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 sidewards 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?

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 `bbbbb`, and the file names as `ccccccccccccc`, `ddddddddddd`, `eeee` and `fffffff`, respectively. The root directory contains the directory `a` and the file `fffffff`; the directory `a` contains the directory `bbbbb` and the file `eeee`; and finally the directory `bbbbb` contains the files `ccccccccccccc` and `ddddddddddd`.

```
/
|-- a
|   |-- bbbbb
|   |   |-- ccccccccccccc
|   |   +-- ddddddddddd
|   +-- eeee
+-- fffffff
```

In the first case, the absolute file path `/a/bbbbb/ccccccccccccc` 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/bbbbb/ddddddddddd`. 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