Valid（有效的） Parentheses（括号）

Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

Open brackets must be closed by the same type of brackets.

Open brackets must be closed in the correct order.

Note that an empty string is also considered valid.

------------------------------------------------------------------------------

Java 解决方法：

public boolean isValid(String s) {

Stack<Character> stack = new Stack<Character>();

for (char c : s.toCharArray()) {

if (c == '(')

stack.push(')');

else if (c == '{')

stack.push('}');

else if (c == '[')

stack.push(']');

else if (stack.isEmpty() || stack.pop() != c)

return false;

}

return stack.isEmpty();

}

C++解决方法：

bool isValid(string s) {

stack p;

int l=s.size();

for(int i=0;i<l;i++)

{

if(s[i]=='[' || s[i]=='{' || s[i]=='(')

{

p.push(s[i]);

continue;

}

else if(s[i]=='}' || s[i]==']' || s[i]==')')

{

if(p.empty())

{

return false;

break;

}

else if(p.top()=='{'&&s[i]=='}'

|| p.top()=='['&&s[i]==']'

|| p.top()=='('&&s[i]==')'){

p.pop();

continue;

}

else

{

return false;

}

}

}

return p.empty()?true:false;

}

Repetitive code but I guess this is clean, and easy to understand. This solution also accepts (and ignores) any characters other than parenthesis in the string. Hence, it can be used to check if the parenthesis matches in an equation for example.

#include <stack>

class Solution {

public:

bool isValid(string s) {

stack<char> paren;

for (char& c : s) {

switch (c) {

case '(':

case '{':

case '[': paren.push(c); break;

case ')': if (paren.empty() || paren.top()!='(') return false; else paren.pop();

break;

case '}': if (paren.empty() || paren.top()!='{') return false; else paren.pop();

break;

case ']': if (paren.empty() || paren.top()!='[')return false;else paren.pop();

break;

default: ; // pass

}

}

return paren.empty() ;

}

};