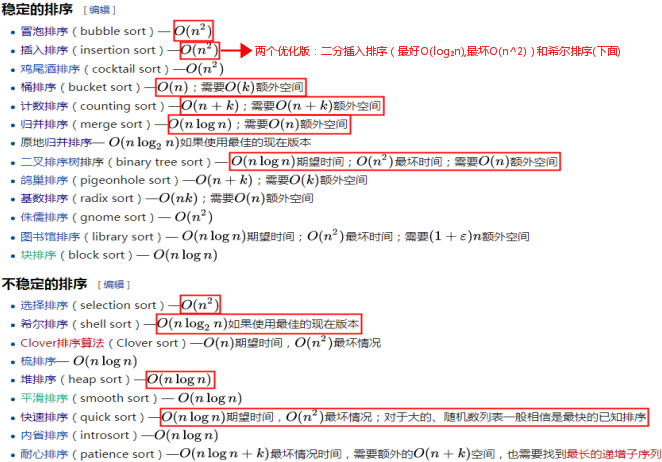
算法思想

* 时间复杂度
* 空间复杂度

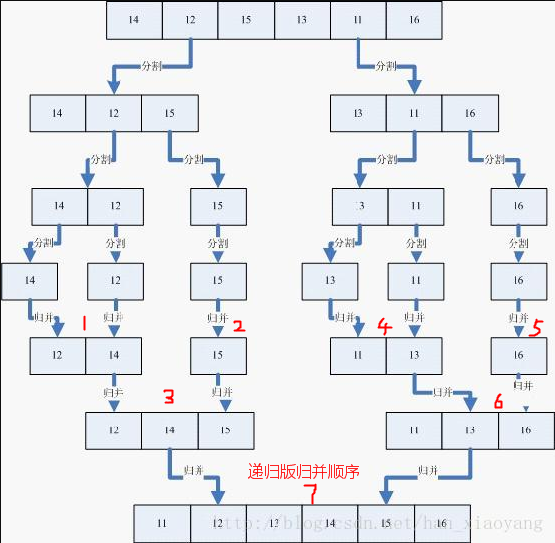
O(1) < O(logn) < O(n) < O(nlogn) < O(n^k) < O(n!) < O(n^n)

# 排序

## 常用的排序算法（维基百科）



注：归并排序理解图



## Java中的排序算法

### Arrays.sort()

如果数组长度大于等于286且连续性（连续升序和连续降序性）好的话，就用**归并排序**，如果大于等于286且连续性不好的话就用**双轴快速排序**。如果长度小于286且大于等于47的话就用**双轴快速排序**，如果长度小于47的话就用**插入排序**。

数据轴 -> **插入排序**【47】**双轴快速排序**【286】连续性好？**归并排序**：**双轴快速排序。**

### Collections.sort()

如果LegacyMergeSort.userRequested为true的话就会使用**归并排序**，可以通过代码设置为true（System.setProperty("java.util.Arrays.useLegacyMergeSort", "true");）如果不为true的话就会用一个叫**TimSort**的排序算法。

#### TimSort

<https://blog.csdn.net/yangzhongblog/article/details/8184707>

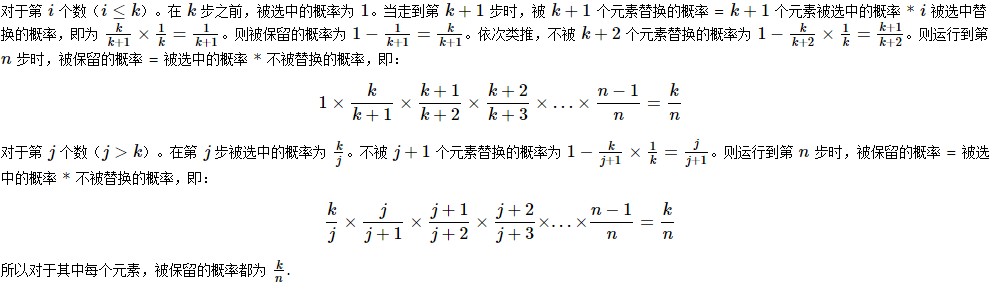
# 查找

# 蓄水池抽样

### 步骤：

1. 假设数据序列的规模为n，需要采样的数量的为k；
2. 首先构建一个可容纳k个元素的数组，将序列的前k个元素放入数组中；
3. 然后从第i=k+1个元素开始遍历池，随机从[0,i+1)抽取一个下标r，若r<k，则将r位置的抽样替换为i位置的样本；否则什么都不做。当遍历完所有元素之后，数组中剩下的元素即为所需采取的样本。

### 证明



### 场景

蓄水池算法适用于对一个不清楚规模的数据集进行采样。以前在某个地方看到过一个面试题，说是从一个字符流中进行采样，最后保留 10 个字符，而并不知道这个流什么时候结束，且须保证每个字符被采样到的几率相同。用的就是这个算法。