is guaranteed that each pair of cities has at most one road between them.

The last two lines contains three integers each, s_1, t_1, l_1 and s_2, t_2, l_2, respectively ($1 \le s_i, t_i \le n$, $0 \le l_i \le n$).

Output

Print a single number — the answer to the problem. If the it is impossible to meet the conditions, print -1.

Sample test(s)

input

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

output

```
0
```

input

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

output

```
1
```

input

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

output

```
-1
```

E. Remembering Strings

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have multiset of n strings of the same length, consisting of lowercase English letters. We will say that those strings are easy to remember if for each string there is some position i and some letter c of the English alphabet, such that this string is the only string in the multiset that has letter c in position i.

For example, a multiset of strings {"abc", "aba", "adc", "ada"} are not easy to remember. And multiset {"abc", "ada", "ssa"} is easy to remember because:

- the first string is the only string that has character c in position 3;
- the second string is the only string that has character d in position 2;
- the third string is the only string that has character s in position 2.

You want to change your multiset a little so that it is easy to remember. For a_{ij} coins, you can change character in the j-th position of the i-th string into any other lowercase letter of the English alphabet. Find what is the minimum sum you should pay in order to make the multiset of strings easy to remember.

Input

The first line contains two integers n, m $(1 \le n, m \le 20)$ — the number of strings in the multiset and the length of the strings respectively. Next n lines contain the strings of the multiset, consisting only of lowercase English letters, each string's length is m.

Next n lines contain m integers each, the i-th of them contains integers $a_{i1}, a_{i2}, ..., a_{im}$ $(0 \le a_{ij} \le 10^6)$.

Output

Print a single number — the answer to the problem.

Sample test(s)

input

```
4 5
abcde
abcde
abcde
abcde
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

output

```
3
```

input

```
4 3
abc
aba
adc
ada
10 10 10
10 1 10
10 10 10
10 1 10
```

output

```
2
```

input

```
3 3
abc
ada
ssa
1 1 1
1 1 1
1 1 1
```

output

```
0
```