# 直接插入排序

## 基本思想

直接插入排序（straight insertion sort）有时简称为插入排序（insertion sort），是增量法的一种典型应用实例，其基本思想如下：

1. 对于序列A区间[0,n-1]的排序问题，假设区间[0,n-2]的排序问题已经解决。
2. 遍历区间[0,n-2]，寻找一个合适的位置插入元素A[n-1]。
3. 区间[0.n-1]的排序任务完成。 （假设序列长度为length，则2≤n≤length）

最坏情况下（对严格逆序的序列进行排序），需要比较和移动元素的次数为n(n-1)/2；最好情况下（序列已经有序），需要比较的次数为n-1，移动元素的次数为0。

2020年3月9日：将待排序序列划分成两个区间——已排序区间和待排序区间，不断从待排序区间选择元素插入已排序区间。

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | ... | i-1 | i | i+1 | i+2 | ... | n-3 | n-2 | n-1 |
| 5 | 6 | 9 | 11 | ... | 25 | 34 | 8 | 16 | ... | 61 | 1 | 27 |
|  | | | | | | | | | | | | |
| 5 | 6 | 8 | 9 | 11 | ... | 25 | 34 | 16 | ... | 61 | 1 | 27 |
| ... | | | | | | | | | | | | |
|
| 1 | 5 | 6 | 8 | 9 | 11 | 16 | ... | 25 | 27 | 34 | 61 | ... |

## 评价

### 时间复杂度

O( n2 )

### 空间复杂度

O( 1 )

### 算法稳定性

直接插入排序是一种稳定的排序算法。

## 实例

适用直接插入排序算法对序列89 45 54 34 29 90 34 68进行升序排序。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 第1次增量  [89] 45 54 34 29 90 34 68  [45 89] 54 34 29 90 34 68 | 第2次增量  [45 89] 54 34 29 90 34 68  [45 54 89] 34 29 90 34 68 | 第3次增量  [45 54 89] 34 29 90 34 68  [34 45 54 89] 29 90 34 68 | 第4次增量  [34 45 54 89] 29 90 34 68  [29 34 45 54 89] 90 34 68 | 第5次增量  [29 34 45 54 89] 90 34 68  [29 34 45 54 89 90] 34 68 |
| 第6次增量  [29 34 45 54 89 90] 34 68  [29 34 34 45 54 89 90] 68 | 第7次增量  [29 34 34 45 54 89 90] 68  [29 34 34 45 54 68 89 90] | 排序结束  [29 34 34 45 54 68 89 90] |  |  |

## 优化

### 设置哨兵位

仔细分析直接插入排序的代码，会发现虽然每次都需要将数组向后移位，但是在此之前的判断却是可以优化的。不难发现，每次都是从有序数组的最后一位开始，向前扫描的，这意味着，如果当前值比有序数组的第一位还要小，那就必须比较有序数组的长度n次。这个比较次数，在不影响算法稳定性的情况下，是可以简化的：记录上一次插入的值和位置，与当前插入值比较。若当前值小于上个值，将上个值插入的位置之后的数，全部向后移位，从上个值插入的位置作为比较的起点；反之，仍然从有序数组的最后一位开始比较。

### 二分查找法

优化直接插入排序的核心在于：快速定位当前数字待插入的位置。在一个有序数组中查找一个给定的值，最快的方法无疑是二分查找法，这种方式有一个致命的缺点，导致虽然效率高出普通的直接插入排序法很多，但是却不被使用，就是这种定位方式找到的位置，最终形成的数组会打破排序算法的稳定性。既然一定会打破稳定性，那么为什么不使用更优秀的希尔排序呢？