A - Document Compression

Source file name: `compress.py`
Time limit: x seconds

Alice needs to send text documents to Bob using the Internet. They want to use the communication channel as efficiently as possible, thus they decided to use a document codification scheme based on a set of *basis documents*. The scheme works as follows:

- Each document to be transmitted is represented using a set of terms. This ignores the frequency of each term and its position in the original document, i.e., a document is represented by the list of different terms that it contains.
- There is a set of basis documents that has been previously agreed upon by Alice and Bob. The basis documents are numbered and they are also represented using sets of terms. Both Alice and Bob have a copy of the set, so it does not need to be transmitted.
- Before transmitting a particular document, Alice finds the basis documents that need to be combined to obtain the original content of the document. The documents are combined by uniting their corresponding sets of terms.
- Alice transmits to Bob the list of indexes of the basis documents that represent the original document.
- Bob rebuilds the original document by combining the corresponding basis documents specified by Alice.

For example, suppose that the basis document set contains the following documents, each one specified by the list of different terms that it contains:

Document 1: $\{2, 5, 7, 10\}$

Document 2: $\{8, 9, 10\}$

Document 3: $\{1, 2, 3, 4\}$

Now, suppose that Alice wants to transmit the document $\{1, 2, 3, 4, 5, 7, 10\}$. This document can be obtained by combining the basis documents 1 and 3, so Alice transmits this list of two indices and Bob uses it to rebuild the original document.

Your work is, given a set of basis documents and a document to be transmitted, to find the minimum number of basis documents needed to represent it, if it can be represented; otherwise, you have to indicate that it is impossible to attain a representation.

Input

Each test case is described as follows:

- A line with two integer numbers M and N ($1 \leq M \leq 100$, $1 \leq N \leq 100$), separated by blanks. M corresponds to the number of basis documents and N corresponds to the number of documents to codify.
- Each one of the following $M + N$ lines contains the numbers $k, t_1, t_2, \ldots, t_k$, separated by blanks ($1 \leq k \leq 16$, $1 \leq t_i \leq 16$ for each $1 \leq i \leq k$). The first value indicates the number of different terms in the document and the following numbers correspond to the different terms in the document. The first M lines describe the basis documents and the next N lines describe the documents to codify.

The last test case is followed by a line containing two zeros.

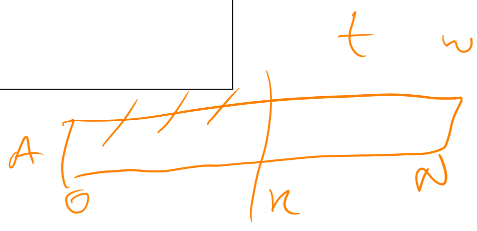
The input must be read from standard input.

Output

For each test case you must print a line containing the integers $r_1, r_2, \ldots, r_N$ (with a blank between each two numbers), where r_i is the minimum number of basis documents required to represent the i-th document of the input ($1 \leq i \leq N$). If it is impossible to represent the i-th document, then r_i must be the number 0.

The output must be written to standard output.

Sample Input	**Sample Output**
3 1	2
4 2 5 7 10	1 0 2
3 8 9 10	
4 1 2 3 4	
7 1 2 3 4 5 7 10	
2 3	
2 2 1	
2 4 3	
2 3 4	
3 3 1 2	
4 3 1 4 2	
0 0	



Entrada: $A[0..N)$, $0 \leq T \leq N$, $0 \leq w \leq \overline{W}$

Para $0 \leq n \leq N$, $0 \leq t \leq T$)

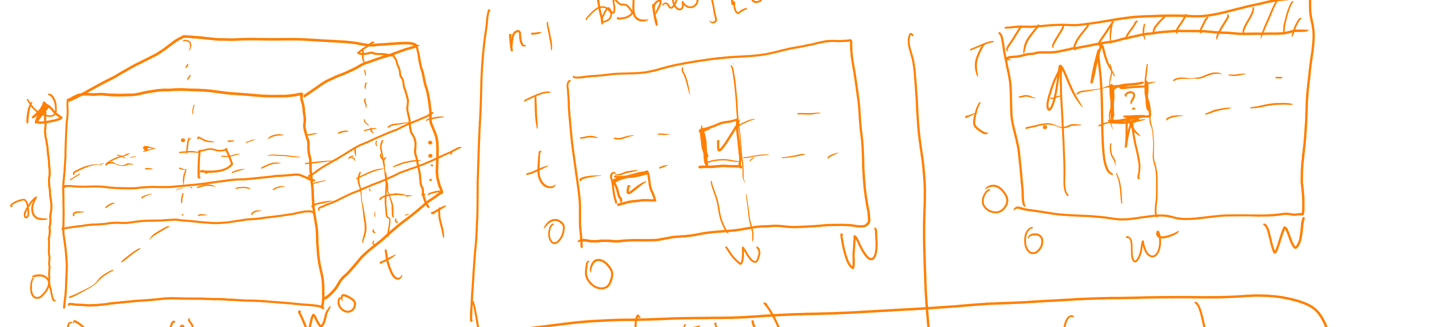
$$\phi(n,t,w) = \begin{cases} \bot & , n=0 \\ & , n<t \\ w=0 & , n \geq t \wedge t=0 \\ \phi(n-1,t,w) \vee \phi(n-1,t-1,w-A[n-1]) & , n \geq t \wedge t \neq 0 \wedge n \neq 0 \end{cases}$$

$100 \cdot 50 \cdot 2500 \approx 12.5 \times 10^6$

$100 \cdot 2500 = 250000$

$\phi(N,T,-)$

$n-1$ tab[prev][0..w][0..T] , n tab[act][0..w][0..T]



Espacio: $\boxed{\Theta(N \cdot T \cdot W) \quad \Theta(T \cdot W) \quad \Theta(T \cdot W)}$

$S = \Sigma A$.

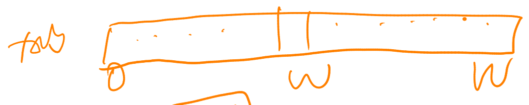
$A = [\,-, -, \ldots, -\,]$

minA : mínimo en A

≥ 0 $A' = [a_0 - \min A, a_1 - \min A, \ldots, a_{N-1} - \min A]$

tab[w] = T

N, T



✓ $w' \cdot (S - w')$

$w \mapsto \boxed{\underbrace{w + T \cdot \min A}_{w'}}$

$\phi(n,t,w)$:

$A[0..n$

B - Expression Again

Source file name: `expression.py`
Time limit: 3 seconds

You are given an algebraic expression of the form $(x_1 + x_2 + x_3 + \ldots + x_n) * (y_1 + y_2 + \ldots + y_m)$ and $(n + m)$ integers. You have to find the maximum and minimum value of the expression using the given integers. For example if you are given $(x_1 + x_2) \times (y_1 + y_2)$ and you are given 1, 2, 3 and 44. Then maximum value is $(1 + 4) \times (2 + 3) = 25$ where as minimum value is $(4 + 3) \times (2 + 1) = 21$.

Input

Each input set starts with two positive integers N, M $\boxed{(1 \le N, M \le 50)}$. Next line follows $(N + M)$ integers which are in the range of -50 to 50. Input is terminated by end of file.

$\overset{N}{4}$ $\overset{T}{2}$ $\overset{A}{[1,2,3,4]}$

$(X, A \setminus X)$

$\left(\sum_{a \in X} a\right)\left(\sum_{a \in A \setminus X} a\right)$ máximo

$|X| = 2$

The input must be read from standard input.

Output

Output is one line for each case, maximum value followed by minimum value.

$\boxed{\begin{matrix} X = A \\ S = \sum_{a \in A} a \end{matrix}}$

$$X' \xrightarrow{a} A \setminus X'$$

$-a$ $\quad$ $+a$

The output must be written to standard output.

Sample Input	Sample Output

2 2
1 2 3 4
3 1
1 2 3 4
2 2
2 2 2 2
	25 21
	24 9
	16 16

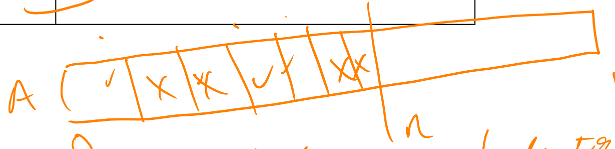
$t \le n$

T

~~A~~ t

?

S'



$\phi(n, \underline{\underline{t}}, s)$: " Máximo producto que resulte de sumar t elementos en $A[0..n)$

$\overset{0}{}$

$t = 0, 1$ $\quad$ $\boxed{s \cdot (S - s)}$

$A = \{a_0, \ldots, a_{N-1}\}$ $\qquad$ $a_i \ge 0$ $\qquad$ $SA = \sum_{i=0}^{N-1} a_i$

¿Qué números puede obtener sumando los elementos de A?

$\boxed{\phi(n, t, w)}$: "En $A[0..n)$ hay una subsecuencia de tamaño t cuya suma es w" $(T/\bot)$

-5000

$100(-50)$

$100(50)$

T $\quad$ $\overset{0}{\vdots}$ $\quad$ N

.10 $\quad$ $\underline{45}$

5001

$\overline{W} = [T|T|\bot|\bot| \ldots |T| \ldots |\bot]$

~~-2500~~ $\quad$ 0 $\quad$ $(10)(45-10)$ $\quad$ N

$\phi(N, T, w)$

C - Fewest Flops

Source file name: `flops.py`
Time limit: x seconds

A common way to uniquely encode a string is by replacing its consecutive repeating characters (or "chunks") by the number of times the character occurs followed by the character itself. For example, the string "aabbbaabaaaa" may be encoded as "2a3b2a1b4a". (Note for this problem even a single character "b" is replaced by "1b").

Suppose we have a string S and a number k such that k divides the length of S. Let S_1 be the substring of S from 1 to k, S_2 be the substring of S from $k+1$ to $2k$, and so on. We wish to rearrange the characters of each block S_i independently so that the concatenation of those permutations S' has as few chunks of the same character as possible. Output the fewest number of chunks.

For example, let S be "uuvuwwuv" and k be 4. Then S_1 is "uuvu" and has three chunks, but may be rearranged to "uuuv" which has two chunks. Similarly, S_2 may be rearranged to "vuww". Then S', or S_1S_2, is "uuuvvuww" which is 4 chunks, indeed the minimum number of chunks.

Input

The input begins with a line containing t ($1 \leq t \leq 100$), the number of test cases. The following t lines contain an integer k and a string S made of no more than 1000 lowercase English alphabet letters. It is guaranteed that k will divide the length of S.

The input must be read from standard input.

Output

For each test case, output a single line containing the minimum number of chunks after we rearrange S as described above.

The output must be written to standard output.

Sample Input	**Sample Output**

`2`
`5 helloworld`
`7 thefewestflops`
	`8`
	`10`

D - Hyperactive Girl

Source file name: `girl.py`
Time limit: x seconds

Helen is a hyperactive girl. She wants to schedule her activities so that at any moment of the day there is at least one thing she can do. She does not care if her activities overlap in time, as long as every moment of her day has an activity scheduled.

Helen divided the day in a particular way. The day starts at time 0 and finishes at time M. Each moment of the day is represented by a real number between 0 and M, inclusive. Helen made a list of all possible activities, with their start and finish times. Now she must decide which subset of activities to schedule.

If an activity starts at time S and finishes at time F, then we say that it covers all moments between S and F, inclusive. Helen does not want to waste any activities, so she will only choose minimal subsets of activities that cover the day to be scheduled. A subset of activities is a minimal subset that covers the day if and only if:

1. every moment of the day is covered by at least one activity of the subset; and
2. removing any of the activities from the subset would leave at least one moment of the day uncovered.

Note that some moments of the day may be covered by more than one activity. Given the list of possible activities for one day, you must help Helen by determining how many distinct minimal subsets cover the day.

Input

Each test case is given using several lines. The first line contains two integers M and N, representing respectively the highest value for a moment in the day ($1 \leq M \leq 10^9$) and the number of possible activities for the day ($1 \leq N \leq 100$). Each of the next N lines describes one possible activity and contains two integers S and F, representing respectively the start and finish times of the activity ($0 \leq S \leq F \leq M$).

The last test case is followed by a line containing two zeros.

The input must be read from standard input.

Output

For each test case output a single line with a single integer representing the number of minimal subsets that cover the day. To make your life easier, print the remainder of dividing the solution by 10^8.

The output must be written to standard output.

Sample Input	Sample Output

```
8 7
0 3
2 5
5 8
1 3
3 6
4 6
0 2
1 1
0 1
2 1
0 1
0 0
```
	```
	4
	1
	0
	```

E - Weights and Measures

Source file name: `weights.py`
Time limit: 4 seconds

I know, up on top you are seeing great sights,
But down at the bottom, we, too, should have rights.
We turtles can't stand it. Our shells will all crack!
Besides, we need food. We are starving!" groaned Mack.

Mack, in an effort to avoid being cracked, has enlisted your advice as to the order in which turtles should be dispatched to form Yertle's throne. Each of the five thousand, six hundred and seven turtles ordered by Yertle has a different weight and strength. Your task is to build the largest stack of turtles possible.

Input

The input consists of several lines, each containing a pair of integers separated by one or more space characters, specifying the weight and strength of a turtle. The weight of the turtle is in grams. The strength, also in grams, is the turtle's overall carrying capacity, including its own weight. That is, a turtle weighing 300g with a strength of 1000g could carry 700g of turtles on its back. There are at most 5607 turtles.

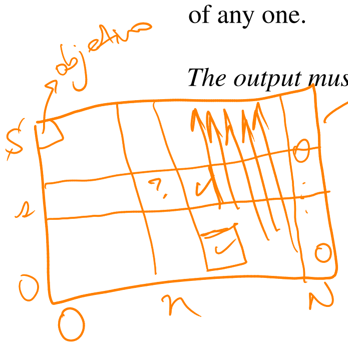
The input must be read from standard input.

Output

Your output is a single integer indicating the maximum number of turtles that can be stacked without exceeding the strength of any one.

$[(1000,1200),(300,1000),(200,600),(100,101)]$.

The output must be written to standard output.



Sample Input	Sample Output

w s
`300 1000`
`1000 1200`
`200 600`
`100 101`
	`3`

S, 1, 0 presente n, pasando n-1

$400 \leq 500 \downarrow 400$

$(700-200) \downarrow (600-200)$ 700

s $A[n].w$ $A[n].S$ $A[n.w]$

$(100,101)$

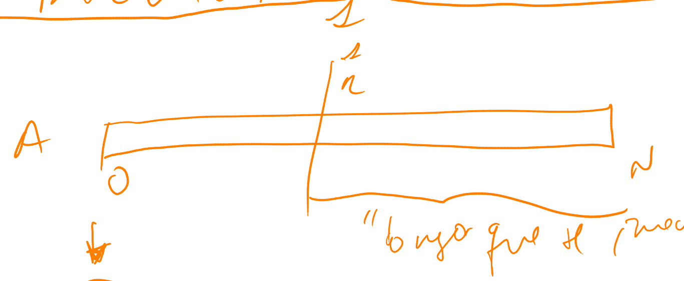
$(200,600)$

$(300,1000)$

Para $0 \leq n < N$ y $0 \leq s \leq S$

$\phi(n,s)$: "máxima cantidad de tortugas en $A[n..N)$ que se pueden apilar con capacidad s".

$(\forall n \mid 0 \leq n < N : A[n].S \geq A[n].w)$



Objetivo: $\phi(0, \boxed{+\infty})$ S

$$\phi(n,s) = \begin{cases} 0 & , n = N \\ \phi(n+1, s) & , n \neq N \wedge A[n].w > s \\ \phi(n+1, s) \uparrow (1+\phi(n+1, (s - A[n].w) \downarrow (A[n].s - A[n].w))) & , n \neq N \wedge A[n].w \leq s \end{cases}$$