# Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

**AY: 2024-25**

| | | | |
|---|---|---|---|
| **Class:** | **SE** | **Semester:** | **IV** |
| **Course Code:** | **CSL401** | **Course Name:** | **Analysis of Algorithm Lab** |

| | |
|---|---|
| **Name of Student:** | **Shravani Sandeep Raut** |
| **Roll No. :** | **48** |
| **Experiment No.:** | **2** |
| **Title of the Experiment:** | **Selection Sort** |
| **Date of Performance:** | **23/01/2025** |
| **Date of Submission:** | **23/01/2025** |

## Evaluation

| Performance Indicator | Max. Marks | Marks Obtained |
|---|---|---|
| Performance | 5 | |
| Understanding | 5 | |
| Journal work and timely submission | 10 | |
| Total | 20 | |

| Performance Indicator | Exceed Expectations (EE) | Meet Expectations (ME) | Below Expectations (BE) |
|---|---|---|---|
| Performance | 4-5 | 2-3 | 1 |
| Understanding | 4-5 | 2-3 | 1 |
| Journal work and timely submission | 8-10 | 5-8 | 1-4 |

**Checked by**

**Name of Faculty : Mrs. Sneha Yadav**

**Signature :**

**Date:**

**Aim**: To implement Selection Comparative analysis for large values of 'n'

**Objective:** To introduce the methods of designing and analyzing algorithms

**Theory**:

Selection sort is a sorting algorithm, specifically an in-place comparison sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited.

The algorithm divides the input list into two parts: the sub list of items already sorted, which is built up from left to right at the front (left) of the list, and the sub list of items remaining to be sorted that occupy the rest of the list. Initially, the sorted sub list is empty and the unsorted sub list is the entire input list. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted sub list, exchanging it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

**Example**:

arr[] = 64 25 12 22 11

// Find the minimum element in arr[0...4] // and place it at beginning

**11** 25 12 22 64

// Find the minimum element in arr[1...4] // and place it at beginning of arr[1...4]

11 12 25 22 64

// Find the minimum element in arr[2...4] // and place it at beginning of arr[2...4]

11 12 **22** 25 64

// Find the minimum element in arr[3...4] // and place it at beginning of arr[3...4]

11 12 22 **25** 64

**Algorithm and Complexity**:

| Alg.: SELECTION-SORT(A) | | |
|---|---|---|
| | cost | Times |
| n ← length[A] | $c_1$ | 1 |
| for j ← 1 to n - 1 | $c_2$ | n-1 |
| do smallest ← j | $c_3$ | n-1 |
| for i ← j + 1 to n | $c_4$ | $\sum_{j=1}^{n-1}(n-j+1)$ |
| ≈ n2/2 comparisons, do if A[i]<A[smallest] | $c_5$ | $\sum_{j=1}^{n-1}(n-j)$ |
| then smallest ← i | $c_6$ | $\sum_{j=1}^{n-1}(n-j)$ |
| ≈ n exchanges, exchange A[j] ↔ A[smallest] | $c_7$ | n-1 |

**Implementation:**

```
#include <stdio.h>
int n, i, j, A[20], minIndex;

void Selection_Sort(int A[], int n);
void Swap(int A[], int i, int minIndex);


void main()
{
   int i, j;
   printf("Enter the size of array: ");
   scanf("%d", &n);
   printf("Enter the elements of array: \n");
   for(i=0; i<n; i++)
   {
      printf("Enter value: ");
      scanf("%d", &A[i]);
   }
   printf("The unsorted array is: ");
   for(i=0; i<n; i++)
   {
      printf("%d\t",A[i]);
   }
   Selection_Sort(A,n);
   printf("\nAfter sorting array is: ");
   for(i=0; i<n; i++)
   {
      printf("%d\t",A[i]);
   }
}
```

```
void Selection_Sort(int A[], int n)
{
   for(i= 0; i<=n-2; i++)
   {
      minIndex = i;
      for(j=i+1; j<= n-1; j++)
      {
         if(A[j]< A[minIndex])
         {
            minIndex = j;
         }
      }

      Swap(A, i, minIndex);
   }
}


void Swap(int A[], int i, int minIndex)
{
   int temp;
   temp = A[i];
   A[i] = A[minIndex];
   A[minIndex] = temp;
}
```

**Output -**

```
Enter the size of array: 5
Enter the elements of array:
Enter value: 12
Enter value: 23
Enter value: 3
Enter value: 6
Enter value: 1
The unsorted array is: 12       23      3       6       1
After sorting array is: 1       3       6       12      23
```

**Conclusion:**

Selection Sort is a straightforward sorting algorithm that repeatedly selects the smallest (or largest) element from the unsorted portion of the list and moves it to the beginning (or end). The algorithm divides the list into a sorted and an unsorted part. It repeatedly finds the minimum element from the unsorted part and swaps it with the first unsorted element.

- Time Complexity:
  - Best Case: $O(n^2)$
  - Average Case:$O(n^2)$
  - Worst Case: $O(n^2)$

- Space Complexity: O(1) (in-place sorting)

- Advantages:
  - Simple to understand and implement.
  - Performs well on small lists.
  - Does not require additional memory (in-place).

- Disadvantages:
  - Inefficient for large datasets due to its quadratic time complexity.
  - Not stable (equal elements may not maintain their relative order).