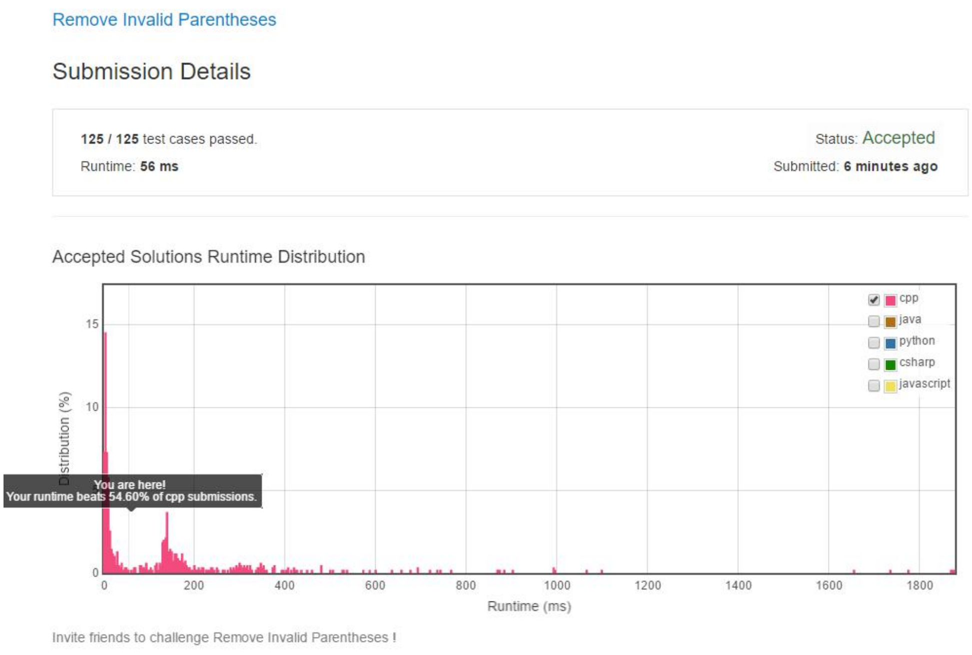
PA6 唐天成 515030910287

Leetcode 301: Remove Invalid Parentheses



Remove Invalid Parentheses

Submission Details

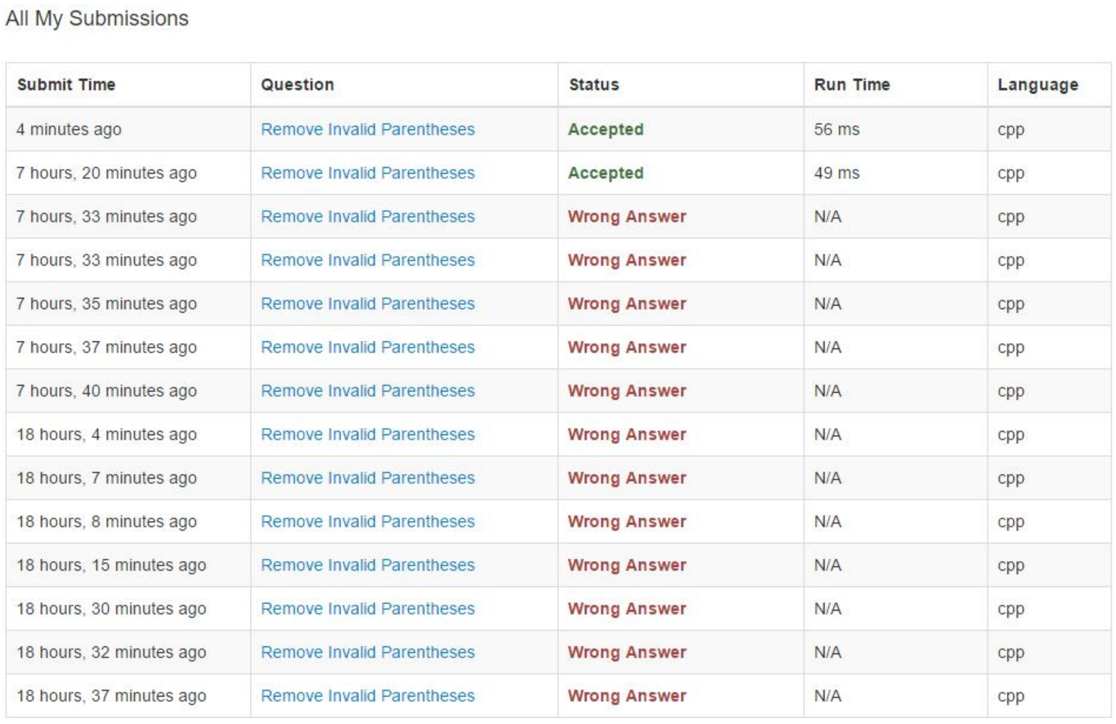
125 / 125 test cases passed. Status: Accepted

Runtime: **56 ms** Submitted: **6 minutes ago**

Accepted Solutions Runtime Distribution

You are here!
Your runtime beats 54.60% of cpp submissions.

Invite friends to challenge Remove Invalid Parentheses !



All My Submissions

Submit Time	Question	Status	Run Time	Language
4 minutes ago	Remove Invalid Parentheses	Accepted	56 ms	cpp
7 hours, 20 minutes ago	Remove Invalid Parentheses	Accepted	49 ms	cpp
7 hours, 33 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
7 hours, 33 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
7 hours, 35 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
7 hours, 37 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
7 hours, 40 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
18 hours, 4 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
18 hours, 7 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
18 hours, 8 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
18 hours, 15 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
18 hours, 30 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
18 hours, 32 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp
18 hours, 37 minutes ago	Remove Invalid Parentheses	Wrong Answer	N/A	cpp

Code(C++):

```
#include <iostream>
#include <vector>
#include <string>
#include <queue>
```

```
#include <stack>
#include <set>
using namespace std;

class Solution {
public:
	int who_has_more(string s) { // return 1 if left has more parentheses,if right has return 2,if
equal return 0
		int size = s.size();
		int left = 0;
		int right = 0;
		for (int i = 0; i < size; i++) {
			if (s[i] == '(') {
				++left;
			}
			else if (s[i] == ')') {
				++right;
			}
		}
		if (left == right) {
			return 0;
		}
		else if (left > right) {
			return 1;
		}
		else {
			return 2;
		}
	}

	bool is_valid(string s) { // valid ----return true;else ----return false
		stack<char> left;
		int size = s.size();
		if (size == 0) {
			return true;
		}
		if (s[0] == ')') {
			return false;
		}
		if (s[0] == '(') {
			left.push(s[0]);
		}
		for (int i = 1; i < size; i++) {
			if (s[i] == '(') {
```

```
                left.push(s[i]);
            }
            else if (s[i] == ')') {
                if (!left.empty()) {
                    left.pop();
                }
                else {
                    return false;
                }
            }
        }
        if (left.empty()) {
            return true;
        }
        else {
            return false;
        }
    }

    vector<string> removeInvalidParentheses(string s) {
        char par[3];
        int done = 0;
        set<string> delete_same;
        par[1] = '(';
        par[2] = ')';
        queue<string> all_possible;
        queue<string> next_possible;
        vector<string> result;
        vector<string> temp;
        all_possible.push(s);

        while (!all_possible.empty()) {
            string test = all_possible.front();
            all_possible.pop();
            if (is_valid(test)) {
                result.push_back(test);
                done = 1;
            }
            else if (done == 0) {
                int left_or_right = who_has_more(test);
                if (left_or_right == 0) {
                    left_or_right = 1;
                }
                char del_par = par[left_or_right];
                int size = test.size();
                for (int i = 0; i < size; i++) { // delete one char which is the extra parenthese
                    if (test[i] == del_par) {
                        string add = test.substr(0, i) + test.substr(i + 1);
                        if (delete_same.count(add) == 0) {
                            next_possible.push(add);
                            delete_same.insert(add);
                        }
                    }
                }
            }

                if (all_possible.empty() && done == 0) {
                    while (!next_possible.empty()) {
                        all_possible.push(next_possible.front());
                        next_possible.pop();
                    }
                }

        }

        if (result.empty()) {
            int size = temp.size();
            for (int i = 0; i < size; i++) {
                result.push_back(temp[i]);
            }
        }
        if (result.empty()) {
            result.push_back("");
        }


        return result;
    }
};
```