

# Practical-01

## Linear Search:

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

int linearsearch(int a[], int key, int n){

    int i = 0;
    for (i = 0; i < n; i++)
    {
        if (key == a[i])
        {
            printf("%d found at %d position!\n",key,i+1);
            return i+1;
            break;
        }
    }

    if (i == n-1)
    {
        printf("Element not found!!");
    }

}

int main()
{

    int key = rand();
    float count = 0.0;
    srand(time(0));

    int n;
    printf("Enter total no. of elements u want to store: ");
    scanf("%d", &n);
    int a[n];

    for (int i = 0; i < n; i++)
    {
        a[i] = i +1;
    }
}
```

```

    }

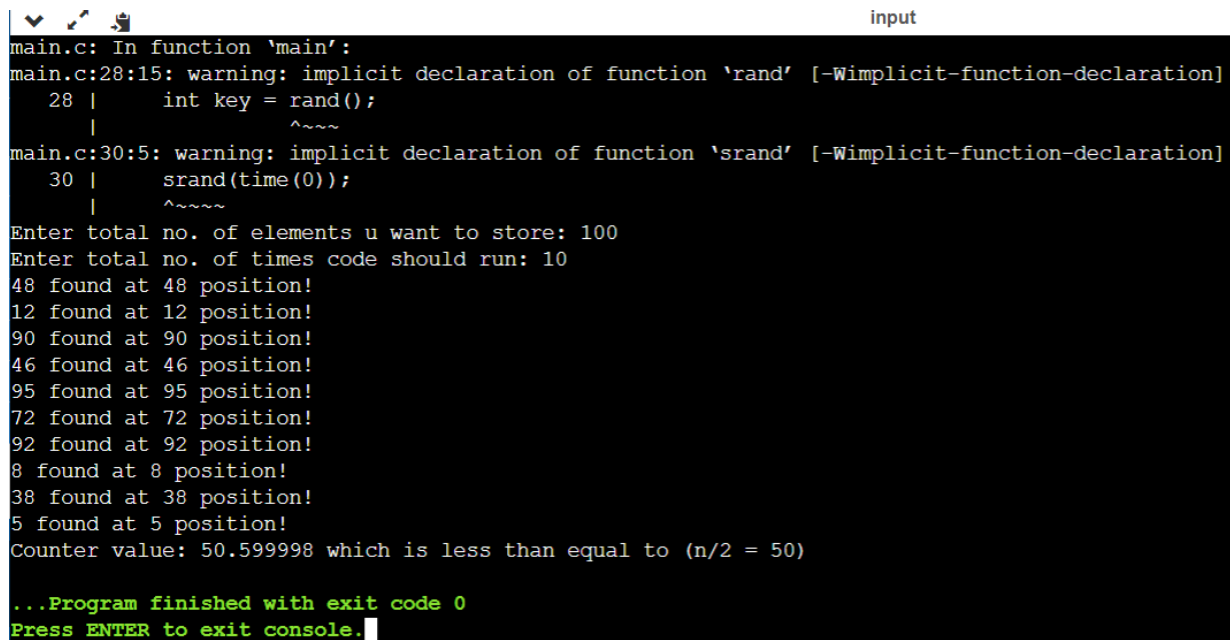
    int run;
    printf("Enter total no. of times code should run: ");
    scanf("%d", &run);

    for (int i = 0; i < run; i++)
    {
        key = rand()%100 + 1 ;
        count += (float) linearsearch(a, key, n)/ run;
    }

    printf("Counter value: %f which is less than equal to (n/2 = %d)",
count, n/2);

    return 0;
}

```



```

main.c: In function 'main':
main.c:28:15: warning: implicit declaration of function 'rand' [-Wimplicit-function-declaration]
   28 |     int key = rand();
      |             ^~~~
main.c:30:5: warning: implicit declaration of function 'srand' [-Wimplicit-function-declaration]
   30 |     srand(time(0));
      |     ^~~~~
Enter total no. of elements u want to store: 100
Enter total no. of times code should run: 10
48 found at 48 position!
12 found at 12 position!
90 found at 90 position!
46 found at 46 position!
95 found at 95 position!
72 found at 72 position!
92 found at 92 position!
8 found at 8 position!
38 found at 38 position!
5 found at 5 position!
Counter value: 50.599998 which is less than equal to (n/2 = 50)

...Program finished with exit code 0
Press ENTER to exit console.

```

## Binary Search:

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

int binary(int array[], int upper, int lower, int key){
    int m = (upper + lower)/2;

    if (array[m] == key)
    {
        printf("%d found at position %d\n", array[m], m+1);
        return m+1;
    }

    else if(key < array[m]){
        return binary(array, upper - 1, lower, key);
    }

    else{
        return binary(array, upper, lower + 1 , key);
    }
}

int main()
{
    int key = rand();
    float count = 0.0;
    srand(time(0));

    int n;
    printf("Enter total no. of elements u want to store: ");
    scanf("%d", &n);
    int a[n];

    for (int i = 0; i < n; i++)
    {
        a[i] = i +1;
    }

    int run;
    printf("Enter total no. of times code should run: ");
```

```

scanf("%d", &run);

for (int i = 0; i < run; i++)
{
    key = rand()%100 + 1 ;
    count += (float) ( binary(a,99,0,key) )/n;
}

printf("Average Time Taken: %f Time Complexiity = %f\n", count,
log2(n));

return 0;
}

```

```

Enter total no. of elements u want to store: 100
Enter total no. of times code should run: 15
96 found at position 96
45 found at position 45
97 found at position 97
61 found at position 61
87 found at position 87
9 found at position 9
9 found at position 9
9 found at position 9
9 found at position 9
99 found at position 99
43 found at position 43
49 found at position 49
40 found at position 40
87 found at position 87
82 found at position 82
Counter value: 8.220000 which is less than equal to ( $\log_2(n) = 6.643856$ )

...Program finished with exit code 0
Press ENTER to exit console.

```

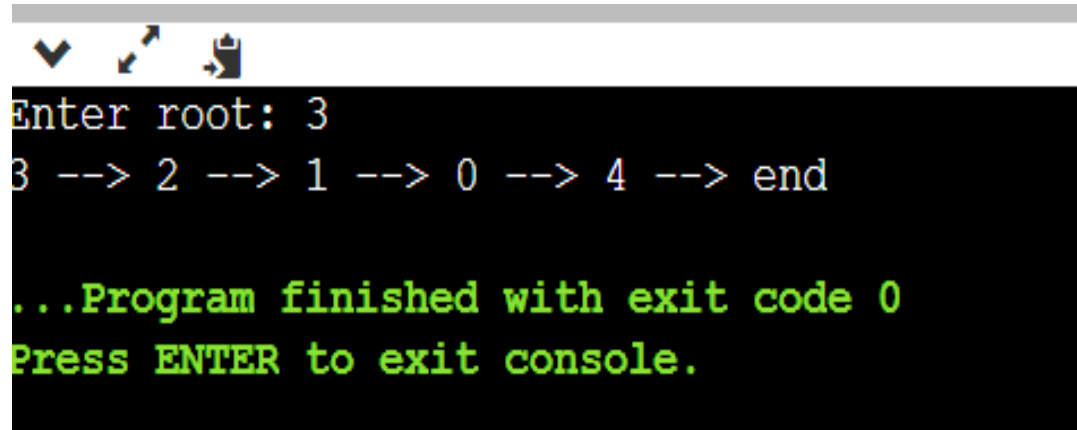
## Practical-02

Q01. 5 nodes Graph Traversal using DFS and Adjacency matrix

```
#include<stdio.h>
int nodes = 5;
int A[5][5] = {{0, 1, 0, 0, 1},
               {1, 0, 1, 0, 0},
               {0, 1, 0, 1, 0},
               {0, 0, 1, 0, 1},
               {1, 0, 0, 1, 0}};
int visited[5] = {0, 0, 0, 0, 0};

void DFS(int i){
    int j;
    printf("%d ", i);
    visited[i] = 1;
    for ( j = 0; j < 5; j++)
    {
        if (A[i][j] == 1 && !visited[j])
        {
            DFS(j);
        }
    }
}

int main(){
    int root;
    printf("Enter root node: ");
    scanf("%d", &root);
    DFS(root);
    return 0;
}
```

A terminal window with a black background and white and green text. At the top, there are three small icons: a checkmark, a cursor, and a clipboard. The text in the terminal reads: "Enter root: 3", "3 --> 2 --> 1 --> 0 --> 4 --> end", "...Program finished with exit code 0", and "Press ENTER to exit console." in green.

```
Enter root: 3
3 --> 2 --> 1 --> 0 --> 4 --> end

...Program finished with exit code 0
Press ENTER to exit console.
```

Q02. 5 nodes Graph Traversal using BFS and Adjacency List

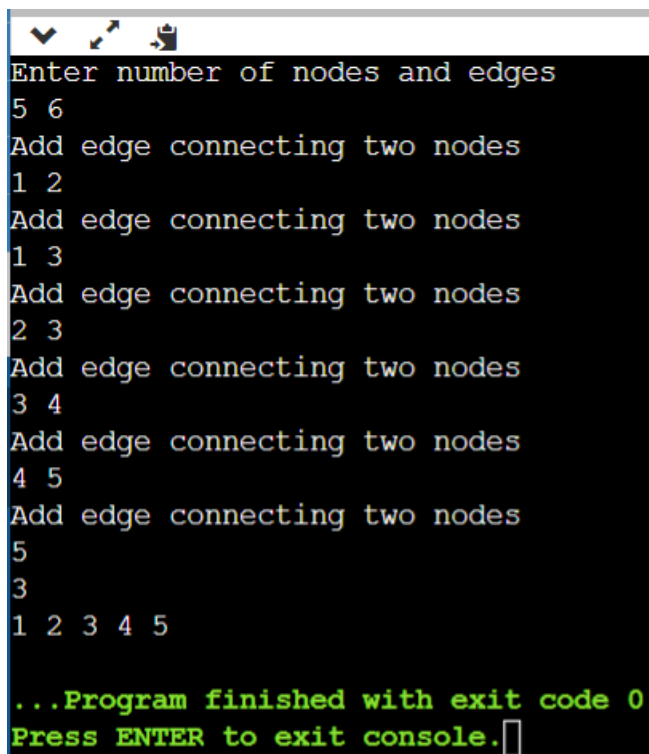
```
//CODE IN C++
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define tej ios_base::sync_with_stdio(false), cin.tie(0), cout.tie(0);
#define ve vector<int>
#define vl vector<long long int>
#define pb push_back
#define fo(i, n) for (i = 0; i < n; i++)
void solve()
{
    ll n, m, i, t, a, b;
    cout << "Enter number of nodes and edges" << endl;
    cin >> n >> m;
```

```

v1 v[n + 1];
while (m--)
{
    cout << "Add edge connecting two nodes" << endl;
    cin >> a >> b;
    v[a].pb(b);
    v[b].pb(a);
}
queue<ll> q;
ll vis[n + 1] = {0};
vis[1] = 1;
q.push(1);
cout << 1 << " ";
while (!q.empty()){
    t = q.front();
    q.pop();
    for (i = 0; i < v[t].size(); i++){
        if (vis[v[t][i]] == 0){
            q.push(v[t][i]);
            vis[v[t][i]] = 1;
            cout << v[t][i] << " ";
        }
    }
}
}

```

```
}  
int main()  
{  
    tej int t = 1;  
    while (t--)  
  
    {  
        solve();  
    }  
    return 0;  
}
```



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are three small icons: a downward arrow, a double-headed arrow, and a clipboard. The text in the terminal is as follows:

```
Enter number of nodes and edges  
5 6  
Add edge connecting two nodes  
1 2  
Add edge connecting two nodes  
1 3  
Add edge connecting two nodes  
2 3  
Add edge connecting two nodes  
3 4  
Add edge connecting two nodes  
4 5  
Add edge connecting two nodes  
5  
3  
1 2 3 4 5  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```



## Practical-03

### Q01. Merge Sort

```
#include <stdio.h>
#include <stdlib.h>
int counter = 0;

void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0;
    j = 0;
    k = l;

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
```

```
        i++;
    }
    else {
        arr[k] = R[j];
        j++;
    }
    k++;
}
```

```
while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
}
```

```
while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
}
counter++;
}
```

```

void mergeSort(int arr[], int l, int r)
{
    if (l < r) {

        int m = l + (r - l) / 2;
        counter++;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);

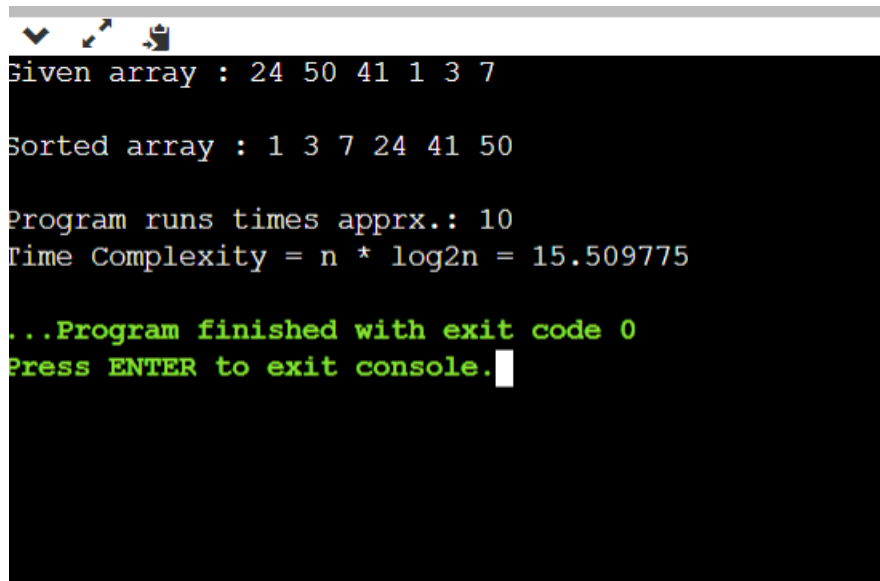
    }
}

void printArray(int A[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

int main()
{
    int arr[] = { 24, 50, 41, 1, 3, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);

```

```
printf("Given array : ");  
printArray(arr, arr_size);  
  
mergeSort(arr, 0, arr_size - 1);  
  
printf("\nSorted array : ");  
printArray(arr, arr_size);  
printf("\nProgram runs times apprx.: %d", counter);  
printf("\nTime Complexity =  $n * \log_2 n = 6 * \log_2(6)$ ");  
return 0;  
}
```



```
Given array : 24 50 41 1 3 7  
  
Sorted array : 1 3 7 24 41 50  
  
Program runs times apprx.: 10  
Time Complexity =  $n * \log_2 n = 15.509775$   
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

## Q02.Quick Sort

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

int counter = 0;

int count = 0, flag = 0;
void swap(int arr[], int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

int partition(int arr[], int l, int x)
{
    int pivot = arr[x];
    int i = l - 1;

    for (int j = l; j < x; j++)
    {
        if (arr[j] < pivot)
        {
            i++;
            swap(arr, i, j);
        }
    }

    swap(arr, i + 1, x);
    flag++;
    return (i + 1);
}

void quicksort(int arr[], int l, int x)
{
    if (l < x)
```

```

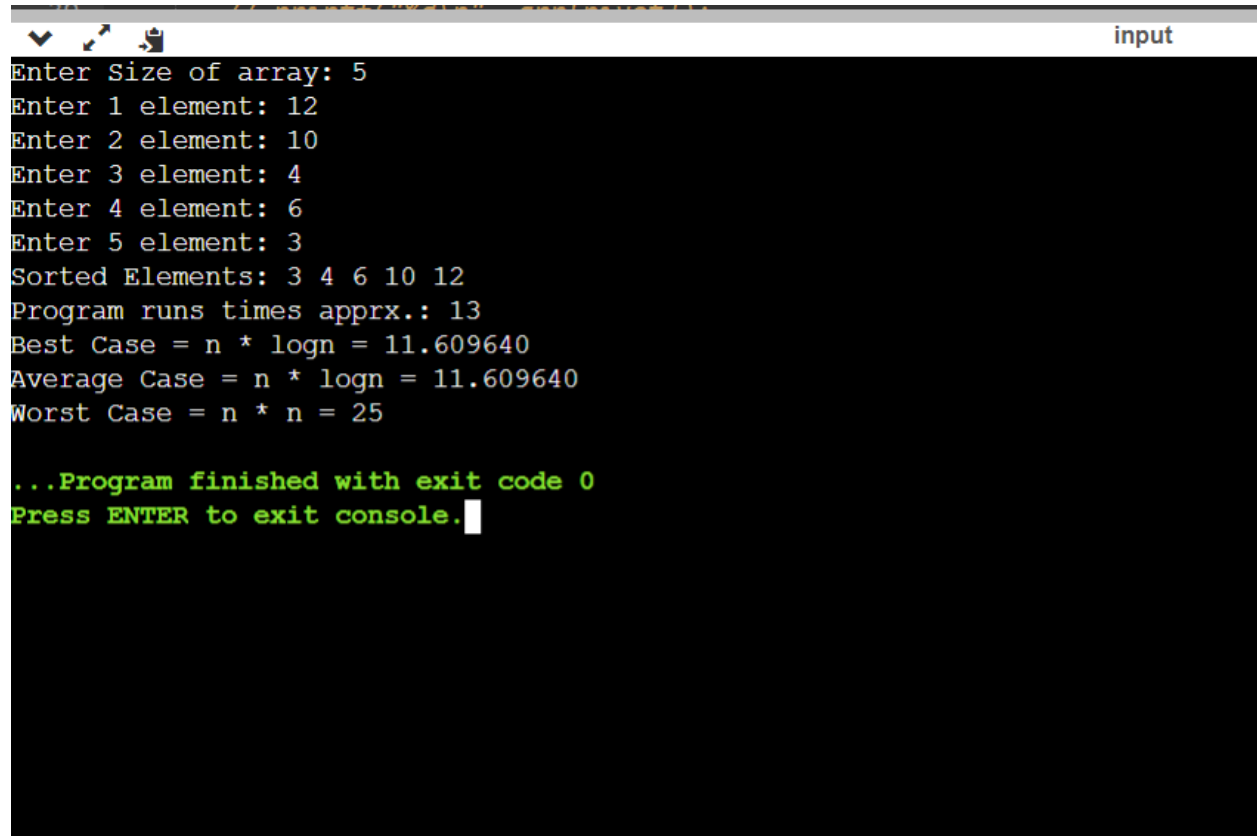
    {
        int pivot = partition(arr, l, x);
        // printf("%d\n", arr[pivot]);
        //get(pivot);
        counter++;
        quicksort(arr, l, pivot - 1);
        quicksort(arr, pivot + 1, x);
    }
}
int get(int p)
{
    return p;
}
int main()
{
    int arr[20], n;
    printf("Enter Size of array: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++)
    {
        printf("Enter %d element: ", i + 1);
        scanf("%d", &arr[i]);
        counter++;
    }
    //printf("Pivot Elements\n");

    quicksort(arr, 0, n - 1);
    printf("Sorted Elements: ");
    for (int k = 0; k < n; k++)
    {
        printf("%d ", arr[k]);
        counter++;
    }
    printf("\nProgram runs times apprx.: %d", counter);
    printf("\nBest Case =  $n * \log n = %2f$ ", n * log2 (n));
    printf("\nAverage Case =  $n * \log n = %2f$ ", n * log2 (n));
    printf("\nWorst Case =  $n * n = %d$ ", n * n);
    return 0;
}

```

}



A screenshot of a terminal window with a dark background and light-colored text. The window title bar at the top shows standard window controls (minimize, maximize, close) and the title "input". The terminal displays the following text:

```
Enter Size of array: 5
Enter 1 element: 12
Enter 2 element: 10
Enter 3 element: 4
Enter 4 element: 6
Enter 5 element: 3
Sorted Elements: 3 4 6 10 12
Program runs times aprx.: 13
Best Case = n * logn = 11.609640
Average Case = n * logn = 11.609640
Worst Case = n * n = 25

...Program finished with exit code 0
Press ENTER to exit console.
```

## Practical-04

### Knapsack Problem

```
#include <stdio.h>

int main()
{
    int obj, W, remain, i, flag = 0;
    printf("Enter total objects and total capacity : ");
    scanf("%d %d", &obj, &W);
    int weight[obj], profit[obj];
    float x[obj];
    for (int i = 0; i < obj; i++)
    {
        printf("Enter weights in ascending order\n");
        printf("for %d object, Enter its weight and profit : ", i+1);
        scanf("%d %d", &weight[i], &profit[i]);
    }
    remain = W;
    i = 0;
    while(remain > 0 && i < obj){

        if (weight[i] <= remain && remain > 0)
        { x[i] = 1;
          remain -= weight[i];
          flag++;
        }
        else if (weight[i] > remain && remain > 0)
```



```

        {
            x[i] = (float) remain / weight[i];
            remain = 0;
            flag++;
        }
i++;
    }
for (int j = flag + 1; j < obj; j++)
{
    x[j] = 0;
}

float profitMax = 0;
for(int a = 0; a < obj; a++){
    printf("%2f  %d\n", x[a], profit[a]);
}

for (int j = 0; j < obj; j++)
{
    profitMax = profitMax + (x[j] * profit[j]) ;
}

printf("Total sumation of pi, xi : %2f ", profitMax);
return 0;
}

```

```
Enter total objects and total capacity : 5
50
Enter weights in ascending order
for 1 object, Enter its weight and profit : 10 3
Enter weights in ascending order
for 2 object, Enter its weight and profit : 12
4
Enter weights in ascending order
for 3 object, Enter its weight and profit : 15 5
Enter weights in ascending order
for 4 object, Enter its weight and profit : 16 6
Enter weights in ascending order
for 5 object, Enter its weight and profit : 25 7
1.000000 3
1.000000 4
1.000000 5
0.812500 6
0.000000 7
Total sumation of pi, xi : 16.875000

...Program finished with exit code 0
Press ENTER to exit console.
```