八种排序算法：



#### 说明

* 当原表有序或基本有序时，直接插入排序和冒泡排序将大大减少比较次数和移动记录的次数，时间复杂度可降至O（n）；
* 而快速排序则相反，当原表基本有序时，将蜕化为冒泡排序，时间复杂度提高为O（n2）；
* 原表是否有序，对简单选择排序、堆排序、归并排序和基数排序的时间复杂度影响不大。

稳定的排序算法：冒泡排序、插入排序、归并排序和基数排序

不稳定的排序算法：选择排序、快速排序、希尔排序、堆排序

<https://www.jianshu.com/p/3ea5f8f49ca8>

堆排序：<http://cmsblogs.com/?p=4695>

堆排序需要两个过程，一是建立堆，二是堆顶与堆的最后一个元素交换位置。所以堆排序有两个函数组成。一是建堆函数，二是反复调用建堆函数以选择出剩余未排元素中最大的数来实现排序的函数。

总结起来就是定义了以下几种操作：

* 最大堆调整（Max\_Heapify）：将堆的末端子节点作调整，使得子节点永远小于父节点
* 创建最大堆（Build\_Max\_Heap）：将堆所有数据重新排序
* 堆排序（HeapSort）：移除位在第一个数据的根节点，并做最大堆调整的递归运算

对于堆节点的访问：

* 父节点i的左子节点在位置：(2\*i+1);
* 父节点i的右子节点在位置：(2\*i+2);
* 子节点i的父节点在位置：floor((i-1)/2);

**注:** x>>1 是位运算中的右移运算, 表示右移一位, 等同于x除以2再取整, 即 x>>1 == Math.floor(x/2) .

以上,  
①. 建立堆的过程, 从length/2 一直处理到0, 时间复杂度为O(n);  
②. 调整堆的过程是沿着堆的父子节点进行调整, 执行次数为堆的深度, 时间复杂度为O(lgn);  
③. 堆排序的过程由n次第②步完成, 时间复杂度为O(nlgn).

| 平均时间复杂度 | 最好情况 | 最坏情况 | 空间复杂度 |
| --- | --- | --- | --- |
| O(nlog2n)O(nlog2⁡n) | O(nlog2n)O(nlog2⁡n) | O(nlog2n)O(nlog2⁡n) | O(1) |

Tips: **由于堆排序中初始化堆的过程比较次数较多, 因此它不太适用于小序列.** 同时由于多次任意下标相互交换位置, 相同元素之间原本相对的顺序被破坏了, 因此, 它是不稳定的排序.