**Selection Sort**

**IDEA:**

The selection sort is an In- place sorting technique.

The sorted part of the array is kept on the left side of the array and the unsorted part in the right side of the array. Initially the unsorted part is null while the unsorted part is the entire list. The minimum number is found from the array and is swapped with the first element of the array. The process is repeated iteratively until the whole array gets sorted.

Array

**IMPLEMENTATION:**

Here the array of numbers are divided into sub-arrays. The minimum number in the array is found out and is placed on the left side of the array by swapping(during the first iteration).

The same process gets repeated for ‘n’ number of times (where n is the total number of elements in the array).

After every iteration the length of the array gets decreased by 1 size and minimum number checking and swapping is done for the rest of array. After completion of all iterations we get an ascending order sorted array(to get descending order sort we have to just change searching from minimum number to maximum number).

**ADVANTAGES:**

* The main advantage of the selection sorting technique is that it is best suited for smaller array.
* No extra array or storing space is required to store other than the main array.

**DISADVANTAGES:**

* For larger number of array the time complexity of this sorting technique is very poor. As the complexity is(n2) with the increase in number of element the technique proves less favourable.

**ALGORITHM**

* Select the minimum element form the array and swap it with to the left hand side of array( to base address 0 ).
* The base address of array is now shifted to 1.
* Again search for minimum element in the remaining array and swapping.
* The above processes are repeated until the whole array gets sorted.

During the implementation of above algorithm two for loops will be used. The first one to traverse the array and the second one to swap and shift the elements.

**PSUEDO CODE:**

array = n

i = 0

for i to n

min = i //select current element in array as minimum element

for j=i+1 to n

if arr[j] < arr[min] then //to search for minimum element in the array

min = j

end if

end for

if min!= i then //to swap the minimum element with current element

swap arr[min] and arr[i]

end if

end for

**SOURCE CODE**

#include <stdio.h>

void selectionSort(int arr[], int n)

{

     int i, j, min\_idx;

     for (i = 0; i < n-1; i++)   //One by one move the size of unsorted subarray

     {

         min\_idx = i; //Find the minimum element in unsorted array

         for (j = i+1; j < n; j++)

           if (arr[j] < arr[min\_idx])

             min\_idx = j;

         swap(&arr[min\_idx], &arr[i]); //Swap the found minimum      } //element with the first element

for (i=0; i < size; i++)

         printf("%d ", arr[i]);

}

void swap(int \*x, int \*y)

{

     int temp = \*x;

     \*x = \*y;

     \*y = temp;

}

int main()

{

     int arr[10] , n;

printf(“Enter the size of array: “);

     scanf(“%d”,&n);

printf(“Performing the Selection Sort”);

     selectionSort(arr, n);

     printf("Sorted array: \n");

     return 0;

}

**TIME COMPLEXITY:**

The recurrence relation can be given as:

T(n) = T(n-1) + c ..........(i)

Now, for n = n-1

T(n-1) = T(n-2) + c ..........(ii)

Similarly, T(n-2) = T(n-3) + c ..........(iii)

Putting values of equation (ii) and (iii) in (i)

T(n) = T(n-1) + T(n-2) +........ + T(1) + c

T(n) = n + n-1 + n-2 +.......... + 1

T(n) = [n(n-1)]/2

Hence, The time complexity for Selection Sort technique is O(n2).

**EXAMPLE:**

Input Array

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 15 | 7 | 19 | 9 |

0 1 2 3 4

**Iteration 1** – Selecting the minimum number form the array

Here minimum number = 4. Swap it with current element i.e to base address. 4 is already there so no change

The array will be -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 15 | 7 | 19 | 9 |

0 1 2 3 4

**Iteration 2** – Here the base address is now 1. Again finding for minimum element and swapping.

New minimum number = 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 7 | 15 | 19 | 9 |

0 1 2 3 4

**Iteration 3** – Here the base address is now 2. Again finding for minimum element and swapping.

New minimum number = 9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 7 | 9 | 19 | 15 |

0 1 2 3 4

**Iteration 4** – Here the base address is now 3. Again finding for minimum element and swapping.

New minimum number = 15

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 7 | 9 | 15 | 19 |

0 1 2 3 4

**Iteration 5** – Here the base address is now 4. Again finding for minimum element and swapping.

As no other numbers for comparison are left Hence the remaining number will be places at it’s position.

Hence the sorted array will be

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 7 | 9 | 15 | 19 |

0 1 2 3 4