Problem 1462: Course Schedule IV

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

There are a total of

numCourses

courses you have to take, labeled from

0

to

numCourses - 1

. You are given an array

prerequisites

where

prerequisites[i] = [a

i

, b

i

]

indicates that you

must

take course

a

i

first if you want to take course

b

i

.

For example, the pair

[0, 1]

indicates that you have to take course

0

before you can take course

1

.

Prerequisites can also be

indirect

. If course

a

is a prerequisite of course

b

, and course

b

is a prerequisite of course

c

, then course

a

is a prerequisite of course

c

.

You are also given an array

queries

where

queries[j] = [u

j

, v

j

]

. For the

j

th

query, you should answer whether course

u

j

is a prerequisite of course

v

j

or not.

Return

a boolean array

answer

, where

answer[j]

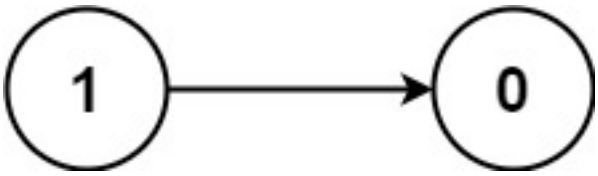
is the answer to the

j

th

query.

Example 1:



Input:

numCourses = 2, prerequisites = [[1,0]], queries = [[0,1],[1,0]]

Output:

[false,true]

Explanation:

The pair [1, 0] indicates that you have to take course 1 before you can take course 0. Course 0 is not a prerequisite of course 1, but the opposite is true.

Example 2:

Input:

numCourses = 2, prerequisites = [], queries = [[1,0],[0,1]]

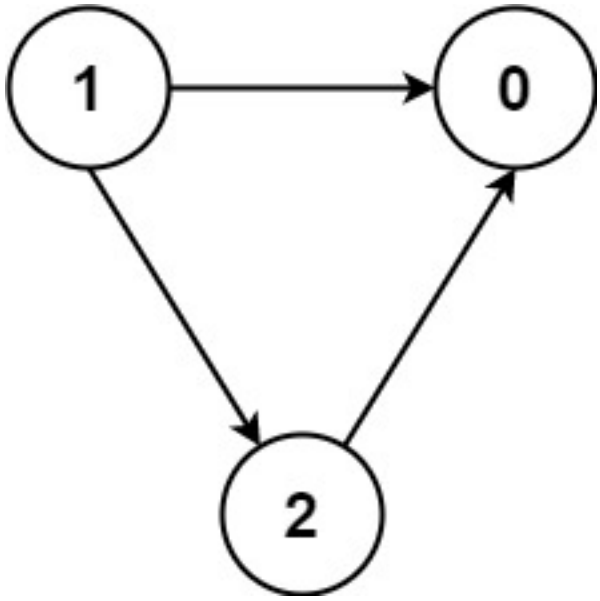
Output:

[false,false]

Explanation:

There are no prerequisites, and each course is independent.

Example 3:



Input:

numCourses = 3, prerequisites = [[1,2],[1,0],[2,0]], queries = [[1,0],[1,2]]

Output:

[true,true]

Constraints:

2 <= numCourses <= 100

0 <= prerequisites.length <= (numCourses * (numCourses - 1) / 2)

prerequisites[i].length == 2

0 <= a

i

, b

i

<= numCourses - 1

a

i

!= b

i

All the pairs

[a

i

, b

i

]

are

unique

.

The prerequisites graph has no cycles.

1 <= queries.length <= 10

4

0 <= u

i

, v

i

<= numCourses - 1

u

i

!= v

i

Code Snippets

C++:

```
class Solution {
public:
vector<bool> checkIfPrerequisite(int numCourses, vector<vector<int>>&
prerequisites, vector<vector<int>>& queries) {

}
};
```

Java:

```
class Solution {
public List<Boolean> checkIfPrerequisite(int numCourses, int[][]
prerequisites, int[][] queries) {

}
}
```

Python3:

```
class Solution:
def checkIfPrerequisite(self, numCourses: int, prerequisites:
List[List[int]], queries: List[List[int]]) -> List[bool]:
```

Python:

```
class Solution(object):
def checkIfPrerequisite(self, numCourses, prerequisites, queries):
"""
:type numCourses: int
:type prerequisites: List[List[int]]
:type queries: List[List[int]]
:rtype: List[bool]
"""
```

JavaScript:

```
/**
* @param {number} numCourses
* @param {number[][]} prerequisites
* @param {number[][]} queries
* @return {boolean[]}
*/
var checkIfPrerequisite = function(numCourses, prerequisites, queries) {

};
```

TypeScript:

```
function checkIfPrerequisite(numCourses: number, prerequisites: number[][],
queries: number[][]): boolean[] {

};
```

C#:

```
public class Solution {
public IList<bool> CheckIfPrerequisite(int numCourses, int[][] prerequisites,
int[][] queries) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
bool* checkIfPrerequisite(int numCourses, int** prerequisites, int
```

```
prerequisitesSize, int* prerequisitesColSize, int** queries, int queriesSize,
int* queriesColSize, int* returnSize) {

}
```

Go:

```
func checkIfPrerequisite(numCourses int, prerequisites [][]int, queries
[][]int) []bool {

}
```

Kotlin:

```
class Solution {
fun checkIfPrerequisite(numCourses: Int, prerequisites: Array<IntArray>,
queries: Array<IntArray>): List<Boolean> {

}
}
```

Swift:

```
class Solution {
func checkIfPrerequisite(_ numCourses: Int, _ prerequisites: [[Int]], _
queries: [[Int]]) -> [Bool] {

}
}
```

Rust:

```
impl Solution {
pub fn check_if_prerequisite(num_courses: i32, prerequisites: Vec<Vec<i32>>,
queries: Vec<Vec<i32>>) -> Vec<bool> {

}
}
```

Ruby:

```
# @param {Integer} num_courses
# @param {Integer[][]} prerequisites
```

```
# @param {Integer[][]} queries
# @return {Boolean[]}
def check_if_prerequisite(num_courses, prerequisites, queries)

end
```

PHP:

```
class Solution {

/**
* @param Integer $numCourses
* @param Integer[][] $prerequisites
* @param Integer[][] $queries
* @return Boolean[]
*/
function checkIfPrerequisite($numCourses, $prerequisites, $queries) {

}
}
```

Dart:

```
class Solution {
List<bool> checkIfPrerequisite(int numCourses, List<List<int>> prerequisites,
List<List<int>> queries) {

}
}
```

Scala:

```
object Solution {
def checkIfPrerequisite(numCourses: Int, prerequisites: Array[Array[Int]],
queries: Array[Array[Int]]): List[Boolean] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec check_if_prerequisite(num_courses :: integer, prerequisites ::
[[integer]], queries :: [[integer]]) :: [boolean]
def check_if_prerequisite(num_courses, prerequisites, queries) do

end
end
```

Erlang:

```
-spec check_if_prerequisite(NumCourses :: integer(), Prerequisites ::
[[integer()]], Queries :: [[integer()]]) -> [boolean()].
check_if_prerequisite(NumCourses, Prerequisites, Queries) ->
.
```

Racket:

```
(define/contract (check-if-prerequisite numCourses prerequisites queries)
(-> exact-integer? (listof (listof exact-integer?)) (listof (listof
exact-integer?)) (listof boolean?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Course Schedule IV
* Difficulty: Medium
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
vector<bool> checkIfPrerequisite(int numCourses, vector<vector<int>>&
prerequisites, vector<vector<int>>& queries) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Course Schedule IV
* Difficulty: Medium
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public List<Boolean> checkIfPrerequisite(int numCourses, int[][]
prerequisites, int[][] queries) {

}
}
```

Python3 Solution:

```
"""
Problem: Course Schedule IV
Difficulty: Medium
Tags: array, graph, sort, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def checkIfPrerequisite(self, numCourses: int, prerequisites:
List[List[int]], queries: List[List[int]]) -> List[bool]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def checkIfPrerequisite(self, numCourses, prerequisites, queries):
"""
:type numCourses: int
:type prerequisites: List[List[int]]
:type queries: List[List[int]]
:rtype: List[bool]
"""
```

JavaScript Solution:

```
/**
* Problem: Course Schedule IV
* Difficulty: Medium
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} numCourses
* @param {number[][]} prerequisites
* @param {number[][]} queries
* @return {boolean[]}
*/
var checkIfPrerequisite = function(numCourses, prerequisites, queries) {

};
```

TypeScript Solution:

```
/**
* Problem: Course Schedule IV
* Difficulty: Medium
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
function checkIfPrerequisite(numCourses: number, prerequisites: number[][],
queries: number[][]): boolean[] {

};
```

C# Solution:

```
/*
* Problem: Course Schedule IV
* Difficulty: Medium
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public IList<bool> CheckIfPrerequisite(int numCourses, int[][] prerequisites,
int[][] queries) {

}
}
```

C Solution:

```
/*
* Problem: Course Schedule IV
* Difficulty: Medium
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
bool* checkIfPrerequisite(int numCourses, int** prerequisites, int
```

```
prerequisitesSize, int* prerequisitesColSize, int** queries, int queriesSize,
int* queriesColSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Course Schedule IV
// Difficulty: Medium
// Tags: array, graph, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func checkIfPrerequisite(numCourses int, prerequisites [][]int, queries
[][]int) []bool {

}
```

Kotlin Solution:

```
class Solution {
fun checkIfPrerequisite(numCourses: Int, prerequisites: Array<IntArray>,
queries: Array<IntArray>): List<Boolean> {

}
}
```

Swift Solution:

```
class Solution {
func checkIfPrerequisite(_ numCourses: Int, _ prerequisites: [[Int]], _
queries: [[Int]]) -> [Bool] {

}
}
```

Rust Solution:

```
// Problem: Course Schedule IV
// Difficulty: Medium
```

```
// Tags: array, graph, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn check_if_prerequisite(num_courses: i32, prerequisites: Vec<Vec<i32>>,
queries: Vec<Vec<i32>>) -> Vec<bool> {

}
}
```

Ruby Solution:

```
# @param {Integer} num_courses
# @param {Integer[][]} prerequisites
# @param {Integer[][]} queries
# @return {Boolean[]}
def check_if_prerequisite(num_courses, prerequisites, queries)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $numCourses
* @param Integer[][] $prerequisites
* @param Integer[][] $queries
* @return Boolean[]
*/
function checkIfPrerequisite($numCourses, $prerequisites, $queries) {

}
}
```

Dart Solution:

```
class Solution {
List<bool> checkIfPrerequisite(int numCourses, List<List<int>> prerequisites,
List<List<int>> queries) {

}
}
```

Scala Solution:

```
object Solution {
def checkIfPrerequisite(numCourses: Int, prerequisites: Array[Array[Int]],
queries: Array[Array[Int]]): List[Boolean] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec check_if_prerequisite(num_courses :: integer, prerequisites ::
[[integer]], queries :: [[integer]]) :: [boolean]
def check_if_prerequisite(num_courses, prerequisites, queries) do

end
end
```

Erlang Solution:

```
-spec check_if_prerequisite(NumCourses :: integer(), Prerequisites ::
[[integer()]], Queries :: [[integer()]]) -> [boolean()].
check_if_prerequisite(NumCourses, Prerequisites, Queries) ->
.
```

Racket Solution:

```
(define/contract (check-if-prerequisite numCourses prerequisites queries)
(-> exact-integer? (listof (listof exact-integer?)) (listof (listof
exact-integer?)) (listof boolean?))
)
```