CE2101/ CZ2101: Algorithm Design and Analysis

Week 1: Review Lecture

Ke Yiping, Kelly

Content

- Introduction to Sorting
- Insertion Sort

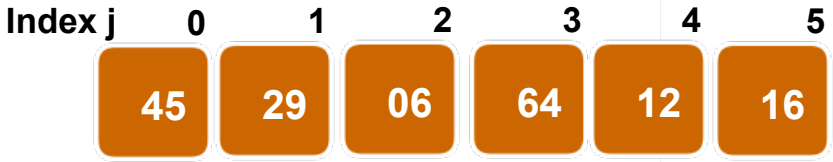
Introduction to Sorting

- Sorting is to arrange a set of records so that their key values are in ascending or descending order.
- It is important to learn sorting, because:
  - Sorting has important applications
  - Ideas of sorting can be used for other algorithms
- Objective is to design sorting algorithms with:
  - Minimum number of key comparisons or swaps
  - Minimum usage of memory

usually contradicting

Insertion Sort (Recap)

- Insertion sort uses the incremental approach.
- **Main idea:** Repeatedly pick up an element *x* to insert into a sorted sub-array on the left side, by comparing *x* with its left neighbour. If they are out of order, swap them; otherwise, insert *x* there.



Insertion Sort (Recap)

- Insertion sort uses the incremental approach.
- **Main idea:** Repeatedly pick up an element *x* to insert into a sorted sub-array on the left side, by comparing *x* with its left neighbour. If they are out of order, swap them; otherwise, insert *x* there.
- **Time complexity analysis:**
  - Best case: □(*n*), when input array is already sorted.
  - Worst case: □(*n*2), when input array is reversely sorted.
  - Average case: □(*n*2).

Insertion Sort Performance

- **Strengths:**
  - Good when the unordered list is almost sorted.
  - Need minimum time to verify if the list is sorted.
  - Fast with linked storage implementation: no movement of data.
- **Weaknesses:**
  - When an entry is inserted, it may still not be in the final position yet.
  - Every new insertion necessitates movements for some inserted entries in ordered list.
  - When each slot is large (e.g., a slot contains a large record of 10Mb), movement is expensive.
  - Less suitable with contiguous storage implementation.

Exercise

5 5 7 9 13

- Insertion Sort is used to sort an input array A = [7, 5, 13, 9, 5, 2, 10] in ascending order. How many key comparisons and swaps are performed respectively when the element "2" reaches the first position of the array? **[AY1920S1]**

write down intermediate steps

key	comparison	swap
5	1	1
13	1	0
9	2	1
5	4	3
2	5	5
total	13	10

Exercise

- A sorting algorithm is called stable if elements of equal keys remain their original order after being sorted by the algorithm. Decide if Insertion Sort is stable. If you believe it is not stable, give an example to show that the order of two elements with equal keys is changed by the algorithm. **[AY1718S1]**

stable

input: [9, 7a, 2, 7b, 6]

stable: [2, 6, 7a, 7b, 9]

unstable: [2, 6, 7b, 7a, 9]

Exercise

- Given an array A of n elements, an inversion in A is defined as a pair of indices (i, j) such that i < j and A[i] > A[j].
- Show that the running time of Insertion sort algorithm, measured by the number of swaps between array elements, is O(n + I), where I is the number of inversions in the input array of n elements. **[AY1516S2]**

```
for (int i=1; i < n; i++)
    for (int j=i; j > 0; j--) {
      if (slot[j].key < slot[j-1].key)
         swap(slot[j], slot[j-1]);
      else break;
    }
```

i 此时为 index+1.

1 3 6 7 9 2 4 8

- Show that the average-case time complexity of Insertion sort, measured in the number of key comparisons performed on an array of size n, is approximately n2/4. **[AY1617S2]**

1. for each element: (i-th element)

$$\text{Time Complexity} = \frac{1}{i}\sum_{x=1}^{i} x = \frac{1}{i} \times \frac{(1+i)\times i}{2} = \frac{i+1}{2}$$

2. i can be from 1 to n.

$$\text{Tin total} = \sum_{y=1}^{n-1} \frac{y+1}{2} = \frac{\left(\frac{1+1}{2} + \frac{n+1}{2}\right)\times(n-1)}{2} = \frac{\frac{n+3}{2}\times(n-1)}{2} = \frac{n^2+2n-3}{4} \in \frac{n^2}{4}$$

首项 末项 项数