Table of Contents

==========

《Problem Solving with Algorithms and Data Structures using Python》中文版

1.英文原版下载地址：http://pan.baidu.com/s/1mibqAP6

2.英文网页交互版：http://interactivepython.org/runestone/static/pythonds/index.html

目标

- 回顾计算机科学的概念，编程和解决问题
- 理解抽象在解决问题中的作用
- 理解和实现抽象数据类型
- 回顾Python编程语言

1.2快速开始

这些年来我们思考编程的方式经历了许多变化，自从人类到机器以及第一台电子计算机需要电缆和开关传达指令开始起。在社会的许多方面,计算机技术的变化为计算机科学家提供越来越多的工具和平台来实践他们的想法。在先进方向如更快的处理器、高速网络和更大容量内存即使有计算机科学家参与，创造的复杂程度螺旋上升。在所有这些快速发展中,一些基本的原则是始终不变的。计算涉及的科学使用电脑来解决问题。

You have no doubt spent considerable time learning the basics of problem-solving and hopefully feel confident in your ability to take a problem statement and develop a solution. 毫无疑问你花了大量时间学习解决问题的基本技能后,用自己来描述问题和。 你也知道写计算机程序一般是困难的。 The complexity of large problems and the corresponding complexity of the solutions can tend to overshadow the fundamental ideas related to the problem-solving process.

本章详述了两个重要领域的内容。首先,它回顾了计算机科学、算法研究、数据结构在内的适应框架体系,特别是,为什么我们需要学习这些方面和理解这些方面是如何帮助我们成为解决问题效率更高的人。第二,我们复习了Python编程语言。虽然我们不能提供一份详细的,详尽的参考,但我们将会在剩下的章节里为基本的结构和解法给出例子和解释。

目标

- 理解算法分析的重要性
- 可以用"大O"描述执行时间
- 了解Python常见操作列表和字典的“大O”执行时间
- 了解如何用Python实现算法分析
- 了解如何去评测一个简单的python程序.

什么是算法分析

计算机科学系的学生比较他们的程序是非常常见的。你可能注意到计算程序看起来非常相似，特别是简单的那个。 一个有趣的问题经常出现。当两个看起来不同的程序解决相同的问题时，是否一个程序比另外一个更好？

为了能够回答这个问题, we need to remember that there is an important difference between a program and the underlying algorithm that the program is representing. As we stated in Chapter 1, an algorithm is a generic, step-by-step list of instructions for solving a problem. It is a method for solving any instance of the problem such that given a particular input, the algorithm produces the desired result. A program, on the other hand, is an algorithm that has been encoded into some programming language. There may be many programs for the same algorithm, depending on the programmer and the programming language being used.

进一步探讨这种差异, 思考下面的函数。这个函数解决了一个常见的问题，计算从1到整数n的和。 The algorithm uses the idea of an accumulator variable that is initialized to 0. The solution then iterates through the n integers, adding each to the accumulator.

```
def sumOfN(n):
   theSum = 0
   for i in range(1,n+1):
       theSum = theSum + i

   return theSum

print(sumOfN(10))
```

目标

- 理解栈、队列、双端队列、和链表的抽象数据类型
- 能够用Python列表实现栈、队列、双端队列的抽象数据类型
- 理解实现的基础线性数据结构的性能
- 理解前缀、中缀、后缀表达式的格式
- 用栈来计算后缀表达式
- 用栈实现中缀表达式转后缀表达式
- 用队列实现基本的时间仿真
- 学会根据问题性质，选择使用栈队双向队等合适的数据结构
- 能够用列表的ADT实现链表的节点和指针
- 能够比较链表和list的性能

什么是线性数据结构?

我们开始数据结构的学习，从四种简单而功能强大的结构开始。栈、队、双向队和列表是一种数据的集合，它的元素根据自己被加入或删除的的顺序排列。当一个元素加入集合之后，它就与之前和之后加入的元素保持一个固定的相对位置。这种数据集合叫做线性数据结构。

既然是线性，就有两头。有时叫做“左侧”或“右侧”，有时叫做“前端”“后端”，叫做"顶部“和”底部“也无不可。因为名字不重要。重要的是数据结构增加删除数据的方式，特别是增删的位置。例如，一种结构可能只允许从一头增加成员，另一种结构则两头都行。

这种方式的变化在计算机科学中构成了多种非常有用的数据结构，他们出现在各种算法和重要问题的解决方案中。

3.3 什么是栈？

栈（也叫下推栈）一种线性有序的数据元素集合，它的特点是，数据的增加删除操作都在同一端进行。进行操作的这一端，我们一般叫做“顶”，另一端叫做“底”。

栈的底部很有象征性，因为元素越接近底部，就意味着在栈里的时间越长。最近进来的，总是最早被移走，这种排列规律叫做先进后出，综合为LIFO。所以栈的排序是按时间长短来排列元素的。新来的在栈顶，老家伙们在栈底。

栈的例子经常看到。比如自助餐厅的盘，人们总是从上面拿盘子，拿走一个后面的人再拿下面的一个，（服务员端来一些新的，又堆在上面了）。又如一堆书，（图1）你只能看到最上面一本的封面，要看下面一本，就要把上面的先拿 走。图2是另一种栈，这保存了同个主要的python语言数据对象。

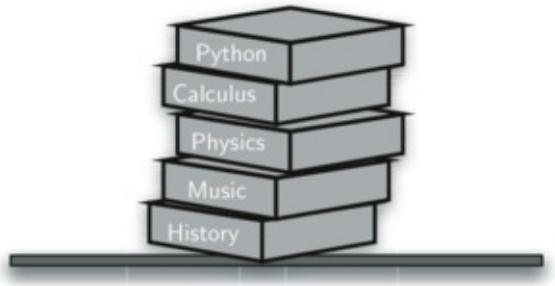

图1：Figure 1: A Stack of Books

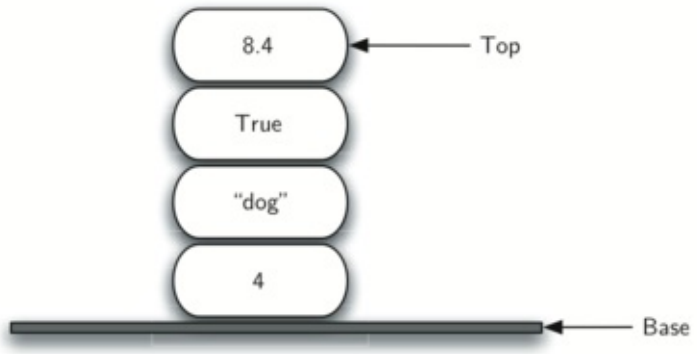

图2：A Stack of Primitive Python Objects

与栈有关的思想来自生活中的观察，假设你从一张干净的桌子开始，一本一本地放上书，这就是在建立栈。当你一本一本地拿走，想像一下，是不是先进后出？由于这种结构具有翻转顺序的作用，所以非常重要。图3显示了Python数据栈，加入和移走的顺序。

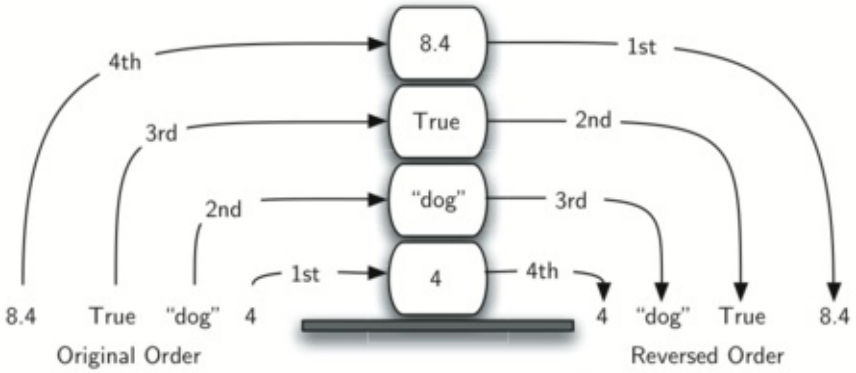

图3：The Reversal Property of Stacks

栈这种翻转性，在你用电脑上网的时候也用到了。浏览器软件上都有“返回”按钮，当你从一个链接到另一个链接，这时网址（URL）就被存进了栈。正在浏览的页就存在栈顶，点“返回”的时候，返回到刚刚浏览的页面。最早浏览的页面，要一直到最后才能看到。

3.4 栈的抽象数据类型

下面的结构和操作定义了栈的抽象数据类型。如前所述，栈是结构化的，有序的的数据集，它的增删操作都在叫在”栈顶“的一端进行，存储顺序是LIFO。栈的操作方法如下：

- `Stack()` ,构造方法，创建一个空栈，无参数，返回值是空栈。
- `Push(item)` 向栈顶压入一个新数据项，需要一个数据项参数，无返回值。
- `pop()` 抛出栈顶数据项，无参数，返回被抛出的数据项，栈本身发生变化。
- `Peek()` 返回栈顶数据项，但不删除。不需要参数，栈不变。
- `isEmpty()` 测试栈是否空栈。不需要参数，返回布尔值。
- `size()` 返回栈内数据项的数目，不需要参数，返回值是整数。

例如，s是一个空栈，表1是一系列的操作，栈内数据和返回值。注意栈顶在右侧。

Stack Operation	Stack Contents	Return Value
`s.isEmpty()`	`[]`	`True`
`s.push(4)`	`[4]`	
`s.push('dog')`	`[4,'dog']`	
`s.peek()`	`[4,'dog']`	`'dog'`
`s.push(True)`	`[4,'dog',True]`	
`s.size()`	`[4,'dog',True]`	`3`
`s.isEmpty()`	`[4,'dog',True]`	`False`
`s.push(8.4)`	`[4,'dog',True,8.4]`	
`s.pop()`	`[4,'dog',True]`	`8.4`
`s.pop()`	`[4,'dog']`	`True`
`s.size()`	`[4,'dog']`	`2`