算法作业 1

张昊　　1927405160

2021 年 9 月 26 日

2.1-2　如算法1所示。

算法 1 非升序的插入排序

输入: n 个数的一个序列 $A < a_1, a_2, \cdots, a_n >$

输出: 输入序列的一个排列 $< a'_1, a'_2, \cdots, a'_n >$，满足 $a'_1 \geq a'_2 \geq \cdots \geq a'_n$。

```
for j = 2 to A.length do
    key = A[j]
    // 将 A[j] 插入到已排序序列 A[1 ··· j − 1] 中
    i = j − 1
    while i > 0 and A[i] < key do
        A[i + 1] = A[i]
        i = i − 1
    end while
    A[i + 1] = key
end for
```

2.1-4　如算法2所示。

算法 2 二进制整数相加

输入: n 元数组 $A =< a_1, a_2, \cdots, a_n >$，$n$ 元数组 $B =< b_1, b_2, \cdots, b_n >$，分别代表二进制整数 a, b。其中，$a_i, b_i \in \{0, 1\}$，$1 \leq i \leq n$，二进制整数采用高位优先法 (big-endian) 存储。

输出: $(n + 1)$ 元数组 $C =< c_1, c_2, \cdots, c_n >$，满足 $c = a + b$。其中 a, b, c 分别为 n 元数组 A, B, C 所表示的二进制数。

```
C = array[A.length + 1]
ex = 0  // 进位
for i = A.length to 1 do
    number = A[i] + B[i] + ex
    C[i + 1] = number%2
    ex = number/2
end for
C[1] = ex
return C
```

2.2-2 如算法3所示。

算法 3 选择排序

输入: n 个数的一个序列 $A < a_1, a_2, \cdots, a_n >$

输出: 输入序列的一个排列 $< a'_1, a'_2, \cdots, a'_n >$ ，满足 $a'_1 \leq a'_2 \leq \cdots \leq a'_n$。

```
1: for i = 1 to A.length − 1 do
2:     small = i
3:     for j = i+1 to A.length do        // 找到 A[i + 1 · · · A.length] 中最小元素的下标
4:         if A[j] < A[small] then
5:             small = j
6:         end if
7:     end for
8:     SWAP(A[i], A[small])        // 交换 A[i], A[small]
9: end for
```

2.3-6 不可以。修改后的插入排序算法可以用算法4描述（二分查找算法略）。

算法 4 使用二分查找的插入排序

输入: n 个数的一个序列 $A < a_1, a_2, \cdots, a_n >$

输出: 输入序列的一个排列 $< a'_1, a'_2, \cdots, a'_n >$ ，满足 $a'_1 \leq a'_2 \leq \cdots \leq a'_n$。

```
1: for j = 2 to A.length do        // 循环体执行了 n − 1 次
2:     key = A[j]
3:     // 将 A[j] 插入到已排序序列 A[1 · · · j − 1] 中
4:     i = BINARY-SEARCH(A, 1, j − 1, key)        // Θ(lg n)
5:     while j > i do        // Θ(n)
6:         A[j] = A[j − 1]
7:         j = j − 1
8:     end while
9:     A[i] = key
10: end for
```

可知，影响时间复杂度的因素有二：

- 寻找插入位置（算法4中第 4 行）；
- 插入动作引起的元素移动（算法4中 5-8 行）。

使用二分查找可以使得第一个操作的时间复杂度最坏情况达到 $\Theta(\lg\ n)$；但是元素移动这一操作的时间复杂度最坏仍为 $\Theta(n)$。外层 **for** 循环共执行了 $n-1$ 次，故整个外层 **for** 循环的时间复杂度仍为 $\Theta(n^2)$，即算法的时间复杂度为 $\Theta(n^2)$。