Problem A
Cache Control

Practice Contest II, Tokyo
22 Feb 2009

Mr. Haskins is working on tuning a database system. The database is a simple associative storage that contains key-value pairs. In this database, a key is a distinct identification (ID) number and a value is an object of any type.

In order to boost the performance, the database system has a cache mechanism. The cache can be accessed much faster than the normal storage, but the number of items it can hold at a time is limited. To implement caching, he selected least recently used (LRU) algorithm: when the cache is full and a new item (not in the cache) is being accessed, the cache discards the least recently accessed entry and adds the new item.

You are an assistant of Mr. Haskins. He regards you as a trusted programmer, so he gave you a task. He wants you to investigate the cache entries after a specific sequence of accesses.

► Input

The first line of the input contains two integers N and M. N is the number of accessed IDs, and M is the size of the cache. These values satisfy the following condition: $1 \leq N, M \leq 100000$.

The following N lines, each containing one ID, represent the sequence of the queries. An ID is a positive integer less than or equal to 10^9.

► Output

Print IDs remaining in the cache after executing all queries. Each line should contain exactly one ID. These IDs should appear in the order of their last access time, from the latest to the earliest.

► Sample Input and Output

Input #1:
```
3 2
1
2
3
```

Output #1:
```
3
2
```

Input #2:
```
5 3
1
2
3
4
1
```

Output #2:
```
1
4
3
```

Problem B
Cover Time

Practice Contest II, Tokyo
22 Feb 2009

Let G be a connected undirected graph where N vertices of G are labeled by numbers from 1 to N. G is simple, i.e. G has no self loops or parallel edges.

Let P be a particle walking on vertices of G. At the beginning, P is on the vertex 1. In each step, P moves to one of the adjacent vertices. When there are multiple adjacent vertices, each is selected in the same probability.

The *cover time* is the expected number of steps necessary for P to visit all the vertices.

Your task is to calculate the cover time for each given graph G.

► Input

The input has the following format.

$$\begin{array}{l} N\ M \\ a_1\ b_1 \\ \vdots \\ a_M\ b_M \end{array}$$

N is the number of vertices and M is the number of edges. You can assume that $2 \le N \le 10$. a_i and b_i ($1 \le i \le M$) are positive integers less than or equal to N, which represent the two vertices connected by the i-th edge. You can assume that the input satisfies the constraints written in the problem description, that is, the given graph G is connected and simple.

► Output

There should be one line containing the cover time in the output.

The answer should be printed with six digits after the decimal point, and should not have an error greater than 10^{-6}.

► Sample Input and Output

Input #1:
```
3 2
1 2
2 3
```

Output #1:
```
4.000000
```

Input #2:
```
4 6
1 2
1 3
1 4
2 3
2 4
3 4
```

Output #2:
```
5.500000
```

Problem C
Craftsman

Practice Contest II, Tokyo
22 Feb 2009

Takeshi, a famous craftsman, accepts many offers from all over Japan. However, the tools which he is using now has become already too old. So he is planning to buy new tools and to replace the old ones before next use of the tools. Some offers may incur him monetary cost, if the offer requires the tools to be replaced. Thus, it is not necessarily best to accept all the orders he has received. Now, you are one of his disciples. Your task is to calculate the set of orders to be accepted, that maximizes his earning for a given list of orders and prices of tools. His earning may shift up and down due to sale income and replacement cost.

He always purchases tools from his friend's shop. The shop discounts prices for some pairs of items when the pair is purchased at the same time. You have to take the discount into account. The total price to pay may be not equal to the simple sum of individual prices.

You may assume that all the tools at the shop are tough enough. Takeshi can complete all orders with replaced tools at this time. Thus you have to buy at most one tool for each kind of tool.

▶ Input

The input conforms to the following format:

N M P
X_1 K_1 $I_{1,1}$ ... I_{1,K_1}
...
X_N K_N $I_{N,1}$ ... I_{N,K_N}
Y_1
...
Y_M
$J_{1,1}$ $J_{1,2}$ D_1
...
$J_{P,1}$ $J_{P,2}$ D_P

where N, M, P are the numbers of orders, tools sold in the shop and pairs of discountable items, respectively.

The following N lines specify the details of orders. X_i is an integer indicating the compensation for the i-th order, and K_i is the number of tools required to complete the order. The remaining part of each line describes the tools required for completing the order. Tools are specified by integers from 1 through M.

The next M lines are the price list at the shop of Takeshi's friend. An integer Y_i represents the price of the i-th tool.

The last P lines of each test case represent the pairs of items to be discounted. When Takeshi buys the $J_{i,1}$-th and the $J_{i,2}$-th tool at the same time, he has to pay only D_i yen, instead of the sum of their individual prices. It is guaranteed that no tool appears more than once in the discount list, and that $\max\{Y_i, Y_j\} < D_{i,j} < Y_i + Y_j$ for every discount prices, where $D_{i,j}$ is the discount price of i-th and j-th tools bought at the same time.

Also it is guaranteed that $1 \le N \le 100$, $2 \le M \le 100$, $1 \le K_i \le 100$, $1 \le P \le M/2$ and $1 \le X_i, Y_i \le 1000$.

► Output

Output the maximum possible earning of Takeshi to the standard output.

► Sample Input and Output

Input #1:

```
3 4 2
100 2 1 2
100 1 3
100 1 4
20
20
50
150
1 2 30
3 4 180
```

Output #1:

```
120
```

Input #2:

```
1 2 1
100 1 2
20
40
1 2 51
```

Output #2:

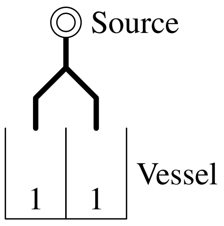
```
60
```

Problem D
Divide the Water

Practice Contest II, Tokyo
22 Feb 2009

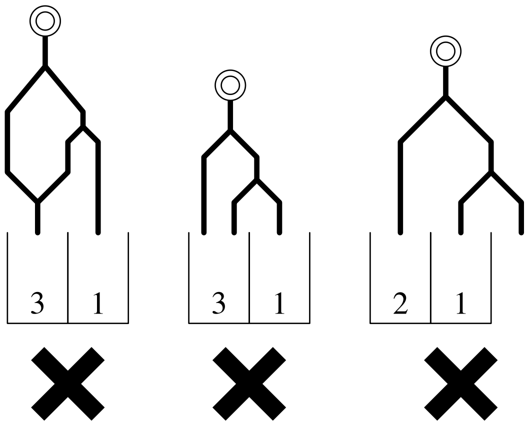
A scientist Arthur C. McDonell is conducting a very complex chemical experiment. This experiment requires a large number of very simple operations to pour water into every column of the vessel at the predetermined ratio. Tired of boring manual work, he was trying to automate the operation.

One day, he came upon the idea to use three-pronged tubes to divide the water flow. For example, when you want to pour water into two columns at the ratio of 1 : 1, you can use one three-pronged tube to split one source into two.

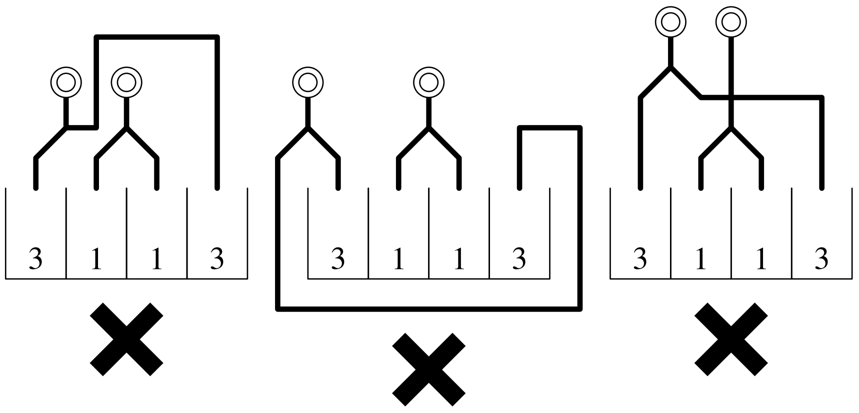


He wanted to apply this idea to split water from only one faucet to the experiment vessel at an arbitrary ratio, but it has gradually turned out to be impossible to set the configuration in general, due to the limitations coming from the following conditions:

1. The vessel he is using has several columns aligned horizontally, and each column has its specific capacity. He cannot rearrange the order of the columns.
2. There are enough number of glass tubes, rubber tubes and three-pronged tubes. A three-pronged tube always divides the water flow at the ratio of 1 : 1.
3. Also there are enough number of fater faucets in his laboratory, but they are in the same height.
4. A three-pronged tube cannot be used to combine two water flows. Moreover, each flow of water going out of the tubes must be poured into exactly one of the columns; he cannot discard water to the sewage, nor pour water into one column from two or more tubes.



5. The water flows only downward. So he cannot place the tubes over the faucets, nor under the exit of another tubes. Moreover, the tubes cannot be crossed.



Still, Arthur did not want to give up. Although one-to-many division at an arbitrary ratio is impossible, he decided to minimize the number of faucets used. He asked you for a help to write a program to calculate the minimum number of faucets required to pour water into the vessel, for the number of columns and the ratio at which each column is going to be filled.

► Input

The input consists of an integer sequence.

The first integer indicates N, the number of columns in the vessel. Then the following N integers describe the capacity by which each column should be filled. The i-th integer in this part specifies the ratio at which he is going to pour water into the i-th column.

You may assume that $N \leq 100$, and for all i, $v_i \leq 1000000$.

► Output

Output the number of the minimum faucet required to complete the operation.

► Sample Input and Output

Input #1:
```
2 1 1
```
Output #1:
```
1
```

Input #2:
```
2 3 1
```
Output #2:
```
2
```

Input #3:
```
2 2 1
```
Output #3:
```
2
```

Input #4:
```
4 3 1 1 3
```
Output #4:
```
3
```

Input #5:
```
3 1 2 1
```
Output #5:
```
3
```

Input #6:
```
5 1 2 3 4 5
```
Output #6:
```
5
```

Input #7:
```
7 8 8 1 1 2 4 8
```
Output #7:
```
1
```

Problem E
Mickle's Beam

Practice Contest II, Tokyo
22 Feb 2009

Major Mickle is commanded to protect a secret base which is being attacked by N tanks. Mickle has an ultimate laser rifle called *Mickle's beam*. Any object shot by this beam will be immediately destroyed, and there is no way for protection. Furthermore, the beam keeps going through even after destroying objects. The beam is such a powerful and horrible weapon, but there is one drawback: the beam consumes a huge amount of energy. For this reason, the number of shots should be always minimized.

Your job is to write a program that calculates the minimum number of shots required to destroy all the enemy tanks.

You can assume the following:

- the base is located on the origin $(0, 0)$;
- each tank has a rectangular shape with edges parallel to the x- and y-axes;
- no tank may touch or include the origin;
- no two tanks share the same point;
- the width of beam is negligible; and
- a beam destroys every tank it touches or crosses.

► Input

The input consists of an integer sequence.

The first integer indicates N ($N \leq 2,000$). Each of the following N lines contains four integers which indicates the x- and y-coordinates of the lower-left corner and the upper-right corner of a tank respectively. You can assume that every coordinate may not exceed 10,000 in absolute value.

► Output

Output the minimum number of shots of the beam.

► Sample Input and Output

Input #1:
```
4
2 -1 3 1
-1 2 1 3
-3 -1 -2 1
-1 -3 1 -2
```

Output #1:
```
4
```

Input #2:
```
2
1 0 2 1
3 -1 4 0
```

Output #2:
```
1
```

Problem F
Poor Computer

Practice Contest II, Tokyo
22 Feb 2009

Brian Fulk is struggling with a really poor computer for a couple of days. Now he is trying to write a very simple program which receives a positive integer x and returns some of its multiples, say, $a_1x, \dots, a_Nx$. However, even such a simple task is not easy on this computer since it has only three arithmetic operations: addition, subtraction and left shift.

Let us describe the situation in more details. Initially the computer stores only a given positive integer x. The program Brian is writing will produce $a_1x, \dots, a_Nx$, where $a_1, \dots, a_N$ are given multipliers, using only the following operations:

- addition of two values,
- subtraction of two values, and
- bitwise left shift (left shift by n bits is equivalent to multiplication by 2^n).

The program should not generate any value greater than $42x$; under this constraint he can assume that no overflow occurs. Also, since this computer cannot represent negative values, there should not be subtraction of a greater value from a smaller value.

Some of you may wonder where the number 42 comes from. There is a deep reason related to the answer to life, the universe and everything, but we don't have enough space and time to describe it.

Your task is to write a program that finds the shortest sequence of operations to produce the multiples $a_1x, \dots, a_Nx$ and reports the length of the sequence. These numbers may be produced in any order.

Here we give an example sequence for the first sample input, in a C++/Java-like language:

```
a = x << 1;     // 2x
b = x + a;      // 3x
c = a + b;      // 5x
d = c << 2;     // 20x
e = d - b;      // 18x
```

► Input

The first line specifies N, the number of multipliers. The second line contains N integers $a_1, \dots, a_N$, each of which represents a multiplier of x.

You can assume that $N \leq 41$ and $2 \leq a_i \leq 42$ ($1 \leq i \leq N$). Furthermore, $a_1, \dots, a_N$ are all distinct.

► Output

Output in a line the minimum number of operations you need to produce the values $a_1x, \dots, a_Nx$.

► Sample Input and Output

Input #1:

```
3
3 5 18
```

Output #1:

```
5
```

Input #2:

```
1
29
```

Output #2:

```
4
```

Input #3:

```
4
12 19 41 42
```

Output #3:

```
8
```

Problem G
Riffle Swap

Practice Contest II, Tokyo
22 Feb 2009

You have a deck of 2^N cards ($1 \leq N \leq 1000000$) and want to have them shuffled.

The most popular shuffling technique is probably the riffle shuffle, in which you split a deck into two halves, place them in your left and right hands, and then interleave the cards alternatively from those hands. The shuffle is called perfect when the deck is divided exactly in half and the cards are interleaved one-by-one from the bottom half. For example, the perfect riffle shuffle of a deck of eight cards $\langle 0, 1, 2, 3, 4, 5, 6, 7 \rangle$ will result in a deck $\langle 0, 4, 1, 5, 2, 6, 3, 7 \rangle$.

Since you are not so good at shuffling that you can perform the perfect riffle shuffle, you have decided to simulate the shuffle by swapping two cards as many times as needed. How many times you will have to perform swapping at least? As the resultant number will obviously become quite huge, your program should report the number modulo $M = 1000003$.

► Input

The input just contains a single integer N.

► Output

Print the number of swaps in a line. No extra space or empty line should occur.

► Sample Input and Output

Input #1:
```
1
```
Output #1:
```
0
```

Input #2:
```
2
```
Output #2:
```
1
```

Input #3:
```
3
```
Output #3:
```
4
```

Input #4:
```
4
```
Output #4:
```
10
```

Input #5:
```
10
```
Output #5:
```
916
```

Problem H
Round Table

Practice Contest II, Tokyo
22 Feb 2009

You are the owner of a restaurant, and you are serving for N customers seating in a round table.

You will distribute M menus to them. Each customer receiving a menu will make the order of plates, and then pass the menu to the customer on the right unless he or she has not make the order. The customer i takes L_i unit time for the ordering.

Your job is to write a program to calculate the minimum time until all customers to call their orders, so you can improve your business performance.

► Input

The input consists of a sequence of positive integers.

The first line of the input contains two positive integers N ($N \leq 50,000$) and M ($M \leq N$). The second line contains N positive integers L_1, L_2, …, L_N ($L_i \leq 600$).

► Output

Output the minimum possible time required for them to finish ordering.

► Sample Input and Output

Input #1:
```
3 2
1 5 10
```

Output #1:
```
10
```

Input #2:
```
4 2
1 2 3 4
```

Output #2:
```
5
```

Problem I
Strange Currency System

Practice Contest II, Tokyo
22 Feb 2009

The currency system in the Kingdom of Yoax-Musty is strange and fairly inefficient. Like other countries, the kingdom has its own currencty unit denoted by K$ (kingdom dollar). However, the Ministry of Finance issues bills for every value between 1 K$ and ($2^{31} - 1$) K$ worth.

On the other hand, this system often enables people to make many different values just with a small number of bills. For example, if you have four bills of 1 K$, 2 K$, 4 K$, and 8 K$ worth respectively, you can make any values from 1 K$ to 15 K$.

In this problem, you are requested to write a program that finds the minimum value that cannot be made with a given set (multiset in a mathematical sense) of bills. For the case with the four bills (1 K$, 2 K$, 4 K$, and 8 K$), since you can make any values up to 15 K$, your program should report 16 K$.

► Input

The input consists of two lines. The first line contains an integer N ($1 \leq N \leq 10000$), the number of bills. The second line contains N integers, each of which represents the value of a bill in K$. There may be multiple bills of the same value.

► Output

Print the minimum value unable to be made on a line. The value should be given in K$ and without any currency sign or name.

► Sample Input and Output

Input #1:
```
4
1 2 4 8
```

Output #1:
```
16
```

Input #2:
```
5
1 1 3 11 2
```

Output #2:
```
8
```

Problem J
Tangram

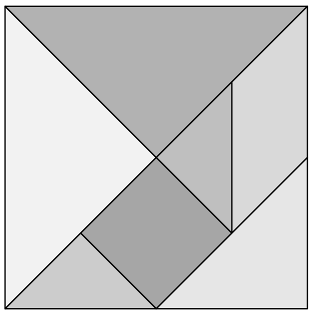
Practice Contest II, Tokyo
22 Feb 2009

Chen, your pretty sister, got absorbed in a tiling puzzle called the tangram. Suddenly, she rushed to you and asked for your help, saying ”Please, please help me solve this problem! I’ve been stuck on this for an hour, but I can’t figure out how to solve it!” Your pretty, cute, lovely sister is asking for your help, so you have to resolve the problem for her immediately, but... can you really do that?

The tangram is a puzzle considered to be rooted in China. It involves seven pieces, each of which has its specific shape. The objective of this puzzle is to form a given shape using all seven pieces. Each piece can be placed anywhere, and rotated and flipped in any way, but cannot overlap with another one.

The seven pieces consist of the following (see figure):

- Two small right triangles (the lengths of the edges are $25\sqrt{2}$, $25\sqrt{2}$ and 50),
- One medium right triangle (the lengths of the edges are 50, 50 and $50\sqrt{2}$),
- Two big right triangles (the lengths of the edges are $50\sqrt{2}$, $50\sqrt{2}$ and 100),
- One square (the length of the side is $25\sqrt{2}$), and
- One parallelogram (the lengths of the sides are $25\sqrt{2}$ and 50).



Given the shape of a polygon P, your program should determine whether the shape can be made by the seven pieces. To make the problem simpler, you may assume the following:

- Each edge of P is parallel to one of the x-axis, the y-axis, $y = x$ or $y = -x$.
- P is connected, and has no hole in it. Furthermore, P is a simple polygon, i.e. it has neither self-crossing edges nor self-touching edges.

► Input

The input begins with a line that contains a single integer N ($3 \le N \le 30$), which indicates the number of the vertices in P. The next N lines denote the coordinates of the vertices of P in clockwise order. Each line contains exactly four non-negative integers a, b, c and d, separated by a space. It represents that the corresponding vertex is at $(a + b\sqrt{2},\ c + d\sqrt{2})$. No coordinate exceeds 1000.

► Output

Print “Yes” in a line if the given shape is solvable, or “No” otherwise.

► Sample Input and Output

Input #1:

```
4
0 0 0 0
0 0 100 0
100 0 100 0
100 0 0 0
```

Output #1:

```
Yes
```

Input #2:

```
4
0 50 0 0
0 0 0 50
0 50 0 100
0 100 0 50
```

Output #2:

```
Yes
```

Problem K
Trading Ship

Practice Contest II, Tokyo
22 Feb 2009

You are on board a trading ship as a crew.

The ship is now going to pass through a strait notorious for many pirates often robbing ships. The Maritime Police has attempted to expel those pirates many times, but failed their attempts as the pirates are fairly strong. For this reason, every ship passing through the strait needs to defend themselves from the pirates.

The navigator has obtained a sea map on which the location of every hideout of pirates is shown. The strait is considered to be a rectangle of $W \times H$ on an xy-plane, where the two opposite corners have the coordinates of $(0, 0)$ and (W, H). The ship is going to enter and exit the strait at arbitrary points on $y = 0$ and $y = H$ respectively.

To minimize the risk of attack, the navigator has decided to take a route as distant from the hideouts as possible. As a talented programmer, you are asked by the navigator to write a program that finds the best route, that is, the route with the maximum possible distance to the closest hideouts. For simplicity, your program just needs to report the distance in this problem.

► Input

The input begins with a line containing three integers W, H, and N. Here, N indicates the number of hideouts on the strait. Then N lines follow, each of which contains two integers x_i and y_i, which denote the coordinates the i-th hideout is located on.

The input satisfies the following conditions: $1 \le W, H \le 10^9$, $1 \le N \le 500$, $0 \le x_i \le W$, $0 \le y_i \le H$.

► Output

There should be a line containing the distance from the best route to the closest hideout(s). The distance should be printed with three fractional digits and should not contain an absolute error greater than 10^{-3}.

► Sample Input and Output

Input #1:
```
10 10 1
3 5
```

Output #1:
```
7.000
```

Input #2:
```
10 10 2
2 2
8 8
```

Output #2:
```
4.243
```

Input #3:
```
10 10 3
0 1
4 4
8 1
```

Output #3:
```
2.500
```