

Principles of Computer Science II

Sorting Algorithms

Ioannis Chatzigiannakis

Sapienza University of Rome

Lecture 5

This algorithm first finds the smallest element in the array and exchanges it with the element in the first position, then find the second smallest element and exchange it with the element in the second position, and continues in this way until the entire array is sorted.



Selection Sort: Example

Original Array	After 1st pass	After 2nd pass	After 3rd pass	After 4th pass	After 5th pass
3 6 8 4 5	1 6 8 4 5	1 3 6 8 5	1 3 4 8 5	1 3 4 5 8	1 3 4 5 8

Selection Sort Code

```
a = [5, 1, 6, 2, 4, 3]
for i in range(0, len(a)):
    min = i
    for j in range(i + 1, len(a) - 1):
        if a[j] < a[min]:
            min = j

    temp = a[j]
    a[j] = a[min]
    a[min] = temp
```



How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?

How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...



How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1) + (n-2) + (n-3) + \dots + 3 + 2 + 1$ comparisons are required



How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1) + (n-2) + (n-3) + \dots + 3 + 2 + 1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required



How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1) + (n-2) + (n-3) + \dots + 3 + 2 + 1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ How much memory is needed ?

How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1) + (n-2) + (n-3) + \dots + 3 + 2 + 1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ How much memory is needed ?
 - ▶ 1 additional slot.

Bubble Sort Algorithm

Bubble Sort is an algorithm which is used to sort N elements that are given in a memory. Bubble Sort compares all the element one by one and sort them based on their values.

- ▶ It is called Bubble sort, because with each iteration the largest element in the list bubbles up towards the last place, just like a water bubble rises up to the water surface.
- ▶ Sorting takes place by stepping through all the data items one-by-one in pairs and comparing adjacent data items and swapping each pair that is out of order.

Bubble Sorting: Example

5	1	6	2	4	3
---	---	---	---	---	---

Lets take this Array.

5	1	6	2	4	3
1	5	6	2	4	3
1	5	2	6	4	3
1	5	2	4	6	3
1	5	2	4	3	6

Here we can see the Array after the first iteration.

Similarly, after other consecutive iterations, this array will get sorted.

Bubble Sort Code

```
a = [5, 1, 6, 2, 4, 3]
for i in range(0, len(a)):
    for j in range(0, len(a) - i - 1):
        if a[j] > a[j+1]:
            temp = a[j]
            a[j] = a[j+1]
            a[j+1] = temp
```

- ▶ The above algorithm is not efficient because as per the above logic, the for-loop will keep executing for six iterations even if the list gets sorted after the second iteration.



How good is Bubble Sort?

- ▶ How many comparisons are required until the list is sorted?



Bubble Sort Code: Version 2

- ▶ We can insert a flag and can keep checking whether swapping of elements is taking place or not in the following iteration.
- ▶ If no swapping is taking place, it means the list is sorted and we can jump out of the for loop, instead executing all the iterations.

```
a = [5, 1, 6, 2, 4, 3]
for i in range(0, len(a)):
    for j in range(0, len(a) - i - 1):
        if a[j] > a[j+1]:
            temp = a[j]
            a[j] = a[j+1]
            a[j+1] = temp
```



How good is Bubble Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1) + (n-2) + (n-3) + \dots + 3 + 2 + 1$ comparisons are required
 - ▶ $\Sigma \frac{n(n-1)}{2}$ comparisons are required



How good is Bubble Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1)+(n-2)+(n-3)+\dots+3+2+1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ Is there a "simple" case ?



How good is Bubble Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1)+(n-2)+(n-3)+\dots+3+2+1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ Is there a "simple" case ?
 - ▶ How many loops are required?



How good is Bubble Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1)+(n-2)+(n-3)+\dots+3+2+1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ Is there a "simple" case ?
 - ▶ How many loops are required?
 - ▶ The list is already sorted



How good is Bubble Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1)+(n-2)+(n-3)+\dots+3+2+1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ Is there a "simple" case ?
 - ▶ How many loops are required?
 - ▶ The list is already sorted
 - ▶ N comparisons are required



How good is Bubble Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1) + (n-2) + (n-3) + \dots + 3 + 2 + 1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ Is there a "simple" case ?
 - ▶ How many loops are required?
 - ▶ The list is already sorted
 - ▶ N comparisons are required
- ▶ How much memory is needed ?

How good is Bubble Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1) + (n-2) + (n-3) + \dots + 3 + 2 + 1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ Is there a "simple" case ?
 - ▶ How many loops are required?
 - ▶ The list is already sorted
 - ▶ N comparisons are required
- ▶ How much memory is needed ?
 - ▶ 1 additional slot.

Insert Sort Algorithm

A simple Sorting algorithm which sorts the list by shifting elements one by one.

- ▶ It has one of the simplest implementation
- ▶ It is efficient for smaller data sets, but very inefficient for larger lists.
- ▶ Insertion Sort is adaptive, that means it reduces its total number of steps if given a partially sorted list, hence it increases its efficiency.
- ▶ It is better than Selection Sort and Bubble Sort algorithms.
- ▶ Like Bubble Sorting, insertion sort also requires a single additional memory space.

Insertion Sort: Example

5	1	6	2	4	3
---	---	---	---	---	---

Lets take this Array.



(Always we start with the second element as key.)

As we can see here, in insertion sort, we pick up a key, and compares it with elements ahead of it, and puts the key in the right place

5 has nothing before it.

1 is compared to 5 and is inserted before 5.

6 is greater than 5 and 1.

2 is smaller than 6 and 5, but greater than 1, so its is inserted after 1.

And this goes on...

Insertion Sort Code

```
a = [5, 1, 6, 2, 4, 3]
for i in range(1, len(a)):
    key = a[i]
    j = i - 1
    while j >= 0 and key < a[j]:
        a[j+1] = a[j]
        j -= 1
    a[j+1] = key
```

- ▶ **key**: we put each element of the list, in each pass, starting from the second element: `a[1]`.
- ▶ using the **while loop**, we iterate, until `j` becomes equal to zero or we find an element which is greater than `key`, and then we insert the `key` at that position.



How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?

How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1) + (n-2) + (n-3) + \dots + 3 + 2 + 1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required



How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1) + (n-2) + (n-3) + \dots + 3 + 2 + 1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ Is there a "simple" case ?



How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1)+(n-2)+(n-3)+\dots+3+2+1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ Is there a "simple" case ?
 - ▶ The list is already sorted
 - ▶ N comparisons are required

How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1)+(n-2)+(n-3)+\dots+3+2+1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ Is there a "simple" case ?
 - ▶ The list is already sorted
 - ▶ N comparisons are required
- ▶ How much memory is needed ?



How good is Insertion Sort?

- ▶ How many comparisons are required until the list is sorted?
 - ▶ 1st loop: $n - 1$
 - ▶ 2nd loop: $n - 2$
 - ▶ ...
 - ▶ $(n-1)+(n-2)+(n-3)+\dots+3+2+1$ comparisons are required
 - ▶ $\sum \frac{n(n-1)}{2}$ comparisons are required
- ▶ Is there a "simple" case ?
 - ▶ The list is already sorted
 - ▶ N comparisons are required
- ▶ How much memory is needed ?
 - ▶ 1 additional slot.

