

Lesson 04

时间复杂度

Time Complexity

均摊分析

Amortized Analysis

背景

```
ArrayList::add(1)
```

时间复杂度 不扩容时 $O(1)$ ，扩容时 $O(n)$

是什么

连续进行 n 次操作，计算平均下来的时间复杂度

计算方法

计算每一次的开销

n 次 的单位时间 求和

再除 n

线性数据结构

栈

Stack

操作

push 加元素
pop 删元素
peek / top 查看顶元素
size 几个元素

要求

先 push 的 最后 pop

举例

麦当劳的外卖纸袋
手机应用

实现

Array / Link

队列

Queue

操作

enqueue 加元素
dequeue 删元素

要求

先 enqueue 的 先 dequeue

举例

叫号系统

实现

Array / Link

训练

特殊情况分析

事实

算法程序的错误

语法错误 / 时间复杂度太高 / 特殊情况没考虑到 / 审题

括号问题

要求

给定一个 非空 字符串 str，
里面由

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

符号组成。

请你确定是否合法

API

```
isValid(str: String) : Bool
```

分析

一般情况：

思路

伪代码

特殊情况：

案例

后果

问题所在

表达式计算

要求

给定一个 表达式 由 个位整数 和 $+-*/$ 组成

比如：

"3+4*2-1"

求出表达式的结果

API

```
calculate(expression: String) : Int
```

算法

字符串末尾添加 "\$"：

3+4*2-1
3+4*2-1\$

准备两个 Stack：

expressionElements

3	+	4	*	2	-	1	\$
---	---	---	---	---	---	---	----

numbers

operators

每次拿出两个元素：

expressionElements

4	*	2	-	1	\$
---	---	---	---	---	----

numbers

3

operators

+

情况 1:

如果 stack 为空

expressionElements

4	*	2	-	1	\$
---	---	---	---	---	----

numbers

3

operators

+

那么 放进去

expressionElements

4	*	2	-	1	\$
---	---	---	---	---	----

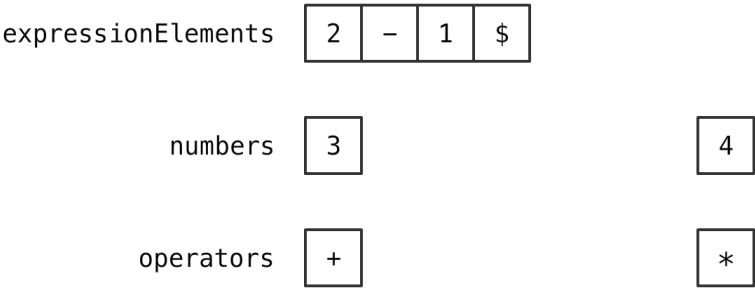
numbers

3

operators

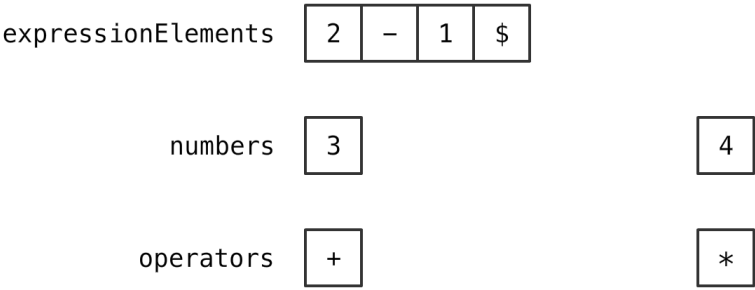
+

再拿出 两个元素

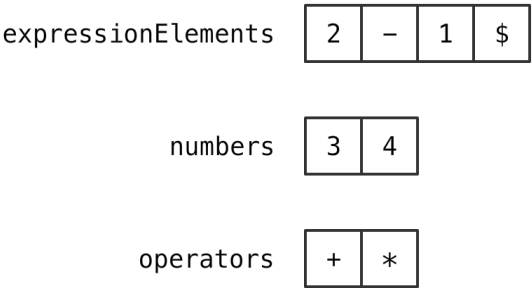


情况 2:

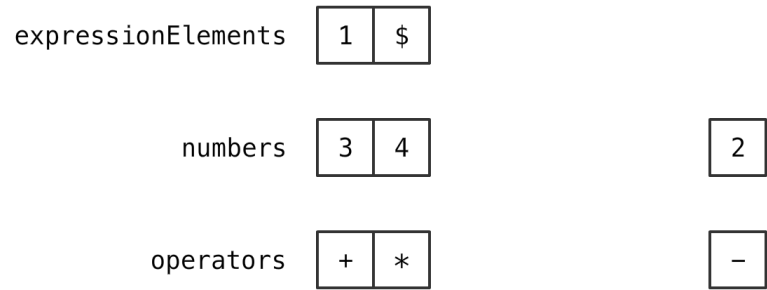
如果 新的运算符 优先级 高于 栈顶的运算符 优先级



那么 放进去

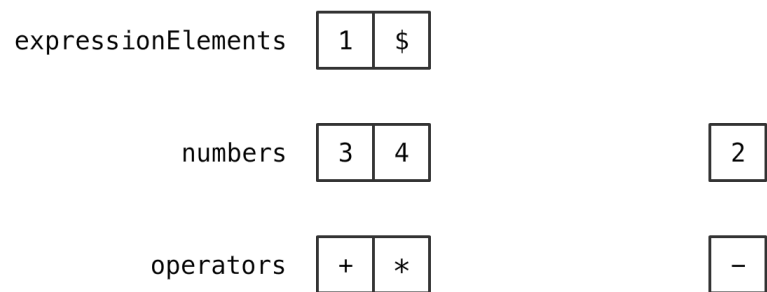


再拿出 两个元素

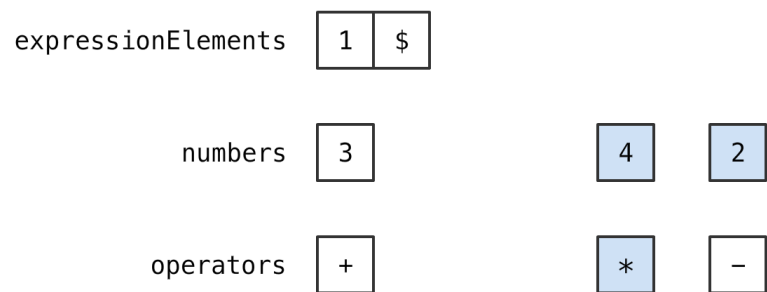


情况 3:

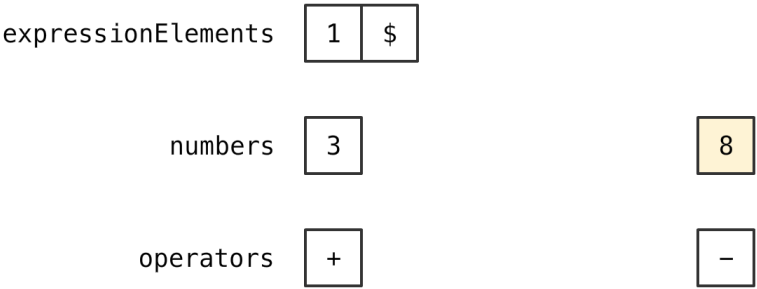
如果 新的运算符 优先级 低于 栈顶的运算符 优先级



那么 从两个 Stack 里 各取出 1 个元素



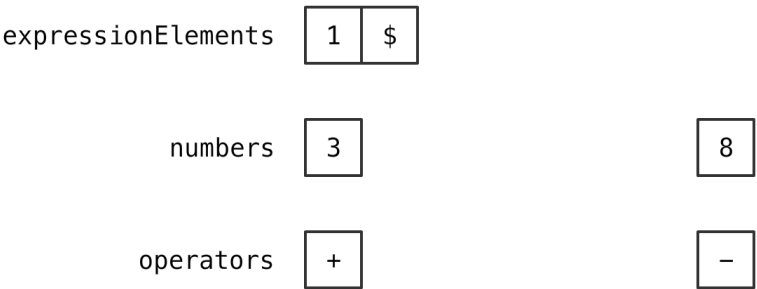
计算



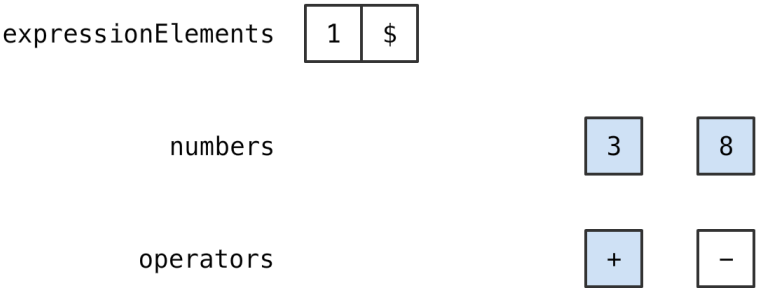
再比较

情况 4:

如果 新的运算符 优先级 等于 栈顶的运算符 优先级



那么 从两个 Stack 里 各取出 1 个元素



计算

expressionElements

1	\$
---	----

numbers

11

operators

-

再比较

走下去:

expressionElements

1	\$
---	----

numbers

11

operators

-

情况 1, 放进去

expressionElements

1	\$
---	----

numbers

11

operators

-

取 2 个

expressionElements

numbers	11	1
operators	-	\$

情况 2, 各取 1 个

expressionElements

numbers	11	1
operators	-	\$

计算

expressionElements

numbers	10
operators	\$

情况 1, 放进去

expressionElements

numbers	10
operators	\$

结束：

numbers 栈顶元素就是结果

分析

特殊情况：

案例

后果

问题所在

修改方案