Assignment No:- 4

Title :- Greedy Search Algorithms.

Problem statement :- Implement Greedy search algorithm for any of the following application.

Objective :-

* To understand the concept of Greedy Search Algorithms.
* To implement algorithm of Selection Sort for given set of Numbers.

Theory :-

Greedy Algorithm :-

The greedy method is one of the strategies like Divide and conquer used to solve the problems. This method is used for solving optimization problems. An optimization problem is a problem that demands either maximum or minimum or minimum results. Let's understand through some terms.

The Greedy method is the simpliest and straightforward approach. It is not an algorithm, but it is a technique. The main function of this approach is that the decision is that the decision is taken on the basis of the currently available information. Whatever the current information is present, the decision is made without worrying about the effect of the current decision in future.

Application of Greedy Algorithm :-

- It is used in finding the shortest path.
- It is used to find the minimum spanning tree using the prim's algorithm or the kruskal's algorithm.
- It is used in a job sequencing with a deadline.
- This algorithm is also used to solve the fractional knapsack problem.

I. Selection sort :-

The selection sort enhance the bubble sort by making only a single swap for each pass through the rundown. In order to do this, a selection sort searches for the biggest values as it make a pass and, after finishing the pass, places it in the best possible area. Similarly, as with a bubble sort after finishing the pass.

Algorithm : SELECTION SORT (A).

```
k ← length[A]
for j ← 1 to ~~a~~ n-1
    smallest ← j
    for I ← j +1 to k
        if A[i] < A [smallest]
            then smallest ← i
    exchange (A[j], A[smallest])
```

How Selection Sort works.

7	4	3	6	5

1st Iteration :-

Set minimum = 7

- compare a_0 and a_1

7	4	3	6	5

As , $a_0 > a_1$, set minimum = 4.

- Compare a_1 and a_2

7	4	3	6	5

As , $a_1 > a_2$, set minimum = 3

- Compare a_2 and a_3

7	4	3	6	5

As , $a_2 > a_3$, set minimum = 3.

- Compare a_2 and a_3

7	4	3	6	5

As, $a_2 < a_3$, set minimum = 3

- Compare a_2 and a_4.

7	4	3	6	5

As, $a_2 < a_4$, set minimum = 3

Since 3 is the smallest element, soo we

will swap a0 and a2.

3	4	7	6	5

2^{nd} Iteration.

Set minimum = 4

- Compare a1 & a2

3	4	7	6	5

As, a1 < a2, set minimum = 4.

- Compare a1, and a3.

3	4	7	6	5

Again a1 < a4, set minimum = 4

since the minimum is already placed in the correct position, so there will be no swapping.

3	4	7	6	5

3^{rd} Iteration.

set minimum = 7

- compare a_2 and a_3

3	4	7	6	5

As, $a_2 > a_3$, set minimum = 6

- Compare a_3 and a_4.

3	4	7	6	5

As, $a_3 > a_4$, set minimum = 5

Since 5 is the smallest element among the leftover unsorted elements, so we will swap 7 and 5.

3	4	5	6	7

4th Iteration :

Set minimum = 6

- Compare a_3 and a_4.

3	4	5	6	7

As $a_3 < a_4$, set minimum =6

Since the minimum is already placed in the correct position, so there will be no swapping.

3	4	5	6	7

Complexity Analysis of Selection Sort.

Input: Given n input elements.

Output: Number of steps incurred to sort a list.

logic : If we are given n element, then in the first pass, it will do n-1 comparisons : in the second pass, it will do n-2, in third pass, it will do n-3 & so on. Thus, the total number of comparisons can be found.

Conclusion:-

We have implemented selection sort for given numbers.