Functions or Not? ★

Points: 553.1800000000001 Rank: 1150

Problem Submissions Leaderboard

RATE THIS CHALLENGE

★ ★ ★ ★ ★

Objective

In this problem, we touch upon a basic concept that is fundamental to Functional Programming: identifying a relation which represents a valid function.

Task

You are given a set of unique (x, y) ordered pairs constituting a relation. The x-values form the domain, and the y-values form the range to which they map. For each of these relations, identify whether they may possibly represent a valid function or not.

Note: You do not have to find the actual function, you just need to determine that the relation may be representative of some valid function.

Input Format

The first line contains an integer, T, denoting the number of test cases. The subsequent lines describe T test cases, and the input for each test case is as follows:

1. The first line contains an integer, N, the number of (x, y) pairs in the test case.
2. The N subsequent lines each contain two space-separated integers describing the respective x and y values for each ordered pair.

Constraints

- $1 \leq T \leq 5$
- $2 \leq N \leq 100$
- $0 \leq x, y \leq 500$
- x and y are both integers.

Output Format

On a new line for each test case, print `YES` if the set of ordered pairs represent a valid function, or `NO` if they do not.

Sample Input

```
2
3
1 1
2 2
3 3
4
1 2
2 4
3 6
4 8
```

Sample Output

```
YES
YES
```

Explanation

Test Case 0:

$N = 3$, Ordered Pairs: $(1, 1), (2, 2), (3, 3)$ The set of ordered pairs represents a relation, which could represent a function such as $f : N \to N$, $f(x) = x$. Thus, we print YES on a new line.

Test Case 1:

$N = 4$, Ordered Pairs: $(1, 2), (2, 4), (3, 6), (4, 8)$

The set of ordered pairs represents a relation, which could represent a function such as $f : N \to N$, $f(x) = 2x$. Thus, we print YES on a new line.

Change Theme Language Haskell

```
import Control.Monad
import qualified Data.Map as M

isValid :: [(Int, Int)] -> Bool
isValid pairs = and . snd  $ foldr g (M.empty, []) pairs
    where g (a, b) (m, bs) = case M.lookup a m of
                                Just v -> if v == b then (m, True:bs) else (m, False:bs)
                                Nothing -> (M.insert a b m, True:bs)

main = do
    cases <- (read::String->Int) `fmap` getLine
    forM [1..cases] $ const $ do
        lines <- (read::String->Int) `fmap` getLine
        pairs <- forM [1..lines] $ const $ do
            a:b:_ <- (fmap (read::String->Int) . words) `fmap` getLine
            return (a,b)
        putStrLn $ if isValid pairs then "YES" else "NO"
```

Line: 23 Col: 1

Upload Code as File Test against custom input Run Code Submit Code