CP MODULE 5(Part 1of 3)

Sorting and Searching : Bubble sort, Selection sort, Linear Search and Binary search.

SEARCHING

Consider an array A. We have to check whether an ***item*** is present in this array(A) or not. This problem is called **searching**.

The problem of searching is to determine whether the **item** is present in the array or not.

If the **item** is present, the positions can be printed.

Two types of search are

- linear search
- binary search

LINEAR SEARCH

Consider array **a.** Let ***item*** be the element to be searched. A simple approach is to do linear search:-

- Start from the leftmost(first) element of the array a[0] .Set ***found=0***
- compare ***item*** with each element of the array one by one
- If ***item*** matches with an element in the array ,
  - Print the position of the element and set ***found =1***.
- If ***item*** doesn't match with any of elements in the array(i.e. If found is 0),
  - print "item is not in the array"

Complexity of linear search is 0(n)

Algorithm for linear search:

Consider a list(array) *L* of *n* elements with items L[0],L[1],.....L[n-1] , and searching value is *item*, linear search is used to find the index(position) of the item in L

1. Set *i* to 0 and Found to 0
2. If L[i] = ***item***, the search terminates successfully; print" item is found at position i" . Found=1
3. Increase *i* by 1.
4. If *i*<*n*, go to step 2.
5. If Found=0 item is not found.

```
include<stdio.h>
main()
{
int a[20],i,n,item,found=0;

printf("\nHow many elements\n");
scanf("%d",&n);

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

printf("\nEnter the element to be searched");
scanf("%d",&item);

for(i=0;i<n;i++)
{
        if(a[i]==item)
        {
        printf("Item is found at %d",i1);
        found=1;
        }
}

if(found==0)
{
printf("\nElement not found");
}

}
```

LINEAR SEARCH USING FUNCTION for linear search

```
#include<stdio.h>
void linear(int a[], int n,int item)
{
int found=0,i;
        for(i=0;i<n;i++)
```

```
        {
          if(a[i]==item)
          {
           printf("Item is found at %d",i);
           found=1;
          }
        }
        if(found==0)
        {
        printf("\nElement not found");
        }
}
main()
{
int a[20],item,i,n;
printf("\nHow many elemnts");
scanf("%d",&n);
printf("\nEnter the elements");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("\nEnter the element\ to be searched");
scanf("%d",&item);
linear(a,n,item);
}
```

Q. Example find the number 8 in following array A using linear search. Show the steps.

9 1 3 8 7 6

i=0
Compare 8 with A[0] 8!=9
i=i+1=0+1=1

Compare 8 with A[1] 8!=1
i=i+1=1+1=2

Compare 8 with A[2] 8!=3
i=i+1=2+1=3

Compare 8 with A[3] 8==8 *found=1*

The element 8is found at position 3

BINARY SEARCH

Binary search can be used for searching elements only if the elements in the array is **sorted (arranged in ascending order)**.

If the elements in the array are sorted, then it is better to perform searching using binary search than linear search. Complexity of binary search is $O(n\log_2 n)$

Algorithm:Consider sorted array A with n elements. Let **item** be the element to be searched.

Step1:Let beg denote the first(0th) and end denote last(n-1) positions in the array.

Step 2:If beg<=end go to step 3 otherwise go to**step 5**

Step 3: compute middle position using mid=(beg+end)/2

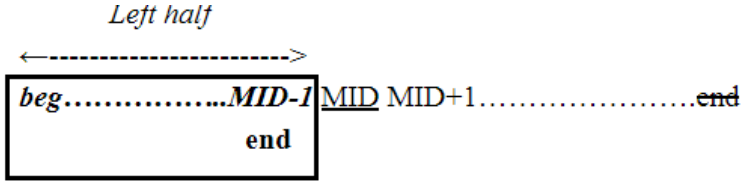
Step 4: To search for an element **item** in the array, compare it with the middle element in the array that is A[mid], where mid denotes the middle position in the array.

- ***Step 4a***:.If it matches(***item is equal to A[mid]***) , then the search is successful and it is terminated. **GO TO STEP 6**
- ***Step 4b***:If **item** is NOT EQUAL to A[mid], the array is divided into two halves.
    - The left half consists of the elements A[beg], A[beg+1], · · · · · · A[mid−1]
    - the right half consists of the elements A[mid+1], A[mid+2], · · · · · · A[end].

(Since the array is sorted, all elements in left half will be less than A[mid] and all elements in the right half will be greater than A[mid].)

- If **item**> A[mid] then key will not be in left half(because left half elements are lesser than middle element) and hence searching is to be done in the right half only.
    - Thus set beg = mid + 1.

- Otherwise(**item<A[mid])**, you need to search in the left half(because righthalf elements are greater than middle element) by setting end= mid − 1.



GO TO STEP 2

Step 5. *If beg>end then ITEM IS NOT IN THE array*

Step 6*:Stop*

```
#include<stdio.h>
main()
{
int a[20], i,n, beg,end, mid, e;
printf("\nHow many elements");
scanf("%d",&n);

printf("\nEnter the elements IN ASCENDING ORDER");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}

printf("\nEnter the element to be searched");
scanf("%d",&e);

beg=0;
end=n-1;
while(beg<=end)
{
        mid=(beg+end)/2;
        if(a[mid]== e)
        {
                printf("\nElement is found at index %d",mid);
                break;
        }
        else if (a[mid]<e)
        {
                beg=mid+1;
        }
        else if (a[mid]>e)
        {
                end=mid-1;
        }
}
 if(beg>end)
{
        printf("\nElement not found");
```

```
}
}
```

BINARY SEARCH USING FUNCTION

```
#include<stdio.h>

void binary(int a[],int beg,int end,int e)
{
int mid;
while(beg<=end)
{
        mid=(beg+end)/2;
        if(a[mid]== e)
        {
        printf("\nElement is found at index %d",mid);
        break;
        }
        else if (a[mid]<e)
        {
        beg=mid+1;
        }
        else if (a[mid]>e)
        {
          end=mid-1;
        }
}

        if(beg>end)
        {
        printf("\nElement not found");
        }
}



main()
{
int a[20], i,n, beg,end, mid, e;

printf("\nHow many elements");
scanf("%d",&n);
```

```
printf("\nEnter the elements IN ASCENDING ORDER");
for(i=0;i<n;i++)
{
        scanf("%d",&a[i]);
}

printf("\nEnter the element to be searched");
scanf("%d",&e);

beg=0;
end=n-1;
binary(a,beg,end,e);
}
```

Q. Example find the number 8 in following array A using binarysearch. Show the steps.

9	**1**	**3**	**8**	**7**	**6**

First sort these numbers in ascending order

1	3	6	7	8	9
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]

Here number of elements N=6
beg=0 end =5 (i.e N-1)
Since beg<=end(0<=5)
So mid=(beg+end)/2=(0+5)/2=2.5= 2
Compare search element 8 with A[mid]i.e A[2]
8>6(i.e element>A[mid])
So element comes after mid
beg=mid+1=3
Since beg<=end i.e 3<=5
mid=(beg+end)/2=(3+5)/2=4
Compare search element 8 with A[mid]i.e A[4]
8=8
So element is found at position mid(4)

SORTING

Sorting means arranging the elements of a list in order. The order could be ascending or descending. Usually sorting means arranging in ***ascending***(smallest to largest) ***order***.
There are many sorting algorithms

- Bubble sort- Bubble the largest first and place in last position. then second largest etc
- Selection sort- Select the smallest first and place in starting (0th)position. then second smallest etc.

//BUBBLE SORT

- The bubble sort algorithm
    - starts by comparing each element in the list with its adjacent element, and swapping them if they are not in ascending order.
    - In this way, when all the elements are compared once, the largest element will move to its correct position(last position).
    - The algorithm then repeats this process until all the elements settle down in their corresponding positions.

Sort the numbers 5, 1, 4, 7, 2, 6 using BUBBLE SORT.

There are 6 numbers in this list. Find first 5 largest numbers in this list.

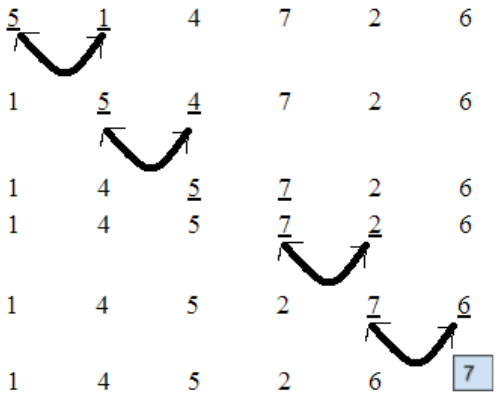
Pass 0 finds 0^{th} largest. Pass 1 finds 1st largest….Pass 4 finds fourth largest.

Compare 0th element and 1st element If they are not in ascending order swap(exchange) them.

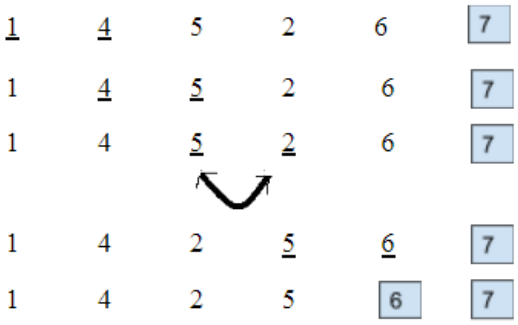
Then compare 1st and 2nd element if not in order swap them etc…

Steps in bubble sort are:- Here n=6(number of elements in list)

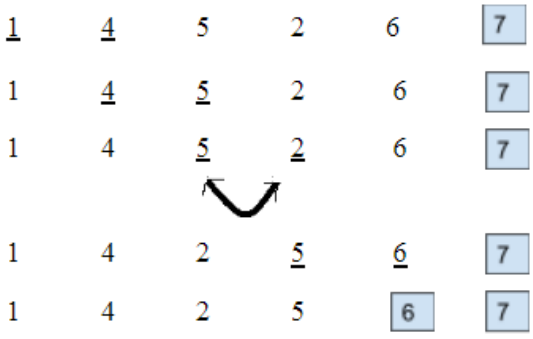
PASS 0 (Find FIRST (0th) LARGEST)----------0th largest



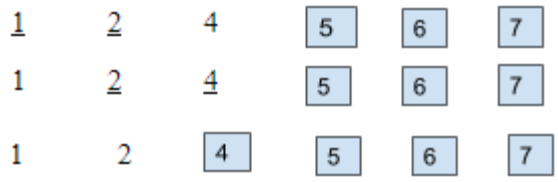
PASS 1 (Find SECOND(1st) LARGEST)



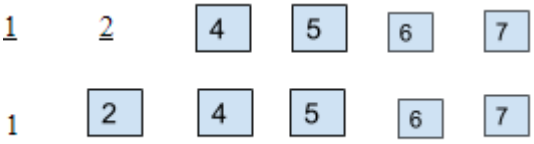
PASS 2 (Find THIRD(2nd) LARGEST)



PASS 3 (Find FOURTH(3rd) LARGEST)



PASS 4 (Find FIFTH(4th) LARGEST)----------n-2th largest(Here n=6 numbers)




```
//n numbers//find n-1 largest//n-1 pass// pass 0 to pass n-2
//in each pass compare a[j] with a[j+1]
//in PASS 0 compare a[0] with a[1] ->  if a[0]>a[1] swap a[0] and a[1]//.....  .....
        //compare a[n-2] with a[n-1]  if a[n-2]>a[n-1] swap a[n-2] and a[n-1]
        //Compare(a[j] with a[j+1]) in Pass 0->  j= 0 to n-2-0(passno)
        //After pass 0 a[n-1] is largest. so consider a[0] to a[n-2] in next pass
//in PASS 1 last a[0] with a[1]..... a[n-3] with a[n-2]
        //Compare(a[j] with a[j+1] in Pass 1-> j= 0 to n-2-1(pasno)
//.....
//in PASS n-2 only a[0] and a[1] are remaining to be sorted
        //compare a[0] with a[1] ->  if a[0]>a[1] swap a[0] and a[1]
        //in pass n-2 j=0 (ie j=0 to n-2-(n-2) )
//*********Pass no i ranges from 0 to n-2
//**********in each Pass i the value of j ranges from 0 to n-2-i
//*************inside each pass compare a[j] and a[j+1] if a[j]>a[j+1] swap them
#include<stdio.h>
main()
{
  int i,  j,  k, m,  n,  a[20],      temp;
  printf("\nHow many numbers?");
  scanf("%d",&n);
  printf("\nEnter the numbers..");
 for(i=0;  i<n;  i++)
 {       scanf("%d",&a[i]);
 }
printf("\nBUBBLE SORT\n");
m=n;
for(i=0;i<=n-2;i++)
```

```
{
        for(j=0; j<=n-2-i; j++)
        {
                if(a[j]>a[j+1])
                {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
         }
        }
}
        printf("\nSorted List\n");
for(i=0;i<m;i++)
        {printf("%d\t",a[i]);
        }       }
```

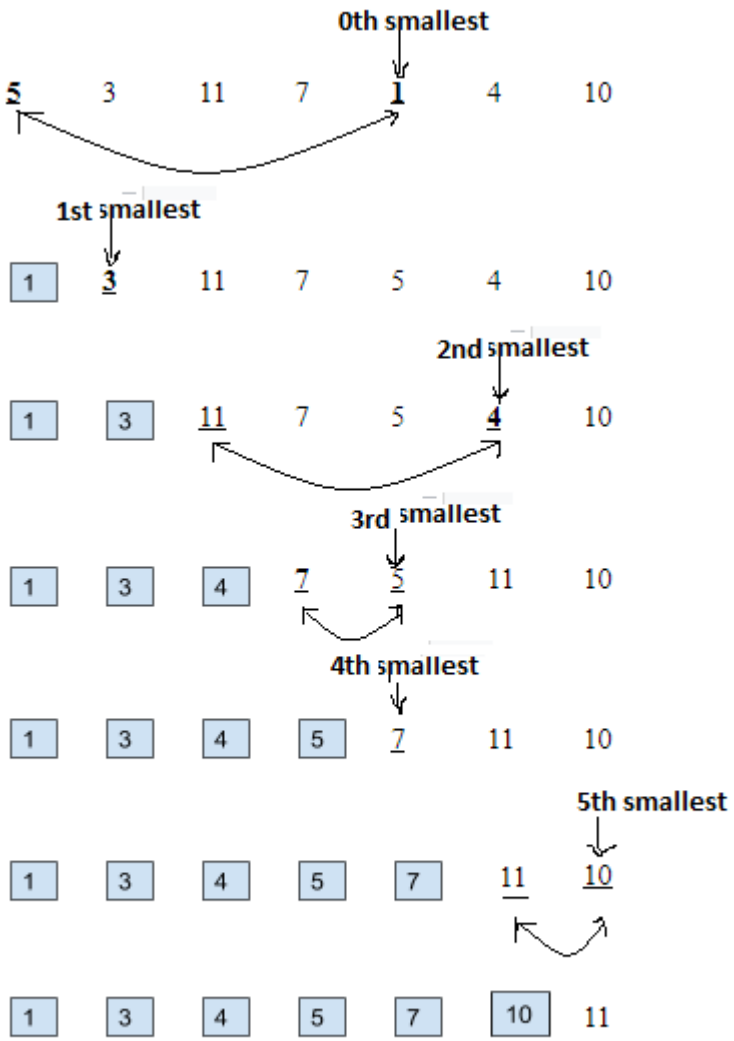
SELECTION SORT

This algorithm first finds the smallest element in the array and exchanges it with the element in the zeroth position. Next it finds next smallest element in the remaining array (an array without the zeroth element) and swaps it with the element at first position. Then it finds the smallest element in the remaining array (an array without zeroth and first elements) and swaps it with the element at second position, and continues in this way until the entire array is sorted.
Consider array A with N elements(A[0] A[1]...A[N-1]). Selection sort algorithm aims to find first N-1(0th largest to N-2th largest) largest elements. It first places 0th smallest in 0th position, 1st smallest in 1st position…..N-2th in N-2th position

```
#include<stdio.h>
 main()
{
        int i, j,  k,  n,  a[20],  smallestloc,  temp;
        printf("\nHow many numbers?");
        scanf("%d",&n);
 printf("\nEnter the numbers..");
for(i=0;  i<n;  i++)
{
    scanf("%d",&a[i]);
}
printf("\n\nSELECTION SORT\n");
for(i=0; i<=n-2; i++)
{
```

```
        smallestloc=i;                  // Let i be the location of ith smallest number
       for(j=i+1;j<n;j++)               // to find the number smaller than ith number
       {
            if(a[smallestloc]>a[j])   // if jth number is the smaller than smallestnumber(assumed)
            {
                smallestloc=j;    //now location of smaller number is at jth location
            }
        }
        if(i!=smallestloc)    //swap the actual smallest element with ith element(assumed smallest)
        {
                    temp=a[i];
                    a[i]=a[smallestloc];
                    a[smallestloc]=temp;
        }

}
printf("\nSorted List\n");
                    for(i=0;i<n;i++)
                    {
                    printf("%d\t",a[i]);
                    }
}
```



Sort the following numbers using selection sort. Show steps

5, 3, 11, 7, 1, 4, 10

SELECTION SORT

```
for(i=0;i<=n-2;i++)
{
 smallestloc=i;
      for(j=i+1;j<=n-1;j++)
      {
          if(a[smallestloc]>a[j])
          {
          smallestloc=j;
          }
       }
       if(i!=smallestloc)
        {
            temp=a[i];
            a[i]=a[smallestloc];
            a[smallestloc]=temp;
          }
}
```

BUBBLE SORT

```
for(i=0;i<=n-2;i++)
{
   for(j=0; j<=n-2-i; j++)
  {
      if(a[j]>a[j+1])
     {
           temp=a[j];
           a[j]=a[j+1];
           a[j+1]=temp;
        }
    }
}
```

Perform binary search after sorting the numbers using bubble sort

```
#include<stdio.h>
int * bubble(int a[],int n)
{
int temp,j;
```

```
printf("\nBUBBLE SORT\n");
        while(n!=0)
        {
                for(j=0; j<n-1; j++)
                {
                        if(a[j]>a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                         }
                }n--;
        }
return a;
}
void binary(int a[],int beg,int end,int e)
{
int mid;
while(beg<=end)
{
        mid=(beg+end)/2;
        if(a[mid]== e)
        {
        printf("\nElement is found at index %d",mid);
        break;
        }
        else if (a[mid]<e)
        {
        beg=mid+1;
        }
        else if (a[mid]>e)
        {
          end=mid-1;
        }
}

        if(beg>end)
        {
        printf("\nElement not found");
        }
```

```
}


main()
{
int a[20], i,n, beg,end, mid, e;
int *p;

printf("\nHow many elements");
scanf("%d",&n);
printf("\nEnter the elements ");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
```

p=bubble(a,n);

```
printf("\n Sorted List\n");
        for(i=0;i<n;i++)
        {
                a[i]=p[i];

        }
printf("\nEnter the element to be searched");
scanf("%d",&e);

beg=0;
end=n-1;
 binary(a,beg,end,e);
}
```

Perform binary search after sorting numbers using selection sort

```
#include<stdio.h>
```

int * selsort(int a[],int n)

```
{
int temp,i,j,smallestloc;

        printf("\n\nSELECTION SORT\n");
        for(i=0; i<=n-2; i++)
        {
        smallestloc=i;
```

```
        for(j=i+1;j<n;j++)
        {
                        if(a[smallestloc]>a[j])
                {
                smallestloc=j;
                 }
        }
                if(i!=smallestloc)
                {
        temp=a[i];
        a[i]=a[smallestloc];
        a[smallestloc]=temp;
                 }
        }
return a;
}
```

void binary(int a[],int beg,int end,int e)

```
{
int mid;
while(beg<=end)
{
        mid=(beg+end)/2;
        if(a[mid]== e)
        {
        printf("\nElement is found at index %d",mid);
        break;
        }
        else if (a[mid]<e)
        {
        beg=mid+1;
        }
        else if (a[mid]>e)
        {
          end=mid-1;
        }
}

if(beg>end)
{
printf("\nElement not found");
```

```
}
}
```

main()

```
{
int a[20], i,n, beg,end, mid, e;
int *p;
clrscr();
printf("\nHow many elements");
scanf("%d",&n);

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

p=selsort(a,n);

```
printf("\n Sorted List\n");
        for(i=0;i<n;i++)
        {
                a[i]=p[i];
        }
printf("\nEnter the elementto be searched");
scanf("%d",&e);

beg=0;
end=n-1;
```

binary(a,beg,end,e);

```
}
```