Searching techniques in data structure:

1)linear search

2)binary search

-->linear search:it is a method where we are searching from 0th index to n-1. If the search element is found in the middle we will stop

Searching and display message as ‘found’ otherwise we will print not found. Since we are searching the entire array it is slower.

#include <stdio.h>

int main()

{

int a[20],n,i,element;

printf("enter size :");

scanf("%d",&n);

printf("enter numbers %d :",n);

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

scanf("%d",&a[i]);

printf("enter the element to be searched");

scanf("%d",&element);

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

{

if(element==a[i])

{

printf("element is found at %d",i);

break;

}

}

if(i==n)

printf("given element is not found");

}

(Using count)

#include <stdio.h>

int main()

{

int a[10],n,i,count=0;

printf("enter 10 values:");

for(i=0;i<10;i++)

scanf("%d",&a[i]);

printf("enter the element to be searched");

scanf("%d",&n);

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

{

if(a[i]==n)

count++;

}

if(count!=0)

printf("\n %d is found for %d times",n,count);

else

printf("\n %d is not found",n);

}

enter 10 values:1

2

3

5

6

4

7

8

9

2

enter the element to be searched 4

4 is not found

#include <stdio.h>

int main()

{

int a[10],n,i;

printf("enter 10 values:");

for(i=0;i<10;i++)

scanf("%d",&a[i]);

printf("enter the element to be searched");

scanf("%d",&n);

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

{

if(a[i]==n)

{

printf("\n %d is found at a[%d]",n,i);

break;

}

else if(i==10)

printf("\n %d is not found",n);

}

}

enter 10 values:1

2

3

4

5

6

7

8

9

10

enter the element to be searched1

1 is found at a[0]

#include <stdio.h>

int main()

{

int a[100],n,i,even=0,odd=0;

printf("enter 10 values:");

for(i=0;i<10;i++)

scanf("%d",&a[i]);

for(i=0;i<10;i++)

{

if(a[i]%2==0)

even++;

else

odd++;

}

printf("\n even elements:%d",even);

printf("\n odd elements:%d",odd);

}

enter 10 values:1

2

3

4

5

6

7

8

9

10

even elements:5

odd elements:5

Binary search:

#include <stdio.h>

int main()

{

int arr[50],n,i,x,first,last,mid,flag=0;

printf("enter size of an array");

scanf("%d",&n);

printf("\n enter array elements(ascending order)\n");

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

scanf("%d",&arr[i]);

printf("\n enter the element to search:");

scanf("%d",&x);

first=0;

last=n-1;

while(first<=last)

{

mid=(first+last)/2;

if(x==arr[mid])

{

flag=1;

break;

}

else

if(x>arr[mid])

first=mid+1;

else

last=mid-1;

}

if(flag==1)

printf("\n element is found at arr[%d]",mid);

else

printf("\n element not found");

return 0;

}

enter size of an array5

enter array elements(ascending order)

2

4

6

8

10

enter the element to search:6

element is found at arr[2]

#include <stdio.h>

int main()

{

int arr[50],n,i,x,first,last,mid,flag=0;

printf("enter size of an array");

scanf("%d",&n);

printf("\n enter array elements(ascending order)\n");

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

scanf("%d",&arr[i]);

printf("\n enter the element to search:");

scanf("%d",&x);

first=0;

last=n-1;

while(first<=last)

{

mid=(first+last)/2;

printf("\n mid:%d",mid);

if(x==arr[mid])

{

flag=1;

break;

}

else

if(x>arr[mid])

first=mid+1;

else

last=mid-1;

}

if(flag==1)

printf("\n element is found at arr[%d]",mid);

else

printf("\n element not found");

return 0;

}

enter size of an array5

enter array elements(ascending order)

10

20

30

40

50

enter the element to search:30

mid:2

element is found at arr[2]

Selection sort:

this algorithm will first find the smallest element in the array and swap it with the element in the first position,then it will find the second smallest element and swap it with the element in the second position,and it will keep on doing this until the entire array is sorted.

-->it is called selection sort because it repeatedly selects the next-smallest element and swaps into the right place.

Time complexity:

Worst case:O(n^2)

Best case:O(n^2)

Average case:O(n^2)

Space complexity:O(1)

#include <stdio.h>

int main()

{

int arr[10],i,j,temp;

printf("\n enter 10 elements:");

for(i=0;i<10;++i)

scanf("%d",&arr[i]);

printf("\n before sort:");

for(i=0;i<10;i++)

printf("%d",arr[i]);

for(i=0;i<10;i++)

{

for(j=i+1;j<10;j++)

{

if(arr[i]>arr[j])

{

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

}

printf("\n after sort:");

for(i=0;i<10;i++)

printf("%d,",arr[i]);

return 0;

}

enter 10 elements:3

4

7

2

9

4

0

1

6

3

before sort:3472940163

after sort:0,1,2,3,3,4,4,6,7,9,

Bubble sort:

It is a simplest algorithm which is used to sort a given set of n elements provided in form of an array with n number of elements provided in form of an array with n number of elements. Bubble sort compares all the elements one by one and sort them based on their values

If the given array has to be sorted in ascending order,then bubble sort will start by comparing the first element of array with the second element if the first element is greater than second element it will swap both the elements and then move on to compare the second and third element, and so on.

This algorithm is not so good for large data

Steps to implement :

->moving along the row ,compare 2 items at a time

->if they are in wrong order,swap them

->when u reach the end of row,repeat from step1

->keep repeating them until they are in order

TIME COMPLEXITY:

Worst case:O(n^2)

Best case:O(n)

Average case:O(n^2)

Space complexity:O(1)

#include <stdio.h>

int noofitems,x[100],i,j;

void sort();

void display();

void input();

main()

{

int i;

printf("\n enter how many elements do u want to store\n");

scanf("%d",&noofitems);

printf("\n enter %d elements",noofitems);

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

scanf("%d",&x[i]);

sort();

display();

}

void display()

{

int i;

printf("\n sorted elements are:\n");

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

printf("%5d",x[i]);

}

void sort()

{

int swap=1,temp;

for(i=0;i<noofitems && swap==1;++i)

{

swap=0;

for(j=0;j<noofitems-(i+1);++j)

if(x[j]>x[j+1])

{

temp=x[j];

x[j]=x[j+1];

x[j+1]=temp;

swap=1;

}

}

}

enter how many elements do u want to store:

5

enter 5 elements:

9

5

3

7

2

sorted elements are:

2 3 5 7 9

Insertion sort:

It is comparison based sorting algorithm.an element which is to be “insert”ed in this sorted sub-list,has to find its appropriate place and then it has to be inserted there.hence the name,insertion sort.

Steps to implement:

->if it is the first element,it is already sorted.return 1;

->pick next element

->compare with all elements in the sorted sub-list

->shift all the elements in the sorted sub-list that is greater than the value to be sorted

->insert the value

->repeat until list is sorted

Time complexity:

Worst case:O(n^2)

Best case:O(n)

Avg case:O(n^2)

Space complexity:O(1)

--->we use insertion sort for short lists,for sorting small sub-lists in quick sort

#include <stdio.h>

main()

{

int a[100],i,j,n,temp;

printf("\n enter no of elements to store\n");

scanf("%d",&n);

printf("\n enter %d values",n);

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

scanf("%d",&a[i]);

for(i=1;i<n;i++)

{

temp=a[i];

j=i-1;

while(j>0 && a[j]>temp)

{

a[j+1]=a[j];

j--;

}

a[j+1]=temp;

}

printf("\n elements after sort");

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

printf("\n%d",a[i]);

}

enter no of elements to store

5

enter 5 values

1

2

37

9

3

elements after sort

1

2

3

9

37

To find largest element in an array

#include <stdio.h>

int main()

{

int i, n,arr[100];

printf("Enter the number of elements :");

scanf("%d", &n);

for (i = 0; i < n; ++i) {

printf("Enter elements %d:",i+1);

scanf("%d",&arr[i]);

}

for (i = 1; i < n; ++i) {

if (arr[0] < arr[i])

arr[0] = arr[i];

}

printf("Largest element = %d", arr[0]);

return 0;

}

Enter the number of elements :5

Enter elements 1:2

Enter elements 2:5

Enter elements 3:8

Enter elements 4:3

Enter elements 5:7

Largest element = 8