# 算法简介

## 引言

算法是一组完成任务的指令；二分查找是一种算法，其输入是一个有序的元素列表，每次判断大小进行折半查找；每次查找都会排除一半的元素，二分查找算法最多需要，前提是有序的列表；对数运算是幂运算的逆运算；的逆运算是；

表示线性时间；是对数时间。

## 大O表示法

不同的算法随着运算规模的增加，算法的运行时间的增速是不同的，使用大O表示法用来表示算法运行时间的增速。就是使用O表示在规模n线性增加时，运行时间相对于规模的函数。O只是一个标记。大O表示法指示的是最糟糕的运行时间；常见的运行时间：对数时间，O(n)线性时间，O(n\*logn),；

# 选择排序

## 2.1 内存的工作原理

内存地址是线性的。

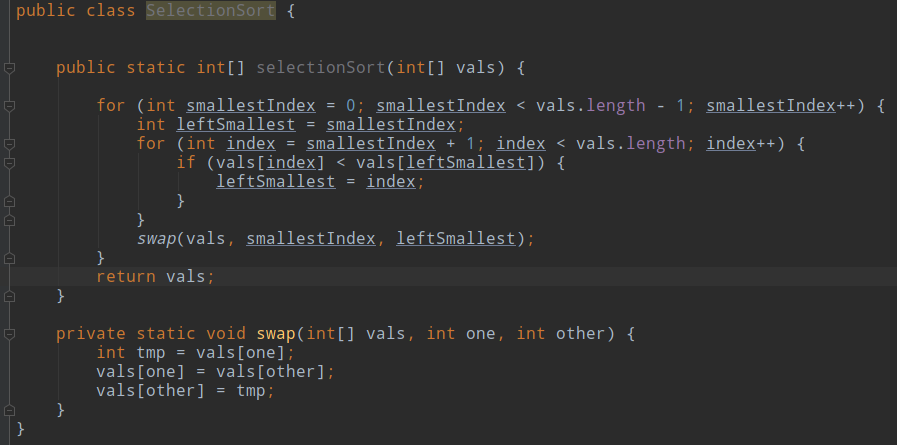
## 2.2 数组与链表

数组需要预留空闲内存，满时需要迁移扩容；还有内存浪费，主要的缺点就是迁移与内存浪费。

链表记录下一个节点的地址，方便新增。链表不支持跳跃读取，只能顺序访问，数组可以。

随机访问/顺序访问；Map 混合的数据结构。

## 2.3 选择排序



# 递归

## 3.1 递归

所有的递归都可以用循环重写。递归让代码更容易理解，循环的性能更高。

## 3.2 基线条件与递归条件

因为递归是自己调用自己，为了避免无限循环，需要定义终止条件，所以每个递归函数都分成2个部分：

* 基线条件；
* 递归条件；

## 3.3 栈

函数调用栈的概念。

# 快速排序

分而治之（divide and conquer, D & C）。

## 4.1 分而治之

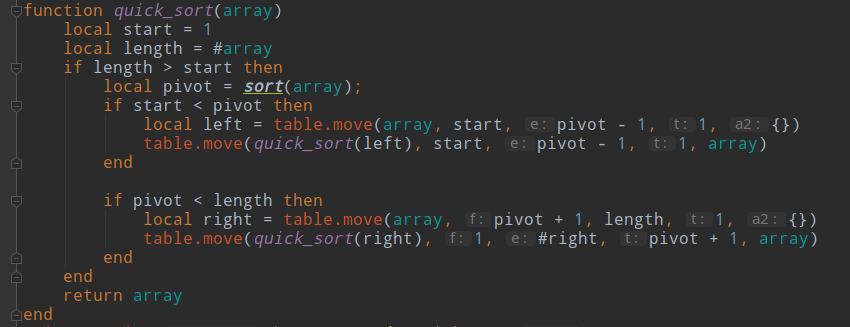
DC是递归的，解决问题包含2个步骤：

* 找出基线条件，这种条件尽可能简单；
* 不断将问题分解（缩小规模），直到符合基线条件；

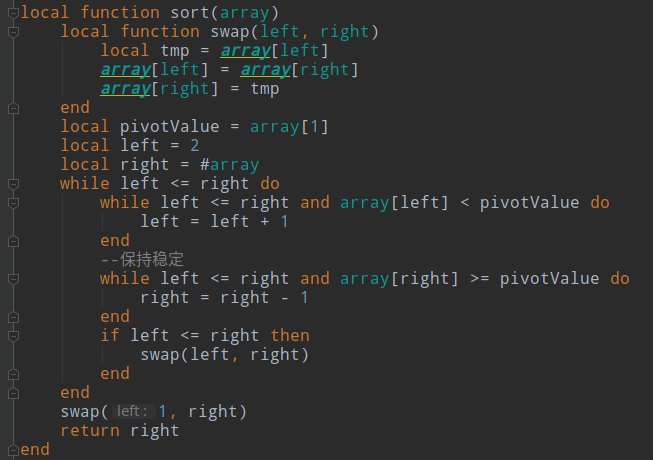
分解的问题模式与元问题相同。

## 4.2 快速排序

快速排序就是DC模式，这种递归的排序方式可以用归纳证明的方式证明。



递归调用，每次都有一个sort函数获取pivot值。



## 4.3 再谈大O表示法

# 第五章 散列表

## 5.1 散列函数

散列表就是散列函数+数组；也叫做字典，映射表；散列函数特点：

* + 同样的输入映射到同样的索引；
  + 不同的输入映射不同的所以；
  + 散列函数知道数组的大小。

## 5.3 冲突

拉链法解决散列冲突；

## 5.4 性能

良好的散列表需要有：

* + 较低的填装因子；
  + 良好的散列函数。

填装因子=元素数/位置总数，通常来说要<1，如果过大，代表链表就比较长，影响性能；需要调整长度，通常是扩大一倍；重新将元素hash到新的散列表中。

# 广度优先搜索

解决最短路径的算法是广度优先算法。

## 6.3 广度优先算法

就是使用队列完成遍历，注意的是，需要记录遍历完的结点，否则碰到环的时候会无限循环。

广度优先搜索就是寻找目标值，按照从近到远的顺序依次遍历，直到寻找到目标值。因为涉及到顺序，所以，使用到队列。

拓扑排序。

# 第七章 迪克斯特拉算法

计算带权有向图的单元最短路径。

算法步骤：

1. 找出最便宜的节点；
2. 对于节点的邻居，找出是否有前往的最短的路径，如果有，更新其开销；
3. 重复这个过程，直到对图中每个节点都这么做；
4. 计算最终路径。

迪克斯特拉算法，是按照层次遍历所有的边然后计算到每个节点的最短路径，然后依次计算到最终的节点。

## 7.2 术语

加权图与非加权图，非加权图最短的路径是用广度优先搜索，加权图是使用迪克斯特拉算法；迪克斯特拉算法只能作用于有向无环图。

## 7.3 换钢琴

这个算法不能处理负权边的情况，需要使用贝尔曼-福德算法。

# 贪婪算法

贪婪算法是用所有的局部最优解来构造出全局的最优解，但是这个可能不是真正的解，但是是接近于最优解的解。他的典型思想是通过分拆为2部分，一部分是可以直接计算的部分，一部分是子问题部分，可以直接计算的部分算出最优解，子问题部分重复最优解的计算。实际还是一种分治算法。

NP完全问题的含义是：没有快速的解，必须计算出所有情况，然后找出最优解。

NP完全问题的特点：

* 随着元素数量的增加，速度变得特别慢；
* 涉及所有的组合；
* 不能将问题分解为小问题，必须考虑各种可能的情况；
* 涉及序列或者集合。

# 动态规划

## 9.1 背包问题

给定背包的重量取最大价值的问题，如果通过排列组合的方式，时间复杂度是，如果用贪婪算法的方式，只能找到接近最优解的结局方案，动态规划先解决子问题。

## 9.3 最长公共子串

动态规划可以在给定的约束条件下找到最优解，问题可以分解为彼此独立且离散的子问题时，使用动态规划。

* 动态规划都涉及网格；
* 网格中的值就是要优化的值；
* 每个网格都是一个子问题。

使用网格解决了（特别巧妙）。

# K最近临算法