**难度: 简单**

给定一个只包括 '('，')'，'{'，'}'，'['，']' 的字符串，判断字符串是否有效。

有效字符串需满足：

左括号必须用相同类型的右括号闭合。

左括号必须以正确的顺序闭合。

注意空字符串可被认为是有效字符串。

示例 1:

输入: "()"

输出: true

示例 2:

输入: "()[]{}"

输出: true

示例 3:

输入: "(]"

输出: false

示例 4:

输入: "([)]"

输出: false

示例 5:

输入: "{[]}"

输出: true

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/valid-parentheses

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路(用栈 + for + if else):**

1). 遇到(、 { 、[、就入栈

2). 遇到 ) 、 }、 ]就退栈,并且判断是否是对应的(、 { 、[、,如果不是就返回false

3). 还需要判断当遇到 ) 、 }、 ]时,栈不为空,如果不空,直接返回false;

4). 最后判断栈是否为空s

|  |
| --- |
| public boolean isValid(String s) {  Deque<Character> stack = new ArrayDeque<Character>();  char[] chars = s.toCharArray();  for (int i = 0; i < chars.length; i++) {  if (chars[i] == '(' || chars[i] == '{' || chars[i] == '[') {  stack.push(chars[i]);  continue;  }  if ( (chars[i] == ')' || chars[i] == '}' || chars[i] == ']' ) && (stack.size() == 0)) {  return false;  }  if (chars[i] == ')') {  if (! (stack.pop() == '(')) {  return false;  }  }else if (chars[i] == '}') {  if (! (stack.pop() == '{')) {  return false;  }  }else {  if (! (stack.pop() == '[')) {  return false;  }  }  }  return stack.size() == 0;  } |

**思路(用栈 + for + witch):**

1). 遇到(、 { 、[、就入栈

2). 遇到 ) 、 }、 ]就退栈,并且判断是否是对应的(、 { 、[、,如果不是就返回false

3). 还需要判断当遇到 ) 、 }、 ]时,栈不为空,如果不空,直接返回false;

4). 最后判断栈是否为空s

|  |
| --- |
| public boolean isValid2(String s) {  Deque<Character> stack = new ArrayDeque<Character>();  char[] chars = s.toCharArray();  for (int i = 0; i < chars.length; i++) {  switch (chars[i]) {  case '(' :  case '{' :  case '[' :  stack.push(chars[i]);  break;  case ')' :  if (stack.size() == 0 || !(stack.pop() == '(')) {  return false;  }  break;  case '}' :  if (stack.size() == 0 || !(stack.pop() == '{')) {  return false;  }  break;  case ']' :  if (stack.size() == 0 || !(stack.pop() == '[')) {  return false;  }  break;  }  }  return stack.size() == 0;  } |