Experiment No.1
To implement Selection Sort
Name : Vaidehi D. Gadag
Branch/Div.: Comps-1 (C47)
Date of Performance: 08/02/2024
Date of Submission: 15/02/2024

Experiment No. 1

Title: To implement selection sort.

Aim: To study, implement and Analyze Selection Sort Algorithm

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 algorithmsin 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 sub-list boundaries one elementto the right.

Example:

Sort the given array using selection sort. arr[] = 64 25 12 22 11

11 25 12 22 64	Find the minimum element in arr[0...4] and place it at beginning
11 12 25 22 64	Find the minimum element in arr[1...4] and place it at beginning of arr[1...4]
11 12 **22** 25 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]

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

The recurrence relation for selection sort is:

T(n) = 1 for n=0

= T(n – 1) + n for n>0 ---- 1

From above equation,

T (n – 1) = T(n – 2) + (n – 1)

Use above in equation 1

T(n) = T(n – 2) + (n – 1) + n----2

Let T(n – 2) = T(n – 3) + n – 2

Use above in equation 2

T(n) = T(n – 3) + (n – 2) + (n – 1) + n

After k iterations,

T(n) = T(n – k) + (n – k + 1) + (n – k + 2) + ..… + (n – 1) + n

When k approaches to n,

T(n) = T(0) + 1 + 2 + 3 + … + (n –1) + n

T(0) = 0,

T(n) = 1 + 2 + 3 + … + n

= n(n + 1) / 2

= $(n^2/2) + (n/2)$

T(n) = O(max($(n^2/2) + (n/2)$))

= $O(n^2 / 2)$

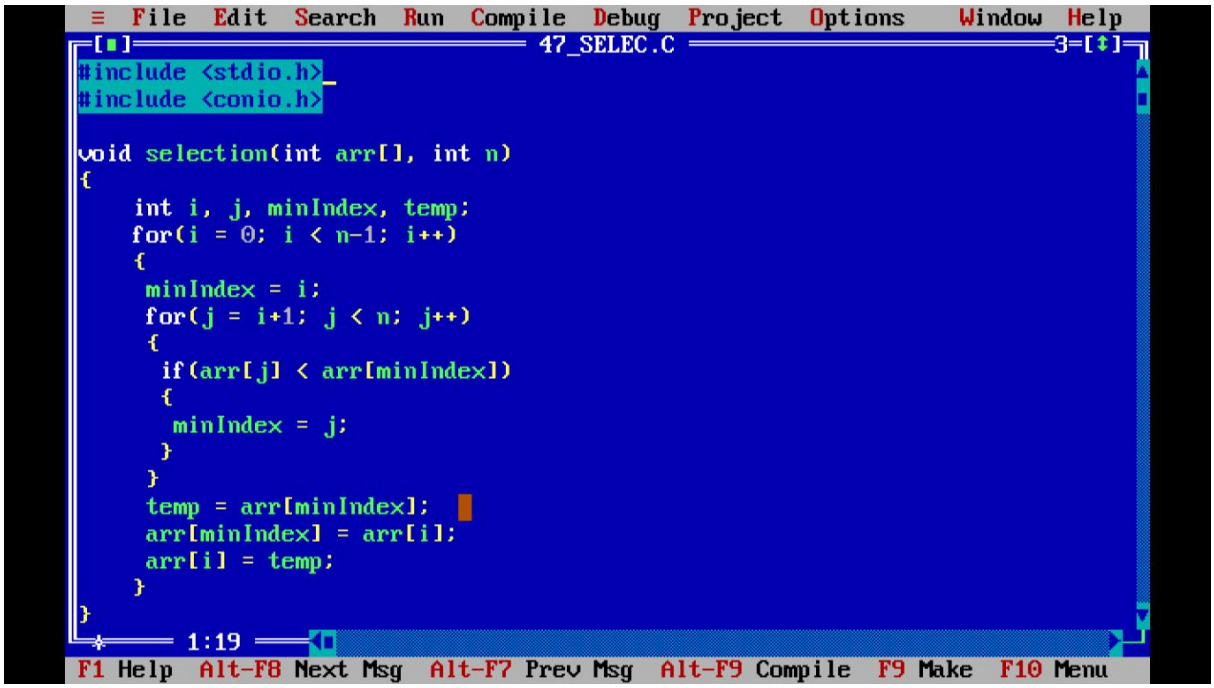
= $O(n^2)$

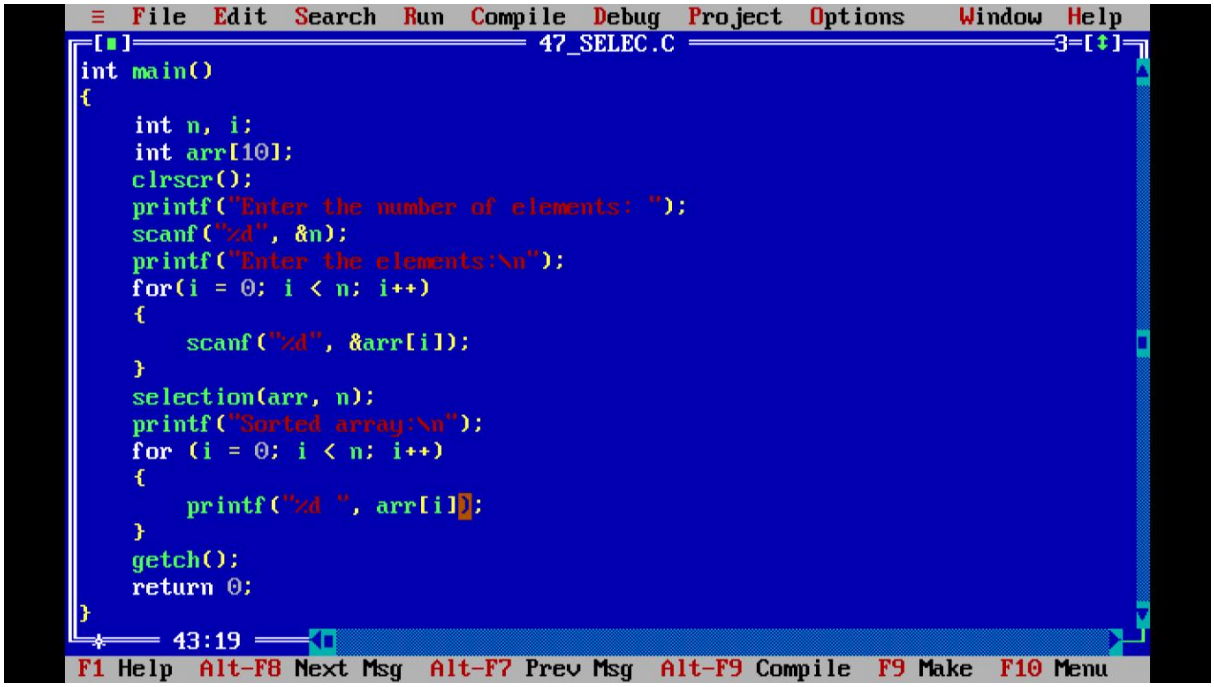
T(n) = $O(n^2)$

Code:

```
#include <stdio.h>
#include <conio.h>
void selection(int arr[], int n)
{
   int i, j, minIndex, temp;
   for(i = 0; i < n-1; i++)
   {
    minIndex = i;
    for(j = i+1; j < n; j++)
    {
     if(arr[j] < arr[minIndex])
     {
      minIndex = j;
     }
    }

    temp = arr[minIndex];
    arr[minIndex] = arr[i];
    arr[i] = temp;
   }
}
int main()
{
   int n, i;
   int arr[10];
   clrscr();
   printf("Enter the number of elements: ");
   scanf("%d", &n);
   printf("Enter the elements:\n");
   for(i = 0; i < n; i++)
   {
           scanf("%d", &arr[i]);
   }
   selection(arr, n);
   printf("Sorted array:\n");
   for (i = 0; i < n; i++)
   {
           printf("%d ", arr[i]);
   }
   getch();
   return 0;
}
```


File Edit Search Run Compile Debug Project Options Window Help
47_SELEC.C
#include <stdio.h>
#include <conio.h>
void selection(int arr[], int n)
{
int i, j, minIndex, temp;
for(i = 0; i < n-1; i++)
{
minIndex = i;
for(j = i+1; j < n; j++)
{
if(arr[j] < arr[minIndex])
{
minIndex = j;
}
}
temp = arr[minIndex];
arr[minIndex] = arr[i];
arr[i] = temp;
}
}
1:19
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
47_SELEC.C
int main()
{
int n, i;
int arr[10];
clrscr();
printf("Enter the number of elements: ");
scanf("%d", &n);
printf("Enter the elements:\n");
for(i = 0; i < n; i++)
{
scanf("%d", &arr[i]);
}
selection(arr, n);
printf("Sorted array:\n");
for (i = 0; i < n; i++)
{
printf("%d ", arr[i]);
}
getch();
return 0;
}
43:19
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

Output:

Enter the number of elements: 7

Enter the elements:

1 30 13 18 8 4 9

Sorted array:

1 4 8 9 13 18 30

Conclusion:

Selection sort is a simple sorting algorithm that works by repeatedly finding the minimum element from an unsorted portion of the array and putting it at the beginning of the sorted portion. This process continues until the entire array is sorted. The algorithm has a time complexity of O(n^2) in the average and worst cases, making it less efficient for large data sets. However, it has a space complexity of O(1), which means it requires constant space and is an in-place sorting algorithm.