算法设计与分析第一次作业

2127405048 方浩楠

2023 年 9 月 21 日

Algorithm 1: Rewrite the INSERTION-SORT procedure to sort into nonincreasing instead of non-decreasing order.
重写过程 INSERTION-SORT, 使之按非升序 (而不是非降序) 排序

Data: A sequence of n numbers $< a_1, a_2, \cdots, a_n >$

Result: A permutation (reordering) $< a_1^{'}, a_2^{'}, \cdots, a_n^{'} >$of the input sequence such that $a_1^{'} \geq a_2^{'} \geq \cdots a_n^{'}$

for $j \leftarrow 2$ **to** $A.length$ **do**
  $key \leftarrow A[j]$;
  $i \leftarrow j - 1$;
  while $i > 0$ *and* $A[i] < key$ **do**
    `// just need to change` $A[i] > key$ `to` $A[i] < key$
    $A[i+1] \leftarrow A[i]$;
    $i \leftarrow i - 1$;
  end
  $A[i+1] \leftarrow key$
end

Algorithm 2: Consider the problem of adding two n-bit binary integers, stored in two n-element arrays A and B. The sum of the two integers should be stored in binary form in an (n + 1) element array C. State the problem formally and write pseudocode for adding the two integers.
考虑把两个 n 位二进制整数加起来的问题, 这两个整数分别存储在两个 n 元数组 A 和 B 中, 这两个整数的和应按二进制形势存储在一个 (n+1) 元数组 C 中. 请给出该问题的形式化描述, 并写出伪代码

Data: A and B integer arrays, A.length = B.length = n
Result: C integer array,C.length = n+1

```
carry ← 0;
for i ← n to 1 do
    C[i + 1] = (carry + A[i] + B[i])mod2;
    if carry + A[i] + B[i] ≥ 2 then
        carry ← 1;
        else
            carry ← 0;
        end
    end
end
C[1] = carry
```

题目 2 的形式化描述:

$$\sum_{i=1}^{n+1} C[i] * 2^{i-1} = \sum_{i=1}^{n} A[i] * 2^{i-1} + \sum_{i=1}^{n} B[i] * 2^{i-1} + carry$$

Algorithm 3: Selection Sort 插入排序

Data: An n-element array A

for $i = 1$ **to** $n - 1$ **do**
    $min = i$;
    for $j \leftarrow i + 1$ **to** n **do**
        if $A[j] < A[min]$ **then**
            $min = j$
        end
    end
    swap $A[min]$ and $A[i]$
end

Algorithm 4: Observe that the while loop of lines 5–7 of the INSERTION-SORT procedure in Section 2.1 uses a linear search to scan (backward) through the sorted subarray A[i$\cdots$ j-1]. Can we use a binary search (see Exercise 2.3-5) instead to improve the overall worst-case running time of insertion sort to θ(nlgn)?
能否使用二分查找来改进插入排序?

for $i = 2$ **to** n **do**
    $position$ =**BinarySearch**$(i, n - 1, A[i])$;
    for $i = i - 1$ **to** $j = position$ **do**
        swap$(A[j], A[j + 1])$
    end
end

二分查找改进插入排序无法降低插入排序的时间复杂度, 时间复杂度 $\Theta(n)$ 仍为 n^2
虽然二分查找可以使我们寻找元素所在位置的复杂度从 $\Theta(n)$ 降低到 $\Theta(log(n))$, 但是移动该元素依旧需要 $\Theta(n)$ 的时间复杂度