Long Distance Coach

There is a long-distance coach between the city I and the city O. It has a water supply machine. Passengers and the driver can drink water from it. The coach departs from the city I at time 0, and arrives the city O at time X. On the route, there are N refilling points where we can put water into the machine. The coach will arrive at the i-th refilling point ($1 \leq i \leq N$) at time S_i.

In the beginning, the machine does not have water in it. We can put water into it before departure. Also, we can put water into it when the coach is at a refilling point. The cost of water is W yen per liter regardless of the position of the coach.

At the city I, M passengers get on the coach. The passengers are numbered from 1 to M. No passengers get on the coach at places other than the city I. The j-th passenger ($1 \leq j \leq M$) needs a liter of water at time D_j. If he drinks water, he will need a liter of water after a lapse of time T. In other words, the j-th passenger needs water at time $D_j + kT$ ($k = 0, 1, 2, \ldots$). Here, $1 \leq D_j < T$ is satisfied, and the value of T is the same for all passengers. If the machine does not have water in it when a passenger needs water, the passenger leaves the coach. If the j-th passenger leaves the coach before getting to the city O, we need to refund the fare. The cost for refund is C_j yen.

The driver also needs water. If he drinks water, he will need a liter of water after a lapse of time T, just in the same way as passengers. In other words, the driver needs water at time kT ($k = 0, 1, 2, \ldots$). If the machine does not have water in it when the driver needs water, the operation of the coach is stopped

No two people will need water at the same time. When the coach arrives at the city O or a refilling point, neither passengers nor the driver need water.

Adjusting the amount of water put into machine at refilling points, we want to minimize the sum of the cost of water and the cost of refund, and to operate the coach until the city O. Your task is to decide where and how much we should put water into the water supply machine during the travel.

Task

Given the traveling time for the coach, information on refilling points, and information on passengers and the driver, write a program which calculates the minimum of the sum of the cost of water and the cost of refund assuming the coach gets to the city O.

Input

Read the following data from the standard input.

- The first line of input contains five space separated integers X, N, M, W, T. This means the coach will arrive at the city O at time X, there are N refilling points, there are M passengers in the coach, the cost of water is W yen per liter, and the interval of time of water is T for passengers and the driver.

- The i-th line ($1 \leq i \leq N$) of the following N lines contains an integer S_i. This means the coach will arrive at the i-th refilling point at time S_i.
- The j-th line ($1 \leq j \leq M$) of the following M lines contains two space separated integers D_j, C_j. This means the j-th passenger will need water at time D_j for the first time, and the cost for refund for the j-th passenger is C_j.

Output

Write one line to the standard output. The output contains an integer, the minimum total cost.

Constraints

All input data satisfy the following conditions.

- $1 \leq X \leq 1\,000\,000\,000\,000$.
- $1 \leq N \leq 200\,000$.
- $1 \leq M \leq 200\,000$.
- $1 \leq W \leq 1\,000\,000$.
- $1 \leq T \leq X$.
- $1 \leq S_i < X$ ($1 \leq i \leq N$).
- $1 \leq D_j < T$ ($1 \leq j \leq M$).
- $1 \leq C_j \leq 1\,000\,000\,000$ ($1 \leq j \leq M$).
- D_j ($1 \leq j \leq M$) are different from each other.
- When the coach arrives at the city O or a refilling point, neither passengers nor the driver need water.

Subtask

There are 4 subtasks. The score and additional constraints of each subtask are as follows:

Subtask 1 [16 points]

- $N \leq 8$.
- $M \leq 8$.

Subtask 2 [30 points]

- $N \leq 100$.
- $M \leq 100$.

Subtask 3 [25 points]

- $N \leq 2\,000$.
- $M \leq 2\,000$.

Subtask 4 [29 points]

There are no additional constraints.

Sample Input and Output

Sample Input 1	Sample Output 1

`19 1 4 8 7`
`10`
`1 20`
`2 10`
`4 5`
`6 5`
	`103`

In this sample input, if we put 7 liters of water into the machine before departure, and 4 liters of water into the machine at the first refilling point, the coach will be operated as follows:

1. The coach departs from the city I. At this time, the water supply machine has 7 liters of water.
2. The driver and the passengers $1, 2, 3, 4$ drink 1 liter of water at time $0, 1, 2, 4, 6$, respectively. The remaining amount of water is 2 liters.
3. The driver and the passenger 1 drink 1 liter of water at time $7, 8$, respectively. The remaining amount of water is 0 liter.
4. At time 9, the passenger 2 needs water. But he leaves the coach because the machine does not have water in it.
5. At time 10, we put 4 liters of water into the machine at the first refilling point. The remaining amount of water is 4 liters.
6. The passengers $3, 4$, the driver, and the passenger 1 drink 1 liter of water at time $11, 13, 14, 15$, respectively. The remaining amount of water is 0 liter.

7. At time 18, the passenger 3 needs water. But he leaves the coach because the machine does not have water in it.

8. At time 19, the coach arrives at the city O.

Total amount of water used is 11 liters. The cost of water is 88 yen. The sum of the cost of refund for the passengers $2, 3$ is 15 yen. The total sum is 103 yen.

We output 103 because it is impossible to operate the coach if the total cost is less than or equal to 102 yen,

Sample Input 2	Sample Output 2

`105 3 5 9 10`
`59`
`68`
`71`
`4 71`
`6 32`
`7 29`
`3 62`
`2 35`
	`547`

Sample Input 3	Sample Output 3

`1000000000000 1 1 1000000 6`
`999999259244`
`1 123456789`
	`333333209997456789`

Long Mansion

There is a wide mansion near JOI-kun's house. The mansion has N rooms located in a row from east to west. The i-th room from the eastmost room is called the room i. For each i with $1 \leq i \leq N - 1$, the room i and the room $i + 1$ are connected by a corridor. We can pass corridors in both directions. We need a key to enter a corridor from a room. Each key has a number called the *type*. More than one keys can have the same type.

From the room i or the room $i + 1$, we need a key of type C_i to enter a corridor between them.

There are B_i keys in the room i. Their types are $A_{i,j}$ ($1 \leq j \leq B_i$). If JOI-kun enters a room, he will pick up all the keys in that room. After that, he can use them to enter corridors.

JOI-kun can use keys as many times as he wants. Sometimes, he gets several keys of the same type. But, he has no special advantage to have several keys of the same type compared with the case where he has only one key of that type.

To deal with the situation where he gets lost in the mansion, JOI-kun plans to write a program which answers the following queries:

- If JOI-kun comes into the room x without any keys, can he move to the room y?

Your task is to write a program which answers the above queries, instead of JOI-kun.

Task

Given information of the mansion and the queries, write a program which determines, for each query, whether he can move from a room to another room assuming he is now in the mansion without any keys.

Input

Read the following data from the standard input.

- The first line of input contains an integer N, the number of rooms in the mansion.
- The second line of input contains $N - 1$ space separated integers $C_1, C_2, \ldots, C_{N-1}$. This means we need a key of type C_i to enter a corridor connecting the room i and the room $i + 1$.
- The i-th line ($1 \leq i \leq N$) of the following N lines contains a positive integer B_i, and B_i space separated integers $A_{i,1}, A_{i,2}, \ldots, A_{i,B_i}$. This means there are B_i keys in the room i, and their types are $A_{i,j}$ ($1 \leq j \leq B_i$).
- The following line contains an integer Q, the number of queries.
- The k-th line ($1 \leq k \leq Q$) of the following Q lines contains two space separated integers X_k, Y_k. This means the k-th query asks whether JOI-kun can move from the room X_k to the room Y_k assuming he is now in the room X_k without any keys.

Output

Write Q lines to the standard output. The k-th line ($1 \leq k \leq Q$) of the Q lines contains `YES` if he can move from the room X_k to the room Y_k assuming he is now in the room X_k without any keys. Otherwise, it contains `NO`.

Constraints

All input data satisfy the following conditions.

- $2 \leq N \leq 500\,000$.
- $1 \leq Q \leq 500\,000$.
- $1 \leq B_1 + B_2 + \cdots + B_N \leq 500\,000$.
- $1 \leq B_i \leq N$ $(1 \leq i \leq N)$.
- $1 \leq C_i \leq N$ $(1 \leq i \leq N-1)$.
- $1 \leq A_{i,j} \leq N$ $(1 \leq i \leq N,\ 1 \leq j \leq B_i)$.
- The B_i integers $A_{i,1}, \ldots, A_{i,B_i}$ are different from each other $(1 \leq i \leq N)$.
- $1 \leq X_k \leq N$ $(1 \leq k \leq Q)$.
- $1 \leq Y_k \leq N$ $(1 \leq k \leq Q)$.
- $X_k \neq Y_k$ $(1 \leq k \leq Q)$.

Subtask

This task has 4 subtasks in total. The score and the additional constraints of each subtask are as follows:

Subtask 1 [5 points]

- $N \leq 5\,000$.
- $Q \leq 5\,000$.
- $B_1 + B_2 + \cdots + B_N \leq 5\,000$.

Subtask 2 [5 points]

- $N \leq 5\,000$.
- $B_1 + B_2 + \cdots + B_N \leq 5\,000$.

Subtask 3 [15 points]

- $N \leq 100\,000$.
- $C_i \leq 20$ $(1 \leq i \leq N-1)$.
- $A_{i,j} \leq 20$ $(1 \leq i \leq N,\ 1 \leq j \leq B_i)$.

Subtask 4 [75 points]

There are no additional constraints.

Sample Input and Output

Sample Input 1	Sample Output 1

5
1 2 3 4
2 2 3
1 1
1 1
1 3
1 4
4
2 4
4 2
1 5
5 3
	YES
	NO
	NO
	YES

- In the first query, if JOI-kun visits the rooms 2, 1, 2, 3, 4 in this order, he gets to the room 4.
- In the second query, he can visit the rooms 3, 4 only. Since he can get keys of type 1, 3 only, he can not get to the room 2.
- In the third query, he can not get a key of type 4 from the room 5 to the room 4. Hence he can not get to the room 5
- In the fourth query, if he visits the rooms 5, 4, 3 in this order, he gets to the room 3.

Sample Input 2	Sample Output 2

```
5
2 3 1 3
1 3
1 2
1 1
1 3
1 2
4
1 3
3 1
4 3
2 5
```
	```
	NO
	YES
	NO
	YES
	```

Sample Input 3	Sample Output 3

```
7
6 3 4 1 2 5
1 1
1 5
1 1
1 1
2 2 3
1 4
1 6
3
4 1
5 3
4 7
```
	```
	YES
	NO
	YES
	```

Natural Park

The JOI island is a sightseeing area. The whole island is designated as a natural park.

There are N places and several roads in the JOI island. The places are numbered from 0 to $N - 1$. Every road connects two different places, and we can pass it in both directions. For each place, there are at most 7 roads connecting it with other places. For each pair of two different places, there is at most one road connecting them. We can travel from every place to any other places if we pass through several roads.

You and your friend IOI-chan will investigate the JOI island. To investigate it efficiently, you need to figure out the structure of the JOI island. The JOI island is dangerous because there are many wild animals. Since IOI-chan has high athletic ability, she will explore the JOI island, and you will specify the structure of the JOI island according to IOI-chan's reports.

You will give two places A, B and several possibilities of intermediate places to IOI-chan, and ask whether it is possible to travel from the place A to the place B if we can pass through some of the given intermediate places only. Then, IOI-chan will explore the JOI island, and reports the results to you.

Since they can not take too much time for investigation, the number of queries should be less than or equal to 45 000.

Task

Write a program which communicates with IOI-chan and specifies the structure of the JOI island.

Implementation Details

You need to write a program which implements the way to specify the structure of the JOI island. Your program should include `park.h`.

Your program should implement the following routine.

- `void Detect(int T, int N)`

  This function is called only once.

  - The parameter `T` is the subtask number, and `N` is the number of places.

Your program should output the structure of the JOI island specified by it using the following function:

- `void Answer(int A, int B)`

  The number of calls to this function should be the same as the number of roads in the JOI island.

  - The parameters `A`, `B` denote there is a road connecting the place `A` and the place `B`.

The parameters should satisfy the following conditions:

- `A`, `B` should satisfy $0 \leq$ `A` $<$ `B` $\leq N - 1$. If this condition is not satisfied, your program is considered as **Wrong Answer[1]**.
- If the function is called with parameters (`A`, `B`), there must be a road connecting the place `A` and the place `B`. If this condition is not satisfied, your program is considered as **Wrong Answer[2]**.
- The function should not be called more than once with the same parameter (`A`, `B`). If this condition is not satisfied, your program is considered as **Wrong Answer[3]**.

Moreover, your program can call the following function:

- `int Ask(int A, int B, int Place[])`

  This function is used to ask IOI-chan.

  - `Place` is the pointer of an array of possible intermediate places. For each i ($0 \leqq i \leqq N - 1$), `Place[`i`]` = 1 is we can pass through the place i, and `Place[`i`]` = 0 is we can not pass through the place i.
  - The return value of this function is 1 if we can travel from the place `A` to the place `B` if we can pass through some of the places specified by the array `Place[]` only. Otherwise, the return value is 0.

  The parameters should satisfy the following conditions:

  - $0 \leq$ `A` $<$ `B` $\leq N - 1$.
  - $0 \leq$ `Place[i]` ≤ 1 ($0 \leqq$ `i` $\leqq N - 1$).
  - `Place[A]` = 1.
  - `Place[B]` = 1.

  If these conditions are not satisfied, your program is considered as **Wrong Answer[4]**. However, the behavior of the function is not guaranteed if the length of the array `Place[]` is not equal to N.

  The function `Ask` should not be called more than 45 000 times. If it exceeds, your program is considered as **Wrong Answer[5]**.

When the function `Detect` terminates, if there is a road which is not a parameter of the previous calls to the function `Answer`, your program is considered as **Wrong Answer[6]**.

Your program can implement other functions for internal use, or use global variables. Your program should not use standard input and standard output. Your program should not communicate with other files by any methods.

Compilation and Test Run

You can download an archive file from the contest webpage which contains a sample grader to test your program. The archive file also contains a sample source file of your program.

A sample grader consists of one source file which is either `grader.c` or `grader.cpp`. For example, if your program is `park.c` or `park.cpp`, you run the following commands to compile your program.

- C

  ```
  gcc -std=c11 -O2 -o grader grader.c park.c -lm
  ```

- C++

  ```
  g++ -std=c++14 -O2 -o grader grader.cpp park.cpp
  ```

When the compilation succeeds, the executable file `grader` is generated.
Note that the actual grader is different from the sample grader. The sample grader will be executed as a single process, which will read input data from the standard input and write the results to the standard output.

Input for the Sample Grader

The sample grader reads the following data from the standard input.

- The first line of input contains an integer T, the subtask number.
- The second line of input contains an integer N, the number of places.
- The third line of input contains an integer M, the number of roads.
- In the i-th line ($1 \leq i \leq M$) of the following M lines contains two space separated integers A_i, B_i. This means there is a road connecting the place A_i and the place B_i, and we can pass it in both directions.

Output of the Sample Grader

When the program terminates successfully, the sample grader writes the following information to the standard output. (The quotation mark is not written actually.)

- If your program is considered as correct, the sample grader writes “`Accepted`.”
- If your program is considered as Wrong Answer, the sample grader writes its type in the following form “Wrong Answer [1],” and your program is terminated.

If your program is considered as several types of Wrong Answer, the sample grader reports one of them only.

Constraints

All input data satisfy the following conditions. For the meaning of T, N, M, see Input for the Sample Grader.

- $1 \leq T \leq 5$.
- $2 \leq N \leq 1\,400$.

- $1 \leq M \leq 1\,500$.
- For each place, there are at most 7 roads connecting it with other places.
- We can travel from every place to any other places if we pass through several roads.
- For each pair of two different places, there is at most one road connecting them.

Subtask

There are 5 subtasks. The score and additional constraints of each subtask are as follows:

Subtask 1 [10 points]

- $T = 1$.
- $N \leq 250$.

Subtask 2 [10 points]

- $T = 2$.
- $M = N - 1$.
- For the place 0 or $N - 1$, there is exactly one road connecting it with another place. For every other place, there are exactly 2 roads connecting it with other places.

Subtask 3 [27 points]

- $T = 3$.
- $M = N - 1$.
- For every i ($1 \leq i \leq N - 1$), we can travel from the place 0 to the place i if we pass through at most 8 other places.

Subtask 4 [30 points]

- $T = 4$.
- $M = N - 1$.

Subtask 5 [23 points]

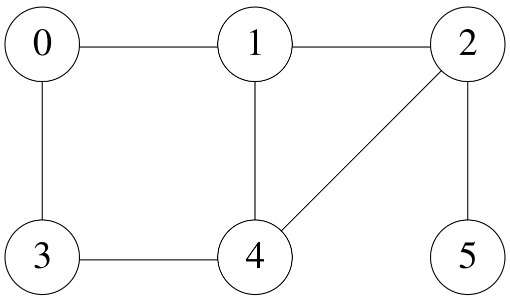
- $T = 5$.

Sample Communication

Here is a sample input for sample grader and corresponding function calls.

Sample Input	Sample Calls	
	Call	Return
1	`Ask(3, 5, {0,0,1,1,1,1})`	1
6	`Answer(2, 4)`	
7	`Answer(2, 5)`	
0 1	`Answer(3, 4)`	
0 3	`Ask(0, 4, {1,0,1,0,1,0})`	0
1 2	`Answer(0, 1)`	
1 4	`Answer(0, 3)`	
2 4	`Answer(1, 4)`	
2 5	`Answer(1, 2)`	
3 4		

Note that the function calls in this example do not necessarily have meaning.
In this example, the function `Detect` is called with parameters `T` = 1, `N` = 6.
In this example, the structure of the JOI island is as follows:



The structure of the JOI island.
The circles and numbers denote the places and their numbers. The line segments denote the roads.

- The first call to the function `Ask` asks whether we can travel from the place 3 to the place 5 if we can pass through the places 2, 3, 4, 5 only. Because it is possible, the function `Ask` returns 1.
- The second call to the function `Ask` asks whether we can travel from the place 0 to the place 4 if we can pass through the places 0, 2, 4 only. Because it is impossible, the function `Ask` returns 0.