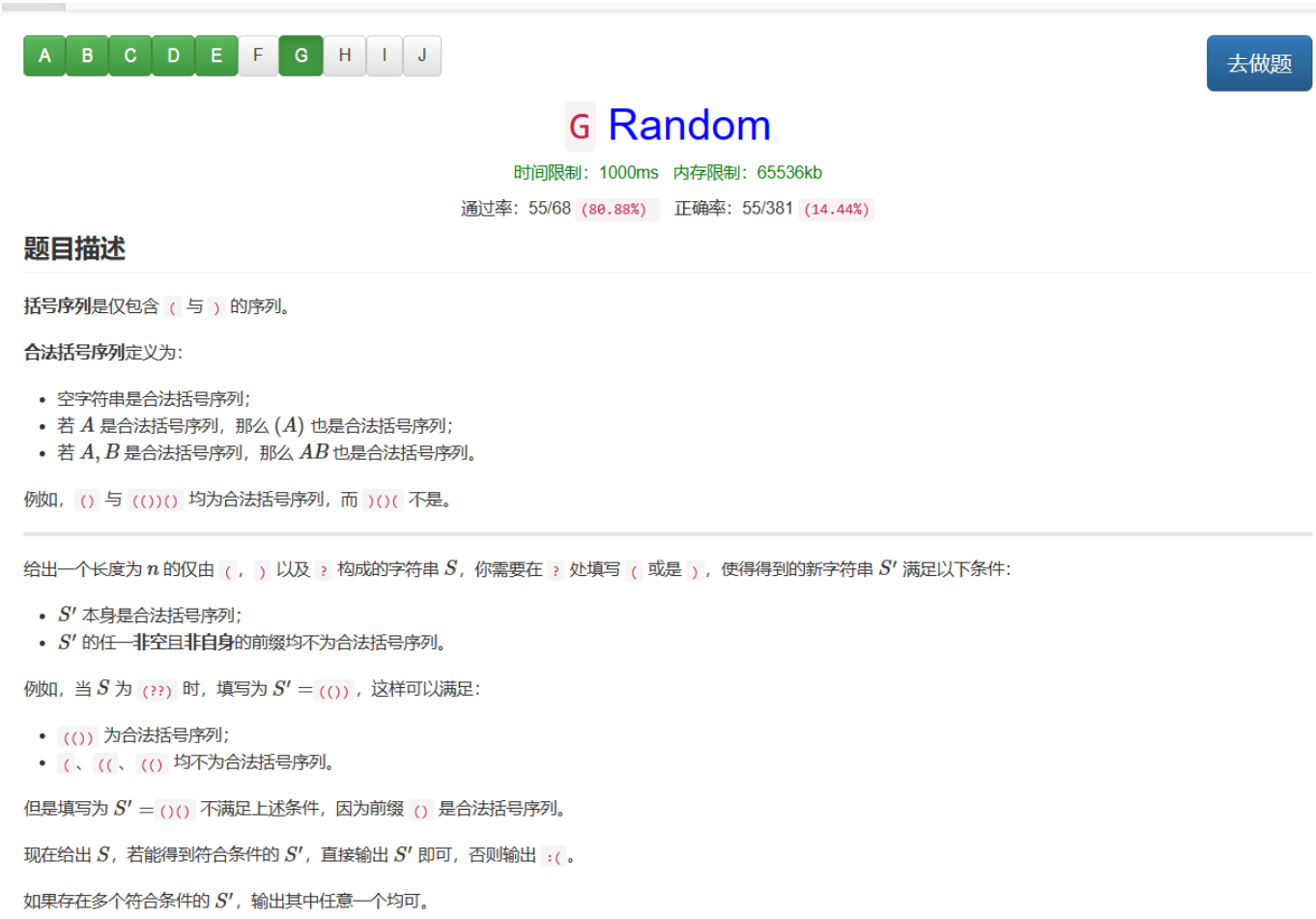
题目



A B C D E F G H I J

去做题

G Random

时间限制：1000ms 内存限制：65536kb

通过率：55/68 (80.88%) 正确率：55/381 (14.44%)

题目描述

括号序列是仅包含 `(` 与 `)` 的序列。

合法括号序列定义为：

- 空字符串是合法括号序列；
- 若 A 是合法括号序列，那么 (A) 也是合法括号序列；
- 若 A, B 是合法括号序列，那么 AB 也是合法括号序列。

例如，`()` 与 `(()())` 均为合法括号序列，而 `)()(` 不是。

给出一个长度为 n 的仅由 `(`，`)` 以及 `?` 构成的字符串 S，你需要在 `?` 处填写 `(` 或是 `)`，使得得到的新字符串 S' 满足以下条件：

- S' 本身是合法括号序列；
- S' 的任一**非空且非自身**的前缀均不为合法括号序列。

例如，当 S 为 `(??)` 时，填写为 $S' =$ `(())`，这样可以满足：

- `(())` 为合法括号序列；
- `(`、`((`、`(()` 均不为合法括号序列。

但是填写为 $S' =$ `()()` 不满足上述条件，因为前缀 `()` 是合法括号序列。

现在给出 S，若能得到符合条件的 S'，直接输出 S' 即可，否则输出 `:(`。

如果存在多个符合条件的 S'，输出其中任意一个均可。

思路

贪心思路

从左到右遍历输入字符串 s。 如果遇到左括号 (，就将其添加到答案序列 ans 中，并增加左括号计数 leftCount。 如果遇到右括号)，就将其添加到答案序列 ans 中，并增加右括号计数 rightCount。 如果遇到问号 ?，贪心地选择将其替换为左括号或右括号： 如果左括号的数量 leftCount 小于总括号数量的一半，就将问号替换为左括号，并增加 leftCount。 否则，将问号替换为右括号，并增加 rightCount。 在每一步中，贪心地根据当前情况选择左括号或右括号或问号的替代项。 代码的贪心思想在于尽可能匹配括号，确保左右括号的数量尽量平衡，同时利用问号来填补空缺。如果成功构建出有效的括号序列，那么左右括号的数量将完全匹配，且在遍历整个字符串后，所有字符都已经处理。如果不能成功构建有效的括号序列，代码会及早停止，输出 "😞"。

代码实现

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

```
int main() {
    int n;
    cin >> n;
    string s;
    cin >> s;

    vector<char> ans(n, ' '); // 初始化一个大小为n的字符数组，用于存储答案

    int leftCount = 0; // 统计左括号的数量
    int rightCount = 0; // 统计右括号的数量

    // 检查特殊情况
    if (n % 2 != 0 || s[0] == ')' || s[n - 1] == '(') {
        cout << ":(" << endl;
        return 0;
    }
    for (int i = 0; i < n; i++) {
        if (s[i] == '(') {
            ans[i] = '(';
            leftCount++;
        } else if (s[i] == ')') {
            ans[i] = ')';
            rightCount++;
        }


    }
    int i;
    // 遍历输入字符串s
    int cl = 0,cr = 0;
    for (i = 0; i < n; i++) {
        if (s[i] == '(') {

            cl++;
        } else if (s[i] == ')') {
            cr ++;
            if(cr >= cl){
                    break;
            }
        }

        // 当遇到'?'时，优先将其替换为左括号
        if (s[i] == '?') {
            if (leftCount < n / 2 ) {
                ans[i] = '(';
                leftCount++;
                cl ++;
            } else {
                ans[i] = ')';
                rightCount++;
                cr ++;
                if(cr >= cl){
                    break;
```

```
                }
            }
        }
    }

    // 检查左右括号的数量是否匹配
    if (leftCount == rightCount && i == n - 1) {
        for (int i = 0; i < n; i++) {
            cout << ans[i];
        }
        cout << endl;
    } else {
        cout << ":(" << endl;
    }

    return 0;
}
```

要注意的细节

必须先遍历一遍，得到左括号的数量，否则有些情况无法正确得出答案