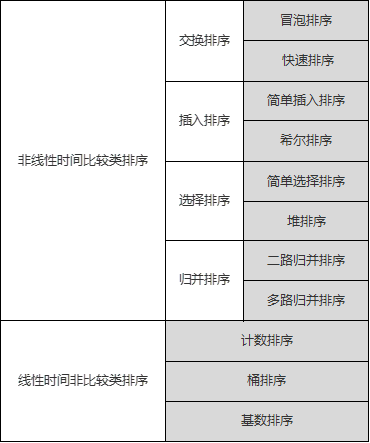
**1. 排序算法概述**

**1.1 概念**

**排序算法：**如何使得记录按照要求排列的方法。   
排序算法在很多领域得到相当地重视，尤其是在大量数据的处理方面。一个优秀的算法可以节省大量的资源。

**1.2 分类**

**非线性时间比较类排序：**通过比较来决定元素间相对次序，其时间复杂度不能突破 O。   
**线性时间非比较类排序：**不能通过比较来决定元素间相对次序，可以突破基于比较排序的时间下限，以线性时间运行。





**稳定排序：**冒泡、插入、归并

**2. 排序种类**

**2.1 冒泡排序**

原理：每次比较相邻两个数据的大小，若前面比后面的大，则将二者交换顺序。   
实现：需要两层 for 循环，第一层从数组第一个到倒数第二个，第二层数组第二个到最后一个。   
代码：

function bubbleSort(arr) {

var len = arr.length;

for (var i = 0; i < len - 1; i++) {

for (var j = 0; j < len - 1 - i; j++) {

if (arr[j] > arr[j+1]) { // 相邻元素两两对比

var temp = arr[j+1]; // 元素交换

arr[j+1] = arr[j];

arr[j] = temp;

}

}

}

return arr;

}

**2.2 选择排序**

原理：首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，存放到已排序列末尾，以此类推，直到所有元素排序完毕。   
实现：需要两层 for 循环，第一层从数组第一个到倒数第二个，第二层数组第二个到最后一个。   
代码：

function selectionSort(arr) {

var len = arr.length;

var minIndex, temp;

for (var i = 0; i < len - 1; i++) {

minIndex = i;

for (var j = i + 1; j < len; j++) {

if (arr[j] < arr[minIndex]) { // 寻找最小的数

minIndex = j; // 将最小数的索引保存

}

}

temp = arr[i];

arr[i] = arr[minIndex];

arr[minIndex] = temp;

}

return arr;

}

**2.3 插入排序**

原理：假定第一个就是最小值，然后取第二个值与第一个值比较产生排序后的序列，然后再取第三个值与排序后的序列进行比较插入到对应的位置，依次类推。

function insertionSort(arr) {

for(let i = 1;i<arr.length;i++){

let preIndex = i-1,

current = arr[i];

while(preIndex>=0 && arr[preIndex]>current){

arr[preIndex+1] = arr[preIndex];

preIndex--;

}

arr[preIndex+1] = current;

}

console.log(arr);

return arr;

}