**Assignment No.3 (a)**

Problem Statement : Implement Greedy search algorithm for any of the following application:

1. Selection Sort
2. Minimum Spanning Tree
3. Single-Source Shortest Path Problem
4. Job Scheduling Problem
5. Prim's Minimal Spanning Tree Algorithm
6. Kruskal's Minimal Spanning Tree Algorithm
7. Dijkstra's Minimal Spanning Tree Algorithm
8. **Selection Sort**

In selection sort, the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted. The average and worst-case complexity of selection sort is **O(n2)**, where **n** is the number of items. Due to this, it is not suitable for large data sets.

**Algorithm :**

SELECTION SORT(arr, n)

Step 1: Repeat Steps 2 **and** 3 **for** i = 0 to n-1

Step 2: CALL SMALLEST(arr, i, n, pos)

Step 3: SWAP arr[i] with arr[pos]

[END OF LOOP]

Step 4: EXIT

SMALLEST (arr, i, n, pos)

Step 1: [INITIALIZE] SET SMALL = arr[i]

Step 2: [INITIALIZE] SET pos = i

Step 3: Repeat **for** j = i+1 to n

**if** (SMALL > arr[j])

     SET SMALL = arr[j]

SET pos = j

[END OF **if**]

[END OF LOOP]

Step 4: RETURN pos

## Working of Selection Sort Algorithm :

To understand the working of the Selection sort algorithm, let's take an unsorted array. It will be easier to understand the Selection sort via an example.

Let the elements of array are -

selection Sort Algorithm

Now, for the first position in the sorted array, the entire array is to be scanned sequentially.

At present, **12** is stored at the first position, after searching the entire array, it is found that **8** is the smallest value.

selection Sort Algorithm

So, swap 12 with 8. After the first iteration, 8 will appear at the first position in the sorted array.

selection Sort Algorithm

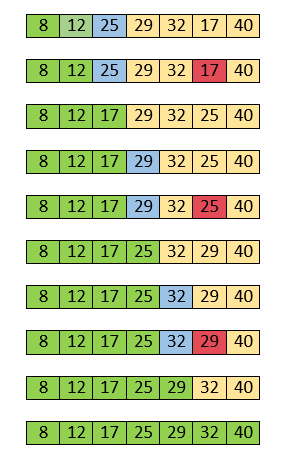
For the second position, where 29 is stored presently, we again sequentially scan the rest of the items of unsorted array. After scanning, we find that 12 is the second lowest element in the array that should be appeared at second position.

selection Sort Algorithm

Now, swap 29 with 12. After the second iteration, 12 will appear at the second position in the sorted array. So, after two iterations, the two smallest values are placed at the beginning in a sorted way.

selection Sort Algorithm

The same process is applied to the rest of the array elements. Now, we are showing a pictorial representation of the entire sorting process.



Now, the array is completely sorted.

**Conclusion :**

## 

## Implementation of selection sort

def selection(a): # Function to implement selection sort

**for** i in range(len(a)): # Traverse through all array elements

        small = i # minimum element in unsorted array

**for** j in range(i+1, len(a)):

**if** a[small] > a[j]:

                small = j

    # Swap the found minimum element with

    # the first element

        a[i], a[small] = a[small], a[i]

def printArr(a): # function to print the array

**for** i in range(len(a)):

        print (a[i], end = " ")

a = [69, 14, 1, 50, 59]

print("Before sorting array elements are - ")

printArr(a)

selection(a)

print("\nAfter sorting array elements are - ")

selection(a)

printArr(a)

**Output:**

