

讲堂 > 数据结构与算法之美 > 文章详情

08 | 栈：如何实现浏览器的前进和后退功能？

2018-10-08 王争



08 | 栈：如何实现浏览器的前进和后退功能？

朗读人：修阳 14'12" | 6.51M

浏览器的前进、后退功能，我想你肯定很熟悉吧？

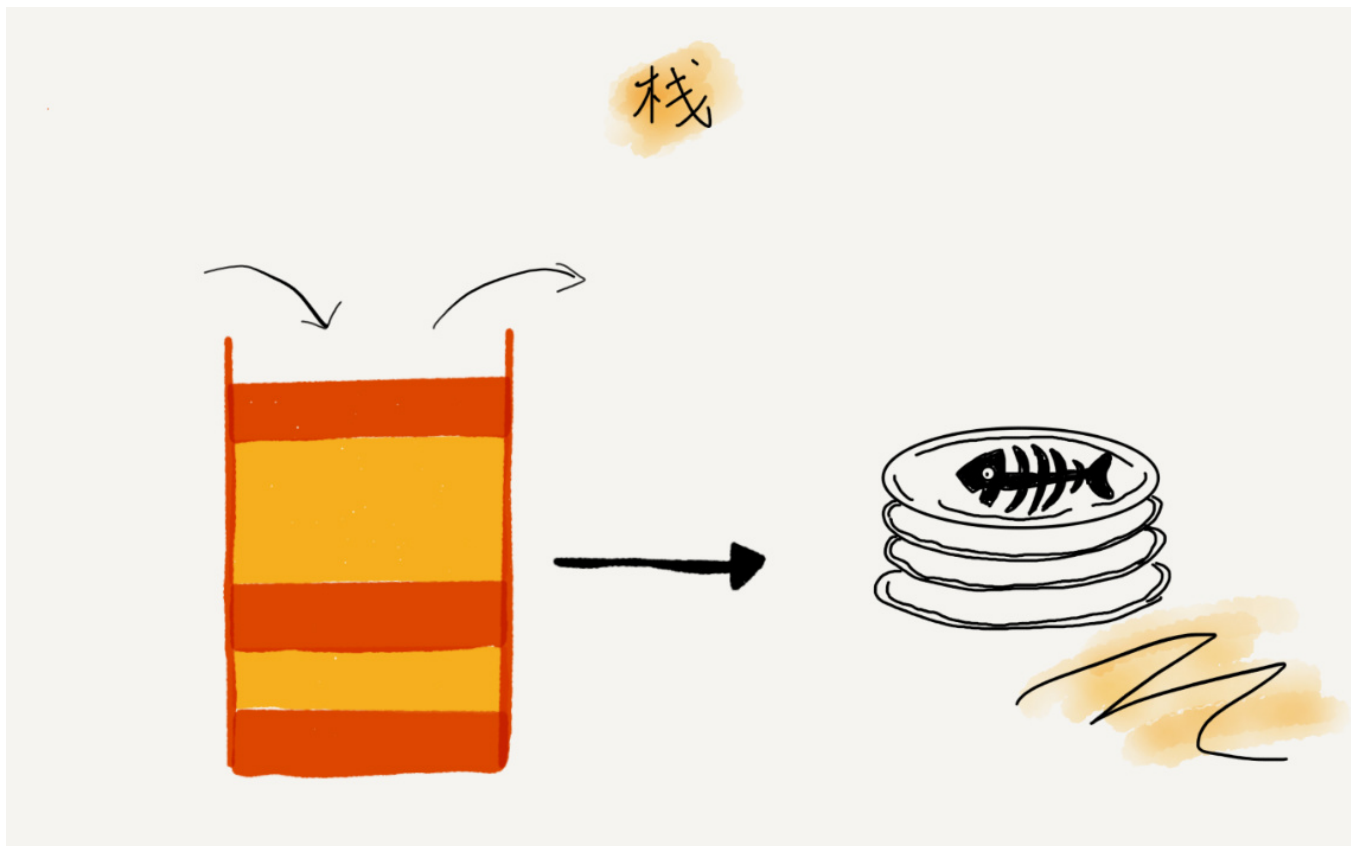
当你依次访问完一串页面 a-b-c 之后，点击浏览器的后退按钮，就可以查看之前浏览过的页面 b 和 a。当你后退到页面 a，点击前进按钮，就可以重新查看页面 b 和 c。但是，如果你后退到页面 b 后，点击了新的页面 d，那就无法再通过前进、后退功能查看页面 c 了。

假设你是 Chrome 浏览器的开发工程师，你会如何实现这个功能呢？

这就要用到我们今天要讲的“栈”这种数据结构。带着这个问题，我们来学习今天的内容。

如何理解“栈”？

关于“栈”，我有一个非常贴切的例子，就是一摞叠在一起的盘子。我们平时放盘子的时候，都是从下往上一个一个放；取的时候，我们也是从上往下一个一个地依次取，不能从中间任意抽出。后进者先出，先进者后出，这就是典型的“栈”结构。



从栈的操作特性上来看，**栈是一种“操作受限”的线性表**，只允许在一端插入和删除数据。

我第一次接触这种数据结构的时候，就对它存在的意义产生了很大的疑惑。因为我觉得，相比数组和链表，栈带给我的只有限制，并没有任何优势。那我直接使用数组或者链表不就好了吗？为什么还要用这个“操作受限”的“栈”呢？

事实上，从功能上来说，数组或链表确实可以替代栈，但你要知道，特定的数据结构是对特定场景的抽象，而且，数组或链表暴露了太多的操作接口，操作上的确灵活自由，但使用时就比较不可控，自然也就更容易出错。

当某个数据集只涉及在一端插入和删除数据，并且满足后进先出、先进后出的特性，我们就应该首选“栈”这种数据结构。


如何实现一个“栈”？

从刚才栈的定义里，我们可以看出，栈主要包含两个操作，入栈和出栈，也就是在栈顶插入一个数据和从栈顶删除一个数据。理解了栈的定义之后，我们来看一看如何用代码实现一个栈。

实际上，栈既可以用数组来实现，也可以用链表来实现。用数组实现的栈，我们叫作**顺序栈**，用链表实现的栈，我们叫作**链式栈**。

我这里实现一个基于数组的顺序栈。基于链表实现的链式栈的代码，你可以自己试着写一下。我会将我写好的代码放到 Github 上，你可以去看一下自己写的是否正确。

我这段代码是用 Java 来实现的，但是不涉及任何高级语法，并且我还用中文做了详细的注释，所以你应该是可以看懂的。

 复制代码

```
1 // 基于数组实现的顺序栈
2 public class ArrayStack {
3     private String[] items; // 数组
4     private int count;      // 栈中元素个数
5     private int n;          // 栈的大小
6
7     // 初始化数组，申请一个大小为 n 的数组空间
8     public ArrayStack(int n) {
9         this.items = new String[n];
10        this.n = n;
11        this.count = 0;
12    }
13
14    // 入栈操作
15    public boolean push(String item) {
16        // 数组空间不够了，直接返回 false，入栈失败。
17        if (count == n) return false;
18        // 将 item 放到下标为 count 的位置，并且 count 加一
19        items[count] = item;
20        ++count;
21        return true;
22    }
23
24    // 出栈操作
25    public String pop() {
26        // 栈为空，则直接返回 null
27        if (count == 0) return null;
28        // 返回下标为 count-1 的数组元素，并且栈中元素个数 count 减一
29        String tmp = items[count-1];
30        --count;
31        return tmp;
32    }
33 }
```

了解了定义和基本操作，那它的操作的时间、空间复杂度是多少呢？

不管是顺序栈还是链式栈，我们存储数据只需要一个大小为 n 的数组就够了。在入栈和出栈过程中，只需要一两个临时变量存储空间，所以空间复杂度是 $O(1)$ 。

注意，这里存储数据需要一个大小为 n 的数组，并不是说空间复杂度就是 $O(n)$ 。因为，这 n 个空间是必须的，无法省掉。所以我们说空间复杂度的时候，是指除了原本的数据存储空间外，算法运行还需要额外的存储空间。

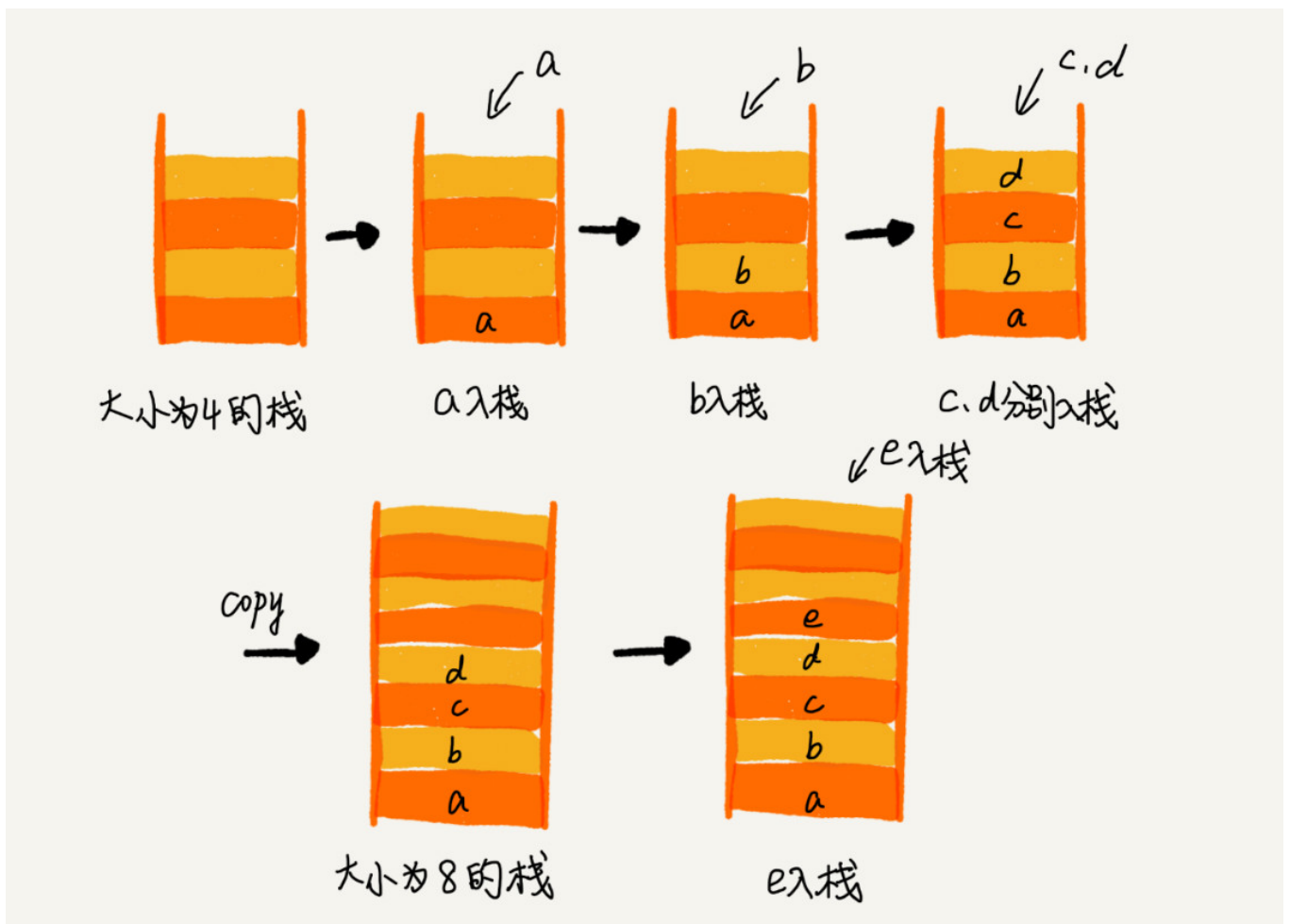
空间复杂度分析是不是很简单？时间复杂度也不难。不管是顺序栈还是链式栈，入栈、出栈只涉及栈顶个别数据的操作，所以时间复杂度都是 $O(1)$ 。

支持动态扩容的顺序栈

刚才那个基于数组实现的栈，是一个固定大小的栈，也就是说，在初始化栈时需要事先指定栈的大小。当栈满之后，就无法再往栈里添加数据了。尽管链式栈的大小不受限，但要存储 next 指针，内存消耗相对较多。那我们如何基于数组实现一个可以支持动态扩容的栈呢？

你还记得，我们在数组那一节，是如何来实现一个支持动态扩容的数组的吗？当数组空间不够时，我们就重新申请一块更大的内存，将原来数组中数据统统拷贝过去。这样就实现了一个支持动态扩容的数组。

所以，如果我们要实现一个支持动态扩容的栈，我们只需要底层依赖一个支持动态扩容的数组就可以了。当栈满了之后，我们就申请一个更大的数组，将原来的数据搬移到新数组中。我画了一张图，你可以对照着理解一下。



实际上，支持动态扩容的顺序栈，我们平时开发中并不常用到。我讲这一块的目的，主要还是希望带你练习一下前面讲的复杂度分析方法。所以这一小节的重点是复杂度分析。

你不用死记硬背入栈、出栈的时间复杂度，你需要掌握的是分析方法。能够自己分析才算是真正掌握了。现在我就带你分析一下支持动态扩容的顺序栈的入栈、出栈操作的时间复杂度。

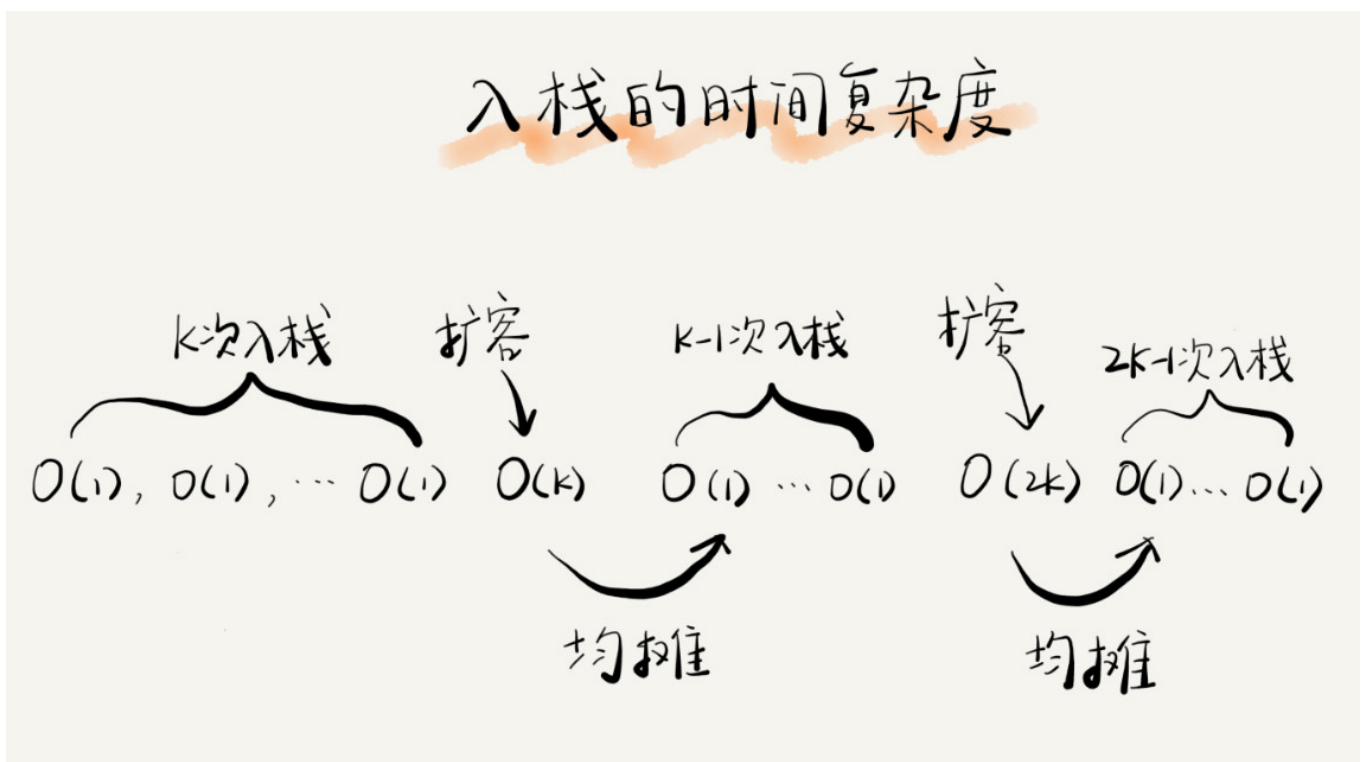
对于出栈操作来说，我们不会涉及内存的重新申请和数据的搬移，所以出栈的时间复杂度仍然是 $O(1)$ 。但是，对于入栈操作来说，情况就不一样了。当栈中有空闲空间时，入栈操作的时间复杂度为 $O(1)$ 。但当空间不够时，就需要重新申请内存和数据搬移，所以时间复杂度就变成了 $O(n)$ 。

也就是说，对于入栈操作来说，最好情况时间复杂度是 $O(1)$ ，最坏情况时间复杂度是 $O(n)$ 。那平均情况下的时间复杂度又是多少呢？还记得我们在复杂度分析那一节中讲的摊还分析法吗？这个入栈操作的平均情况下的时间复杂度可以用摊还分析法来分析。我们也正好借此来实战一下摊还分析法。

为了分析的方便，我们需要事先做一些假设和定义：

- 栈空间不够时，我们重新申请一个是原来大小两倍的数组；
- 为了简化分析，假设只有入栈操作没有出栈操作；
- 定义不涉及内存搬移的入栈操作为 simple-push 操作，时间复杂度为 $O(1)$ 。

如果当前栈大小为 K ，并且已满，当再有新的数据要入栈时，就需要重新申请 2 倍大小的内存，并且做 K 个数据的搬移操作，然后再入栈。但是，接下来的 $K-1$ 次入栈操作，我们都不需要再重新申请内存和搬移数据，所以这 $K-1$ 次入栈操作都只需要一个 simple-push 操作就可以完成。为了让你更加直观地理解这个过程，我画了一张图。



你应该可以看出来，这 K 次入栈操作，总共涉及了 K 个数据的搬移，以及 K 次 simple-push 操作。将 K 个数据搬移均摊到 K 次入栈操作，那每个入栈操作只需要一个数据搬移和一个 simple-push 操作。以此类推，入栈操作的均摊时间复杂度就为 $O(1)$ 。

通过这个例子的实战分析，也印证了前面讲到的，均摊时间复杂度一般都等于最好情况时间复杂度。因为在大部分情况下，入栈操作的时间复杂度 O 都是 $O(1)$ ，只有在个别时刻才会退化为 $O(n)$ ，所以把耗时多的入栈操作的时间均摊到其他入栈操作上，平均情况下的耗时就接近 $O(1)$ 。

栈在函数调用中的应用

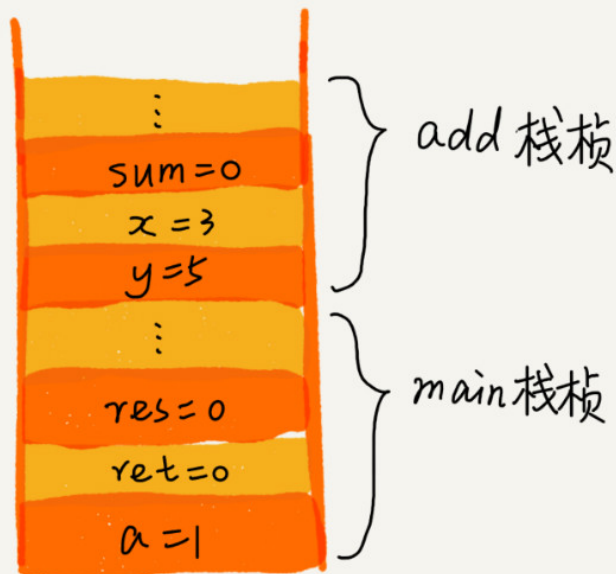
前面我讲的都比较偏理论，我们现在来看下，栈在软件工程中的实际应用。栈作为一个比较基础的数据结构，应用场景还是蛮多的。其中，比较经典的一个应用场景就是**函数调用栈**。

我们知道，操作系统给每个线程分配了一块独立的内存空间，这块内存被组织成“栈”这种结构，用来存储函数调用时的临时变量。每进入一个函数，就会将临时变量作为一个栈帧入栈，当被调用函数执行完成，返回之后，将这个函数对应的栈帧出栈。为了让你更好地理解，我们一块来看下这段代码的执行过程。

```
1 int main() {
2     int a = 1;
3     int ret = 0;
4     int res = 0;
5     ret = add(3, 5);
6     res = a + ret;
7     printf("%d", res);
8     return 0;
9 }
10
11 int add(int x, int y) {
12     int sum = 0;
13     sum = x + y;
14     return sum;
15 }
```

[复制代码](#)

从代码中我们可以看出，main() 函数调用了 add() 函数，获取计算结果，并且与临时变量 a 相加，最后打印 res 的值。为了让你清晰地看到这个过程对应的函数栈里出栈、入栈的操作，我画了一张图。图中显示的是，在执行到 add() 函数时，函数调用栈的情况。



栈在表达式求值中的应用

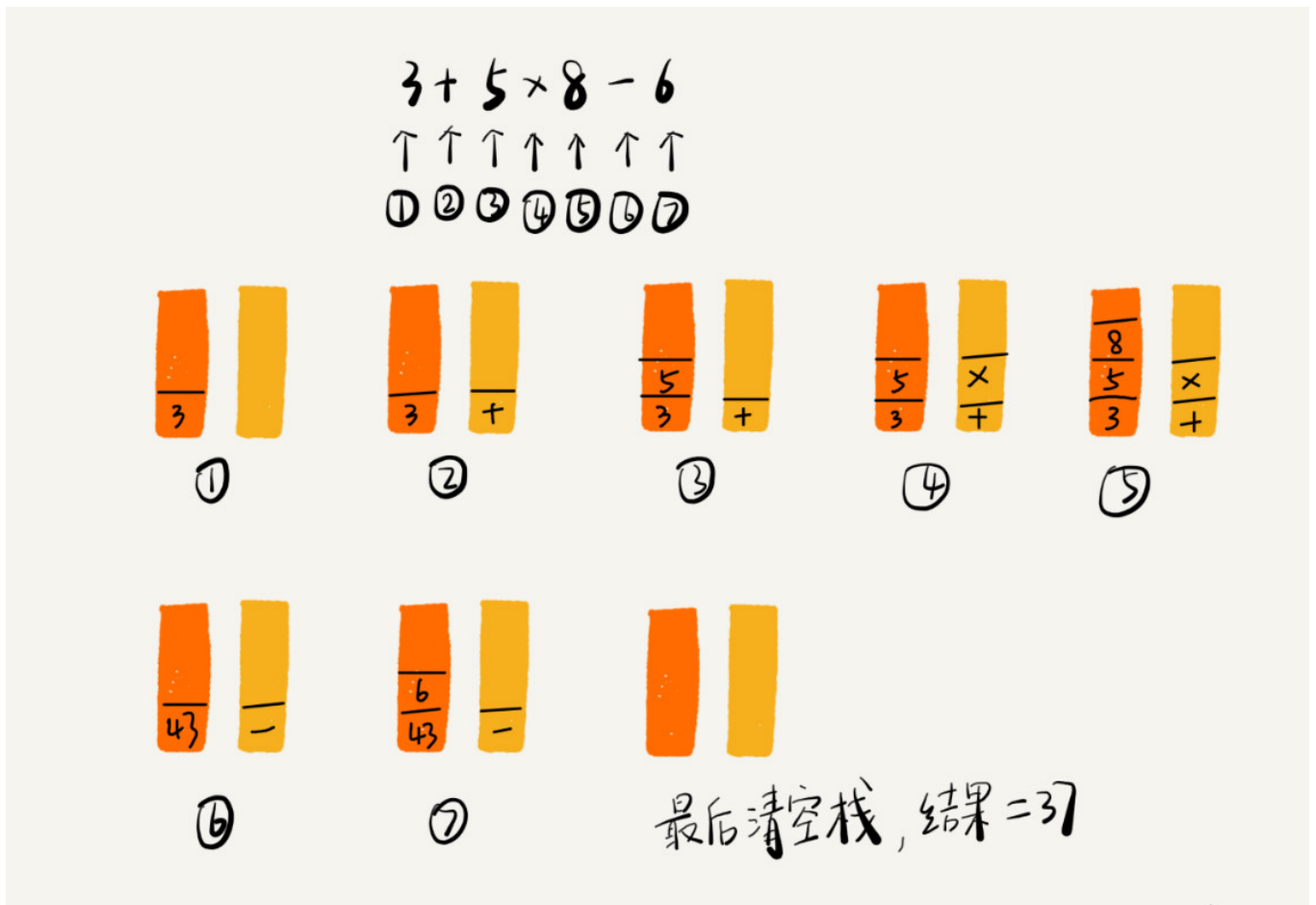
我们再来看栈的另一个常见的应用场景，编译器如何利用栈来实现**表达式求值**。

为了方便解释，我将算术表达式简化为只包含加减乘除四则运算，比如： $34+13*9+44-12/3$ 。对于这个四则运算，我们人脑可以很快求解出答案，但是对于计算机来说，理解这个表达式本身就是个挺难的事儿。如果换作你，让你来实现这样一个表达式求值的功能，你会怎么做呢？

实际上，编译器就是通过两个栈来实现的。其中一个保存操作数的栈，另一个是保存运算符的栈。我们从左向右遍历表达式，当遇到数字，我们就直接压入操作数栈；当遇到运算符，就与运算符栈的栈顶元素进行比较。

如果比运算符栈顶元素的优先级高，就将当前运算符压入栈；如果比运算符栈顶元素的优先级低或者相同，从运算符栈中取栈顶运算符，从操作数栈的栈顶取 2 个操作数，然后进行计算，再把计算完的结果压入操作数栈，继续比较。

我将 $3+5*8-6$ 这个表达式的计算过程画成了一张图，你可以结合图来理解我刚讲的计算过程。



这样用两个栈来解决的思路是不是非常巧妙？你有没有想到呢？

栈在括号匹配中的应用

除了用栈来实现表达式求值，我们还可以借助栈来检查表达式中的括号是否匹配。

我们同样简化一下背景。我们假设表达式中只包含三种括号，圆括号 ()、方括号 [] 和花括号 {}，并且它们可以任意嵌套。比如，{[]} 或 [{()}[]] 等都为合法格式，而 {}() 或 [(())] 为不合法的格式。那我现在给你一个包含三种括号的表达式字符串，如何检查它是否合法呢？

这里也可以用栈来解决。我们用栈来保存未匹配的左括号，从左到右依次扫描字符串。当扫描到左括号时，则将其压入栈中；当扫描到右括号时，从栈顶取出一个左括号。如果能够匹配，比如“(”跟“)”匹配，“[”跟“]”匹配，“{”跟“}”匹配，则继续扫描剩下的字符串。如果扫描的过程中，遇到不能配对的右括号，或者栈中没有数据，则说明为非法格式。

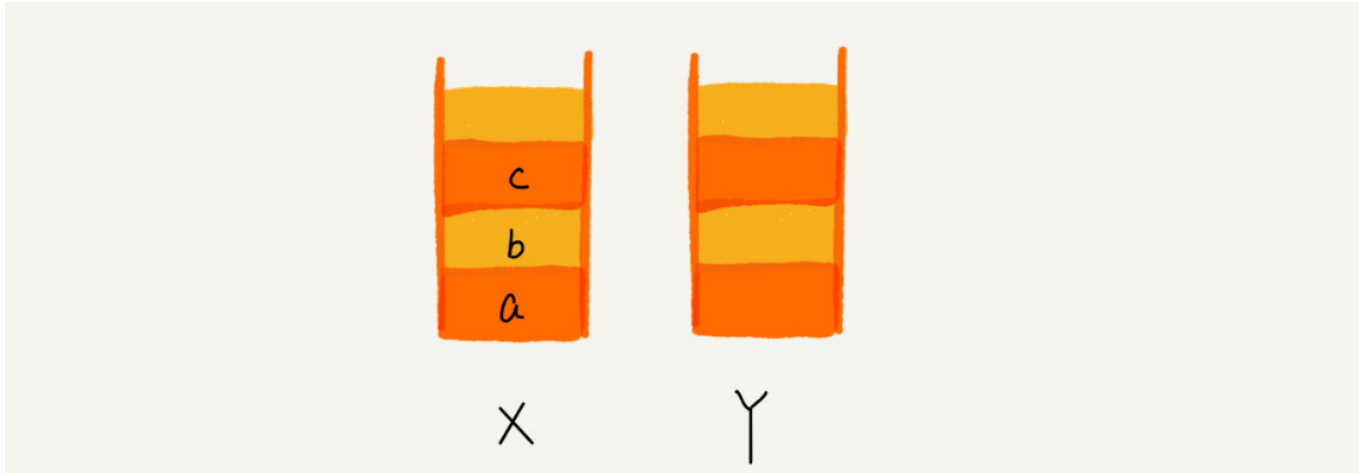
当所有的括号都扫描完成之后，如果栈为空，则说明字符串为合法格式；否则，说明有未匹配的左括号，为非法格式。

解答开篇

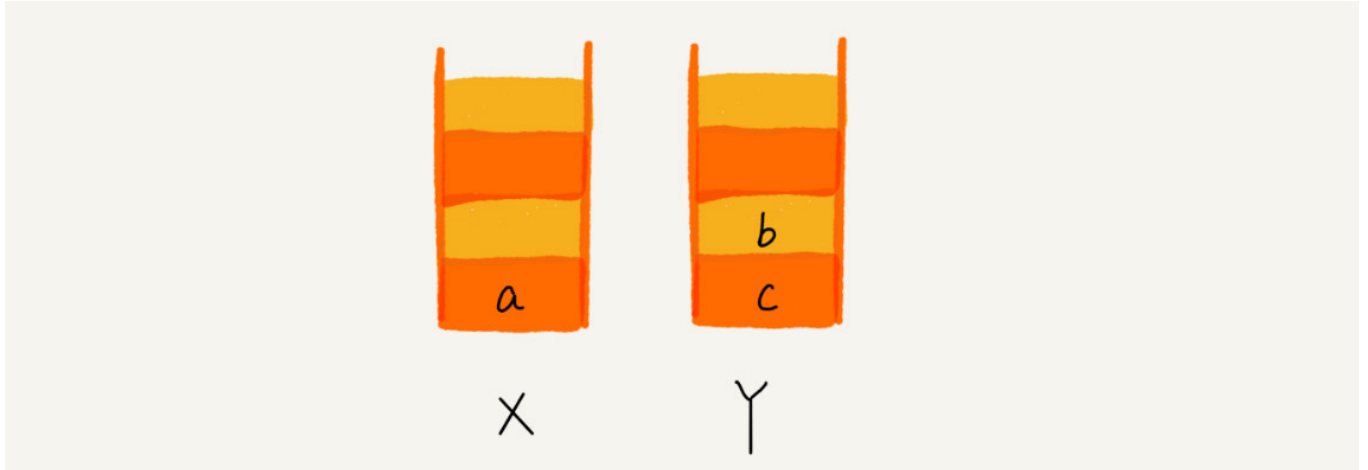
好了，我想现在你已经完全理解了栈的概念。我们再回来看看开篇的思考题，如何实现浏览器的前进、后退功能？其实，用两个栈就可以非常完美地解决这个问题。

我们使用两个栈，X 和 Y，我们把首次浏览的页面依次压入栈 X，当点击后退按钮时，再依次从栈 X 中出栈，并将出栈的数据依次放入栈 Y。当我们点击前进按钮时，我们依次从栈 Y 中取出数据，放入栈 X 中。当栈 X 中没有数据时，那就说明没有页面可以继续后退浏览了。当栈 Y 中没有数据，那就说明没有页面可以点击前进按钮浏览了。

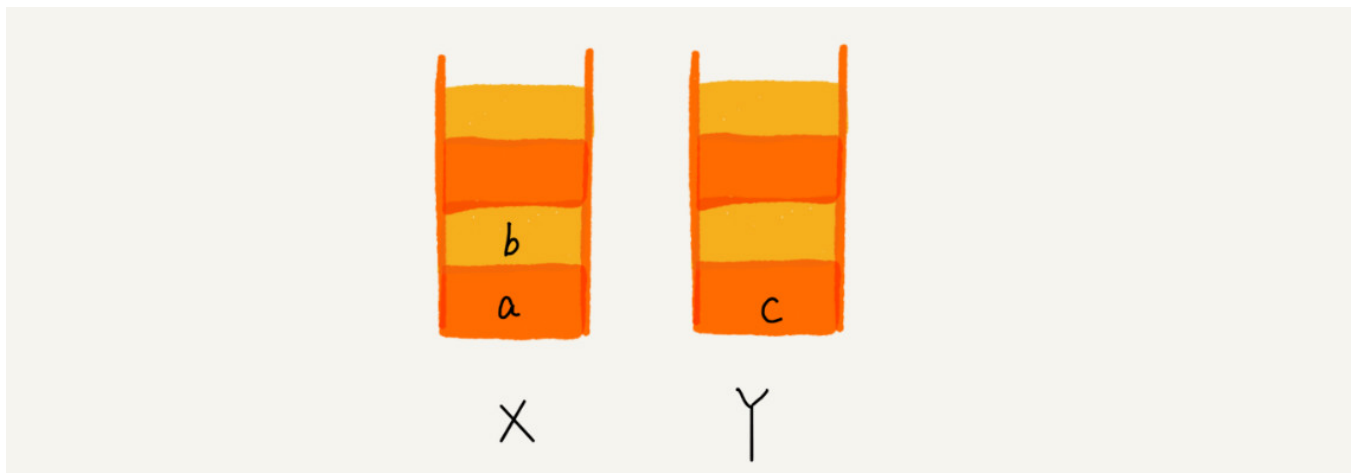
比如你顺序查看了 a, b, c 三个页面，我们就依次把 a, b, c 压入栈，这个时候，两个栈的数据就是这个样子：



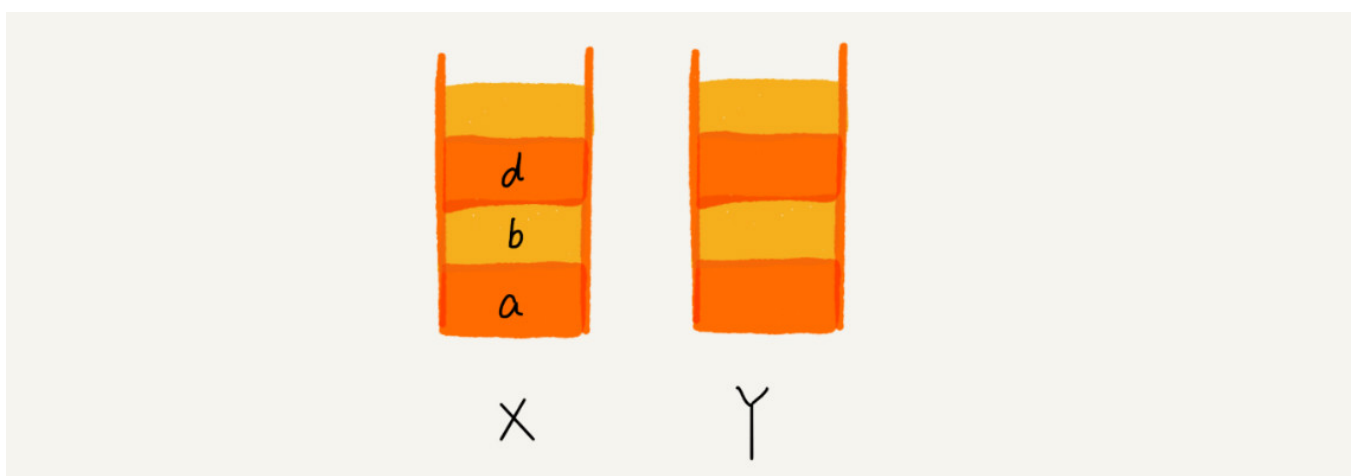
当你通过浏览器的后退按钮，从页面 c 后退到页面 a 之后，我们就依次把 c 和 b 从栈 X 中弹出，并且依次放入到栈 Y。这个时候，两个栈的数据就是这个样子：



这个时候你又想看页面 b，于是你又点击前进按钮回到 b 页面，我们就把 b 再从栈 Y 中出栈，放入栈 X 中。此时两个栈的数据是这个样子：



这个时候，你通过页面 b 又跳转到新的页面 d 了，页面 c 就无法再通过前进、后退按钮重复查看了，所以需要清空栈 Y。此时两个栈的数据这个样子：



内容小结

我们来回顾一下今天讲的内容。栈是一种操作受限的数据结构，只支持入栈和出栈操作。后进先出是它最大的特点。栈既可以通过数组实现，也可以通过链表来实现。不管基于数组还是链表，入栈、出栈的时间复杂度都为 $O(1)$ 。除此之外，我们还讲了一种支持动态扩容的顺序栈，你需要重点掌握它的均摊时间复杂度分析方法。

课后思考

1. 我们在讲栈的应用时，讲到用函数调用栈来保存临时变量，为什么函数调用要用“栈”来保存临时变量呢？用其他数据结构不行吗？
2. 我们都知道，JVM 内存管理中有个“堆栈”的概念。栈内存用来存储局部变量和方法调用，堆内存用来存储 Java 中的对象。那 JVM 里面的“栈”跟我们这里说的“栈”是不是一回事呢？如果不是，那它为什么又叫作“栈”呢？

欢迎留言和我分享，我会第一时间给你反馈。

[戳此查看本节内容相关的详细代码](#)

版权归极客邦科技所有，未经许可不得转载

[写留言](#)

精选留言



阿杜S考特

👍 39

内存中的堆栈和数据结构堆栈不是一个概念，可以说内存中的堆栈是真实存在的物理区，数据结构中的堆栈是抽象的数据存储结构。

内存空间在逻辑上分为三部分：代码区、静态数据区和动态数据区，动态数据区又分为栈区和堆区。

代码区：存储方法体的二进制代码。高级调度（作业调度）、中级调度（内存调度）、低级调度（进程调度）控制代码区执行代码的切换。

静态数据区：存储全局变量、静态变量、常量，常量包括final修饰的常量和String常量。系统自动分配和回收。

栈区：存储运行方法的形参、局部变量、返回值。由系统自动分配和回收。

堆区：new一个对象的引用或地址存储在栈区，指向该对象存储在堆区中的真实数据。

2018-10-08



Rain

👍 35

对于老师的思考题我看到评论区有多种答案，希望老师能够给出权威解释，谢谢！

2018-10-08



姜威

👍 17

一、什么是栈？

1. 后进者先出，先进者后出，这就是典型的“栈”结构。

2.从栈的操作特性来看，是一种“操作受限”的线性表，只允许在端插入和删除数据。

二、为什么需要栈？

1.栈是一种操作受限的数据结构，其操作特性用数组和链表均可实现。

2.但，任何数据结构都是对特定应用场景的抽象，数组和链表虽然使用起来更加灵活，但却暴露了几乎所有的操作，难免会引发错误操作的风险。

3.所以，当某个数据集只涉及在某端插入和删除数据，且满足后进者先出，先进者后出的操作特性时，我们应该首选栈这种数据结构。

三、如何实现栈？

1.栈的API

```
public class Stack<Item> {  
    //压栈  
    public void push(Item item){  
    //弹栈  
    public Item pop(){  
    //是否为空  
    public boolean isEmpty(){  
    //栈中数据的数量  
    public int size(){  
    //返回栈中最近添加的元素而不删除它  
    public Item peek(){  
    }  
}
```

2.数组实现（自动扩容）

时间复杂度分析：根据均摊复杂度的定义，可以得数组实现（自动扩容）符合大多数情况是 $O(1)$ 级别复杂度，个别情况是 $O(n)$ 级别复杂度，比如自动扩容时，会进行完整数据的拷贝。

空间复杂度分析：在入栈和出栈的过程中，只需要一两个临时变量存储空间，所以 $O(1)$ 级别。我们说空间复杂度的时候，是指除了原本的数据存储空间外，算法运行还需要额外的存储空间。

实现代码：（见另一条留言）

3.链表实现

时间复杂度分析：压栈和弹栈的时间复杂度均为 $O(1)$ 级别，因为只需更改单个节点的索引即可。

空间复杂度分析：在入栈和出栈的过程中，只需要一两个临时变量存储空间，所以 $O(1)$ 级别。我们说空间复杂度的时候，是指除了原本的数据存储空间外，算法运行还需要额外的存储空间。

实现代码：（见另一条留言）

四、栈的应用

1.栈在函数调用中的应用

操作系统给每个线程分配了一块独立的内存空间，这块内存被组织成“栈”这种结构，用来存储函数调用时的临时变量。每进入一个函数，就会将其中的临时变量作为栈帧入栈，当被调用函数执行完成，返回之后，将这个函数对应的栈帧出栈。

2.栈在表达式求值中的应用（比如： $34+13*9+44-12/3$ ）

利用两个栈，其中一个用来保存操作数，另一个用来保存运算符。我们从左向右遍历表达式，当遇到数字，我们就直接压入操作数栈；当遇到运算符，就与运算符栈的栈顶元素进行比较，若比运算符栈顶元素优先级高，就将当前运算符压入栈，若比运算符栈顶元素的优先级低或者相同，从运算符栈中取出栈顶运算符，从操作数栈顶取出2个操作数，然后进行计算，把计算完的结果压入操作数栈，继续比较。

3. 栈在括号匹配中的应用（比如：{[{()}]()})

用栈保存为匹配的左括号，从左到右一次扫描字符串，当扫描到左括号时，则将其压入栈中；当扫描到右括号时，从栈顶取出一个左括号，如果能匹配上，则继续扫描剩下的字符串。如果扫描过程中，遇到不能配对的右括号，或者栈中没有数据，则说明为非法格式。

当所有的括号都扫描完成之后，如果栈为空，则说明字符串为合法格式；否则，说明未匹配的左括号为非法格式。

4. 如何实现浏览器的前进后退功能？

我们使用两个栈X和Y，我们把首次浏览的页面依次压如栈X，当点击后退按钮时，再依次从栈X中出栈，并将出栈的数据一次放入Y栈。当点击前进按钮时，我们依次从栈Y中取出数据，放入栈X中。当栈X中没有数据时，说明没有页面可以继续后退浏览了。当Y栈没有数据，那就说明没有页面可以点击前进浏览了。

五、思考

1. 我们在讲栈的应用时，讲到用函数调用栈来保存临时变量，为什么函数调用要用“栈”来保存临时变量呢？用其他数据结构不行吗？

答：因为函数调用的执行顺序符合后进者先出，先进者后出的特点。比如函数中的局部变量的生命周期的长短是先定义的生命周期长，后定义的生命周期短；还有函数中调用函数也是这样，先开始执行的函数只有等到内部调用的其他函数执行完毕，该函数才能执行结束。

正是由于函数调用的这些特点，根据数据结构是特定应用场景的抽象的原则，我们优先考虑栈结构。

2. 我们都知道，JVM 内存管理中有个“堆栈”的概念。栈内存用来存储局部变量和方法调用，堆内存用来存储 Java 中的对象。那 JVM 里面的“栈”跟我们这里说的“栈”是不是一回事呢？如果不是，那它为什么又叫作“栈”呢？

答：JVM里面的栈和我们这里说的是一回事，被称为方法栈。和前面函数调用的作用是一致的，用来存储方法中的局部变量。

2018-10-08



观弈道人

12

感觉在留言区做笔记没多大意义，留言区还是提问问题或回答问题笔记合适，长篇累牍的笔记给谁看啊，占空间~~

2018-10-08

作者回复

还是有很多同学看的 个人喜好吧 不必强求

2018-10-09



小洋洋

9

函数调用之所以用栈，是因为函数调用中经常嵌套，栗子：A调用B，B又调用C，那么就需要先把C执行完，结果赋值给B中的临时变量，B的执行结果再赋值给A的临时变量，嵌套越深的

函数越需要被先执行，这样刚好符合栈的特点，因此每次遇到函数调用，只需要压栈，最后依次从栈顶弹出依次执行即可，这个过程很像文稿中的3+5*8-6//小白之拙见，欢迎拍砖*^o^*

2018-10-08



姜威

👍 7

实现代码：（栈的数组实现）

```
public class StackOfArray<Item> implements Iterable<Item>{
    //存储数据的数组
    Item[] a = (Item[])new Object[1];
    //记录元素个数N
    int N = 0;
    //构造器
    public StackOfArray(){
        //添加元素
        public void push(Item item){
            //自动扩容
            if (N == a.length ) resize(2*a.length );
            a[N++] = item;
        }
        //删除元素
        public Item pop(){
            Item item = a[--N];
            a[N] = null;
            if (N > 0 && N == a.length / 4) resize(a.length / 2);
            return item;
        }
        //是否为空
        public boolean isEmpty(){
            return N == 0;
        }
        //元素个数
        public int size(){
            return N;
        }
        //改变数组容量
        private void resize(int length) {
            Item[] temp = (Item[])new Object[length];
            for (int i = 0; i < N; i++) {
                temp[i] = a[i];
            }
            a = temp;
        }
        //返回栈中最近添加的元素而不删除它
```



```

public Item peek(){
return a[N-1];
}
@Override
public Iterator<Item> iterator() {
return new ArrayIterator();
}
//内部类
class ArrayIterator implements Iterator{
//控制迭代数量
int i = N;
@Override
public boolean hasNext() {
return i > 0;
}
@Override
public Item next() {
return a[--i];
}
}
}

```

实现代码：（栈的链表实现）

```

public class StackOfLinked<Item> implements Iterable<Item> {
//定义一个内部类，就可以直接使用类型参数
private class Node{
Item item;
Node next;
}
private Node first;
private int N;
//构造器
public StackOfLinked(){
//添加
public void push(Item item){
Node oldfirst = first;
first = new Node();
first.item = item;
first.next = oldfirst;
N++;
}
//删除
public Item pop(){

```

```

Item item = first.item;
first = first.next;
N--;
return item;
}
//是否为空
public boolean isEmpty(){
return N == 0;
}
//元素数量
public int size(){
return N;
}
//返回栈中最近添加的元素而不删除它
public Item peek(){
return first.item;
}
@Override
public Iterator<Item> iterator() {
return new LinkedIterator();
}
//内部类：迭代器
class LinkedIterator implements Iterator{
int i = N;
Node t = first;
@Override
public boolean hasNext() {
return i > 0;
}
@Override
public Item next() {
Item item = (Item) t.item;
t = t.next;
i--;
return item;
}
}
}

```

2018-10-08



Liam

👍 6

1 函数调用和返回符合后进先出原则，而局部变量的生命周期应该和函数一致，因此用栈保存局部变量是合适的，函数出栈时同时销毁局部变量

2 jvm的栈就是一种栈数据结构，本质相同

2018-10-08



Smallfly

👍 3

函数调用为什么用栈实现，其中一个原因是为了满足递归的需求。

2018-10-08



刘大侠

👍 3

不一样的栈，java里面栈和堆有数据共享不共享的问题，所以不一样

2018-10-08



kylexu

👍 3

看完再睡!

2018-10-08



飞羽

👍 2

看到有人问小括号的情况怎么办，我写了个JS的示例，抛砖引玉哈。
主要思路就是遇到小括号时，将小括号内的表达式提取出来，然后进行递归调用。

使用ES6写的，Git地址：

<https://github.com/taifu5522/ProblemSet>

2018-10-09

作者回复

👍 不过不需要用递归的

2018-10-09



Monday

👍 2

对于每次留下的思考题都希望老师在 n ($n>1$) 天后给出权威的答案，谢谢。
国庆在家里只看文档和听音频没有记录笔记，回去工作了，一定补上。个人认为本课题是最实惠的知识付费，没有之一。❤

2018-10-09

作者回复

关于思考题 很多同学的留言都已经回答的很好了 关于权威答案 我可以集中写篇文章说说

2018-10-09



乘坐Tornado的线程魔法师

👍 2

想借此机会请教一个问题，同时也算是给大家的一道思考题。如果文中计算表达式的例子 涉及到小括号(默认小括号使用合法) 该如何处理呢？

2018-10-08

作者回复

思路一样的 可以把右括号看作优先级最高的 至于左括号怎么处理 留给你思考吧 😊

2018-10-09



陈华应

👍 2

1, 方法调用是有先后顺序的, 并且一个线程中同一时间点只会处理一个方法, 若方法中调用其他方法, 则被调用方法会入栈并且方法体被执行, 执行完后方法出栈, 并将执行结果相关信息向下一个执行方法传递(方法所有信息被封装为一个栈元素)。方法执行过程中产生的数据存储特点非常符合栈数据结构特点。

2, 概念和用途是一个意思, jvm的栈稍有复杂, 每个栈元素叫做栈帧(一个包含被执行方法所有信息的元素, 栈最顶部是当前栈帧, 也就是正在被执行的方法), 栈元素是一个具体方法, 又包含了操作数栈, 用于方法的具体执行中的数据计算。

2018-10-08



天涯过客

👍 2

课后思考题会给思路吗?

2018-10-08



zyzheng

👍 2

函数调用使用的栈是有硬件基础的, 所有的CPU都有相应的SP寄存器用于存储栈顶指针, 也有相应的入栈出栈指令, 用于实现函数调用栈效率很高, 和软件数据结构的栈有所不同。

如果要回答为什么函数调用要用栈, 个人理解是CPU设计就是如此

2018-10-08



nkulpj

👍 1

#栈实现括号匹配!

```
class Solution:
    def isValid(self, s):
        """
        :type s: str
        :rtype: bool
        """
        #栈
        a=[]
        if len(s)==0:return True

        for i in range(len(s)):
            #入栈
            if s[i]=='(' or s[i]=='{' or s[i]=='[' :a.append(s[i])
            else:
                if len(a)==0:return False
                if s[i]==')' and a[-1]!='(':return False
                if s[i]==']' and a[-1]!='[':return False
                if s[i]=='}' and a[-1]!='{':return False
```

```
a.pop()
return len(a)==0
```

2018-10-10



daydream

👍 1

例子举得不恰当，一堆碟子是可以随意抽取的

2018-10-09

作者回复



还是比较难抽出来吧 摔坏了咋办

2018-10-09



jackeymm

👍 1

java中关于内存的存储分为，静态内存，栈内存，堆内存，静态内存只能存储长度不可变的常量，栈内存用来存储在程序入口处长度可知的局部变量，堆内存用于存储在入口处不可知长度的变量，所以可存储的内存有栈内存和堆内存，又因为堆内存是共享内存，开销比栈内存高，根据最小原则应该存在栈内存中。

第二个问题，我认为我们文章用说的栈主要是说明后进先出的存储逻辑，堆栈中的栈在此基础上还有其他的规则，比如存储的是在程序入口处长度可知的变量等，所以两者最基本的原则是一样的，但堆栈中的栈更丰富

2018-10-09



smx

👍 1

迫不及待想看后面的更新啊，后面的能看看草图，草稿也是极好的啊

2018-10-08



小老鼠

👍 1

嵌套函数中的变量也是用栈的技术吧！

2018-10-08



thewangzl

👍 1

JVM中的“栈”应该有两个。

一个是每个线程中方法调用用到的栈。该栈以栈帧为元素，当调用一个方法时，会把方法相关的局部变量表、操作数栈、方法返回地址等信息封装到栈帧中，把该栈帧入栈；当方法执行结束后，把该栈帧出栈。

第二个栈就是栈帧中的操作数栈。JVM的解释执行引擎是“基于栈的执行引擎”，是因为JVM的指令都是对操作数栈中的元素进行入栈出栈操作。

两者应该都是标准的栈。

2018-10-08



ZerahMu

👍 1

深夜更，占沙发，国庆玩嗨了，落下三节课，明天地铁开始补了，晚安

2018-10-08



404



脚虫

👍 0

对我来说理解有些困难，所以姜威的笔记给了我很大的帮助的。给了我更好完善笔记的构架，以及用不同方式解释加深理解和记忆。真的有的人不喜欢看不看就好，划掉不过两秒的事情。

2018-10-10



勤劳的小胖子-libo

👍 0

1:因为是临时变量，也就是只是同一层的作用范围之内有效，函数返回的时候自动使之无效，所以要使用栈。另外，可能跟底层硬件sp结构相对应。

2:我猜在jvm自己申请了一段内存作栈堆，自己来控制何时何地清理，而不是栈由操作系统默认的，堆由程序员自己控制，主要是出于性能以及出于对指针以及引用更智能化的收集吧

2018-10-09



传说中的成大大

👍 0

哈哈 突然想到 如果不把函数调用放到栈区里面取 那么所用的内存啥的不会释放，也就不会存在什么生命周期之说，竟然不释放，这么搞下去跟内存泄漏没啥区别了呀 不知道理解得对不对

2018-10-09



传说中的成大大

👍 0

操作系统原理书上也写了 线程是进程的组成部分 所以 老师说操作系统给线程分配内存一段独立的内存空间应该是不对的把？

2018-10-09

作者回复

线程栈是不是内存空间 他是分配给进程的 所有线程共享的 还是分配给线程 线程独享的？

2018-10-10



传说中的成大大

👍 0

操作系统中最小的资源分配单位不是进程么？ 线程应该是执行在进程之上吧 不然为啥一个程序的多线程之间可以共享同一个全局变量？

2018-10-09

作者回复

我现在说的是局部变量呢

2018-10-10



狼的诱惑

👍 0

对于每次留下的思考题都希望老师在n (n>1) 天后给出权威的答案，谢谢。

2018-10-09



SunshlnW

👍 0

思考题：

1、为什么使用栈，这个问题有点类似老师前面介绍的，是为了给特定场景的抽象。操作系统函数调用符合后进先出，先进后出等特点，利用栈方便管理，同时可以防止保留太多接口，防止程序执行出错。

2、JVM里的栈是栈这种数据结构的应用吧，本质感觉没有多大区别。
希望老师下节课可以其进行讲解！

2018-10-09



objcoding

👍 0

用单项链表实现了一个链式栈（建议极客开发一个代码风格的留言样式）：

```
public class SinglyLinkedStack<V> {
    private int count; // 栈中元素个数
    private Node<V> bottom = null; // 栈底
    private Node<V> top = null; // 栈顶

    private static class Node<V> {
        V value;
        Node<V> next;
        public Node(V node) {
            this.value = node;
            this.next = null;
        }
    }

    public boolean push(V value) {
        if (bottom == null) {
            this.bottom = new Node<>(value);
            this.top = bottom;
        } else {
            Node<V> l = top;
            Node<V> newNode = new Node<>(value);
            this.top = newNode;
            l.next = newNode;
        }
        count++;
        return true;
    }

    public V pop() {
        if (top == null) {
            return null;
        }
        V value = top.value;
        if (bottom == top) {
            bottom.value = null;
            bottom = null;
        }
    }
}
```

```
top.value = null;
top = null; // help gc
count--;
return value;
}
int pos = 1;
Node<V> n = bottom;
while (pos != (count - 1)) { // 查找到倒数第二个节点
pos++;
n = n.next;
}
top.value = null; // help gc
top.next = null;
n.next = null;
top = n;
count--;
return value;
}
}
```

2018-10-09



polk

👍 0

用2个栈实现计算公式，实现浏览器返回前进，挺巧妙的。问题是，我想不到啊。

2018-10-09

作者回复

应计算公式的不好想 但是浏览器的还好吧 多看多想 慢慢就能想到了

2018-10-09



王威人 🤔

👍 0

函数的递归调用也是用的栈技术吧

2018-10-09



zixuan

👍 0

为什么函数要用栈来保存变量，原因很简单：在调用链中，后被调(call)的函数要先返回(return)，亦即后入先出！

打破这一规则的有其他场景有异常处理、协程等。

2018-10-09



青柠

👍 0

1.函数调用满足后进先出的原理。比如main->a->b,这样的话就是b先执行，然后退出栈，接下来一次是a,main。如果用其它数据结构，比如数组和链表，可以随机访问，没有单方向操作这个限制条件，多次操作容易出错。时间复杂度方面，数组还好，如果是按调用顺序存储的，可以直接根据下标取最后进入的值，但是链表就要每次都遍历一次，找到当前结点的前结

点，时间复杂度变为 $O(n)$ 。

2.jvm的栈和这里的栈是一个概念，调用方法和存储局部变量。就像其它语言存储调用函数和局部变量一样的概念。

2018-10-09



良辰美景

0

为什么要用栈:

- 1: 栈操作简单，只需要维护一个栈顶指针，而队列需要维护队首与队尾
- 2: 函数中的操作数入栈顺序应该是经过编译器调整的吧？比如下面的代码：

```
int a = 11;
int b = 12;
int c = 13;
int res = a + c;
int ret = b + res;
```

2018-10-09



yaxin

0

顺序栈扩容，最好情况时间复杂度为1，最坏情况时间复杂度是 n 。

假如原始栈大小为 k 。

栈满时，假如扩容至原来容量的2倍，即扩容后容量为 $2k$ 。

执行扩容操作，需要申请大小为 $2k$ 的内存，然后执行 k 次的搬移操作，时间复杂度为 k 。

扩容后的 $k-1$ 次入栈操作，时间复杂度为1。

将扩容操作的时间复杂度 k ，均摊在之后的 $k-1$ 次入栈操作上，可知均摊复杂度为1,即最好情况时间复杂度。

如 $2k$ 容量的栈满员，就要再次进行扩容操作，扩容后栈容量为 $4k$ 。

扩容操作的时间复杂度是 $2k$ ，但接下来的 $2k-1$ 次操作的时间复杂度为1，所以入栈操作的均摊时间复杂度是1。

2018-10-09



objcoding

0

老师， $3+5*8-6$ ，为什么不是把所有操作数栈和运算符栈都入栈了在进行计算呢？像您讲解的先乘法和加法对比优先顺序计算完再将减号入栈再计算，这样做是不是可以节省栈内存空间了？

2018-10-09

作者回复

都先入栈好像不对吧

2018-10-09



孙立中

0

我觉得就是一个东西。文中讲到的加减乘除运算，和Java的函数调用栈很像啊。运算符类比方法，数字类比变量。还是一个道理的，方法可以嵌套，类比运算符的优先级。就是

2018-10-09



objcoding

通俗易懂，感谢老师！

2018-10-08

0



Never too late

希望每一次新的一课之中可以包含前一节课问题的答案

2018-10-08

0



罗爱军

老师，动态扩容顺序栈例子也解释了04节的课后习题了，数组拷贝均摊后时间复杂度为 $O(1)$ ，数组插入的时间复杂度也为 $O(1)$ ，所以整体的时间复杂度为 $O(1)$

2018-10-08

0

| 作者回复

是的 套路都是一样的

2018-10-09



Zix

局部变量的声明顺序要和使用顺序相反，效率最高？老师，是这样的吗？

2018-10-08

0

| 作者回复

没听说过这个。应该没啥影响吧

2018-10-08



传说中的成大大

编译器就是通过两个栈来实现的。其中一个保存操作数的栈，另一个另一个是保存运算符的栈。我们从左向右遍历表达式，当遇到数字，我们就直接压入操作数栈；当遇到运算符，就与运算符栈的栈顶元素的栈顶元素进行比较

根据你讲的遇到操作数入栈 $3+5*8-6$ 首先 将3入操作数栈，此时操作数栈中就只有一个元素 向右读到+号 此时 运算符栈内还没有运算符 怎么比较优先级，比较了优先级以后，假设认定优先级高了 此时操作数栈中就一个元素 怎么取得两个元素 这一块一来就懵了 还需要老师和大佬们指教一下

2018-10-08

0

| 作者回复

第一个操作符直接入栈 不用比较

2018-10-09



wean

基础不够，对课后思考不是很明白，希望老师可以讲解一下

2018-10-08

0



junwen.luo

stackoverflow

2018-10-08

0



BitlNit

👍 0

JVM中的栈的本质和文中栈是一样的，JVM中栈是线程不共享，每个线程都有一个栈，栈中存储的是函数调用链，每一个函数在栈中组成一个栈帧，栈帧中存储该函数的局部变量值，引用，返回地址等。

对于一个调用链main->a->b，即main调用a，a再调用b，则在栈中从栈低到栈顶存储main的栈帧，a的栈帧和b的栈帧，当b运行完成后，则b栈帧就出栈。

2018-10-08



🐱 您的好友William 🐱

👍 0

老师讲的公式实现的那道题其实也可以用分而治之的思想，在函数里面嵌套自己实现，其实也是栈，但是应该是一个栈。。。遥想当年面试，实习面试官问了这道题，我写了无数个函数，悔不当初啊。。。哎。。。

2018-10-08



吕宁

👍 0

老师好，关于平均复杂度和摊还分析可不可以这么理解？前者是极端情况 $O(1)$ ，所以不能平均分摊到 n 个case上。典型例子为线性查找；后者是极端情况为 $O(n)$ ，而且紧跟着 n 个 $O(1)$ 的操作，可以均摊，故平均为 $O(1)$ 。典型例子为可扩容数组的插入？

2018-10-08



熊先生口

👍 0

打卡，理论已学完，下面就是实操了，下午抽时间，完成顺序栈和链式栈，然后基于自己的栈，实现四则运算和括号嵌套合法性检测。

2018-10-08



徐凯

👍 0

楼主 局部变量和临时变量的区别 c++返回对象时会有个临时对象 这个应该和函数内的局部变量区分开来。

2018-10-08



涛

👍 0

对于思考题 栈的本质是解决 需要有顺序访问的一种数据结构，程序的运行是按照顺序执行的。当然其他的数据结构也能实现顺序访问，只是代价比栈高而已😄

2018-10-08



五岳寻仙

👍 0

函数调用为何要用栈保存临时变量？

我觉得是基于函数调用“先进后出”的逻辑：主调函数先放入内存中，当遇到被调函数时，被调函数被放入内存，创建变量，执行完运算并销毁局部变量后返回结果，主调函数才能继续往下运行。

2018-10-08



Reis



浏览器的前进后退很像命令模式中的命令栈的设计。

课后思考题都不确定，等老师和大牛来解惑。

2018-10-08