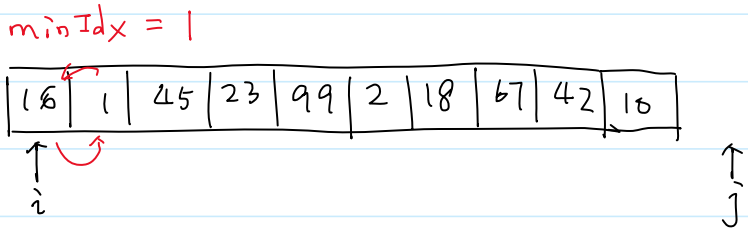
选择排序

2024年5月9日 9:37

数列[16, 1, 45, 23, 99, 2, 18, 67, 42, 10]

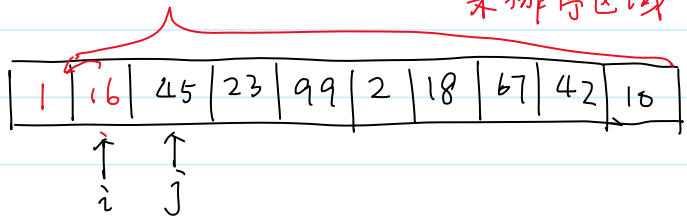
(小→大)

第一次选择



swap(i, minIdx)

第二次选择



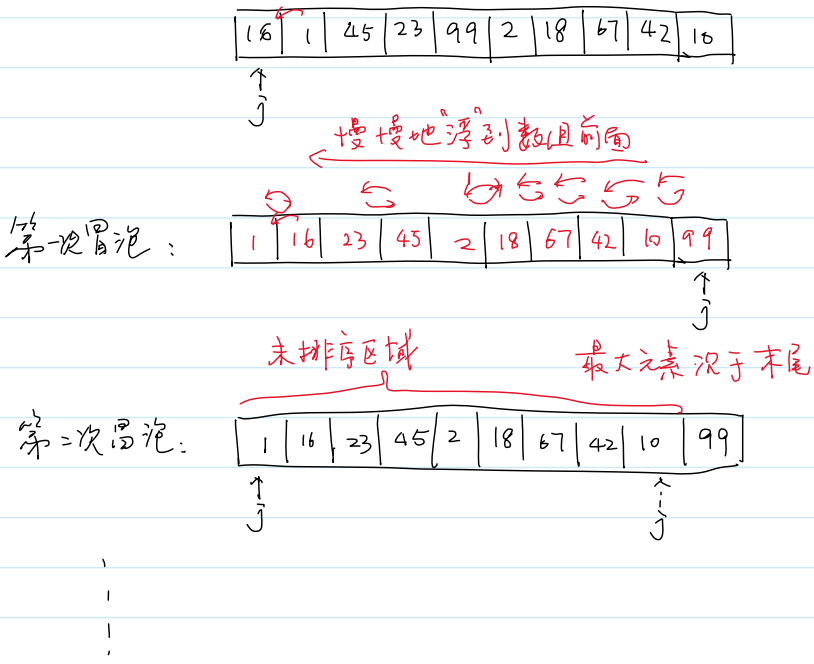
⋮

第n-1次选择

冒泡排序

2024年5月9日 10:02

[16, 1, 45, 23, 99, 2, 18, 67, 42, 10]



顺序时，i<j，arr[i] ≤ arr[j]

逆序时，i<j，arr[i] >arr[j]

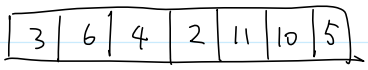
排序，将逆序时减为0

插入排序

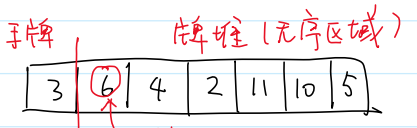
2024年5月9日 10:18

#1. 算法步骤

数列 [3, 6, 4, 2 ,11, 10, 5]

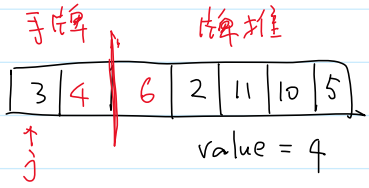


第一次插入：

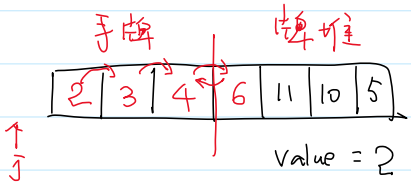


比较，找第一个小于等于 value 的元素，插入到它的后面.

第二次插入.



第三次插入



第 n-1 次插入

#2. 实现

```c
/*****************************************************************************/
/*                              插入排序                                     */
/*****************************************************************************/
void insertion_sort(int arr[], int n) {
    for (int i = 1; i < n; i++) {   // i: 要插入元素的索引
        int value = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > value) {
            arr[j + 1] = arr[j];
            j--;
        }   // j == -1 || arr[j] <= value
        arr[j + 1] = value;

        // print_array(arr, n);
    }
}
```

#3. 分析

时间复杂度

最好：原数组有序　　$O(n)$　　+判定

比较：$n-1$

交换　0

最坏：原数组逆序　　$O(n^2)$

比较：$1+2+\cdots+(n-1)=\frac{n(n-1)}{2}$

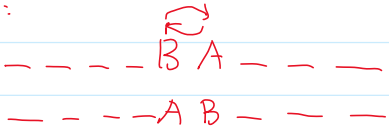
交换：$1+2+\cdots+(n-1)=\frac{n(n-1)}{2}$

平均：逆序对 = 顺序对 = $C_n^2/2=\frac{n(n-1)}{4}$　　$O(n^2)$

比较：大于等于交换次数，小于等于 $\frac{n(n-1)}{2}$

交换：逆序对 = $\frac{n(n-1)}{4}$

交换相邻元素：



逆序对--

有序对++

空间复杂度：$O(1)$

↓

原地排序

稳定性：稳定

（交换相邻的元素）

#4. 使用场景

① 数组长度比较小。

② 当原数组基本有序（离最终的位置很近），插入排序可以在 $O(n)$ 时间内完成排序。

希尔排序

2024年5月9日 11:27

数列 [16, 1, 45, 23, 99, 2, 18, 67, 42, 10] n = 10

5, 2, 1, 0 → gap 序列 (n/2, n/4, n/8, ---, 1, 0) 希尔序列

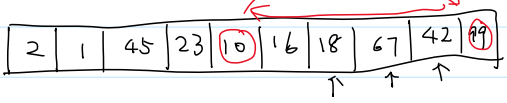
10, 99 (E)

2, 16 (A) (D) 23, 42

每个人的手牌是有序的

(B) 1, 18 (C) 45, 67

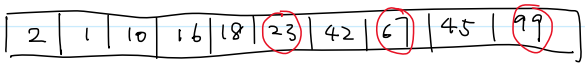
先距离交换，减少交换的次数



gap = 5

让数组基本有序

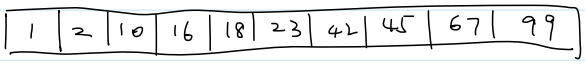
2, 10, 18, 42, 45 (A) 1, 16, 23, 67, 99 (B)



gap = 2

(A) ⇒ 简单的插入排序

O(n)



gap = 1

#2、实现.

```c
void shell_sort(int arr[], int n) {
    // gap: n/2, n/4, ..., 1, 0
    int gap = n >> 1;
    while (gap) {
        // 组间插入排序（gap个人轮流抓拍）
        for (int i = gap; i < n; i++) { // 牌堆[gap, n)
            int value = arr[i];
            // 保证手牌有序
            int j = i - gap;
            while (j >= 0 && arr[j] > value) {
                arr[j + gap] = arr[j];
                j -= gap;
            } // j < 0 || arr[j] <= value
            arr[j + gap] = value;
        }
        print_array(arr, n);
        // 缩小gap
        gap >>= 1;
    } // gap == 0
}
```

#3. 分析

时间复杂度：和 gap 序列相关，一般来说，平均情况小于 $O(n^2)$。

空间复杂度：$O(1)$

稳定性：不稳定

（发生长距离的交换）

牛顿性不稳定一点，换取了时间