算法简介算法基础知识（主要是时间复杂度的大 O 表示法）以及选择排序

一、算法简介：算法是算法（Algorithm）是指解题方案的准确而完整的描述，是一系列解决问题的清晰指令 ,算法代表着用系统的方法描述解决问题的策略机制。也就是说，能够对一定规范的输入，在有限时间内获得所要求的输出。如果一个算法有缺陷，或不适合于某个问题，执行这个算法将不会解决这个问题。不同的算法可能用不同的时间、空间或效率来完成同样的任务。一个算法的优劣可以用空间复杂度与时间复杂度来衡量。（摘自百度百科）

二、大 O 表示法：在定义中已经提到了，算法的优劣往往用时间复杂度和空间复杂度表示，我们今天主要讲的就是表示时间渐进复杂度的大 O 表示法。

编程者都希望自己的算法高效，但是不是只有在算法写完用数据检验后，才可以知道自己的算法高效与否呢？显然不是，我们可以在实际编写算法前进行估测。这就要用到大 O 表示法了。（毕竟我们都不希望自己写的程序又复杂又贼慢）

那么，时间复杂度该如何计算呢？（以下部分文字，图片摘自网络）

时间复杂度与语句的执行次数有着很大的关联。

具体方法如下：

大 O 推导法：

1. 用常数 1 取代运行时间中的所有加法常数
2. 在修改后的运行函数中，只保留最高阶项
3. 如果最高阶项存在且不是 1，则去除与这个项相乘的常数

执行次数函数	阶	术语描述
12	O(1)	常数阶
2n+3	O(n)	线性阶
$3n^2+2n+1$	$O(n^2)$	平方阶
$5\log_2^n+20$	$O(\log_2^n)$	对数阶
$2n+3n\log_2^n+19$	O(nlogn)	$n\log_2^n$阶
$6n^3+2n^2+3n+4$	$O(n^3)$	立方阶
2^n	$O(2^n)$	指数阶

时间复杂度所耗费的时间是：

$O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3) < O(2^n) < O(n!) < O(n^n)$

具体例子会在课上给出。

三．选择排序

之前的学习让我们明白了时间复杂度，现在让我们真正的用所学知识分析一个很有名的算法吧。

选择排序的思路很简单，来让我说明一下。假设你有一本 n 页散了的书，你十分想把它按照顺序排列起来。你不妨将它从头到尾翻一遍，第一遍挑出页数最大的一页，并将它抽离放在一边，在第二次翻的时候同样将页数最大的一页抽离（之前已经把第一大的抽走了），放在第一次抽离的上面，以此类推，重复 n 次后就可以将它排好序。

Assignment:

1.1　假设有一个包含 128 个名字的有序列表，你要使用二分查找在其中查找一个名字，请 问最多需要几步才能找到？

1.2　上面列表的长度翻倍后，最多需要几步？

1.3　在电话簿中根据名字查找电话号码。（电话簿有序排列）

1.4　在电话簿中根据电话号码找人。（提示：你必须查找整个电话簿。）

1.5　阅读电话簿中每个人的电话号码。

1.6　阅读电话簿中姓名以 A 打头的人的电话号码。这个问题比较棘手。

1.7　给出选择排序的时间复杂度。

1.8　请使用数组在课后实现选择排序。（现在是选做，但它是你之后学习中绕不开的坎，相信我）