

Task: FIL

File Paths

english

BOI 2015, day 2. Available memory: 256 MB.

1.05.2015

Byteasar likes to live risky. He runs with scissors, submits solutions to contest problems without testing on example inputs, and wants all his files to have path names that are exactly as long as the operating system allows (on Linux, for instance, this is 4095 characters).

When Byteasar is working on someone else's computer, it might turn out that not all of the files meet his criterion. In this case he tries to introduce symbolic links (*symlinks*) and use them for creating file paths. You are asked to figure out, for each file in the file system, whether Byteasar can introduce a single symbolic link (of length chosen by him in advance), so that this file can be referred to by a path name of length exactly k characters.

If a file of name `file` is contained in the chain of directories `dir1, dir2, ..., dirj`, then its *absolute file path* is `/dir1/dir2/.../dirj/file`. The root directory can be referred to as `/` and each file contained directly in this directory has the absolute path of the form `/file`. A symbolic link is a named shortcut to a directory, which can be placed in any directory in the file system. **In this task symlinks to files are not allowed.** Using a symlink, we can obtain *alternative* file paths. For example, if we introduced a symlink to `/` with name `hello` in `/`, then `/dir/file`, `/hello/dir/file` and `/hello/hello/dir/file` would all refer to the same file, but have different path name length. As another example, with a symlink to `/` with name `hi` in `/dir`, one could obtain file paths: `/dir/file`, `/dir/hi/dir/file`, `/dir/hi/dir/hi/dir/file`. Note that it is perfectly legal for symlinks to point upwards, downwards or sideways in the file system hierarchy, and even back to the directory they are placed in. For the purpose of this problem, `./` or `../` or `//` path components are *not* considered allowed in path names.

Input

The first line of input contains three positive integers: n (the number of directories other than the root directory), m (the number of files), and k (the desired path name length). The root directory has number 0, and all the remaining directories are numbered 1 through n . Files are numbered 1 through m . The second line contains the length s of the introduced symlink name (we don't care about the name itself, and assume it will not collide with anything else in the file system).

After that follow n lines describing the directories (other than the root directory) that exist in the file system. The i -th of those lines describes the directory number i and consists of two integers: p_i and l_i . They specify that the directory number i has a name of length l_i and its parent directory (i.e. the directory in which the i -th directory is directly contained) has number p_i . It is guaranteed that $p_i < i$.

Finally m lines follow with a description of the files in the file system. The j -th of those lines describes the file number j and consists of two integers: p_j and l_j . They specify that the file number j has a name of length l_j and its parent directory has number p_j .

All files and directories will have names with positive length, and their absolute file paths will be at most k characters long.

Output

Your program should output m lines, one for each file. The j -th line should contain one word, YES or NO, answering the question: is it possible, by introducing a symlink of length s , to create a path name of length exactly k which refers to the file number j ?

v. 3.1

File Paths 1/2

Examples

For the input data:

```
2 4 22
2
0 1
1 5
2 13
2 10
1 4
0 7
```

a correct result is:

```
YES
YES
YES
NO
```

Explanation to the examples: Let us refer to the symlink as LL, the directory names as **a** and **bbbb**, and the file names as **cccccccccccc**, **dddddddddd**, **eeee** and **ffffff**, respectively. The root directory contains the directory **a** and the file **ffffff**; the directory **a** contains the directory **bbbb** and the file **eeee**; and finally the directory **bbbb** contains the files **cccccccccccc** and **dddddddddd**.

```
/
|-- a
|  |-- bbbb
|  |  |-- cccccccccccc
|  |  +-- ddddddddd
|  +-- eeee
+-- fffffff
```

In the first case, the absolute file path `/a/bbbb/cccccccccccc` already has the desired length, so we don't even have to use the symlink. In the second case we can introduce the symlink `/a/LL -> /a`, and refer to `/a/LL/bbbb/dddddddd`. In the third case, we should introduce the symlink `/a/LL -> /`, and refer to `/a/LL/a/LL/a/LL/a/eeee`. In the fourth case we cannot reach the goal regardless of where we introduce the symlink.

Grading

In all subtasks $1 \leq k, s \leq 1\,000\,000$.

Subtask	Conditions	Points
1	$n, m \leq 500$	33
2	$n, m \leq 3000$ and for each file if the answer is positive, one can introduce a symlink that has to be traversed at most once	33
3	$n, m \leq 3000$	34

Task: HAC Hacker

english

BOI 2015, day 2. Available memory: 256 MB.

1.05.2015

Byteasar the hacker has qualified for this year's IHO, the International Hacking Olympiad. One of the tasks in the Olympiad involves competing against a system operator. There are n computers numbered from 1 to n , connected in a ring topology, i.e. computers i and $i + 1$ are connected (for $i = 1, \dots, n - 1$), and also computers n and 1 are connected.

The competition is performed as a game between the hacker and the system operator:

- Byteasar moves first. Afterwards, the operator and Byteasar move alternately.
- In his first move, Byteasar chooses any computer and hacks it (for instance, by exploiting some operating system vulnerabilities).
- In his first move, the operator chooses any non-hacked computer and protects it (for instance, by installing latest security upgrades).
- In all his following moves, Byteasar either (a) does nothing or (b) chooses any computer that is neither hacked nor protected and is directly linked to any hacked computer, and hacks it.
- In all his following moves, the operator either (a) does nothing or (b) chooses any computer that is neither hacked nor protected and is directly linked to any protected computer, and protects it.
- The game ends as soon as both have done nothing in two subsequent moves.

At the beginning of the game none of the computers are hacked or protected.

Every computer i has a certain value v_i which specifies the value of the data which is stored on it. For each hacked computer i , Byteasar scores its value v_i . Byteasar is quite a good hacker, but has no idea of algorithms. That is why he asks you to write a program that computes his maximum possible score, assuming that the operator plays optimally.

Input

The first line of input contains a positive integer n ($n \geq 2$), specifying the number of computers. The second line contains a sequence of n integers v_1, v_2, \dots, v_n ($1 \leq v_i \leq 2000$); number v_i specifies the value of the data stored on computer i .

Output

In the first and only line of output your program should write one integer: Byteasar's maximum possible score against an optimally playing operator.

Examples

For the input data:

4
7 6 8 4

a correct result is:

13

For the input data:

5
1 1 1 1 1

a correct result is:

3

v. 3.2

Hacker 1/2

Explanation to the examples: In the first example, Byteasar in his first move should hack computer 2 (scoring 6). The operator's response will be protecting computer 3. In the next move Byteasar can hack computer 1 (scoring 7). Finally, the operator will protect computer 4.

Grading

Subtask	Conditions	Points
1	$n \leq 300$	20
2	$n \leq 5000$	20
3	$n \leq 500\,000$, hacking computer 1 is optimal as Byteasar's first move	20
4	$n \leq 500\,000$	40

Task: TUG

Tug of War

english

BOI 2015, day 2. Available memory: 256 MB.

1.05.2015

Tug of war is a very popular sport in Byteland. The rules are easy: two teams pull a rope in opposite directions. The annual Byteland tug of war charity is taking place, and a lot of contestants have signed up. As the fair play commissioner, your job is to divide the contestants into two teams, such that the game can go on for a long time.

Since a total of $2n$ contestants have signed up, each team will consist of n contestants. The rope has n spots on the left side and n spots on the right side. The tug of war elite of Byteland are a picky bunch: each contestant has exactly one spot on the left side of the rope and one spot on the right side that he or she wants to use. Furthermore, you know the strength of each contestant.

The organizer has now asked you the following: Given an integer k , is it possible to create two teams, such that each team has n contestants, each contestant uses a spot that he or she wants to use (of course no two contestants share a spot), and the sums of strengths of the two teams differ by at most k ?

Input

The first line of input contains a positive integer n , specifying the number of spots on each side of the rope, and an integer $k \leq 20n$, specifying the maximum difference of teams' strengths. For simplicity, we number the contestants from 1 to $2n$.

Each of the following $2n$ lines describes one contestant: the i -th of these lines contains three positive integers l_i , r_i and s_i ($1 \leq l_i, r_i \leq n$, $1 \leq s_i \leq 20$), which specify that contestant i has strength s_i and wants to use either spot l_i on the left side of the rope or spot r_i on the right side of the rope.

Output

In the first and only line of output your program should write either YES or NO, depending on whether it is possible to create two teams satisfying the requirements stated above.

Examples

For the input data:

```
4 1
1 1 1
2 1 2
2 2 8
1 2 2
3 3 5
3 3 2
4 4 1
4 4 2
```

a correct result is:

```
YES
```

v. 3

Tug of War 1/2

For the input data:

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

a correct result is:

NO

Explanation to the examples: In the first example we can assign contestants 1, 3, 6 and 7 to the left side (which results in a team of strength $1 + 8 + 2 + 1 = 12$) and contestants 2, 4, 5 and 8 to the right side (which results in a team of strength $2 + 2 + 5 + 2 = 11$). The difference of strengths between teams is 1.

In the second example both players of strength 4 have to be in the same team, thus the minimal difference of strengths between teams is 6.

Grading

Subtask	Conditions	Points
1	$n \leq 10$	18
2	$n \leq 2000$	30
3	$n \leq 30\,000, s_i = 1$	23
4	$n \leq 30\,000$	29