PPS Assignment-III

Uppala B Archana

B181867

1)) Implement the following Searching Techniques.

a) Linear search

1.Problem statement

Implement the searching Technique of linear search.

2.Pseudo code/ flow chart

Int Linear search(int[] list, int target)

For(int j=0;j<list.length ; j=j+1)

If(list[j]==target)

Return j

End if

End for

Return -1

End linear search

3.C program

```c
# include <stdio.h>
int main()
{
    int i,data,a[50],size;
    printf("enter the size of array");
    scanf("%d",&size);
    if(size>50)
    {
```

```
        printf("overflow condition");
    }
    else
    {
    printf("enter elements of array:\n");
    for(i=0;i<size;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("enter the element you want to search in the array:");
    scanf("%d",&data);
    for(i=0;i<size;i++)
    {
        if(a[i]==data)
        {
            printf("element found at index:%d",i);
            break;
        }
    }
    if(i==size)
    {
        printf("element not found");
    }
    }
    return 0;
}
```

4. Results (min: 3 IN/OUT)

```
PS C:\Users\user\Desktop\c programming\DSalgo> .\linearsearch
enter the size of array:5
enter elements of array:
1
2
3
4
5
enter the element you want to search in the array:4
element found at index:3
```

```
PS C:\Users\user\Desktop\c programming\DSalgo> .\linearsearch
enter the size of array:3
enter elements of array:
20
535
66
enter the element you want to search in the array:4
element not found
```

```
PS C:\Users\user\Desktop\c programming\DSalgo> .\linearsearch
enter the size of array:9
enter elements of array:
23
32
34
5
67
87
23
45
67
enter the element you want to search in the array:23
element found at index:0
```

5. Observations

Linear search algorithm will search for the every element that is equal to the entered data element and return the index if found.

---

b) Binary search

1.Problem Statement

Implement the searching Technique of binary search.

2.Pseudo code/ flow chart

Binary_search(a.n,key)

Begin

Set start=0,end=n-1,mid=(start+end)/2;

While (start<=end && a[mid]!=key) do

If(key<a[mid] then

Set end=mid-1;

Else

Set end=mid+1;

End if

Set mid=(start+end)/2;

End while

If (start+end)

Return -1;

Return mid;

End

3.C program

```c
# include <stdio.h>
int Binarysearch(int a[],int size,int element);
int main()
{
    int i,data,a[50],size,result;
    printf("enter the size of array");
    scanf("%d",&size);
    if(size>50)
    {
        printf("overflow condition");
    }
    else
    {
    printf("enter elements of array:\n");
    for(i=0;i<size;i++)
    {
        scanf("%d",&a[i]);
    }
    }
    printf("enter the element you want to search in the array:");
    scanf("%d",&data);
    Binarysearch(a,size,data);
     printf("%d is found at Index %d \n",data,Binarysearch(a,size,data));

    return 0;
}
int Binarysearch(int a[],int size, int data)
    {
        int l=0,r=size-1;
        int mid;
```

```
        while(l<r)
        {
            mid=(l+r)/2;
            if(data==a[mid])
            return mid;
            else if (data<a[mid])
            r=mid-1;
            else
            l=mid+1;
        }
        return -1;
    }
```

4. Results (min: 3 IN/OUT)

```
PS C:\Users\user\Desktop\c programming\DSalgo> gcc binarysearch.c -o binarysearch
PS C:\Users\user\Desktop\c programming\DSalgo> .\binarysearch
enter the size of array6
enter elements of array:
12
23
45
76
87
54
enter the element you want to search in the array:12
12 is found at Index 0
```

```
PS C:\Users\user\Desktop\c programming\DSalgo> .\binarysearch
enter the size of array10
enter elements of array:
1
2
3
4
4
5
6
67
78
6
enter the element you want to search in the array:4
4 is found at Index 4
```

5. Observations

- **It is important to define the type of elements that we use in the function definition. Even if it is the array.**

---

2) Implement the following Sorting Techniques.

a) Bubble sort

1.Problem statement

Implement the Bubble sort sorting technique

2.Pseudo code/ flow chart

Begin Bubble sort(list)

For all elements of list

If list[i]>list[i+1]

Swap(list[i],list[i+1])

End if

End for

Return list

End Bubble sort

3.C program

```
# include <stdio.h>
int main()
{
    int i,j,a[50],n,temp;
    printf("enter the size of array");
```

```c
    scanf("%d",&n);
    if(n>50)
    {
        printf("overflow condition");
    }
    else
    {
    printf("enter elements of array:\n");
      for(i=0;i<n;i++)
      {
          scanf("%d",&a[i]);
      }

     for(i=0;i<n-1;i++)
     {
         for(j=0;j<n-1-i;j++)
        {
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
             }
        }
     }
     printf("The sorted array by the bubble sort is:\n");
     for(i=0;i<n;i++)
     {
         printf("%d\t",a[i]);
     }
    }
  return 0;
}
```

4. Results (min: 3 IN/OUT)

```
PS C:\Users\user\Desktop\c programming\DSalgo> gcc bubblesort.c -o bubblesort
PS C:\Users\user\Desktop\c programming\DSalgo> .\bubblesort
enter the size of array5
enter elements of array:
15
16
6
8
5
The sorted array by the bubble sort is:
5       6       8       15      16
```

```
PS C:\Users\user\Desktop\c programming\DSalgo> .\bubblesort
enter the size of array8
enter elements of array:
57
23
12
34
55
67
54
32
The sorted array by the bubble sort is:
12      23      32      34      54      55      57      67
```

```
enter the size of array5
enter elements of array:
5
4
3
2
1
The sorted array by the bubble sort is:
1       2       3       4       5
```

5. Observations

- The main condition is if(a[j]>a[j+1]) to check in the bubble sort

---

b) Insertion sort

1.Problem statement

Implement the Insertion sort sorting technique

2.Pseudo code/ flow chart

Procedure Insertion sort(A)

Int hole position

Int value to insert

For i=1 to length(A) do:

value to insert=A[i]

hole position i

while hole position>0 and A[hole position-1]>value to insert

A[hole position]=A[hole position -1]

hole position - -

end while

end for

end procedure

3.C program

```c
# include <stdio.h>
int main()
{
    int i,j,data,a[50],size ,temp;
    printf("enter the size of array");
    scanf("%d",&size);
    if(size>50)
    {
        printf("overflow condition");
    }
    else
    {
    printf("enter elements of array:\n");
    for(i=0;i<size;i++)
    {
        scanf("%d",&a[i]);
    }

    printf("elements in array are:\n");
     for(i=0;i<size;i++)
     {
         printf("%d\t",a[i]);
     }
    for(i=1;i<size;i++)
    {
        temp=a[i];
        j=i-1;
        while(j>=0 && a[j]>temp)
        {
            a[j+1]=a[j];
            j--;
        }
```

```
        a[j+1]=temp;
    }

    printf("\nthe sorted array by insertion sort is:\n");
     for(i=0;i<size;i++)
     {
         printf("%d\t",a[i]);
     }
    }
    return 0;
}
```

4. Results (min: 3 IN/OUT)

```
PS C:\Users\user\Desktop\c programming\DSalgo> .\insertionsort
enter the size of array5
enter elements of array:
12
23
43
53
5553
elements in array are:
12      23      43      53      5553
the sorted array by insertion sort is:
12      23      43      53      5553
```

```
PS C:\Users\user\Desktop\c programming\DSalgo> .\insertionsort
enter the size of array6
enter elements of array:
1
6
5
43
2
1
elements in array are:
1       6       5       43      2       1
the sorted array by insertion sort is:
1       1       2       5       6       43
```

5. Observations

- **Insertion sort divides the array into subarray one as sorted subarray and the other as unsorted subarray.**
- **The sorted subarray contains one element at first.**

c) Selection sort

1.Problem statement

Implement the Selection sort sorting technique

2.Pseudo code/ flow chart

```
Procedure Selection sort
    List : array of items
    n : size of list
  for i=1 to n
       min=i
      for j=i+1 to n
        if list[j]<list[min] then
            min=j
         end if
       end for
    if indexmin!=I then
       swap list[min] and list[i]
     end if
  end for
end procedure
```

3.C program

```c
# include <stdio.h>
int main()
{
    int i,j,data,a[50],size ,temp;
    printf("enter the size of array");
    scanf("%d",&size);
    if(size>50)
    {
        printf("overflow condition");
    }
    else
    {
    printf("enter elements of array:\n");
    for(i=0;i<size;i++)
    {                                       //to take input from user
        scanf("%d",&a[i]);
    }

    printf("elements in array are:\n");
     for(i=0;i<size;i++)
     {                                        // to output the initial array
         printf("%d\t",a[i]);
     }

     for(i=0;i<size-1;i++)
     {
         int min=i;
         for(j=i+1;j<size;j++)
         {
             if(a[j]<a[min])
             min=j;
         }
         if(min!=i)
         {
             temp=a[min];
             a[min]=a[i];
             a[i]=temp;
         }
     }


    printf("\nthe sorted array by selection sort is:\n");
     for(i=0;i<size;i++)
     {
         printf("%d\t",a[i]);
     }
```

```
    }
    return 0;
}
```

4. Results (min: 3 IN/OUT)

```
PS C:\Users\user\Desktop\c programming\DSalgo> gcc selectionsort.c -o selectionsort
PS C:\Users\user\Desktop\c programming\DSalgo> .\selectionsort
enter the size of array5
enter elements of array:
5
3
43
2
43
elements in array are:
5       3       43      2       43
the sorted array by selection sort is:
2       3       5       43      43
```

```
PS C:\Users\user\Desktop\c programming\DSalgo> .\selectionsort
enter the size of array3
enter elements of array:
2333332
434454
13243
elements in array are:
2333332 434454  13243
the sorted array by selection sort is:
13243   434454  2333332
```

```
PS C:\Users\user\Desktop\c programming\DSalgo> .\selectionsort
enter the size of array6
enter elements of array:
6
5
4
3
2
1
elements in array are:
6       5       4       3       2       1
the sorted array by selection sort is:
1       2       3       4       5       6
```

5. Observations

- **Selection sort divides the array into subarray one as sorted subarray and the other as unsorted subarray.**
- **The sorted subarray is empty here in selection sort.**

---

d) Quick sort(using recursion)

1.Problem statement

Implement the Quick sort sorting technique using recursion.

2.Pseudo code/ flow chart

```
Quick sort( arr ,beg ,end)
 If(beg<end)
 Pivot index=position(arr,beg,end)
Quick sort(arr ,beg , pivotindex)
Quick sort(arr,pivotindex+1,end)
Partition(arr ,beg,end)
Set end as pivot
Pindex=beg-1
For i=beg to end-1
  If arr[i]<pivot
    Swap arr[i] and arr[pindex]
     Pindex++
    Swap pivot and arr[pindex+1]
Return pindex+1
```

3.C program

```c
# include <stdio.h>
int partition(int [],int ,int);
int quicksort(int[],int, int);
int  partition(int a[25],int lb,int ub)
{   int pivot,start,end,temp;
     pivot=a[lb];
     start=lb;
     end=ub;
     while(start<end)
     {
         while(a[start]<=pivot)
         start++;
         while(a[end]>pivot)
         end--;
         if(start<end)
         {
             temp=a[start];
             a[start]=a[end];
             a[end]=temp;
         }

     }
     temp=a[lb];
     a[lb]=a[end];
     a[end]=temp;
     return end;
}
int quicksort(int a[25],int lb,int ub)
{    int loc ,count;
    if(lb<ub)
    {
        loc=partition(a,lb,ub);
        quicksort(a,lb,loc-1);
        quicksort(a,loc+1,ub);
    }
}
int main()
{
    int i,count,a[25];
    printf("how many numbers do you want to enter:");
```

```
    scanf("%d",&count);
    printf("enter %d elements:\n",count);
    for(i=0;i<count;i++)
    {
        scanf("%d",&a[i]);
    }

    quicksort(a,0,count-1);

    printf("order of sorted elements:\n");
    for(i=0;i<count;i++)
      printf("%d\t",a[i]);
    return 0;
}
```

4. Results (min: 3 IN/OUT)

```
PS C:\Users\user\Desktop\c programming\Dsalgo> .\quicksortt
how many numbers do you want to enter:5
enter 5 elements:
-9
2
3
1
2
order of sorted elements:
-9      1       2       2       3
```

```
PS C:\Users\user\Desktop\c programming\Dsalgo> .\quicksortt
how many numbers do you want to enter:9
enter 9 elements:
7
6
10
5
9
2
1
15
7
order of sorted elements:
1       2       5       6       7       7       9       10      15
```

```
PS C:\Users\user\Desktop\c programming\Dsalgo> .\quicksortt
how many numbers do you want to enter:4
enter 4 elements:
-1
2
0
1
order of sorted elements:
-1      0       1       2
```

5. Observations

- **We are going to choose one pivot element.**
- **Pivot element can be anything.**
- **Partition the array in such a way that all the elements<pivots would be to the left side and all the elements>pivot would be to the right side of the pivot.**
- **Passing the arguments into the function should be taken care of..**

e) Merge sort(using recursion)

1.Problem statement

Implement the Merge sort sorting technique using recursion.

2.Pseudo code/ flow chart

```
MergeSort(arr[],left,right)
   If left>right
      return
    mid=(left+right)/2
    MergeSort(arr,left,mid)
    MergeSort(arr,mid,right)
End
```

3.C program

```
# include <stdio.h>
int merge(int[],int,int,int);
int mergesort(int[],int,int);
int mergesort(int a[25],int lb,int ub)
{   int mid;
    if(lb<ub)
    {
        mid=(lb+ub)/2;
        mergesort(a,lb,mid);
        mergesort(a,mid+1,ub);
        merge(a,lb,mid,ub);
    }
}
int merge(int a[25],int lb,int mid,int ub)
 {  int i,j,k,b[25];
    i=lb;
    j=mid+1;
    k=lb;
    while(i<=mid && j<=ub)
    {
      if(a[i]<=a[j])
      {
          b[k]=a[i];
          i++;
      }
      else
      {
          b[k]=a[j];
          j++;
```

```
      }
      k++;
    }
    if(i>mid)
    {
        while(j<=ub)
        {
            b[k]=a[j];
            j++;
            k++;
        }
    }
    else
    {
        while(i<=mid)
        {
            b[k]=a[i];
            i++;
            k++;
        }
    }
    for(k=lb;k<=ub;k++)
    {
        a[k]=b[k];
    }
 }
int main()
{
    int i,count,a[25];
    printf("how many numbers do you want to ente
r:");
```

```
    scanf("%d",&count);
    printf("enter %d elements:\n",count);
    for(i=0;i<count;i++)
    {
        scanf("%d",&a[i]);
    }

    mergesort(a,0,count-1);

    printf("order of sorted elements:\n");
    for(i=0;i<count;i++)
      printf("%d\t",a[i]);
    return 0;
}
```

4. Results (min: 3 IN/OUT)

```
PS C:\Users\user\Desktop\c programming\Dsalgo> gcc mergesort.c -o mergesort
PS C:\Users\user\Desktop\c programming\Dsalgo> .\mergesort
how many numbers do you want to enter:5
enter 5 elements:
-9
2
3
1
2
order of sorted elements:
-9      1       2       2       3
```

```
PS C:\Users\user\Desktop\c programming\Dsalgo> .\mergesort
how many numbers do you want to enter:9
enter 9 elements:
15
5
24
8
1
3
16
10
20
order of sorted elements:
1       3       5       8       10      15      16      20      24
```

```
PS C:\Users\user\Desktop\c programming\Dsalgo> .\mergesort
how many numbers do you want to enter:5
enter 5 elements:
2
3
-1
0
-3
order of sorted elements:
-3      -1      0       2       3
```

5. Observations

- **Complete array is divided into n sub arrays.**
- **Each subarray is having one element.**
- **We keep on dividing the array into subarray, until we get the subarray containing only one element.**
- **After that, we keep on merging the subarrays to produce a new sorted array**.