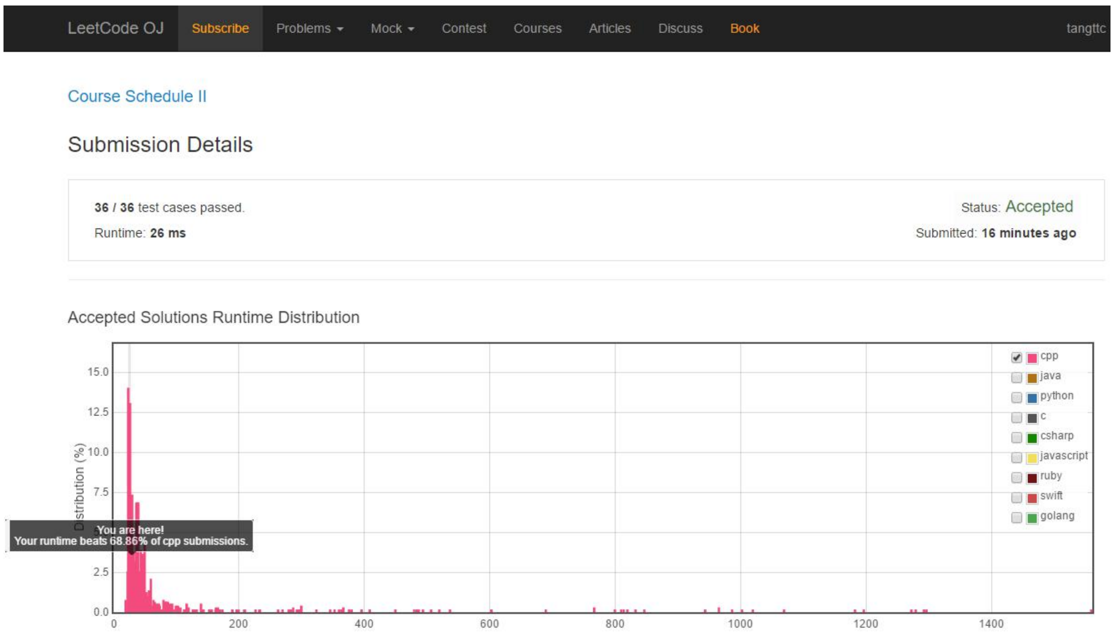
PA5 Course Schedule II 唐天成 515030910287



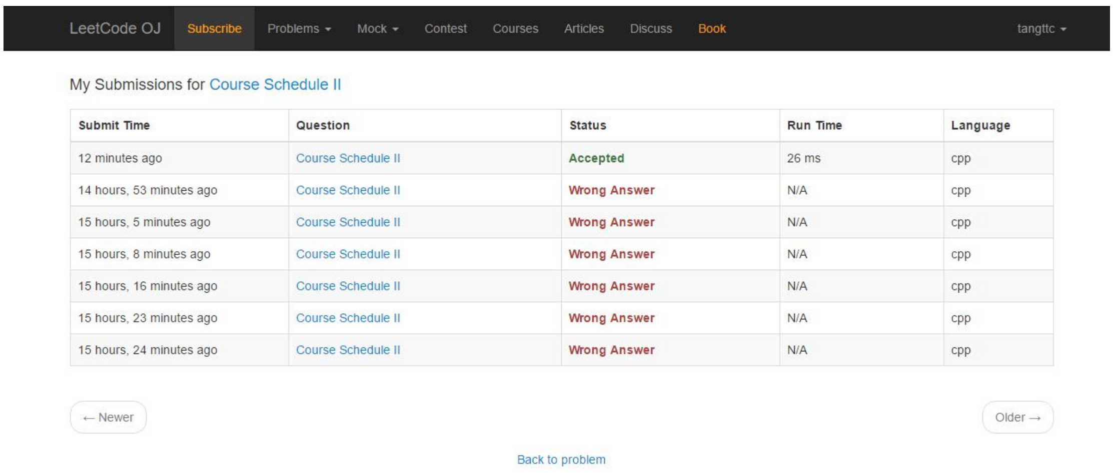
Course Schedule II

Submission Details

36 / 36 test cases passed.
Runtime: **26 ms**

Status: Accepted
Submitted: **16 minutes ago**

Accepted Solutions Runtime Distribution



My Submissions for Course Schedule II

Submit Time	Question	Status	Run Time	Language
12 minutes ago	Course Schedule II	Accepted	26 ms	cpp
14 hours, 53 minutes ago	Course Schedule II	Wrong Answer	N/A	cpp
15 hours, 5 minutes ago	Course Schedule II	Wrong Answer	N/A	cpp
15 hours, 8 minutes ago	Course Schedule II	Wrong Answer	N/A	cpp
15 hours, 16 minutes ago	Course Schedule II	Wrong Answer	N/A	cpp
15 hours, 23 minutes ago	Course Schedule II	Wrong Answer	N/A	cpp
15 hours, 24 minutes ago	Course Schedule II	Wrong Answer	N/A	cpp

← Newer Older →

Back to problem

Codes(C++):

```
#include <iostream>
#include <vector>
#include <list>
using namespace std;

class Solution {
public:
    void init_tree(int numCourses, vector<pair<int, int>>& prerequisites, vector<list<int>>& tree)
{
        for (int i = 0; i < numCourses; i++) {
            list<int> newtree;
            tree.push_back(newtree);
        }
        int total_size = prerequisites.size();
        for (int i = 0; i < total_size; i++) {
            tree[prerequisites[i].second].push_back(prerequisites[i].first);
        }
    }

    void init_vector(vector<int>& vector,int num) {
        for (int i = 0; i < num; i++) {
            vector.push_back(0); // 0 represent not visited
        }
    }

    void DFS(int numCourses, vector<list<int>>& tree, vector<int>& mark,int& possible,
list<int>& to_result, vector<int>& back_edge,int& postdfn) {
        for (int i = 0; i < numCourses; i++) {
            if (mark[i] == 0) {
                dfs(i,-1,tree, mark, possible,to_result, back_edge, postdfn);
                if (possible == 0) {
                    break;
                }
            }
        }
    }

    void dfs(int v, int parent,vector<list<int>>& tree, vector<int>& mark,int& possible, list<int>&
to_result, vector<int>& back_edge, int& postdfn) {
        mark[v] = 1;
        for (list<int>::iterator it = tree[v].begin(); it != tree[v].end(); ++it) {
            if (mark[*it] == 0) {
                dfs(*it,v, tree, mark,possible,to_result, back_edge,postdfn);
            }
            else if (back_edge[*it] == 0) {
                possible = 0;
                break;
            }
        }
        to_result.push_front(v);
        postdfn++;
        back_edge[v] = postdfn;
    }

    void list_to_vector(list<int>& to_result, vector<int>& result) {
        for (list<int>::iterator it = to_result.begin(); it != to_result.end(); ++it) {
            result.push_back(*it);
        }
    }

    vector<int> findOrder(int numCourses, vector<pair<int, int>>& prerequisites) {
        int possible = 1;
        vector<list<int>> tree;
        vector<int> mark;    // visited --- 1 ; not visited --- 0
        vector<int> back_edge; // store postdfn
        list<int> to_result;
        vector<int> result;
        init_tree(numCourses, prerequisites, tree); // put the pair into the tree
        init_vector(mark,numCourses);
        init_vector(back_edge, numCourses);
        int postdfn = 0;
        DFS(numCourses, tree, mark,possible,to_result,back_edge,postdfn); // start dfs
        list_to_vector(to_result,result);
        if (possible == 0) {
            vector<int> empty;
            return empty;
        }
        else {
            return result;
        }
    }
};
```