Roll No. 125
LTCE

# Experiment No. 1

## Selection sort.

**Aim -**

Implementation of selection sort.

**Introduction -**

The selection sort algo is a simple, yet effective sorting algo. A selection-based sorting algo is described as in place comparison-based algo that divides the list into 2 parts, the sorted part on the left & the unsorted part on the right. Initially, the sorted selection is empty, & the unsorted selection contains the entire list.

**Example -**

arr [] = 64, 25 12 22 11

// find the minimum element in arr [0 ... 4]
// And place it at begining
// 25 12 22 64

// Find the minimum element in arr [1 ... 4]
// And place it at begining of arr [1 ..... 4]
// 12 25 22 64
// Find the minimum element in arr [2 ... 4]
// And place it at beging of arr [2 .... 4]
// 12 22 25 64
// Find the minimum element in arr [3 .... 4]
// And place it at begining of arr [3 .... 4]
// 12 22 25 24

Time complexity -

Let's assume T[n] defines the running time to solve the problem of sizen. In selection sort, after each iteration. One element get sorted & problem size reduces by one elements for each problem of sian, inner loop iterates n times. So recurrance eqn for sclection sort can be written as

$T(n) = T(n-1) + n$

put $n = n-1$ in above eqn

$T(n-1) = T(n-2) + (n-1)$

put value of $T(n-1)$ in previous eqn of $T(n)$

$T(n) = T(n-2) + (n-1) + n$

put $n = n-2$ ~~$+n$~~ in above eqn

$T(n-2) = T(n-3) + n-2$

put value of $T(n-2)$ in previous eqn of $T(n)$

$\therefore T(n) = T(n-3) + (n-3) + (n-1) + n$

After K iterations

$T(n-K) = T(n-K-1) + (n-K)$

$T(n) = T(n-K) + (n-K+1) + (n-K+2) + \cdots + (n-1) + n$

when K approaches to n

$T(n) = T(0) + 1 + 2 + 3 + \cdots + (n-1) + n$

$T(0) = 0$, because it is running time of problem of size zero, & no effect it is needed to solve problem.

$$T(n) = 1 + 2 + 3 + \cdots + n$$

$$= \Sigma n = \frac{n(n+1)}{2} = \frac{n^2+n}{2} = \frac{n^2}{2} + \frac{n}{2}$$

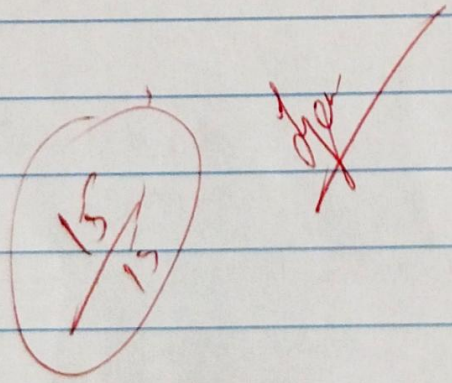
$$= O\left(\max\left(\frac{n^2}{n}, \frac{n}{n}\right)\right) = O\left(\frac{n^2}{2}\right)$$

$$T(n) = O(n^2)$$

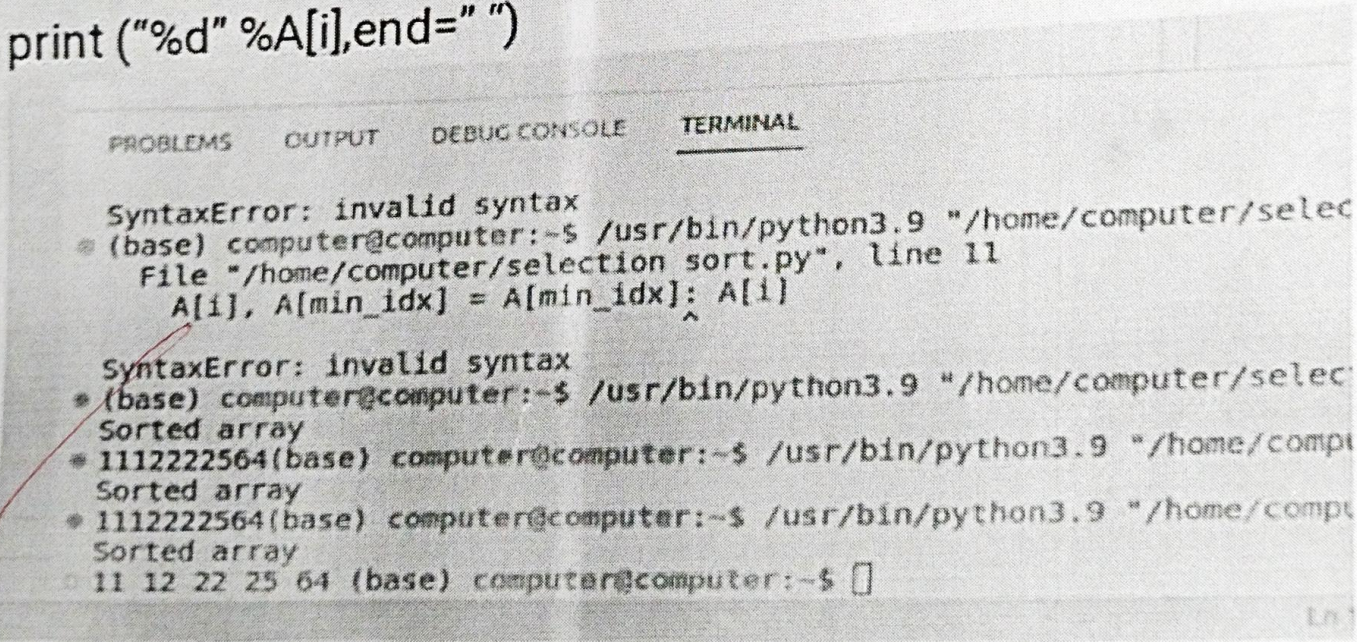
| Best case | Avg case | wrost case |
|---|---|---|
| $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |

Conclusion -

Hence we have successfully implemented algo & analyzed the complexity.



```
import sys

A = [64, 25, 12, 22, 11]

for i in range (len(A)):

min_idx = i

for j in range(i+1, len(A)):

    if  A[min_idx] > A[j]:

        min_idx = j

A[i], A[min_idx] = A[min_idx], A[i]

print ("Sorted array")

for i in range(len(A)):

print ("%d" %A[i],end=" ")
```



LTCE

# Experiment No. 2

## Merge Sort

Aim - Implementation of merge sort.

Theory -

Algo:-

1) If it is only one element in the list it is alread sorted, return
2) Divide the list recurssively into two halves until it can no more divided.
3) Merge the smaller list into new list in sorted order.

Example -

Mergesort (arr[], l, r)

if r > l

1. Find the middle point to divide the array into two halves
   middle m = l + (r-l)/2
2. Call the mergesort for 1st half:
   call mergesort m = (arr, l, m)
3. Call mergesort for 2nd half:
   call mergesort (arr, m+1, r)
4. Merge the 2 halves sorted in step 2 & 3
   call merge (arr, l, m, r)

Time complexity -

| Best | Average | Worst |
|---|---|---|
| $\Omega(n \log(n))$ | $\Theta(n \log(n))$ | $O(n \log(n))$ |

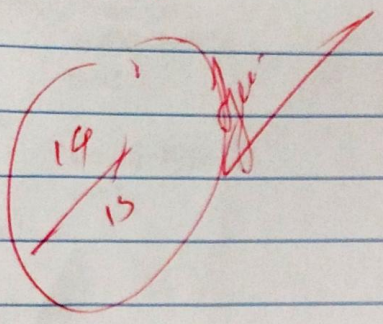
Time complexity -

Merge sort is recursive algo & time complexity can be expressed as-following recurrance relation.

$$T(n) = 2T(n/2) + \Theta(n)$$

Conclusion -

Hence, we have successfully implemented algorithm & analyzed the complexity.



```
#Python program to perform Merge Sort
#Function to merge and sort elements

def merge_sort(arr):
        if len(arr)>1:
        m=len(arr)//2
        Left=arr[:m]
        Right=arr[m:]
        merge_sort(Left)
        merge_sort(Right)

        i=j=k=0

        while i<len(Left) and j<len(Right):
                if Left[i]<Right[j]:
                        arr[k]=Left[i]
                        i+=1
                else :
                        arr[k]=Right[j]
                        j+=1
                k+=1

        while i<len(Left):
                arr[k]=Left[i]
                i+=1
        k+=1

#Inserting an no of elements
n =int(input("Enter no of elements in an array"))

array=[]

#Taking input
print("Enter elements in array")
for i in range(0,n):
        b=int(input())
        array.append(b)

print('Given Array is',array)
merge_sort(array)
print("Sorted array is ",array)
```

```
(base) computer@computer-ThinkCentre:~/Downloads/SIMPLE_INVENTORY_SYSTEM_IN_PYTHON_WITH_SOURCE_CODE/Simple_Inventory PYTHON/Simple Inventory System$ /bin/python3 "/home/co
puter/Downloads/SIMPLE_INVENTORY_SYSTEM_IN_PYTHON_WITH_SOURCE_CODE/Simple_Inventory PYTHON/Simple Inventory System/Merge Sort.py"
Enter no of elements in an array5
Enter elements in array
23
576
12
68
99
Given Array is [23, 576, 12, 68, 99]
Sorted array is  [12, 23, 68, 99, 576]
(base) computer@computer-ThinkCentre:~/Downloads/SIMPLE_INVENTORY_SYSTEM_IN_PYTHON_WITH_SOURCE_CODE/Simple_Inventory PYTHON/Simple Inventory System$
```