

CLASS XII – 2019 CBSE – CS SORTING ALGORITHMS

S. VIGNESH M.E (CS & IS)

PGT [CS]

KV No. 1 MADURAI

SORTING ALGORITHMS IN PYTHON

❖ Bubble Sort

❖ Selection Sort

❖ Insertion Sort

❖ Merge Sort

❖ Heap Sort

❖ Quick Sort

BUBBLE SORT

❖SIMPLE TECHNIQUE:

- ❖Compare and swap

- ❖List of elements are going to be sorted in an ascending order by comparing them and swapping their positions.

- ❖Swap: $\langle A, B - \text{using } 3^{\text{rd}} \text{ variable} \rangle$

- ❖Temp = A $\rightarrow$ Now, A is empty.

- ❖A = B $\rightarrow$ now A stores value of B and B left empty.

- ❖B = Temp $\rightarrow$ B store the value in Temp.

❖ Example: -- List = [55,66,87,41,30,95,65]

❖ Iteration – 1

❖ List = [55, 66, 87, 41, 30, 95, 65]

❖ List = [55, 66, 41, 87, 30, 95, 65]

❖ List = [55, 66, 41, 30, 87, 95, 65]

❖ List = [55, 66, 41, 30, 87, 65, 95]

❖ Iteration – 2

❖ List = [55, 66, 41, 30, 87, 65, 95]

❖ List = [55, 41, 66, 30, 87, 65, 95]

❖ List = [55, 41, 30, 66, 87, 65, 95]

❖ List = [55, 41, 30, 66, 65, 87, 95]

❖ I – 3

❖ List = [55, 41, 30, 66, 65, 87, 95]

❖ List = [41, 55, 30, 66, 65, 87, 95]

❖ List = [41, 30, 55, 66, 65, 87, 95]

❖ List = [41, 30, 55, 65, 66, 87, 95]

❖ I – 4

❖ List = [41, 30, 55, 65, 66, 87, 95]

❖ List = [30, 41, 55, 65, 66, 87, 95]

```
>>> def sort(nums):
...     for i in range(len(nums)-1, 0, -1):
...         for j in range(i):
...             if nums[j] > nums[j+1]:
...                 temp = nums[j]
...                 nums[j] = nums[j+1]
...                 nums[j+1] = temp
...
>>> sort(N)
>>> print(N)
[2, 3, 5, 6, 7, 8]
>>>
```

```
2, 3, 5, 6, 7, 8]
>> N = [88,74,96,110,5,65,258,20]
>> sort(N)
>> print(N)
[5, 20, 65, 74, 88, 96, 110, 258]
>>
```

```
>>> N = []
>>> lt = int(input("ENTER THE LIMIT"))
ENTER THE LIMIT 7
>>> for i in range(0, lt):
...     item = int(input("ENTER ANY INTEGER"))
...     N.append(item)
...
ENTER ANY INTEGER 55
ENTER ANY INTEGER 66
ENTER ANY INTEGER 87
ENTER ANY INTEGER 41
ENTER ANY INTEGER 30
ENTER ANY INTEGER 95
ENTER ANY INTEGER 65
>>> print(N)
[55, 66, 87, 41, 30, 95, 65]
>>> sort(N)
>>> print(N)
[30, 41, 55, 65, 66, 87, 95]
>>>
```

SELECTION SORT

- ❖ The insertion sort method sorts a list of elements by inserting each successive elements in the previously sorted sub list.
- ❖ It performs “ $n - 1$ ” passes.
- ❖ For pass 1 through $n - 1$, insertion sort ensures that the elements in position 0 through p are in sorted order.
- ❖ In pass P , we move the p th element to left until it's correct place is found among the first element.

- Example - List = [55, 66, 87, 41, 30, 95, 65]

• List = [55, 66, 87, 41, 30, 95, 65]
Sorted ← | → Un sorted

- List = [55, 66, 87, 41, 30, 95, 65]
- List = [41, 55, 66, 87, 30, 95, 65]
- List = [30, 41, 55, 66, 87, 95, 65]
- List = [30, 41, 55, 66, 87, 95, 65]
- List = [30, 41, 55, 65, 66, 87, 95]

```
def insertion_sort(num):  
    for index in range(1, len(num)):  
        current_value = num[index]  
        position = index  
        while position > 0 and num[position - 1] > current_value:  
            num[position] = num[position - 1]  
            position = position - 1  
            num[position] = current_value  
        print("LIST AFTER INSERTION:", num)  
  
num = []  
n = int(input("ENTER THE SIZE:"))  
for i in range(n):  
    num.append(int(input()))  
  
insertion_sort(num)
```

ENTER THE SIZE:5

55

98

74

63

1

('LIST AFTER INSERTION:', [55, 74, 98, 63, 1])

('LIST AFTER INSERTION:', [55, 74, 63, 98, 1])

('LIST AFTER INSERTION:', [55, 63, 74, 98, 1])

('LIST AFTER INSERTION:', [55, 63, 74, 1, 98])

('LIST AFTER INSERTION:', [55, 63, 1, 74, 98])

('LIST AFTER INSERTION:', [55, 1, 63, 74, 98])

('LIST AFTER INSERTION:', [1, 55, 63, 74, 98])

ENTER THE SIZE:7

55

66

87

41

30

95

65

List = [55, 66, 87, 41, 30, 95, 65]

('LIST AFTER INSERTION:', [55, 66, 41, 87, 30, 95, 65])

('LIST AFTER INSERTION:', [55, 41, 66, 87, 30, 95, 65])

('LIST AFTER INSERTION:', [41, 55, 66, 87, 30, 95, 65])

('LIST AFTER INSERTION:', [41, 55, 66, 30, 87, 95, 65])

('LIST AFTER INSERTION:', [41, 55, 30, 66, 87, 95, 65])

('LIST AFTER INSERTION:', [41, 30, 55, 66, 87, 95, 65])

('LIST AFTER INSERTION:', [30, 41, 55, 66, 87, 95, 65])

('LIST AFTER INSERTION:', [30, 41, 55, 66, 87, 65, 95])

('LIST AFTER INSERTION:', [30, 41, 55, 66, 65, 87, 95])

('LIST AFTER INSERTION:', [30, 41, 55, 65, 66, 87, 95])

>>>

BUBBLE SORT

A simple sorting algorithm that continuously steps through the list and compares the adjacent pairs to sort the elements

Compares the adjacent elements and swap accordingly

Less efficient

Slower

Uses item exchanging

SELECTION SORT

A sorting algorithm that takes the smallest value (considering ascending order) in the list and moves it to the proper position in the array

Selects the minimum element from the unsorted sub-array and places it at the next position of the sorted subarray

More efficient

Faster

Uses item selection