1136. Parallel Courses

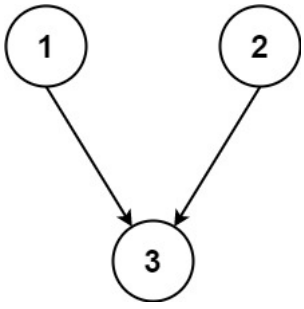
Medium 863 22 Add to List Share

You are given an integer `n`, which indicates that there are `n` courses labeled from `1` to `n`. You are also given an array `relations` where `relations[i] = [prevCourse_i, nextCourse_i]`, representing a prerequisite relationship between course `prevCourse_i` and course `nextCourse_i`: course `prevCourse_i` has to be taken before course `nextCourse_i`.

In one semester, you can take **any number** of courses as long as you have taken all the prerequisites in the **previous** semester for the courses you are taking.

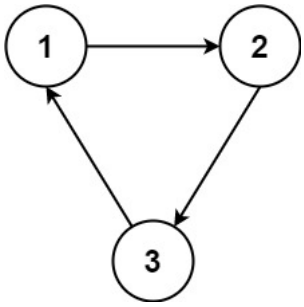
Return *the **minimum** number of semesters needed to take all courses*. If there is no way to take all the courses, return `-1`.

Example 1:




```
Input: n = 3, relations = [[1,3],[2,3]]
Output: 2
Explanation: The figure above represents the given graph.
In the first semester, you can take courses 1 and 2.
In the second semester, you can take course 3.
```

Example 2:




```
Input: n = 3, relations = [[1,2],[2,3],[3,1]]
Output: -1
Explanation: No course can be studied because they are
prerequisites of each other.
```

Constraints:

- `1 <= n <= 5000`
- `1 <= relations.length <= 5000`
- `relations[i].length == 2`
- `1 <= prevCourse_i, nextCourse_i <= n`
- `prevCourse_i != nextCourse_i`

C++ Autocomplete

```cpp
// we have to store the results beause dfs solution won't
always guarantee the longest path
//sometimes it may give correct ans and sometimes wrong ans
also .. it can give worng ans as
// the path from where i am going i have already visited by
some other node resulting in me not
// getting the correct answere so thats why we are storing
result from every node to get the
//greatest value from that path

class Solution {
public:
    bool dfs(int parent,vector<int>adj[],vector<int>
&visited,vector<bool> &isinstack,int &maxLen)
    {
        visited[parent] = 1;
        isinstack[parent] = true;

        for(int children : adj[parent])
        {
            if(!visited[children])
            {

 if(dfs(children,adj,visited,isinstack,maxLen))
                    return true;
            }
            else if(isinstack[children])
                return true;
            visited[parent] =
max(visited[parent],visited[children]+1);
        }
        maxLen = max(maxLen,visited[parent]);

        isinstack[parent] = false;
        return false;
    }

    int minimumSemesters(int n, vector<vector<int>>&
relations) {

        vector<int> adj[n+1];
```

NEW

Your previous code was restored from your local storage. Reset to default

Testcase Run Code Result Debugger

Accepted Runtime: 2 ms

Your input
2
[[1,2]]

stdout
1
2

Output 2 Diff

Expected 2

Console... Use Example Testcases Run Code Submit