1355：字符串匹配问题(strs)

题目描述

字符串中只含有括号 (),[],<>,{},判断输入的字符串中括号是否匹配。如果括号有互相包含的形式，从内到外必须是<>,(),[],{}，例如。输入：[()] 输出:YES，而输入([])，([)]都应该输出NO。

输入格式

第一行为一个整数n，表示以下有多少个由括好组成的字符串。接下来的n行，每行都是一个由括号组成的长度不超过255的字符串。

输出格式

在输出文件中有n行，每行都是YES或NO。

输入样例

```
5
{}{}<><>()()[][]
{{}}{{}}<<>><<>>(())(())[[]][[]]
{{}}{{}}<<>><<>>(())(())[[]][[]]
{<>}{[]}<<<>><<>>>((<>))(())[[(<>)]][[]]
><}{{[]}<<<>><<>>>((<>))(())[[(<>)]][[]]
```

输出样例

```
YES
YES
YES
YES
NO
```

解析

本题相比较前面的几道题就有了一定难度，主要是括号的嵌套有了顺序的要求。所以，当我们读入一个括号符号时，除了要判断它是否成对外，还要判断它的顺序是否正确。

那么，如何判断顺序呢，只需要在向栈中存入数据的时候判断栈顶是否存在不应该存在的数据即可。

编码

```
#include <bits/stdc++.h>
using namespace std;
int num;
string p;
stack<char> s;
//根据输入的括号，返回对应的前括号
char trans(char a) {
    if (a == ')') {
        return '(';
    }
    if (a == ']') {
        return '[';
    }
    if (a == '}') {
        return '{';
    }
    if (a == '>') {
        return '<';
    }
    return '\0';
}
//判断当前的符号能否入栈
bool getOrder(char a) {
    bool res = false;
    switch (a) {
        //随便入栈
      case '<':
            res = true;
            break;
        case '(':
            if (s.top() != '<') {
                res = true;
            }
            break;
        case '[':
            if (s.top() != '<' && s.top() != '(') {
                res = true;
            }
```

```
            break;
        case '{':
            if (s.top() != '<' && s.top() != '(' && s.top() != '[')
                res = true;
            }
            break;
        default:
            //右括号都可以入栈
          res = true;
            break;
    }
    return res;
}
int main() {
    cin >> num;
    //为了在后面使用getline，在这里必须处理回车符号
    getchar();
    for (int i = 0; i < num; ++i) {
        //清空数据
       while (!s.empty()) {
            s.pop();
        }
        getline(cin, p);
        for (int j = 0; j < p.size(); ++j) {
            //栈空间为空，直接将字符串存入
          if (s.empty()) {
                s.push(p[j]);
                continue;
            }
            //判断一下当前是否可以入栈
          bool order = getOrder(p[j]);
            if (!order) {
                break;
            }
            //匹配上了
          if (trans(p[j]) == s.top()) {
                s.pop();
            }
                //没有匹配成功，将数据存入
          else {
                s.push(p[j]);
            }
        }
        //判断栈中是否有残留的括号
       if (s.empty()) {
            cout << "YES" << endl;
        } else {
            cout << "NO" << endl;
```

```
            }
        }
        return 0;
}
```