**Breadth First Search**

#include<stdio.h>

#include<time.h>

#include<math.h>

#include<stdlib.h>

#include<string.h>

int q[100];

int visited[100];

int adj[20][20];

int n;

void enqueue(int v);

int dequeue();

int front=-1;

int rear=-1;

void enqueue(int v)

{

if(front==-1 && rear==-1)

{

front=rear=0;

}

if(rear==n-1)

{

printf("Queue Full\n");

return;

}

q[rear]=v;

rear++;

}

int dequeue()

{

int val;

if(front==-1 || front>rear)

{

return -1;

}

val=q[front];

if(front==rear || front>rear)

{

front=-1;

rear=-1;

}

front++;

return val;

}

void bfs(int v)

{

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

{

if(adj[v][i]==1 && visited[i]==0)

{

enqueue(i);

printf("%d\t",i);

visited[i]=1;

}

}

int val=dequeue();

if(val!=-1)

{

bfs(val);

}

else

{

return;

}

}

int main()

{

clock\_t t;

int flag=1;

int ci=2;

int v;

printf("Enter the Number of the vertex\n");

scanf("%d",&n);

printf("Enter the Entries Of The Adjacent Matrix\n");

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

{

for(int j=0;j<n;j++)

{

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

}

}

printf("Enter the Starting Vertex\n");

scanf("%d",&v);

printf(" BFS TRAVERSAL IS\n");

printf("%d\t",v);

visited[v]=1;

t=clock();

bfs(v);

t=clock()-t;

double time\_taken=((double)t)/CLOCKS\_PER\_SEC;

printf(" \nTime Taken Is%f\n",time\_taken);

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

{

if(visited[i]==0)

{

flag=0;

printf("\n%d\t",i);

visited[i]=1;

printf("\nTRAVERSAL %d\n",ci);

ci++;

bfs(i);

}

}

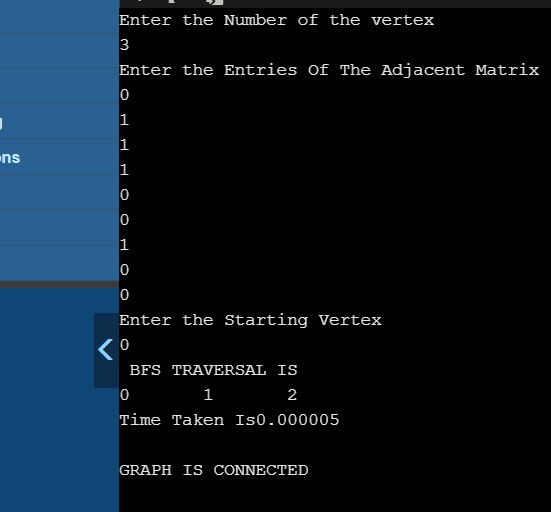
if(flag==1)

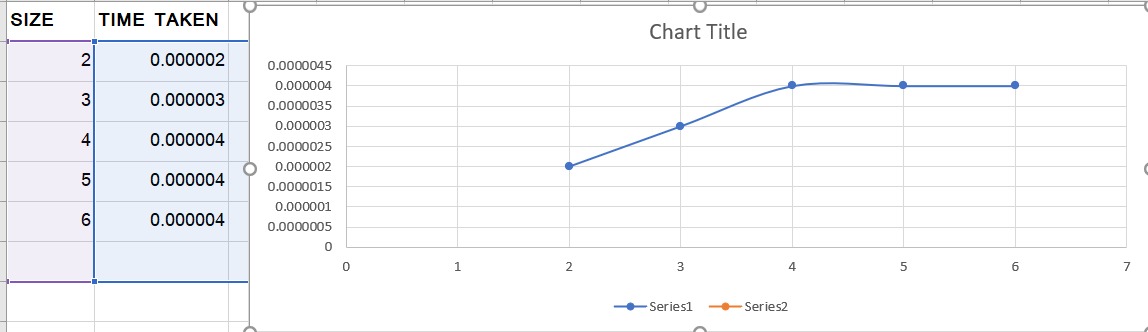
{

printf("\nGRAPH IS CONNECTED\n");

}

}





**Insertion sort**

#include<stdio.h>

#include<time.h>

#include<stdlib.h>

void insertion\_sort(int arr[],int n)

{

int i, key, j;

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

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

int main()

{

clock\_t start, end;

for(;;){

int arr[100000];

int n;

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

scanf("%d",&n);

printf("Enter elements to array:\n");

int r;

srand(time(NULL));

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

{

r = rand();

arr[i] = r;

}

start = clock();

insertion\_sort(arr,n);

end = clock();

printf("Sorted array is:\n");

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

{

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

}

printf("time required is %.8f\n",((double)(end-start)/CLOCKS\_PER\_SEC));

}

}

