Experiment 1 [B]

Problem Statement: Given a sequence of n numbers <a1, a2, ...,an>, find a permutation (reordering) <a1', a2', ...,an'> of the input sequence such that a1'<= a2'<=, ...,<=an', using selection sort algorithm.

Theory

The Selection Sort algorithm is an in-place comparison-based algorithm. The idea behind it is to divide an array into two segments:

- The sorted part at the left end.
- The remaining unsorted part at the right end.

First, the algorithm finds either the minimum or the maximum element in our input array. Then, the algorithm moves this element to its correct position in the array.

Algorithm

➔ Accept n, the number of elements
➔ Accept the data (numbers) into info
➔ Loop pos from 2 to n in steps of 1
  ◆ Y info(pos)
  ◆ Loop i from pos -1 to 1st element steps of -1
    - if y < info(i)
      - x(i+1) x(i)
    - else Break out of loop (goto 3.3)
  ◆ x(i+1) y
➔ Display the sorted array, info
➔ Accept n, the number of elements
➔ Accept the data into array x
➔ loop i from last position i.e. n -1 to 0 in steps of -1
  ◆ large x(0)
  ◆ indx 0
  ◆ loop j from 1 to i in steps of 1
    - if x(j) > large
      - update large x(j)
      - update indx j

- ◆ x(indx) x(i)
- ◆ x(i) large

➔ Display sorted list , i.e. x

Example

Consider an array of size 5

| 5 | 4 | 3 | 2 | 1 |

Pass 1: | 5 | 4 | 3 | 2 | 1 |
Pass 2: | 1 | 4 | 3 | 2 | 5 |
Pass 3: | 1 | 2 | 3 | 4 | 5 |
Pass 4: | 1 | 2 | 3 | 4 | 5 |

Sorted array: | 1 | 2 | 3 | 4 | 5 |

Complexity Analysis

Time Complexity

In general, the time complexity depends on the size and arrangement of values within an array. Therefore, the standard terms like “best-case” and “worst-case” complexities are a measure of time-complexities for algorithms.

Worst-Case: O(n^2) – The scenario when the array is in descending order. Meanwhile, the algorithm tries to sort in the ascending order.

Best-Case: O(n) – The scenario when the array is already in a sorted order.

Overall, the time complexity for Insertion Sort is O(n^2).

Space Complexity

In simpler terms, space complexity refers to the amount of excess space or memory used during the running of the algorithm. Since we are only using extra variables like value, i and j, the space complexity is O(1).

Code

```
#include <stdio.h>
#define MAX 100
int data[MAX], n, i;

void printArray(int array[])
{
   for (i = 0; i < n; i++)
   {
      printf("| %d |", array[i]);
   }
   printf("\n");
}

void selectionSort(int array[])
{
   int i, j, smallest_index, smallest, temp;
   for (i = 0; i < n; i++)
   {
      for (j = i; j < n; j++)
      {
         smallest = array[i];
         if (array[j] < smallest)
         {
            smallest = array[j];
            smallest_index = j;
         }
         if (smallest < array[i])
         {
            temp = array[i];
            array[i] = array[smallest_index];
            array[smallest_index] = temp;
         }
      }
      printf("Pass %d: ", i);
      printArray(array);
      printf("\n");
   }
```

```
}

int main()
{
   printf("\n");
   printf("Enter number of elements: ");
   scanf("%d", &n);
   printf("Start entering %d integers: \n", n);
   for (i = 0; i < n; i++)
   {
      printf("data[%d] = ", i);
      scanf("%d", &data[i]);
   }

   printf("\n------- Array before sorting -------\n");
   printArray(data);
   printf("------- x ------- x ------- x -------\n\n");

   selectionSort(data);

   printf("------- Array after sorting -------\n");
   printArray(data);
   printf("------- x ------- x ------- x -------\n\n");
   return 0;
}
```

Output

```
Enter number of elements: 7
Start entering 7 integers:
data[0] = 7
data[1] = 3
data[2] = 2
data[3] = 5
data[4] = 4
data[5] = 1
data[6] = 6
```

```
------- Array before sorting -------
| 7 || 3 || 2 || 5 || 4 || 1 || 6 |
------- x ------- x ------- x -------

Pass 0: | 1 || 7 || 3 || 5 || 4 || 2 || 6 |

Pass 1: | 1 || 2 || 7 || 5 || 4 || 3 || 6 |

Pass 2: | 1 || 2 || 3 || 7 || 5 || 4 || 6 |

Pass 3: | 1 || 2 || 3 || 4 || 7 || 5 || 6 |

Pass 4: | 1 || 2 || 3 || 4 || 5 || 7 || 6 |

Pass 5: | 1 || 2 || 3 || 4 || 5 || 6 || 7 |

Pass 6: | 1 || 2 || 3 || 4 || 5 || 6 || 7 |

------- Array after sorting -------
| 1 || 2 || 3 || 4 || 5 || 6 || 7 |
------- x ------- x ------- x -------
```