Understanding Sorting

Ascending or Descending order

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

What is Sorting

- It means arranging the elements / values in specific order – Ascending order or Descending order. For example if we have set of values as –

S1 = [7,4,8,3,1,5]

Then its ascending order will be

S1 = [1,3,4,5,7,8]

And its descending order will be

S1 = [8,7,5,4,3,1]

There are many type of sorting methods available like Bubble, Selection, Insertion, Heap, Quick, Shell etc.

In this chapter, we will learn Bubble and Insertion Sorting

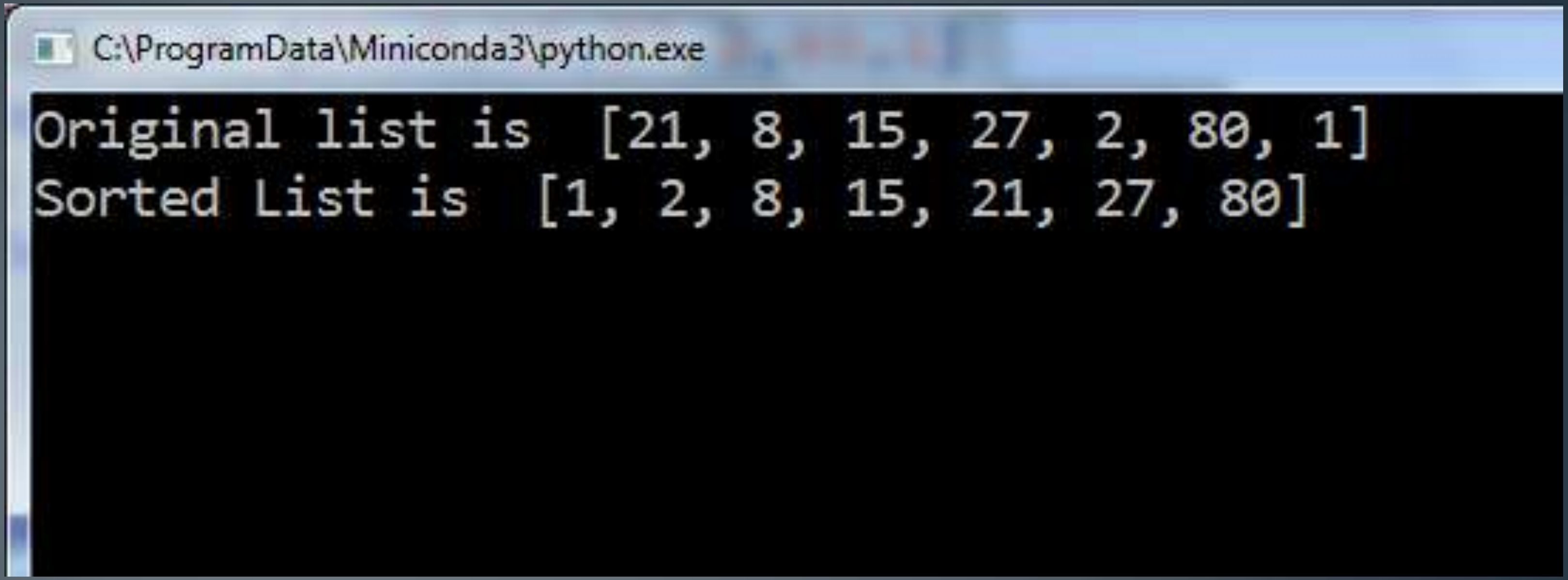
Bubble Sorting

- **You are requested to download Bubble Sorting PDF from this site.**

Python code…

```
aList = [21,8,15,27,2,80,1]
print("Original list is ", aList)
n = len(aList)
for i in range(n):
    for j in range(n-1-i):
        if aList[j]>aList[j+1]:
            aList[j],aList[j+1] = aList[j+1],aList[j]
print("Sorted List is ",aList)
```

Output



```
Original list is  [21, 8, 15, 27, 2, 80, 1]
Sorted List is  [1, 2, 8, 15, 21, 27, 80]
```

Calculating number of operations...

```
1. aList = [21,8,15,27,2,80,1]                      ← 1 OP
2. print("Original list is ", aList)                ← 1 OP
3. n = len(aList)                                   ← 1 OP
4. for i in range(n):                               ← 1 OP AT A TIME (REPEATS 7 TIMES 0-6)
5.     for j in range(n-1-i):                       ← 1 OP AT A TIME, REPEATS 6 TIMES 1ST ,TIME 5, THEN 4,..
6.         if aList[j]>aList[j+1]:                  ← 1 COMP
7.             aList[j],aList[j+1] = aList[j+1],aList[j]   ← 1 SWAP
8. print("Sorted List is ",aList)                   ← 1 OP
```

So, total number of operations will be :

*Operation in Line 1 + Line 2 + Line 3 = **3 Operations***

*+ Operation in Line 4 to 7 = **? (to be calculated)***

*+ Operation in Line 8 = **1 Operations***

Calculating number of operations…

So, total number of operations will be :

Operation in Line 1 + Line 2 + Line 3 = ***3 Operations***

+ Operation in Line 4 to 7 = ***? (to be calculated)***

+ Operation in Line 8 = ***1 Operations***

Let us Calculate number of operations for OUTER FOR LOOP

Operation in Line (4 to 7) x 7 times

= Line 4 = 1 Op

=Line 5 to 7 = Inner for loop

Inner for loop

First time inner loop will execute 6 times, then 5 and so on

So, Line 5 = 1 op
Line 6 = 1 op
Line 7 = 1 op

Inner loop performs 3 operation in single iteration

Now outer for loop operations are (when i= 0)

1 op (Line 4) + 3 op(Line 5 to 7) x 6 times = 19 ops

Calculating number of operations…

Outer for loop operations are (when i= 0)

1 op (Line 4) + 3 op(Line 5 to 7) x 6 times = 19 ops

Outer for loop operations are (when i= 1)

1 op (Line 4) + 3 op(Line 5 to 7) x 5 times = 16 ops

Similarly for (i =2) = 13, (i =3) = 10, (i =4) = 7, (i =5) = 4, (i =6) = 1

Total Operations in outer for loop will be

3 ops + 70 ops + 1 ops = 74 ops

Number of operations increases with the number of elements i.e. n

Total number of comparison in bubble sorting is : $N(N-1)/2 \cong N^2$

Best, worst and average case of Bubble

Best case – when all elements of the sequence are already sorted, in this case no swapping will take place therefor

N^2 comparison + 0 swapping = N^2 operations

Worst case – when all the element of the sequence are in opposite direction, i.e. every comparison result into a swapping.

N^2 comparison + N^2 swapping = $2N^2$ operations

Average case – when the elements are in mixed form

Insertion Sorting

- Have you ever observed your subject teacher after he collected your answer script arrange all the answer sheet in the order of your roll numbers, what does he do often?
- Take first two sheets and put them in correct order – swapping them if needed. Then take the third sheet and put it in the correct position of previous arranged sheets; then take the fourth and so on.. And the process continues till the sheets are arranged in desired order.
- **It is nothing but the INSERTION SORTING…**
- **For More Detail, Please download Insertion Sorting PDF**

Python Code – Insertion Sorting

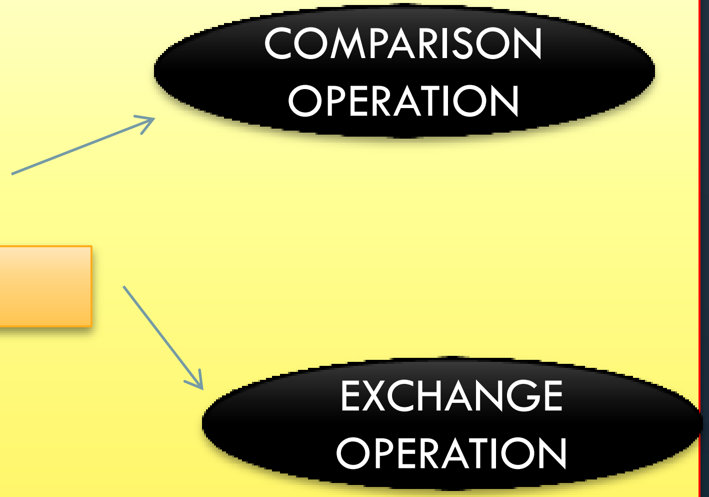
```
aList = [21,8,15,27,2,80,1]
print("Original list is ", aList)
n = len(aList)
for i in range(1,n):
          key = aList[i]
          j=i-1
          while j>=0 and aList[j]>key:
                    aList[j+1]=aList[j]
                    j=j-1
          aList[j+1]=key
          print("After pass", i,":",aList)
print("Sorted List is ",aList)
```

Number of Comparison

- In Bubble sorting we already discussed that the 2 main costly / time taking operations are COMPARISON AND SWAPPING/EXCHANGE. So let us understand these with Insertion Sorting

```
•

•
while j>=0 and aList[j]>key:
            aList[j+1]=aList[j]
                j=j-1
        aList[j+1]=key
print("Sorted List is ",aList)
```



COMPARISON OPERATION

EXCHANGE OPERATION

The number of comparison

- During the first iteration of outer loop in the inner loop there is only 1 comparison
- During the second iteration of outer loop, there will be 2 comparison and so on..
- During the N-1 iteration of outer loop, N-1 comparison
- So, the maximum number of comparison will be:
- $1+2+\ldots(N-2)+(N-1)$
- $= (N \times (N-1))/2$
- $= (N^2 - N)/ 2 < N^2$

- It means there can be maximum N^2 comparison in insertion sort

The number of exchanges

- In first iteration, there is at most 1 exchange
- In second iteration there is at most 2 exchange and so on…
- In N-1 iteration there is at most N-1 exchange
- So,
- 1+2+…..(N-2)+(N-1)
- =(Nx(N-1))/2
- N^2-N < N^2

- It means there can be maximum N^2 exchange in insertion sort

3 Possibilities (Best, Average, Worst) case

- In the best case, the sequence has all the elements in desired sorted order
- In a worst case, no element at the correct position i.e. sequence is completely in reverse order
- In the average case, the sequence will have a mix of these two

Application of Bubble and Insertion Sort

- Bubble is one of the simplest sorting algorithm, but it is not considered as an efficient one because insertion sorting works better than bubble sort.
- Insertion sorting has been preferred choice for smaller datasets. It does not require any additional memory.
- Insertion sorting is fast if the data is almost nearly sorted.
- Bubble sorting is best option when we want to sort data stored in magnetic tape; as we know from Tape we can access data in sequentially only and with bubble sorting it is easier to sort two successive record. Though this situation is rare but for such we can use bubble sort.