数据结构与算法（Python版）

栈的应用：简单括号匹配

陈斌 北京大学 gischen@pku.edu.cn

栈的应用：简单括号匹配

- 我们都写过这样的表达式：

  (5+6)*(7+8)/(4+3)

  这里的括号是用来指定表达式项的计算优先级

- 有些函数式语言，如Lisp，在函数定义的时候会用到**大量**的括号

  比如：(defun square(n)

  (* n n))

  这个语句定义了一个计算平方值的函数

栈的应用：简单括号匹配

- 当然，括号的使用必须遵循 “平衡”规则

  首先，每个开括号要恰好对应一个闭括号；

  其次，每对开闭括号要正确的嵌套

  正确的括号：(()()()())，((((()))))，
  (()((())()))

  错误的括号：((((((())，()))，(()()(()

- 对括号是否正确匹配的识别，是很多语言编译器的基础算法

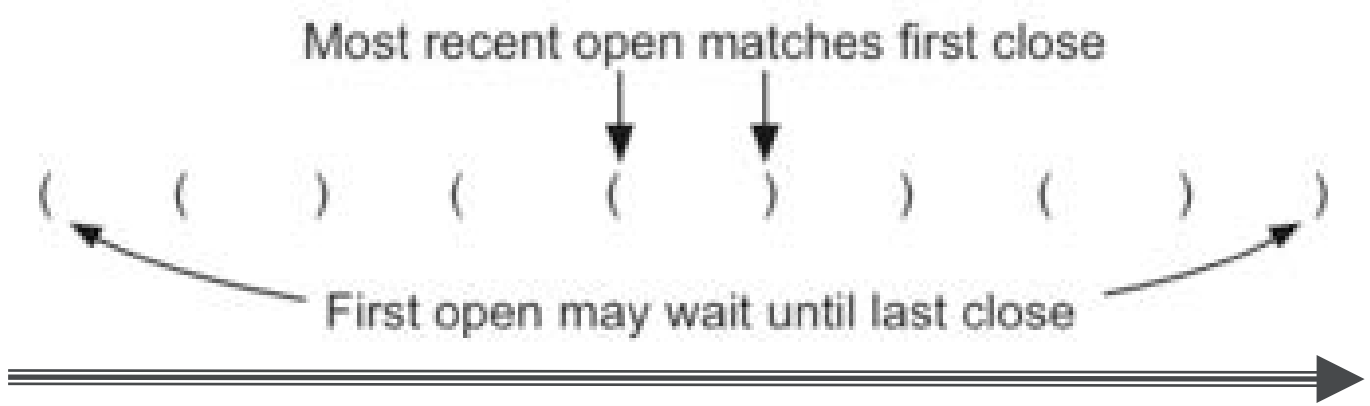
栈的应用：简单括号匹配

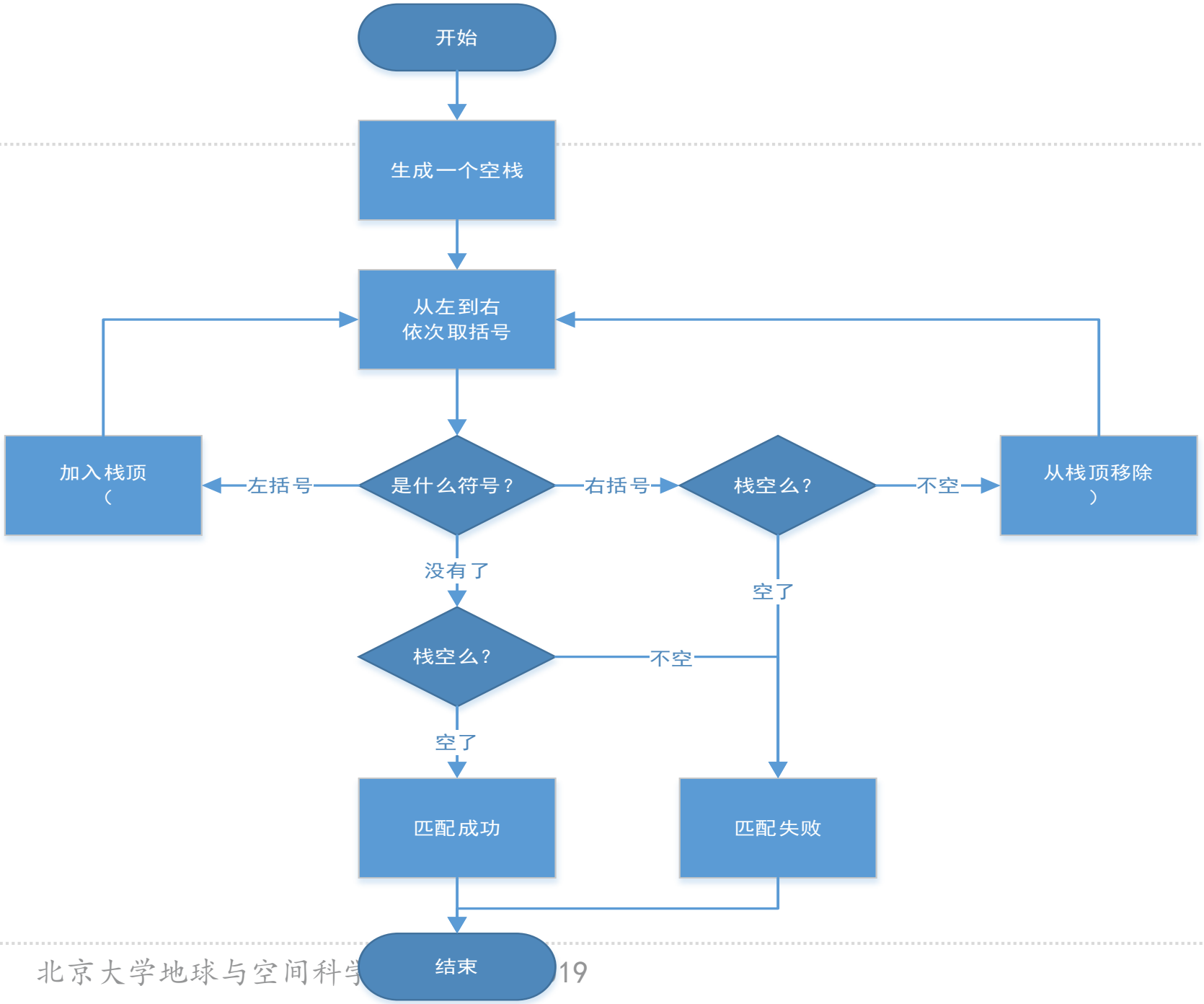
- 下面看看如何构造括号匹配识别算法

从左到右扫描括号串，最新打开的左括号，应该匹配最先遇到的右括号

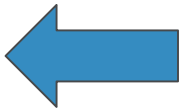
这样，第一个左括号（最早打开），就应该匹配最后一个右括号（最后遇到）

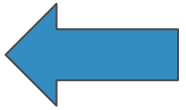
这种次序反转的识别，正好符合栈的特性！




开始
生成一个空栈
从左到右
依次取括号
是什么符号？
左括号
加入栈顶
（
右括号
栈空么？
不空
从栈顶移除
）
没有了
空了
栈空么？
不空
空了
匹配成功
匹配失败
结束

```
from pythonds.basic.stack import Stack

def parChecker(symbolString):
    s = Stack()
    balanced = True
    index = 0
    while index < len(symbolString) and balanced:
        symbol = symbolString[index]
        if symbol == "(":
            s.push(symbol)
        else:
            if s.isEmpty():
                balanced = False
            else:
                s.pop()

        index = index + 1

    if balanced and s.isEmpty():
        return True
    else:
        return False

print(parChecker('((()))'))
print(parChecker('(()'))
```





更多种括号的匹配

- 在实际的应用里，我们会碰到更多种括号
  - 如python中列表所用的方括号“[]”
  - 字典所用的花括号“{}”
  - 元组和表达式所用的圆括号“()”
- 这些不同的括号有可能混合在一起使用，
- 因此就要注意各自的开闭匹配情况

更多种括号的匹配

- 下面这些是匹配的

```
{ { ( [ ] [ ] ) } ( ) }
[ [ { { ( ( ) ) } } ] ]
[ ] [ ] [ ] ( ) { }
```

- 下面这些是不匹配的

```
( [ ) ]                    ( ( ( ) ] ) )
[ { ( ) ]
```

通用括号匹配算法：代码

- 需要修改的地方

碰到各种左括号仍然入栈

碰到各种右括号的时候需要判断栈顶的左括号是否跟之同一种类

```
def parChecker(symbolString):
    s = Stack()
    balanced = True
    index = 0
    while index < len(symbolString) and balanced:
        symbol = symbolString[index]
        if symbol == "(":
            s.push(symbol)
        else:
            if s.isEmpty():
                balanced = False
            else:
                s.pop()

        index = index + 1

    if balanced and s.isEmpty():
        return True
    else:
        return False
```

```
from pythonds.basic.stack import Stack

def parChecker(symbolString):
    s = Stack()
    balanced = True
    index = 0
    while index < len(symbolString) and balanced:
        symbol = symbolString[index]
        if symbol in "([{":
            s.push(symbol)
        else:
            if s.isEmpty():
                balanced = False
            else:
                top = s.pop()
                if not matches(top,symbol):
                    balanced = False
        index = index + 1
    if balanced and s.isEmpty():
        return True
    else:
        return False

def matches(open,close):
    opens = "([{"
    closers = ")]}"
    return opens.index(open) == closers.index(close)


print(parChecker('{{([][])}()}'))
print(parChecker('[{()]'))
```

通用括号匹配算法

- HTML/XML文档也有类似于括号的开闭标记，这种层次结构化文档的校验、操作也可以通过栈来实现

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Hello World</title>
6  </head>
7  <body>
8  <h1>Hello!</h1>
9  </body>
10 </html>
```