NOI 2023 Qualification

18 February 2023

Tasks	Time Limit	Memory Limit
Task 1: Area	1 second	1GB
Task 2: Swords	1 second	1GB
Task 3: Dolls	1 second	1GB
Task 4: Burgers	1 second	1GB
Task 5: Network	2 seconds	1GB

Have Fun!

Task 1: `Area`

Stuart has n rectangular frames, which are numbered from 1 to n. Frame i is a rectangle with height $h[i]$ and width $w[i]$.

The size of a frame is the area that it covers. Stuart wants you to help him find the area covered by the largest size frame that he has.

Input format

Your program must read from standard input.

The first line of input contains exactly 1 integer, n.

The next n lines of input contains two space-separated integers each. The i-th such line of input will contain $h[i]$ and $w[i]$ respectively, representing the height and width of frame i.

Output format

Your program must print to standard output.

The output should contain one integer, the area covered by the largest size frame Stuart has.

The output should contain only a single integer. Do not print any additional text such as `'Enter a number'` or `'The answer is'`.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq n \leq 100$
- $1 \leq h[i], w[i] \leq 1000$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	50	$n = 1$
2	50	No additional restrictions

Sample Testcase 1

This testcase is valid for subtask 2 only.

Input	Output
3 5 9 19 4 8 10	80

Sample Testcase 1 Explanation

The size of frame 1 is $h[1] \times w[1] = 5 \times 9 = 45$.

The size of frame 2 is $h[2] \times w[2] = 19 \times 4 = 76$.

The size of frame 3 is $h[3] \times w[3] = 8 \times 10 = 80$.

Among the above frames, the largest size is 80.

Sample Testcase 2

This testcase is valid for subtask 2 only.

Input	Output
5 8 2 4 9 3 8 1 7 9 4	36

Task 2: `Swords`

Yan Hao has n swords numbered from 1 to n. Sword i has attack $a[i]$ and defence $b[i]$.

Yan Hao thinks that sword i is **useless** if there exists a different sword j ($j \neq i$) such that $a[i] \leq a[j]$ and $b[i] \leq b[j]$. That is, a sword i is **useless** if the attack and defence of another sword j are both at least as good as that of sword i. If a sword is **not useless**, we say that it is **useful**.

Two swords are considered equivalent if they have the same attack and same defence. It is guaranteed that no pair of swords are equivalent.

Help Yan Hao find the number of **useful** swords in his collection.

Input format

Your program must read from standard input.

The first line of input contains exactly 1 integer, n.

The next n lines of input contains two space-separated integers each. The i-th such line of input will contain $a[i]$ and $b[i]$ respectively, indicating the attack and defence of sword i.

Output format

Your program must print to standard output.

The output should contain one integer, the number of **useful** swords.

The output should contain only a single integer. Do not print any additional text such as `'Enter a number'` or `'The answer is'`.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq n \leq 100\,000$
- $1 \leq a[i], b[i] \leq 10^9$

- For all $1 \leq i < j \leq n$, $a[i] \neq a[j]$ or $b[i] \neq b[j]$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	11	$n \leq 500$
2	21	$a[i], b[i] \leq 500$
3	34	$a[i] = i$
4	25	$a[i] \neq a[j]$ for every $1 \leq i < j \leq n$
5	9	No additional constraints

Sample Testcase 1

This testcase is valid for subtasks 1, 2, 4 and 5.

Input	Output
3 2 3 1 3 5 3	1

Sample Testcase 1 Explanation

Comparing sword 1 with sword 3, we have $a[1] = 2 \leq 5 = a[3]$ and $b[1] = 3 \leq 3 = b[3]$, so sword 1 is **useless**.

Comparing sword 2 with sword 1, we have $a[2] = 1 \leq 2 = a[1]$ and $b[2] = 3 \leq 3 = b[1]$, so sword 2 is **useless**.

Sword 3 is the only **useful** sword.

Sample Testcase 2

This testcase is valid for subtasks 1, 2, 4 and 5.

Input	Output
4 5 6 2 5 6 9 1 3	1

Task 3: `Dolls`

Marc is teaching some children about objects with different sizes. To demonstrate this concept, he will be using dolls. These dolls are hollow on the inside, so smaller dolls can be placed inside larger ones.

Each doll has a certain size. A doll of size x can fit inside another doll of size y if $y - x \geq 2$. In other words, a smaller doll can fit in a larger doll if the difference in size between the larger doll and the smaller doll is at least 2.

A doll stack is formed by selecting some dolls that Marc has and repeatedly fitting the smallest doll into the second smallest doll until only one doll is left. The size of a doll stack is the number of dolls used to create it.

There are n days. On the i^{th} ($1 \leq i \leq n$) day, Marc will buy a doll of size $a[i]$. After buying the doll, he will try to construct a doll stack with the maximum number of dolls. Help Marc compute, for each day, the maximum size of a doll stack using the dolls available on that day.

Input format

Your program must read from standard input.

The first line of input contains exactly 1 integer, n.

The second line contain n integers $a[1], a[2], \ldots, a[n]$, representing the sizes of the dolls bought on each of the n days.

Output format

Your program must print to standard output.

The output should contain n integers on a single line and separated by spaces. The i-th integer should be the maximum size of a doll stack using the dolls available on that day.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq n \leq 100\,000$
- $1 \leq a[i] \leq 500\,000$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	23	$n \leq 200$
2	14	$a[i]$ is not a multiple of 2 for all i $(1 \leq i \leq n)$
3	27	$a[i]$ is not a multiple of 4 for all i $(1 \leq i \leq n)$
4	36	No additional restrictions

Sample Testcase 1

This testcase is valid for subtasks 1 and 4.

Input	Output
5 1 2 3 4 5	1 1 2 2 3

Sample Testcase 1 Explanation

Day 1: Marc can only stack doll 1.

Day 2: Marc can stack either doll of size 1 or doll of size 2.

Day 3: Marc can stack doll of size 1 and doll of size 3. He cannot stack all 3 dolls as doll of size 2 does not fit inside doll of size 3.

Day 4: Marc can stack either doll of size 1 and doll of size 3, doll of size 1 and doll of size 4 or doll of size 2 and doll of size 4.

Day 5: Marc can stack doll 1, 3 and 5.

Sample Testcase 2

This testcase is valid for subtasks 1 and 4.

Input	Output
5 2 4 6 8 10	1 2 3 4 5

Sample Testcase 2 Explanation

All dolls can be stacked as each day passes because the gap between adjacent dolls is equal to 2.

Sample Testcase 3

This testcase is valid for subtasks 1, 3 and 4.

Input	Output
5 3 3 1 3 2	1 1 2 2 2

Sample Testcase 3 Explanation

On the first 2 days, Marc can only stack 1 doll of size 3.

On the 3^{rd} day, Marc can stack the doll of size 1.

On the 5^{th} day, Marc **cannot** stack the doll of size 2 because the gap between the doll of size 3 and itself is exactly 1.

Task 4: `Burgers`

Kai the lobster is starting a burger chain selling burgers. He has n ingredients to work with, which are labelled from 1 to n. For each ingredient i, he has $x[i]$ portions of ingredient i.

He has two recipes for burgers. For each ingredient i, the first recipe requires $a[i]$ portions of ingredient i and the second recipe requires $b[i]$ portions of ingredients i.

Can you help Kai compute the maximum total number of burgers he can make?

Input format

Your program must read from standard input.

The first line of input consists of one integer n, the number of different ingredients.

The second line consists of n spaced integers $x[1], x[2], \ldots, x[n-1], x[n]$, the total number of portions Kai has of each ingredient.

The third line consists of n spaced integers $a[1], a[2], \ldots, a[n-1], a[n]$, the number of portions of each ingredient for the first recipe.

The fourth line consists of n spaced integers $b[1], b[2], \ldots, b[n-1], b[n]$, the number of portions of each ingredient for the second recipe.

Output format

Your program must print to standard output.

The output should contain a single integer on a single line, the largest number of burgers Kai can make.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq n \leq 100\,000$
- $1 \leq x[i], a[i], b[i] \leq 10^9$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	9	$a[i] = b[i]$ (i.e. the two recipes are the same)
2	17	$n, x[i] \leq 100$
3	25	$n, x[i] \leq 1500$
4	49	No additional restrictions

Sample Testcase 1

This testcase is valid for subtasks 2, 3 and 4.

Input	Output
3 14 10 100 3 1 1 2 3 1	5

Sample Testcase 1 Explanation

He can make 3 burgers using the first recipe and 2 burgers using the second recipe for a total of 5 burgers.

Sample Testcase 2

This testcase is valid for all subtasks.

Input	Output
2 83 72 1 3 1 3	24

Sample Testcase 2 Explanation

He can make 24 burgers of either type, since both recipes are the same.

Task 5: `Network`

Rui Yuan the Duck is building a new network to run all of his new applications. This network is comprised of n servers numbered from 1 to n. There are also $n-1$ network links, numbered from 1 to $n-1$ connecting these servers.

The i-th of these links connects servers $u[i]$ and $v[i]$ **bidirectionally**. It is guaranteed that it is possible to travel from any server to any other server using only the servers and network links.

Initially, all n servers are active. Rui Yuan has m applications running, which have distinct IDs from 1 to m. Each application has 2 databases. Application j has databases in servers $a[j]$ and $b[j]$ (which may be the same server). It is possible for two different applications to have a database on the same server.

In order for application j to work, both of its databases must be connected through network links and **active** servers. To be precise, application j is **working** if **both** of the following conditions are met:

- Servers $a[j]$ and $b[j]$ are both active.
- One can travel from server $a[j]$ to server $b[j]$ using only network links and active servers.

Rui Yuan has asked Benson the Rabbit to test his network. Using his hacking skills, Benson can select any set of servers and deactivate all of them simultaneously. The robustness of the network is the minimum number of servers that must be deactivated to ensure **all** m applications are not working.

Benson is very busy, so he wants you to help him compute the robustness of the network and the selection of servers he should deactivate so that he doesn't waste time deactivating more servers than needed.

Input format

Your program must read from standard input.

The first line of input contains two integers, n and m.

$n-1$ lines will follow. The i-th line contains two integers, $u[i]$ and $v[i]$ respectively, which are the servers connected by the i-th network link.

Another m lines will follow. The j-th line contains two integers, $a[j]$ and $b[j]$ respectively, which are the servers storing the databases for application j.

Output format

Your program must print to standard output.

Output two lines. The first line should contain a single integer, representing the robustness of the network, which we will denote as x.

The second line should contain x integers, representing the servers that should be deactivated. All x integers must be pairwise distinct. You may output them in any order.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq n \leq 200\,000$
- $1 \leq m \leq 200\,000$
- $1 \leq u[i], v[i] \leq n$, $u[i] \neq v[i]$ (for all $1 \leq i \leq n$).
- $1 \leq a[j], b[j] \leq n$ (for all $1 \leq j \leq m$)
- It is possible to travel from any server to any other server using only the servers and network links

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	4	$a[i] = b[i]$
2	17	$u[i] = i, v[i] = i + 1$
3	14	$n, m \leq 15$
4	29	$n, m \leq 2000$
5	36	No additional restrictions

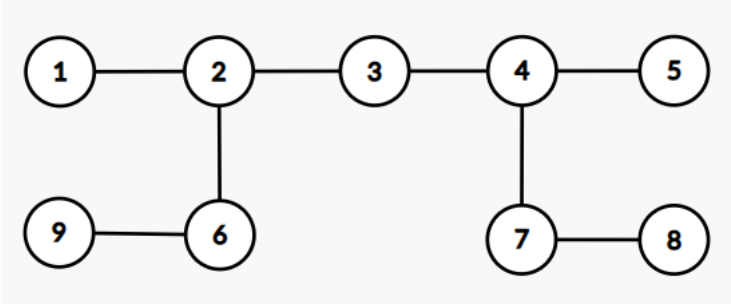
Sample Testcase 1

This testcase is valid for subtasks 3, 4 and 5.

Input	Output
9 4 1 2 2 3 2 6 3 4 4 5 4 7 6 9 7 8 6 2 5 3 4 8 5 9	2 4 2

Sample Testcase 1 Explanation

The network described in the input may be represented by the diagram below.



Suppose servers 4 and 2 are deactivated, then all applications will not be working as shown below.

- $b[1] = 2$, so application 1 is not working.
- A path from server $a[2] = 5$ to server $b[2] = 3$ must pass through server 4, so application 2 is not working.
- $a[3] = 4$, so application 3 is not working.
- A path from server $a[4] = 5$ to server $b[4] = 9$ must pass through both servers 2 and 4, so application 4 is not working.

It can be shown that it is not possible to ensure all applications are not working by deactivating a single server. Hence, the robustness of the network is 2.

Note that you may output the chosen servers in any order, so the following output is also valid.

```
2
2 4
```

Also note that there may exist other selections of 2 servers that cause all applications to not be working.

Sample Testcase 2

This testcase is valid for subtasks 1, 3, 4 and 5.

Input	Output
6 5 1 2 2 3 4 1 3 5 4 6 1 1 2 2 3 3 2 2 4 4	4 3 2 4 1

Sample Testcase 2 Explanation

Note that you may output the chosen servers in any order. For example, the following output is also valid.

```
4
4 3 2 1
```

Sample Testcase 3

This testcase is valid for subtasks 2, 3, 4 and 5.

Input	Output
8 3 1 2 2 3 3 4 4 5 5 6 6 7 7 8 3 5 8 5 4 4	2 4 6

Sample Testcase 4

This testcase is valid for subtasks 3, 4 and 5.

Input	Output
6 2 1 2 1 3 1 4 1 5 1 6 2 2 5 6	2 2 1