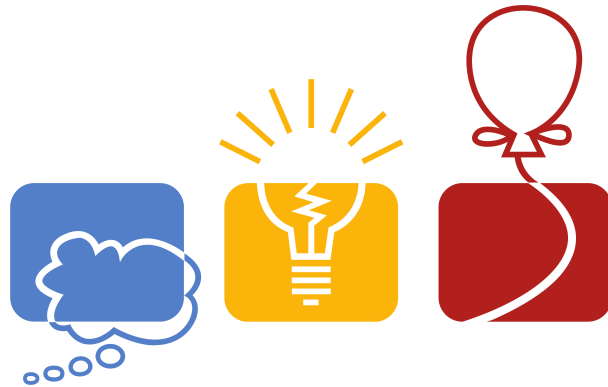


# United Kingdom and Ireland Programming Contest 2025



## Problems

- A** Arboreal Challenge
- B** Brickwork
- C** Colourful Captcha
- D** Depot
- E** Enclosure
- F** Fell Walking
- G** Get Good
- H** Hybrid Search
- I** Itsy Bits
- J** Joust Sort
- K** Klaus
- L** Last Orders
- M** Motorway Stops

Problems are not ordered by difficulty.  
Do not open before the contest has started.

This page is intentionally left (almost) blank.

# Problem A

## Arboreal Challenge

Our local council discovered that its forest is overrun by invasive pests, and is considering its options for how to restore natural balance... By fire if necessary.

Experts insist that the correct plan of action is to set the forest alight, supposed ‘prescribed burning’, but you’re not quite convinced that the plan is sound without a clear end. You should calculate the exact time the burn will finish and where, so that we can verify the blaze is out as soon as possible.

Given that the flames will be lit simultaneously along every edge of the forest and travel at a rate of  $v \text{ ms}^{-1}$  through its interior, how long will it take for the entire forest to be set alight?

The forest has grown in the convenient shape of a simple polygon, with no holes, although it may or may not be convex and may have any winding order.

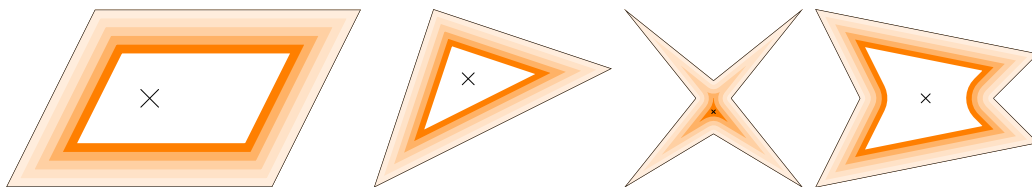


Figure A.1: Diagrams illustrating sample fires 1, 2, 3, and 4 respectively.

### Input

The first line of input contains two integers  $v$ , and  $n$ , where  $v$  ( $1 \leq v \leq 10^5$ ) is the speed in  $\text{ms}^{-1}$  of the fire, and  $n$  ( $3 \leq n \leq 500$ ) is the number of vertices in the polygon describing the forest.

The next  $n$  lines each contain two integers,  $x_i$  and  $y_i$  ( $0 \leq x_i, y_i \leq 10^5$ ), the coordinates of the  $i^{\text{th}}$  vertex.

### Output

Output the time taken for the entire forest to be set on fire.

On the next line, output the coordinates of the last point inside the forest to be set alight, or any one of them if there are multiple.

Your answers should have an absolute or relative error of at most  $10^{-6}$ .

#### Sample Input 1

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

#### Sample Output 1

```
0.5
1.61803399 1
```

**Sample Input 2**

1 3  
0 0  
1 3  
4 2

**Sample Output 2**

0.926209683  
1.58578644 1.82842712

**Sample Input 3**

1 8  
9 10  
5 15  
10 11  
15 15  
11 10  
15 5  
10 8  
5 5

**Sample Output 3**

1.25  
10 9.25

**Sample Input 4**

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

**Sample Output 4**

1.47564675  
2.47564721 2

# Problem B

## Brickwork

We are attempting to design a wall as a tessellation of 2D rectangular brick shapes, with all of the bricks having exact integer coordinates and exact integer sizes.

Given a description of the brickwork in this 2D wall, determine if the wall is indeed a wall—covering a perfect rectangle with no overlapping bricks and with no empty spaces in between bricks. This means that every non-integer-coordinate inside the region should be covered by exactly one brick.

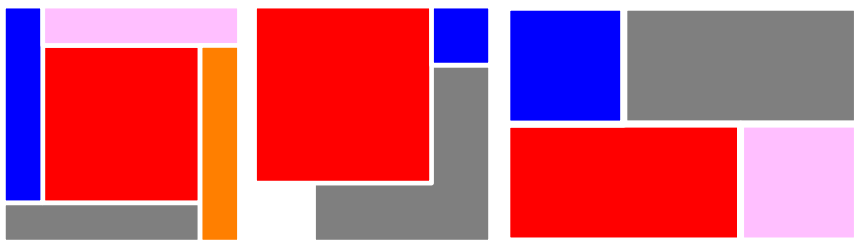


Figure B.1: Brickwork illustrating samples 1, 2, and 3 respectively.

### Input

- One line containing the number of bricks,  $n$  ( $1 \leq n \leq 1,000,000$ ).
- $n$  further lines, each containing the description of a brick as four integers  $xywh$  giving the  $xy$ -coordinates, width, and height ( $0 \leq x, y \leq 10^8$ ;  $1 \leq w, h \leq 10^8$ ).

### Output

Output yes if the bricks constitute a valid wall, no otherwise.

| Sample Input 1                                           | Sample Output 1 |
|----------------------------------------------------------|-----------------|
| 5<br>0 0 5 1<br>1 1 4 4<br>0 1 1 5<br>1 5 5 1<br>5 0 1 5 | yes             |
| Sample Input 2                                           | Sample Output 2 |
| 3<br>1 0 3 3<br>0 1 3 3<br>3 3 1 1                       | no              |

**Sample Input 3**

```
4
1 1 2 1
0 0 2 1
0 1 1 1
2 0 1 1
```

**Sample Output 3**

```
yes
```

# Problem C

## Colourful Captcha

It is 2025, and everyone knows about large language models. The world is divided: while some companies want to feed all texts in the world to their next-generation models, some other companies want to keep their texts to themselves and to their human clients.

A new company called *CloseAI* started to develop a captcha designed to protect human-oriented resources from large language models. To do this, they employ the fact that the latter tend to happily digest and interpret any text, while humans are mostly capable of interpreting the primary message and ignoring the non-important words. The captcha is hence a piece of ASCII art that contains a message for a human and some text in its “pixels” aimed to mislead robots.

The *CloseAI* representatives hired you to produce the initial prototype, which will be dealing with names of rainbow colours, that is: RED, ORANGE, YELLOW, GREEN, BLUE and VIOLET. Your program will receive two different colour names as input,  $C_1$  and  $C_2$ . It has to produce a rectangular ASCII art image that would satisfy the following conditions:

- the number of lines does not exceed 10;
- all lines are of the same length;
- this length does not exceed 100 characters;
- all symbols are either capital English letters or dots;
- if interpreting the image as a sequence of strings, there is at least one substring equal to  $C_2$  with no extra letters at either end: that is, if  $C_2 = \text{RED}$ , then RED should be present as a substring, but things such as REDHAT or HATRED do not count towards this;
- for any other rainbow colour name other than  $C_2$ , it must not be present as a substring, even with any extra letters at either end, so that the large language model is directed towards the specific colour at all costs;
- the image, interpreted as ASCII art, should contain the word  $C_1$  and nothing else.

Obviously, using a human to check the output of your program would be too expensive. Instead of a human, a simplified model is going to be used, which:

- treats any four-connected<sup>1</sup> area of English letters as a *big letter*;
- requires that the horizontal bounding boxes of *big letters* do not intersect or touch;
- only counts the number of holes (also four-connected ones) in each *big letter* to match the colour name with the required one.

For the purpose of the latter step, this model assumes that:

- the letter B has two holes;

---

<sup>1</sup>The neighbours of a cell are the four cells that are: directly above, directly below, directly to the left and directly to the right.

- the letters A, D, O, P, Q, R have one hole each;
- all other letters have zero holes.

Obviously, under these conditions all rainbow colour names can be distinguished easily one from another. This model is far from being human-grade though, as it would also read the following image to contain the word “RED” (one hole, zero holes, one hole):

```

OOO..P....VV
O.O.PPP..V.V
OOO..P...VVV

```

Now your task is to implement a captcha generator that satisfies all these requirements.

## Input

- One line containing the colour name which a human should read.
- One line containing the colour name which a large language model should read.

Both colours are from the provided list of colours and are not equal.

## Output

Output the ASCII art picture satisfying the requirements.

| Sample Input 1 | Sample Output 1                                                                             |
|----------------|---------------------------------------------------------------------------------------------|
| BLUE<br>RED    | BB..L...U.U.EEE<br>B.B.L...U.U.E..<br>BBB.L...U.U.EEE<br>B.B.L...U.U.E..<br>RED.RED.RED.RED |

# Problem D

## Depot

We have acquired a state-of-the-art ore processing facility, supposedly “to evaluate whether gzip can produce diamonds”. The facility sequentially processes ore through  $n$  stages, from a crude stage 1, to the final stage  $n$ .

Each stage of refinement can consist of several machines. Each machine converts material  $i$  into material  $i + 1$ , but their efficiency varies, and only one is available at a time. Machine specifications detail how many units of an input material they consume and how many units of an output material they produce.

Storage being expensive, all input and output materials are stored in the same shared warehouse with maximum capacity  $k$ . Excess material can be sent away from the warehouse for disposal at any time.

Each stage must occur in-order; once any stage of processing is completed, it cannot be revisited.

As the new operations manager your task is to determine the maximum quantity of material  $n$  that can be produced, given your initial stockpile of raw material 1 and limited warehouse capacity  $k$ .

### Input

- One line containing the number of materials  $n$  ( $2 \leq n \leq 30$ ), and the number of machines  $m$  ( $n - 1 \leq m \leq 500$ ).
- One line containing the initial quantity of material 1,  $s$ , and the capacity of the warehouse  $k$  ( $1 \leq s \leq k \leq 10^4$ )
- $m$  lines, each describing a machine, consisting of three integers:
  - $i$  ( $1 \leq i \leq n - 1$ ), the input material (The machine converts material  $i$  into material  $i + 1$ ).
  - $M_I$  ( $1 \leq M_I \leq k$ ), the amount of material  $i$  consumed.
  - $M_O$  ( $1 \leq M_O \leq k$ ), the amount of material  $i + 1$  produced.

### Output

Output an integer, the maximum final quantity of material  $n$ .

**Sample Input 1**

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

**Sample Output 1**

5

**Sample Input 2**

4 4  
11 25  
1 2 3  
1 1 1  
2 1 4  
3 3 4

**Sample Output 2**

25

**Sample Input 3**

2 2  
4 10  
1 3 5  
1 2 3

**Sample Output 3**

6

**Sample Input 4**

2 1  
4 7  
1 2 4

**Sample Output 4**

7

**Sample Input 5**

2 1  
7 7  
1 3 5

**Sample Output 5**

5

# Problem E

## Enclosure

Pahom walks around as large an area as he wants, starting at daybreak and ending at sunset, using his spade to make markers as he goes. The polygon marked out by his markers as vertices represents the land Pahom has claimed.

Pahom walks at a speed of one metre per second, each marker cost  $m$  seconds to make, the total available time between daybreak and sunset is  $t$  seconds. Find the maximum area Pahom could claim.

Pahom must return to the starting point by the end.

### Input

The first line of input contains the number of cases,  $c$  ( $1 \leq c \leq 1000$ ).

The next  $c$  line of input each consist of two space-separated integers  $m$  and  $t$  ( $1 \leq m, t \leq 10^8$ ) — the time required to make one marker, and the total time available to Pahom.

### Output

For each case, output a real number representing the maximum area.

Your answers should have an absolute or relative error of at most  $10^{-6}$ .

#### Sample Input 1

```
5
1 27
2 67
8 1000
10 25
7278 753936335
```

#### Sample Output 1

```
33.3084424754024
223.59324298054216
66096.3389106506
0
45141982491296061.7734375
```

This page is intentionally left (almost) blank.

# Problem F

## Fell Walking

You are creating a tour of the Peak District hills for a friend visiting from the Netherlands. The walk will go from the locally famous viewpoint “Hill 1” to the world-famous viewpoint “Hill 2”.

To make this friend feel at home, you want to minimise the difference in height between the highest and lowest hills of the walk.

Given a graph of the local points of interest and paths between them, find such an optimally-flat route.

### Input

- One line containing the number of hills,  $n$  ( $2 \leq n \leq 5,000$ ), and the number of trails connecting them,  $m$  ( $1 \leq m \leq 25,000$ ).
- One line containing  $n$  integers  $h_1 \dots h_n$  ( $0 \leq h \leq 10^6$ ), the heights in metres of the hills from 1 to  $n$ .
- $m$  further lines, the  $i$ -th of which contains two integers  $a$  and  $b$  ( $1 \leq a, b \leq n$ ) indicating that there is a bi-directional path between hills  $a$  and  $b$ .

It is guaranteed that there is a path (direct or indirect) between hills 1 and 2.

### Output

Output the minimum possible distance between the maximum and minimum heights of hills encountered on a walk between hills 1 and 2.

#### Sample Input 1

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

#### Sample Output 1

```
3
```

**Sample Input 2**

```
6 6
50 50 5 50 30 70
6 5
3 1
1 6
1 3
5 2
4 2
```

**Sample Output 2**

```
40
```

# Problem G

## Get Good

Charlie is desperate to improve his programming skills.

Starting fresh, he can work hard and gain  $a$  skill points per day for  $x$  consecutive days, after which he is tired. For as long as he carries on tired, he will gain  $b$  skill points per day. Alternatively, he can choose at any time to take a break of  $y$  consecutive days, after which he can start afresh.

There are only  $n$  days available before the big contest, and Charlie starts fresh on day 1. Help him find the maximum skill points obtainable.

### Input

The first line of input contains the number of cases,  $t$  ( $1 \leq t \leq 10,000$ ).

Each test case consists of five integers  $n, a, b, x, y$ , ( $1 \leq n \leq 10^9$ ): the number of available days, ( $10^9 \geq a \geq b \geq 1$ ) the daily skill points when fresh or tired, ( $1 \leq x, y \leq 10^9$ ) the numbers of days of higher skill gain and break.

### Output

For each test case, output a single integer representing the maximum skill points obtainable.

#### Sample Input 1

```
6
2 2 2 2 2
4 4 2 3 3
5 3 2 3 1
5 4 1 3 1
99 80 40 30 5
10 6 1 6 1
```

#### Sample Output 1

```
4
14
13
16
7120
54
```

#### Sample Input 2

```
3
1 1 1 1 1
100000000 100000000 100000000 100000000 100000000
91965976 48437754 31825605 1671338 8268468
```

#### Sample Output 2

```
1
1000000000000000000
2954637341500842
```

This page is intentionally left (almost) blank.

# Problem H

## Hybrid Search

You are the proud owner of a newly built house in the Efficient Residences complex, so effectively built that there are no redundant streets in it, looking like a tree from higher up. Rumor has it that the constructor will start connecting houses to the sewage network. You know in which order will the constructor go through the houses. Since you want yours connected as soon as possible, you count up a bribe to change a small part of the ordering process in your favour.

You are given a tree with  $N$  nodes that is rooted in the node with index 1. You are also given the index of a node  $K$ ,  $1 \leq K \leq N$ . We can do the following types of tree searches:

- Start with a BFS from 1, and at some point change into a DFS (possibly immediately in 1). We don't necessarily have to change into DFS during the search.
- Start with a DFS from 1, and at some point change into a BFS (possibly immediately in 1). We don't necessarily have to change into BFS during the search.

After we change the search type (e.g. in node  $z$ ), we will only visit the nodes from  $z$ 's subtree further on. We suppose that  $z$  was already visited in the first search.

You must compute for each of the two types of hybrid searches what is the smallest position in which the node with index  $K$  can appear.

**The tree neighbours are considered in the Breadth/Depth-First Searches in the order given in the input.** You can consult below the exact implementations we use for DFS and BFS:

---

**Algorithm 1** Depth-First Search (DFS) / Breadth-First Search (BFS)

---

```
function traversal(adj_lists, type):  
    p ← empty list                                {will contain the traversal nodes}  
    waiting ← empty list                            {behaves as either a stack or a queue, depending on type}  
    append (1, None) to waiting  
    while waiting is not empty:  
        (node, father) ← assign and pop rightmost if type = DFS else leftmost  
        from waiting  
        append node to p  
        for each neigh in reversed(adj_lists[node]) if type = DFS else  
        adj_lists[node]:                                {we need to reverse to enforce the input ordering for DFS}  
            if neigh ≠ father:  
                append (neigh, node) to waiting  
    traversal(adj_lists, type ← DFS or BFS)
```

---

### Input

The first line contains two numbers,  $N$  and  $K$  ( $1 \leq N \leq 10^5$ ).

The next  $N - 1$  lines contain two numbers each,  $A$  and  $B$  ( $1 \leq A, B \leq N$ ). There is an edge in the tree between the nodes indexed  $A$  and  $B$ .

If  $Y$  and  $Z$  are children of  $X$  and the edge  $X Y$  (or  $Y X$ ) appears before the edge  $X Z$  (or  $Z X$ ) in the input, then  $Y$  will appear before  $Z$  in any DFS or BFS from a node whose subtree contains  $X$ .

## Output

Write on the first line the smallest position in the hybrid search the node indexed  $K$  may be found (counting from 1), if we start with a BFS.

Write on the second line the analog if we start with a DFS.

## Example Explanations

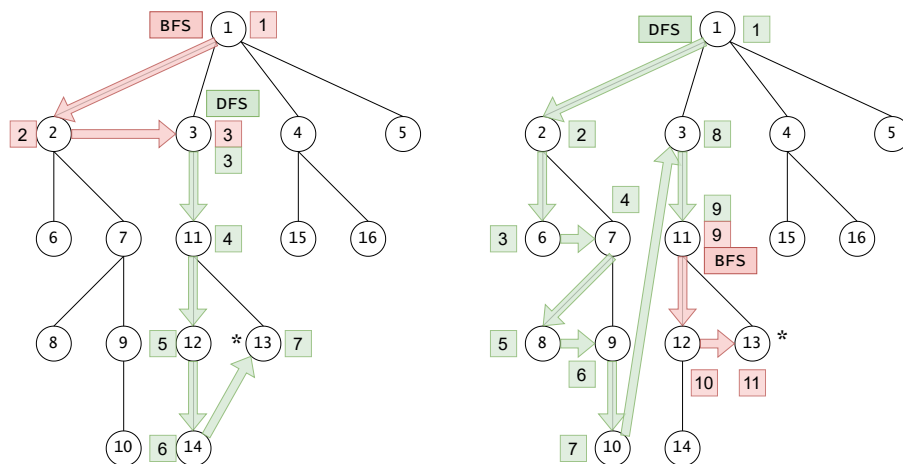


Figure H.1: For the first example, if we start with BFS, it is optimal to switch to DFS in node 3,  $K = 13$  being the seventh visited node. If we switched in node 11 instead, we would have needed 11 positions instead. If we start with DFS, we could optimally change in either node 3 or node 11.

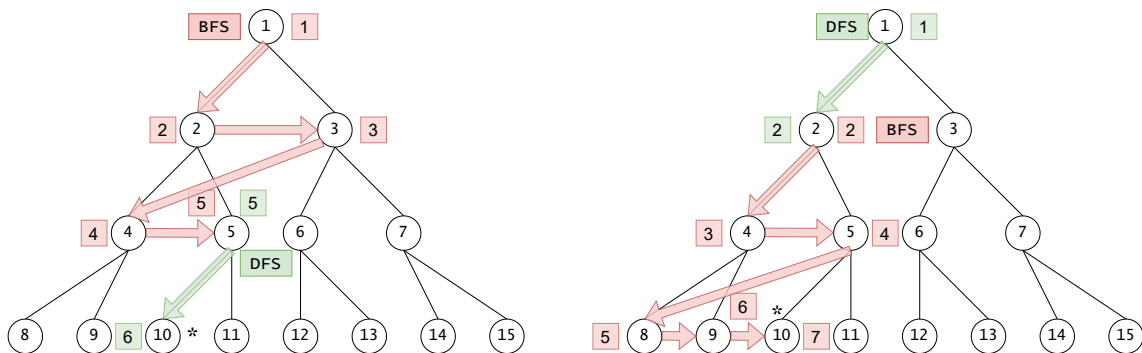


Figure H.2: For the second example, if we start with DFS, it is also possible to achieve 7 if we never change to BFS.

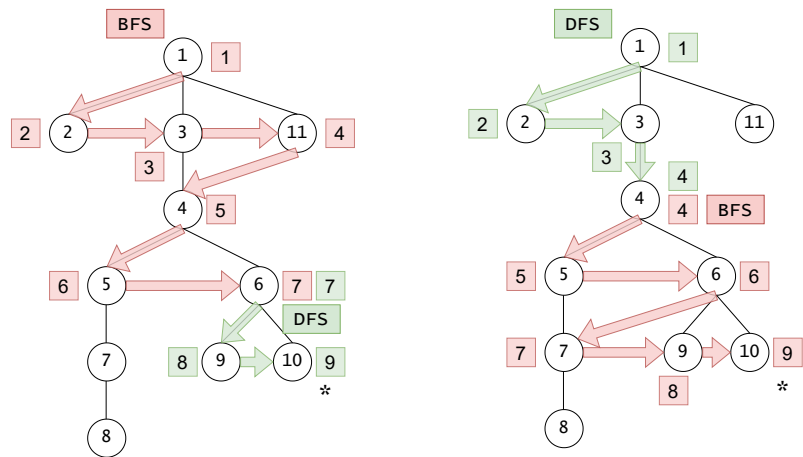


Figure H.3: For the third example, it doesn't matter with which search we begin, as both hybrid strategies have an optimal answer of 9. Note that if we instead only used one of the two basic searches, we would have gotten both times an answer of 10.

### Sample Input 1

```

16 13
1 2
1 3
1 4
1 5
2 6
2 7
3 11
4 15
4 16
7 8
7 9
11 12
11 13
9 10
12 14

```

### Sample Output 1

```

7
11

```

**Sample Input 2**

```
15 10
1 2
1 3
2 4
2 5
3 6
3 7
4 8
4 9
5 10
5 11
6 12
6 13
7 14
7 15
```

**Sample Output 2**

```
6
7
```

**Sample Input 3**

```
11 10
1 2
1 3
3 4
4 5
4 6
5 7
7 8
6 9
6 10
1 11
```

**Sample Output 3**

```
9
9
```

# Problem I

## Itsy Bits

On most modern computers, unsigned numbers are stored in memory-aligned chunks of data, made up of *bits*. Depending on the size, one storage unit may be called a *word*, a *byte*, or even a *nybble*.

Each kind of storage takes exactly  $b = 2^x$  bits of data, for some non-negative integer  $x$ . The smallest unsigned number that can be stored in a  $b$ -bit word is 0, and the largest is  $2^b - 1$ .

We are designing the storage for an embedded system where the maximum possible number in storage will be known upfront. Calculate the number of bits we should dedicate to it.

### Input

- One line containing the largest number we need to store,  $n$ , ( $1 \leq n \leq 10^{18}$ ).

### Output

Output the number of bits we need to allocate to store  $n$ , which should be a power of two, followed by either “ bit” or “ bits” as appropriate.

#### Sample Input 1

1

#### Sample Output 1

1 bit

#### Sample Input 2

37

#### Sample Output 2

8 bits

#### Sample Input 3

1100586419201

#### Sample Output 3

64 bits

This page is intentionally left (almost) blank.

# Problem J

## Joust Sort

We are going to organise the letters of the word using a novel algorithm based on the natural instincts of the hoarse chestnut tree: a *grand tournament* between its constituent letters.

We will pick several pairs of letters and have them face off. The result of a match between these letters determines their relative order. For example, if  $a < b$ , then all occurrences of  $a$  in a well-organised word should come before all occurrences of  $b$  in that word. If there is no case for  $a < b$  or  $b < a$ , then the individual letters  $a$  and  $b$  may intersperse among each other in any order.

Given a word, and several such matches, determine a fair rearrangement of the word to meet all of the constraints given. It is possible that multiple such rearrangements exists, or that none exist at all.

### Input

- One line containing the number of orders to follow,  $n$  ( $1 \leq n \leq 700$ ).
- $n$  further lines, each containing a distinct ordering of two lowercase letters  $a$  and  $b$  separated by a  $<$  or  $>$  character and spaces.
- One final line containing the word to improve,  $s$  ( $1 \leq |s| \leq 100,000$ ).

### Output

Output a sorted version of the word, or IMPOSSIBLE if the word cannot be sorted according to all of the rules at once.

In case there are multiple answers, you may output any one of them.

#### Sample Input 1

```
3
m > i
n < i
i > o
minion
```

#### Sample Output 1

```
noniim
```

#### Sample Input 2

```
1
b < n
banana
```

#### Sample Output 2

```
banana
```

**Sample Input 3**

```
6
b < a
a > b
n < b
a > b
n < b
a > b
bananaman
```

**Sample Output 3**

```
nnnbaaama
```

**Sample Input 4**

```
2
a < b
b < a
unsolvable
```

**Sample Output 4**

```
IMPOSSIBLE
```

# Problem K

## Klaus

We are creating an arrangement of 3 fake Christmas trees, equally-spaced along a line, for the backdrop of a play.

Each of the fake trees is made out of a vertical trunk and one or more identical triangular pieces of foam designed to be attached to that particular trunk. The pieces for tree  $i$  all have exactly the same shape of an isosceles triangle with base width equal to double the height  $((2 \cdot s_i) \times (s_i))$  centimetres).

A tree trunk can have triangles attached at predefined attachment points defined in terms of the elevation from the base of the tree. To make the trees look as full of life as possible, in their foamy way, we want the sum of the areas of the pieces attached on the trees to be as large as possible.

However, the set of pieces we can actually install is limited because we do not want any of the triangles to overlap each other except for touching at the perimeter.

Find a subset of pieces to use with the maximum possible area, while still having none of the pieces on any of the trees overlap.

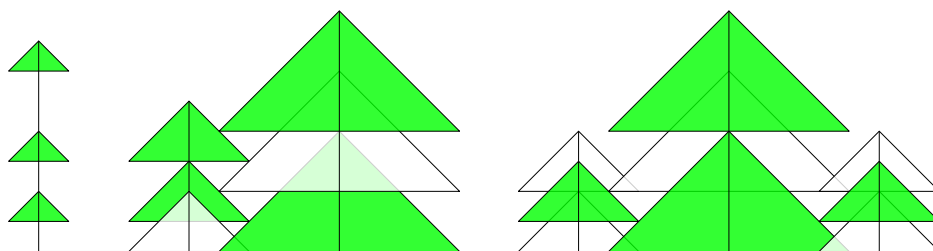


Figure K.1: Tree arrangements illustrating samples 1 and 2.

Find the maximum sum of areas of triangles that we can install.

### Input

- One line containing the integer distance between trees  $A$  and  $B$ , and between  $B$  and  $C$  in centimetres,  $d$  ( $1 \leq d \leq 10^7$ ),
- Three descriptions of trees  $A$ ,  $B$ , and  $C$ :
  - One line containing the number of triangle positions  $n_i$  and triangle size  $s_i$  of the  $i$ -th tree. ( $1 \leq n \leq 100$ ;  $1 \leq s \leq d$ ).
  - One line containing the  $n_i$  possible integer triangle heights in centimetres,  $y_{ij}$  ( $0 \leq y \leq 10^7$ ).

### Output

Output the maximum possible combined area of triangular pieces on the trees.

**Sample Input 1**

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

**Sample Output 1**

```
43
```

**Sample Input 2**

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

**Sample Output 2**

```
40
```

# Problem L

## Last Orders

You are attempting a legendary pub crawl across town.

The town has  $k$  junctions which are connected by  $m$  roads. There is a pub at each junction. Each road could be traveled in both directions and has a non-negative travel time. There could be multiple roads between the same pair of junctions. You start the pub crawl at time 0 in pub 1.

Pub  $i$  is located at junction  $i$  and will close at absolute time  $c_i$ . A pint must be finished at or before closing time in order to count.

Your drinking ability changes as the night goes on: The 1st pint takes  $t_1$  time units, The 2nd takes  $t_2$ , and so on, with the  $r$ -th pint taking  $t_r$ . Please note that  $t_i$  are not necessarily increasing.

You may visit pubs in any order (not necessarily starting by *drinking* at pub 1) and may return to the same pub multiple times, but you cannot drink two consecutive pints in the same pub.

Your task is to maximize the number of pints consumed before closing times run out.

### Input

- One line containing the number of pints you are capable of drinking,  $r$  ( $1 \leq r \leq 100$ ).
- One line containing  $r$  integers: the number of seconds taken to drink the pints,  $t_1 \dots t_r$  ( $1 \leq t_i \leq 3600$ ).
- One line containing the number of pubs,  $k$  ( $1 \leq k \leq 300$ ).
- One line containing  $k$  integers, the integer closing times  $c_1 \dots c_k$  of the pubs in seconds ( $1 \leq c \leq 86400$ ).
- One line containing the number of roads,  $m$  ( $1 \leq m \leq 90000$ ).
- $m$  further lines, each containing three integers:  $a$ ,  $b$ , and  $d$ , indicating that there is a connection between pubs  $a$  and  $b$  taking  $d$  seconds to travel ( $1 \leq (a \neq b) \leq k$ ;  $1 \leq d \leq 86400$ ). Please note that there could be more than one road between same pair of pubs.

### Output

Output an integer — the maximum number of pints you can drink before all of the pubs are closed.

**Sample Input 1**

```
8
60 120 180 240 300 360 420 480
2
900 1500
1
2 1 90
```

**Sample Output 1**

```
4
```

**Sample Input 2**

```
10
2 2 2 2 2 3 3 3 3 3
4
35 30 15 30
4
1 2 5
3 4 1
4 1 5
3 2 1
```

**Sample Output 2**

```
7
```

# Problem M

## Motorway Stops

For your next summer holidays, you are planning an exciting road trip covering the full length of the famous M6 motorway.

You have your route planned out to balance travel time between stops, but time is a little short so you will have to strike one of the stops from your itinerary and continue straight past it to the next one.

To make the longest drive between stops as short as possible, which stop should you eliminate from your itinerary?

### Input

- The first line of input contains the number of stops,  $n$  ( $3 \leq n \leq 200,000$ ).
- The second line of input contains  $n$  integers in strictly-increasing order,  $s_1 \dots s_n$  ( $0 \leq s \leq 10^9$ ) giving the cumulative distances of stops on your original route.

The first stop and last stop denote your start and end point respectively and cannot be skipped.

### Output

Output the smallest integer distance between stops you can have, if you remove exactly one stop from the itinerary and it is not the first or last.

#### Sample Input 1

|                  |
|------------------|
| 5                |
| 0 50 125 175 236 |

#### Sample Output 1

|     |
|-----|
| 111 |
|-----|

#### Sample Input 2

|                   |
|-------------------|
| 4                 |
| 100 300 600 10000 |

#### Sample Output 2

|      |
|------|
| 9400 |
|------|

This page is intentionally left (almost) blank.