Sorting (IV): Insertion Sort

CSCD 300 – Data Structures

Eastern Washington University

Goal

We will learn the mechanism of the Insertion Sort algorithm and then analyze its time complexity in the best as well as in the worst case.

Outline

1. Insertion sort
2. The time complexity
3. Question

Insertion sort

Basic idea

- Incrementally sort the first i numbers, $i = 2, 3, \ldots$
- When we are sorting the first i numbers, the first $i - 1$ numbers have already been sorted, so we only need to find the right position for the ith number in the first $i - 1$ numbers, which is easy to do.
- after the nth number finds its position in the first $n - 1$ numbers, the whole sequence is sorted.

An example

(a) 5 $\boxed{2}$ 4 6 1 3

(b) 2 5 $\boxed{4}$ 6 1 3

(c) 2 4 5 $\boxed{6}$ 1 3

(d) 2 4 5 6 $\boxed{1}$ 3

(e) 1 2 4 5 6 $\boxed{3}$

(f) 1 2 3 4 5 6

Pseudocode [1]

```
INSERTION_SORT(A)
{
   /* Each step sorts A[0...j],
      given A[0...j-1] is sorted already */
   for j = 1 to n-1
      key = A[j]

      /* insert A[j] into the right location in A[0...j-1] */
      // travel toward left for efficiency
      i = j - 1
      while i >= 0 and A[i] > key
         A[i+1] = A[i]
         i = i - 1
      A[i+1] = key
}
```

[1] We use 0-based indexing.

The time complexity

The best case

The best case is where the input sequence is already sorted. In that case, each step of the inner `while` loop will stop immediately, so the total time cost is $O(n)$.

The worst case

The worst case is where the input sequence is in the descending order. In that case, the number of steps of the inner `while` loop for each value of j, $j = 1, 2, \ldots, n-1$, would be: $1, 2, \ldots, n-1$. So the total time cost is:

$$1 + 2 + \ldots + (n-1) = \frac{n(n-1)}{2} = O(n^2)$$

Question

How do you use the Insertion sort if the data sequence is saved in a singly linked list ?