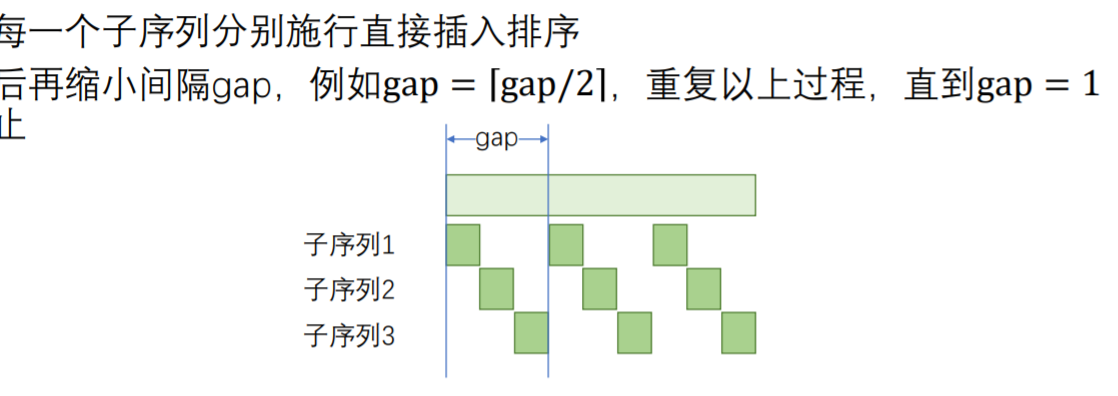
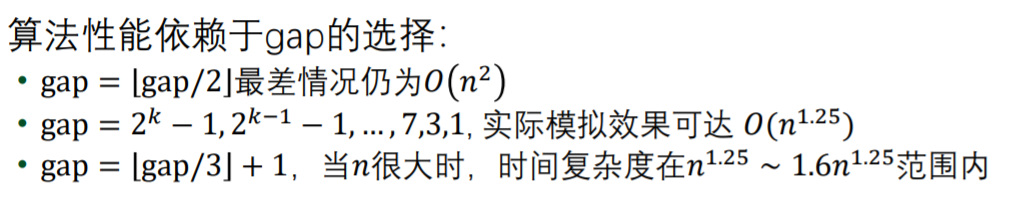
**排序算法的衡量指标**

* 稳定性：关键字相等的对象在排序前后的次序保持不变。
* 内排序与外排序
* 平均比较次数、平均移动、平均辅助存储空间

**插入排序**（代码实现）

* 直接插入排序(Insertion sort)
* 折半插入排序（二分法排序）
* 希尔排序(Shell sort)：不稳定





**交换排序**（代码实现）

* 冒泡排序(Bubble sort)：
* 快速排序(QuickSort)：不稳定

次序统计（查找次序为i的元素）

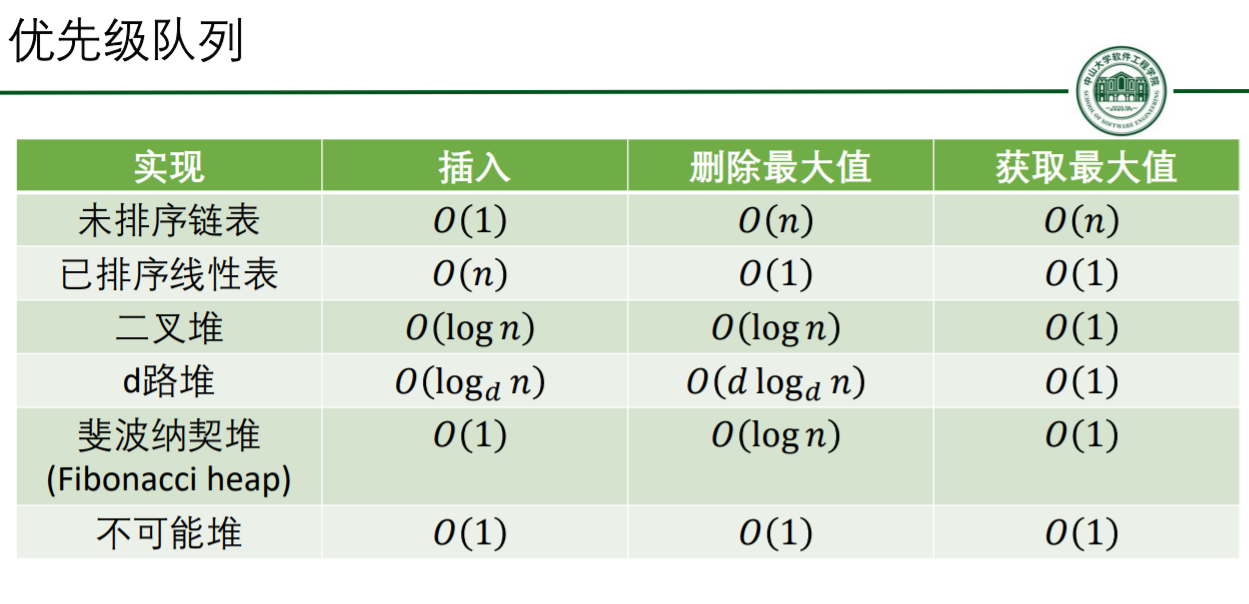
* 分治法：由快速排序算法启发

改进→5元组划分的方法：可保证理论上是线性复杂度但常数项偏大

**堆排序**(heapsort) （代码实现）：最好的排序算法之一，最差情况不会变为平方复杂度

二叉堆：完全二叉树的顺序表表示，索引从1开始

堆顺序：子节点的关键字不能大于父节点的关键字，又分为大根堆和小根堆



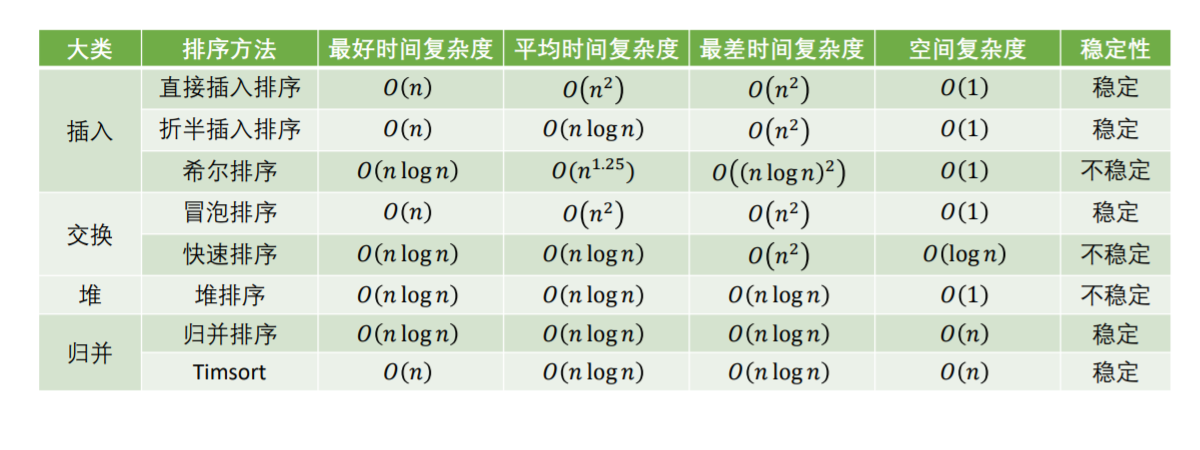
先建一个大根堆，再不断地删除最大值，将其保留在末尾，执行n-1次完成。

**归并排序**：分成多个有序子序列再合并

Timsort排序算法（本质是归并排序）：最快且是稳定的通用排序算法

改进：

1. 不要让大分区和小分区合并，要让小分区之间合并。利用栈实现，将分区长度和起始位置入栈。
2. 设置最小分区长度，如果分区长度小于最小分区长度，和前一个分区合并。
3. 如果出现逆序，直接反转。
4. 为避免多次无效的比较，设置驰骋(Gallop)模式。当一分区中连续min\_gallop个元素小于另一分区某个元素，进入驰骋模式，逐次倍增的方式寻找位置，又称指数搜索。



**计数排序**(Counting Sort) （代码实现）：线性复杂度

步骤：

1. 第一次遍历找到数据范围。
2. 第二次遍历生成计数数组，保存每个数据出现次数。
3. 由计数数组得到累计数组。
4. 第三次遍历结合累计数组得到有序结果。

**桶排序**(Bucket Sort) （代码实现）：元素值为浮点数（0~1）

**基数排序**(Radix sort) ：面向特殊对象

**拓扑排序**：对一个有向图构造拓扑序列的过程称为拓扑排序，常用于顶点活动网络( AOV)

从AOV网中选择一个没有前驱的顶点并且输出（有多个时可用栈辅助），从AOV网中删去该顶点，重复上述步骤直到全部输出。

关键路径：在AOE网中，从始点到终点具有 最大路径长度（该路径上的各个活动所持续的时间之和）的路径。

关键活动（边）：在关键路径中。找法常为，算出每个活动的最早发生时间和最迟发生时间，两者之差（时间余量）为零则为关键活动。而活动的最早发生时间和最迟发生时间常由事件（结点）的最早发生时间和最迟发生时间求得。