**A - 頻出文字調査（Finding frequently occurring characters）**

Time limit : 2sec / Stack limit : 256MB / Memory limit : 256MB

**Question**

You will be given a character string *S*, which only consists of half width characters.

Output each character in descending order of the most frequent occurrence, separating each with a newline.  
However, if there multiple alphabets have same number of occurrence, output them in alphabetical order.

**Input**

Input will be given in the following format from Standard Input:

*S*

A character string *S*(1≦|*S*|≦1,000) will be given as one string.

* Define the length of the string as |*S*|
* *S* consists of 26 alphabets from a to z with half width.

**Output**

Output each character in descending order of the most frequent occurrence, separating each with a newline.  
However, if there multiple alphabets have same number of occurrence, output them in alphabetical order.

Also, make sure to insert a line break at the end of the output.

**Input Example # 1**

dcbadcbadcbadde

**Output Example #1**

d

a

b

c

e

* Output d, the most frequently occurring value, which occurs 5 times.
* Then, a,b, and c occur 3 times, therefore output them in alphabetical order; a,b and c.
* Lastly, output e,which occurs 1 time.
* Do not output that is not contained in the character string *S*.

**Input Example #2**

ajklfajdlkfajsdklfjalljaklsdfjaklsdjf

**Output Example #2**

j

l

a

f

k

d

s

**Input Example #3**

z

**Output Example #3**

z

**B - 線形合同法（Linear Congruential Generators）**

Time limit : 2sec / Stack limit : 256MB / Memory limit : 256MB

**Question**

Examine the following random number generator:

*Xn*+1=(*A*×*Xn*+*B*)%*M* *A*, *B*, *X*1, *K* and *M* will be given.

Output the random numbers from **K-th** to **K+4-th**.

**Input**

Input will be given in the following format from Standard Input

*A* *B* *X*1 *K* *M*

Integer *A*(0≦*A*<*M*), Integer *B*(0≦*B*<*M*), Integer *X*1(0≦*X*1<*M*), Integer *K*(1≦*K*<109) and Integer *M*(2≦*M*≦104) will be given in one string.

**Output**

Output the random numbers from *K* to *K*+4.

Also, make sure to insert a line break at the end of the output.

**Input Example # 1**

3 2 5 1 7

**Output Example #1**

5

3

4

0

2

* The initial value of random number *X*1 is 5.
* *X*2=(3×5+2)%7=3
* *X*3=(3×3+2)%7=4
* *X*4=(3×4+2)%7=0
* *X*5=(3×0+2)%7=2

**Input Example #2**

1234 5678 123 12345 9876

**Output Example #2**

4162

6066

5114

5590

414

YieldSquare

**C - ケーブルチェック（Checking Cable）**

Time limit : 2sec / Stack limit : 256MB / Memory limit : 256MB

**Question**

Your task is to connect *N* computers with cables that can communicate in both directions and to check if those computers can actually interact with each other.  
Your work will follow the two instructions shown below:

1. *make* *A* *B* *time*
   * Connect Computer *A* and Computer *B* with a cable.
   * This cable will become unavailable when the value is bigger than *time* seconds. If the value equals to *time* seconds, the cable is still available.
   * Multiple cables may be connected between aforementioned Computer *A* and *B*.
2. *check* *A* *B* *time*
   * Predict if computer *A* and *B* can communicate within *time* seconds with the connected cable(s).
   * Computer *A* and *B* may be connected through other computers.
   * If Computer *A* and *B* can communicate, output YES in one string. If they can’t, output NO in one string.
   * The cables that are connected by all the *make* instructions existing after particular *check* instruction are not available in time series.

The instructions will be processed in the descending order of input.

Determine if Computer *A* and *B* are able to communicate every time there is *check* instruction. If they can communicate, please output YES in one string. If not, output NO in one string.

**Input**

Input will be given in the following format from Standard Input:

*N* *Q*

*S*1 *A*1 *B*1 *time*1

:

*SQ* *AQ* *BQ* *timeQ*

1. On the first string, *N*(2≦*N*≦20), an integer which shows the number of computers, and *Q*(1≦*Q*≦500), another integer which shows the number of the instructions, will be given with a half-width break.
2. From the second string to the *N*−*th* string, you will be given instructions.
   * 1≦*Ak* , *Bk*≦*N* and 1≦*timek*≦104 are guaranteed.
   * There are only two kinds of *Sk*: *make* or *check*.

**Output**

Determine if Computer *A* and *B* are able to communicate every time there is check instruction. If they can communicate, please output YES in one string. If not, output NO in one string.

Also, make sure to insert a line break at the end of the output.

**Input Example # 1**

3 5

make 1 2 1000

check 1 3 500

make 3 2 2000

check 1 3 500

check 1 3 1500

**Output Example #1**

NO

YES

NO

**Input Example #2**

4 11

make 1 2 2000

make 2 3 3000

make 3 4 2500

check 1 4 1500

check 1 3 2000

check 1 3 2500

make 1 4 3000

check 1 3 2500

check 1 3 3000

make 2 4 3000

check 1 3 3000

**Output Example #2**

YES

YES

NO

YES

NO

YES

**D - 適当な連絡網（Appropriate Contact Network）**

Time limit : 2sec / Stack limit : 256MB / Memory limit : 256MB

**Question**

There are *N* students in a classroom. You will be given *N* of relationships with your friends.

Each student has a student number ranging from 1 to *N*, and the student whose student number is *i* is called *i*.

In this classroom, they have a routine they use to pass on information utilizing their contact network. They will pass on the information following the rules shown below:

1. First, Student 1 will receive the information. At this stage, no one except for Student 1 knows about this information.
2. The person who received the information will choose one person who has not received it yet with equal probability then deliver the info to them.
3. This process ends when all of your friends learned about the info.

Output the expected value of the number of people to deliver the information.  
Errors are acceptable when the value of absolute errors or relative errors are no more than 10−6.

**Input**

Input will be given in the following format from Standard Input:

*N*

*c*11 *c*12 *c*13 … *c*1*N*

*c*21 *c*22 *c*23 … *c*2*N*

:

*cN*1 *cN*2 *cN*3 … *cNN*

1. On the first string, integer that indicates the number of people in the classroom, *N*(1≦*N*≦14), will be given.
2. From the second string to the *N* string, friendships will be given.
3. *cij* indicates the following about integers *i* and *j*:
   * If *cij* is Y, the student *i* and the student *j* are friends.
   * If *cij* is *N*, the student *i* and the student *j* are not friends.

**Output**

Output the expected value of the number of people to deliver the information.  
Errors are acceptable when the value of absolute errors or relative errors are no more than 10−6.

Also, make sure to insert a line break at the end of the output.

**Input Example # 1**

in

4

NYYY

YNNN

YNNY

YNYN

**Output Example #1**

2.66666666666666666

* The number of people who received the information When Student 1 delivers the information to Student 2 is 2 people.
* When Student 3 delivers the information to Student 4, the number of people who received the information is 3 people.

**Input Example #2**

5

NNNNN

NNYYY

NYNYY

NYYNY

NYYYN

**Output Example #2**

1

* Since Student 1 has no friend, the information will never reach anyone except for Student 1.

**Input Example #3**

10

NNNNYNYNNY

NNYYNNYNYY

NYNYYYNYYN

NYYNYYYNYY

YNYYNYYYNN

NNYYYNYYYN

YYNYYYNYYN

NNYNYYYNYN

NYYYNYYYNY

YYNYNNNNYN

**Output Example# 3**

9.5018896919296