AOA Harsh Kashikar
2003085 - C21

Page No. 1
Date

Experiment 1

Aim: Selection and insertion sort

Theory: Selection sort.

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part & putting it at the beginning. The algorithm maintains 2 subarrays in a given array.

i) The subarray which is already sorted.
ii) Remaining subarray which is unsorted.

In every iteration of selection sort the minimum element from unsorted subarray is picked & moved to be sorted subarray.

Algorithm:

```
sel_sort(A,n){
    for i=0 to n{
        j=i
        for k=i+1 to n{
            if(A[k]<A[j]){
                j=k
            }
        }
        t=A[i]
        A[i]=A[j]
        A[j]=t
    }
}
```

Harsh
2003085

Page No.
Date

Selection Sort Analysis:

Constants	Algorithm	Frequency
	selectionSort(A,n)	
c_1	{	
c_2 c_1	for i=1 to n	(n+1) → boundary case.
	{	
c_3 c_2	j=i	n
c_3	for k=i+1 to n	$\sum_{i=1}^{n} n-i+1$
	{	
c_4	if(A[k]<A[j])	$\sum_{i=1}^{n} (n-i)$
	{	
c_5	j=k	$\sum_{i=1}^{n} (n-i)*t_i$
	}	
	}	
c_6	t=A[i]	n
c_7	A[i]=A[j]	n
c_8	A[j]=t	n
	}	
	}	

Example:

arr.[ ] = {64, 25, 12, 22, 11}

1) Find the minimum element in arr.[0 ··· 4] & place it at beginning

11 25 12 22 64

Harsh
2003035

Page No.
Date

iii) Find minimum element in arr [1 ... 4] & place at beginning of arr [1 ... 4]

11 12 25 22 64

iii) Find min. element in arr [2 ... 4] & place at beggining of arr [2 ... 4]

11 12 22 25 64

iv) Find minimum element in arr [3 ... 4] & place at beggining of arr [3 ... 4]

11 12 22 25 64.

* Complexity Analysis

Selection sort requires 2 nested for loops to complete itself, one for top loop it is in the function selection sort & inside the first loop we are making a call to another function index of minimum which has second (inner) for loop.
Hence for a given input size of n following will be the time & space complexities.

a) worst case time complexity : $O(n^2)$ (Big-O)
b) Best case time complexity : $\Omega(n^2)$ (Big-omega)
c) Average time complexity : $\Theta(n^2)$ (Big-theta)
d) space-complexity : $O(1)$

Harsh
2003085

Insertion sort:

It is a sorting algorithm that virtually split the given array into unsorted & sorted parts then values from unsorted parts are picked and placed at correct position in sorted part.

Algorithm:

```
Ins-sort(A,n){
    for j=2 to n{
        key = A[j]
        i=j-1
        while (i>0 & A[i]>key){
        A[j+1] = A[i]
          j=j-1
        }
        A[i+1]=key
    }
}
```

Insertion sort analysis

Constants	Algorithm	Frequency
€	Insertion_sort(A,n)	
	{	
C_1	for j=2 to n{	n
C_2	key = A[j]	n-1
C_3	i=j-1	n-1
C_4	while(i>0 & A[i]>key)	$\sum_{j=2}^{n} t_j$
C_5	{ A[i+1] = A[i]	$\sum_{j=2}^{n} (t_j - 1)$
C_6	i=j-1	$\sum_{j=2}^{n} (t_j - 1)$
C_7	A[i+1]=key	n-1
	}	
	}	

Page No.
Date

Example:
Taking an unsorted array.
12 31 25 8 32 17.

Initially the first 2 elements are compared insertion sort
12 31 25 8 32 17.

Here 31 is greater than 12. That means both elements are already in ascending order.

So, for now 12 is sorted in sorted sub-array.
Now move to the next 2 elements & compare them.
12 31 25 8 32 17.

Here, 25 is smaller than 31. Now swap 31 with 25. Along with swapping insertion sort will also check it with all elements in sorted array.

For now sorted array has only one element i.e 12 so 25 is greater than 12. Hence the sorted array remains sorted after swapping
12 25 31 8 32 17

Now, 2 elements in sorted array are 12 & 25 move forward to the next elements that are 31 & 8.
12 25 31 8 32 17

Both 31 and 8 are not sorted, so swap them
12 25 8 31 32 17.

After swapping elements 25 & 8 are unsorted

12 8 25 31 32 17

Now elements 12 & 8 are unsorted so swap them

8 12 25 31 32 17

Now the sorted array includes 8 12 25 31 32 17

Move to next elements that are 32, 17

17 is smaller that 32 so swap them

8 12 25 31 17 32

Swapping makes 25 & 17 unsorted so swap them

8 12 17 25 31 32

Now, array is completely sorted

Analysis:

1] Time complexity:

(a) Best case complexity:

It occurs when there is no sorting required best case time complexity in O(n)

(b) Worst case complexity:

It occurs when array required to be sorted in reverse order worst case time complexity $O(n^2)$

(c) Average time complexity:

It occurs when array elements are in jumbled order that is neither ascending nor descending.

Average complexity is $O(n^2)$

Harsh
2003089

(d) Space complexity
Because in insertion sort an extra variable is required for swapping.

Conclusion:
Selection sort is easy to implement & performs well on a small list but less efficient than insertion sort.

PROGRAM:

Selection Sort

```
import java.util.*;

public class SelectionSort {
    public static void main(String[] args) {
        int arr[];

        System.out.println("Enter the number of elements: ");
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        arr = new int[n];

        System.out.println("Enter the elements: ");
        for(int i=0; i<n; i++){
            arr[i] = sc.nextInt();
        }

        sc.close();

        int min = 0;

        System.out.println("Unsorted array ==> \n" + Arrays.toString(arr));

        System.out.println("The steps for sorting in SelectionSort Sort are:
\n");

        int totalComparisions,totalSwaps;
        totalComparisions=totalSwaps=0;

        while(min < arr.length-1) {
            int comp,swap;
            comp = swap = 0;
            for(int i = min+1; i < arr.length; i++) {
                if(arr[i] < arr[min]) {
                    int temp = arr[i];
                    arr[i] = arr[min];
                    arr[min] = temp;
                    totalSwaps++;
                    swap++;
                }
                totalComparisions++;
                comp++;
            }
            System.out.println(Arrays.toString(arr)+"\nComparisions:
"+comp+"\nSwaps: "+swap);
```

```
            min++;
        }

        System.out.println("totalComparisions = " + totalComparisions);
        System.out.println("totalSwaps = " + totalSwaps);

        System.out.println("\nSorted array" + Arrays.toString(arr));
    }
}
```

OUTPUT:

```
PS D:\Harsh\SEM 4\AOA\Assignment> cd "d:\Harsh\SEM 4\AOA\Assignment\" ; if ($?)
Enter the number of elements:
5
Enter the elements:
12
14
16
13
17
Unsorted array ==>
[12, 14, 16, 13, 17]
The steps for sorting in SelectionSort Sort are:

[12, 14, 16, 13, 17]
Comparisions: 4
Swaps: 0
[12, 13, 16, 14, 17]
Comparisions: 3
Swaps: 1
[12, 13, 14, 16, 17]
Comparisions: 2
Swaps: 1
[12, 13, 14, 16, 17]
Comparisions: 1
Swaps: 0
totalComparisions = 10
totalSwaps = 2

Sorted array[12, 13, 14, 16, 17]
PS D:\Harsh\SEM 4\AOA\Assignment>
```

Insertion Sort:

```
import java.util.*;

class InsertionSort{

    public static void sort(int arr[])
    {
        int n = arr.length;

        int totalSwaps = 0;
        int totalComparisions = 0;

        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j = i - 1;
            int swaps = 0;
            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j--;
                swaps++;
            }

            arr[j + 1] = key;

            totalSwaps+=swaps;

            System.out.println("\n"+Arrays.toString(arr));

            if(j==i-1){
                totalComparisions+=(1);
                System.out.println("No. of comparisions in this cycle: " +
(1));
            }
            else{
                totalComparisions+=(i-j);
                System.out.println("No. of comparisions in this cycle: " + (i-
j));
            }

            System.out.println("Swaps = " + swaps);

        }

        System.out.println("\nTotal number of comparisions: " +
totalComparisions);
```

```java
    }

    public static void main(String[] args) {

        int arr[];

        System.out.println("Enter the number of elements: ");
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        arr = new int[n];

        System.out.println("\nEnter the element of the array: ");
        for(int i=0; i<n; i++){
            arr[i] = sc.nextInt();
        }

        sc.close();

        System.out.println("Unsorted array ==> " + Arrays.toString(arr));

        sort(arr);

        System.out.println("\nSorted array ==> " + Arrays.toString(arr));
    }
}
```

OUTPUT:

```
PS D:\Harsh\SEM 4\AOA\Assignment> cd "d:\Harsh\SEM 4\AOA\Assignm
Enter the number of elements:
5

Enter the element of the array:
12
45
5
4
6
Unsorted array ==> [12, 45, 5, 4, 6]

[12, 45, 5, 4, 6]
No. of comparisions in this cycle: 1
Swaps = 0

[5, 12, 45, 4, 6]
No. of comparisions in this cycle: 3
Swaps = 2

[4, 5, 12, 45, 6]
No. of comparisions in this cycle: 4
Swaps = 3

[4, 5, 6, 12, 45]
No. of comparisions in this cycle: 3
Swaps = 2

Total number of comparisions: 11

Sorted array ==> [4, 5, 6, 12, 45]
PS D:\Harsh\SEM 4\AOA\Assignment>
```