1 排序

不同排序算法根据时间复杂度可以分为:1) 简单的排序算法 ($O(n^2)$);2) 先进的排序算法 ($O(nlogn)$);3) 基数排序 ($O(d \times n)$).

根据不同原理大体可以分为:1) 插入排序;2) 交换排序;3) 选择排序;4) 归并排序;5) 计数排序.

存储方式有: 数组, 链表, 数组 + 地址向量

1.1 稳定与不稳定

在比较过程中, 当出现两个元素相等的情景. 若出现元素的移动, 则为不稳定排序, 否则则为稳定排序算法.

1.2 内部排序和外部排序

待排序元素数量较小, 在内存中实现的排序过程为内部排序; 当待排序元素数量较多, 内存无法一次性容纳所有元素, 在排序实现过程中需要访问外部存储的排序算法称为外部排序.

1.3 几种排序算法介绍

1.3.1 插入排序

思路 对于一个待排序序列, 维护一个有序的子序列. 不断的将待排元素插入到该有序子序列中, 直至所有元素插入完毕, 最终待排序列为有序序列.

代码

insert sort

```
    template <typename T>
    void InsertSort(T* arr, int n)
    {
f   or(int i=1;i<n;++i){ // for every item.
            T tmp = arr[i];
            int j = i;  // j must be i.
```

```
                for(;j>0 && arr[j-1]>tmp;j--){ // move
                    to right position.
                arr[j] = arr[j-1];
                }
                arr[j] = tmp;
        }
        }
```

复杂度分析

变体

1.3.2 希尔排序

思路 希尔排序法使用了增量的概念. 首先采用比如增量 $h = n/2$, 而 n 为序列总长度. 每一轮只对位置为 $a + h \times i$ 上的元素进行选择排序 (其中 a 为第一个元素, 显然 $0 \leq a < h$. 而 i 为整数, 显然 $0 \leq a + h \times i < n$). 而增量 h 逐渐减小, 直至 $h = 1$. 至此整个序列有序.

代码

代码

insert sort

1.3.3 选择排序法

思路 对于当前位置, 应该从后继元素中, 选择合适的元素 (最大或最小) 插入到当前位置. 不断重复这个过程, 直至整个序列有序.

代码

select sort

```
template <typename T>
void SelectSort(T* arr, int n)
```

```
{
    for(int i=0;i<n;++i)  // for every item
    {
        int min_i=i;  // initial min_idx
        for(int j=i;j<n;++j)  // for every item behind
            current item.
        {
            if(arr[j]<arr[min_i]){  // find min_idx.
                min_i=j;
            }
        }
        T tmp = arr[i];  // exchange current item and
            minimum item.
        arr[i] = arr[min_i];
        arr[min_i]=tmp;
    }
}
```

复杂度分析

变体