

**East West University**

**Report**

**Discrete Mathematics**

**CSE-205**

**Topic –** Undirected graph

**Submitted by-**

Md Ferdous

**Submitted to-**

Yeasir Rayhan

Lecturer

Department of CSE

East West University

**Submission Date:** 04-12-2019

**Big-O of this code**

#include<bits/stdc++.h>

using namespace std;

int grp[7000][7000],t,i,j;

int main()

{

cout<< "Give t:"<<endl;

cin>>t;

double Second;

double NanoSecond;

clock\_t s = clock();

srand (time(NULL));

for(i = 0; i<t; i++) Let “n” be the count for this loop

for(j = i; j<t; j++) Let “m” be the count for this loop

{

grp[i][j] = rand()%2;

grp[j][i] = grp[i][j];

} So the big O for this loop is O(mn).

int c = 0,d=0;

//cout<<"vertex"<< " "<<"Degree"<<endl<<endl;

for(i = 0; i<t; i++) Again, Let “n” be the count for this loop

{

for(j = 0; j<t; j++) Let “m” be the count for this loop

{

if(grp[i][j] == 1)

{

if(i == j){c+=2;

d+=2;}

else {c++;

d++;}

}

} we can see in this loop n = m;

So the big O for this loop is O(n\*n) = O(n2)

//cout<<i<< " "<<d<<endl;

d=0;

//cout<<endl;

}

cout<<"Total degree:"<<c<<endl;

c = 0;

for(i = 0;i<t;i++) Let “n” be the count for this loop

{

for(j = 0;j<=i;j++) Let “m” be the count for this loop

{

if(grp[i][j] == 1)

c++;

}

} So the big O for this loop is O(mn).

cout<< "The edege is:"<<c<<endl;

clock\_t e = clock();

Second = (double) (e - s)/ CLOCKS\_PER\_SEC;

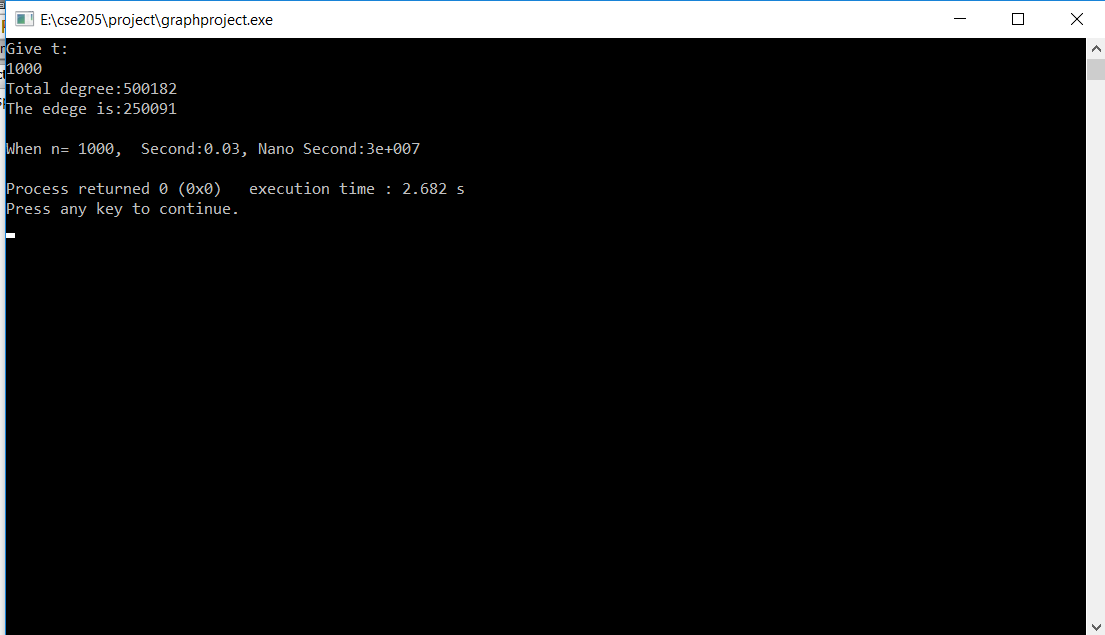
NanoSecond = (double) Second \* 1000000000;

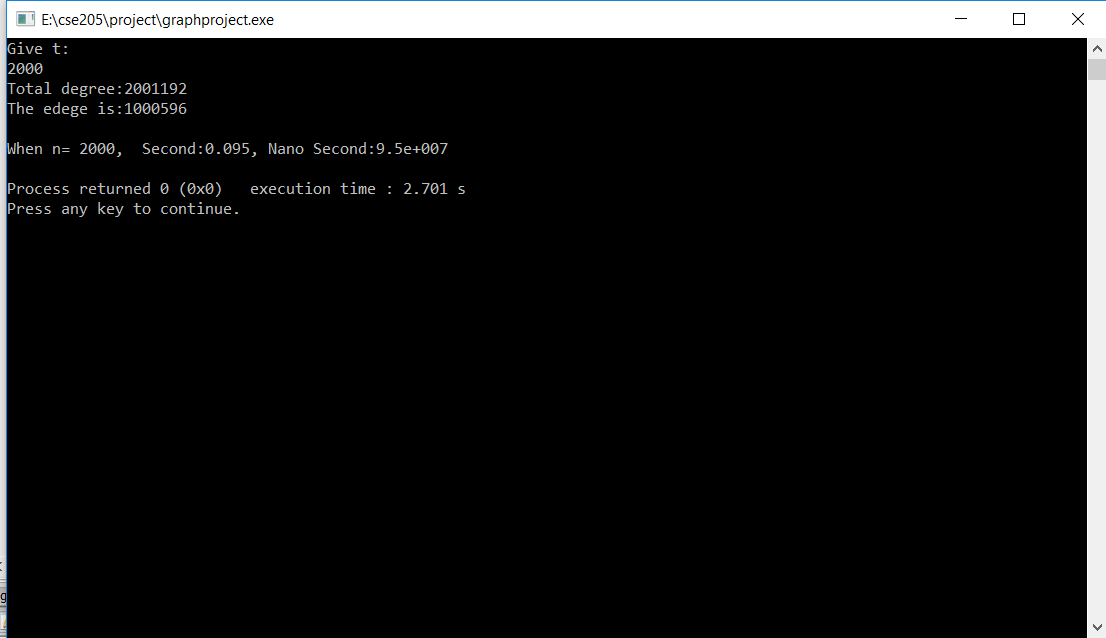
cout<<endl<<"When n= "<<t<<", Second:"<< (double) Second <<", Nano Second:"<< (double) NanoSecond<<endl;

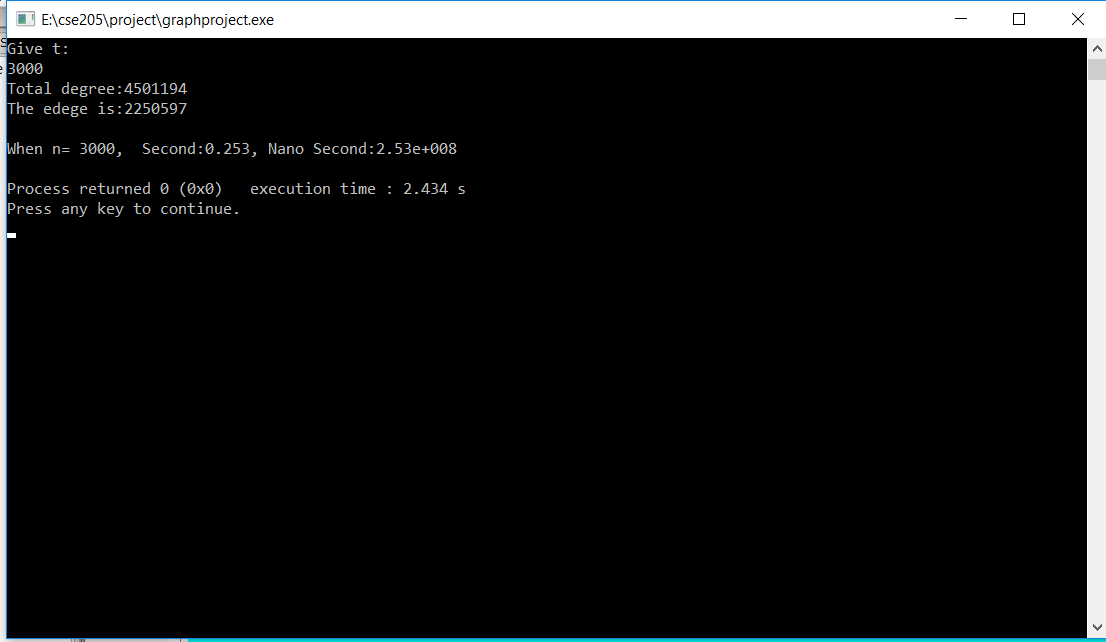
}

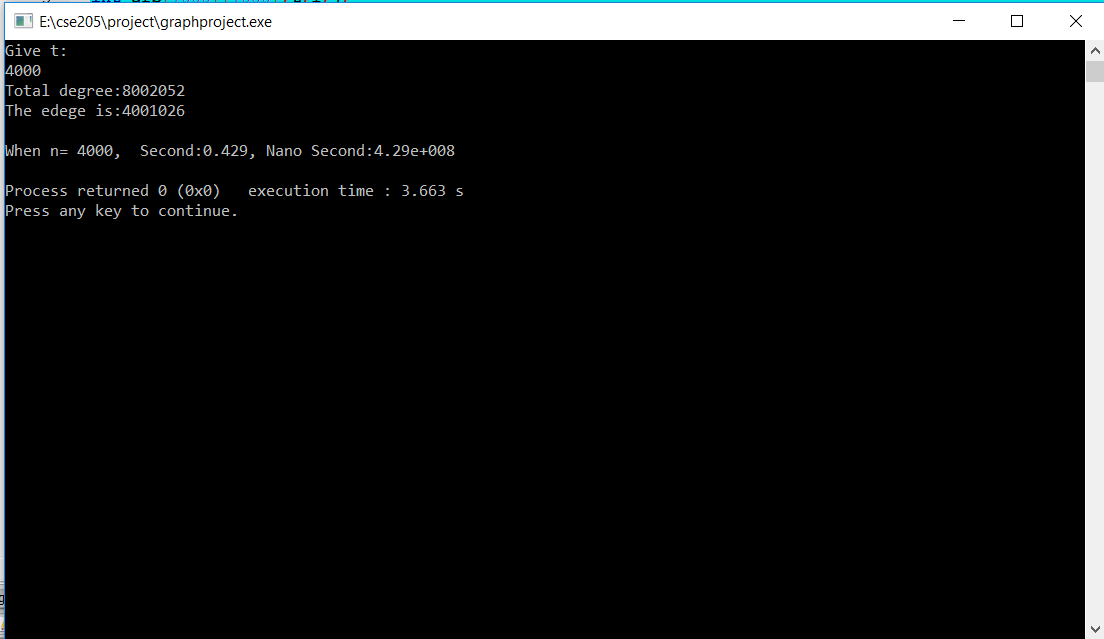
We know, when the input is larger than the smaller statement do not count. In first nested loop the loop is the m is half of n and the last loops m is also half of the n. But the middle loop where we count the degree, there the m is same as n. and we know big O(n2) is most high on time complexity then other time complexity other in this code. So our code complexity is Big O(n2).

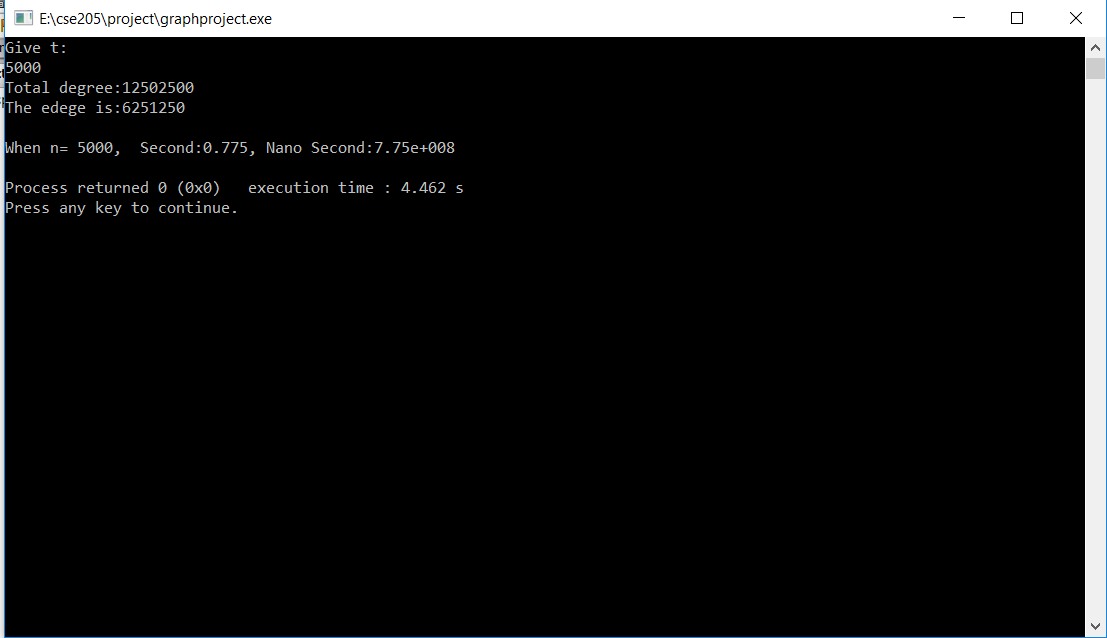
**The time complexity**











**The graph of this codes time complexity**

So we can see when the value is 1000 to 2000 the code takes time less than 1.00E+8. When value grows to 5000 the time goes up to 7.00E+8. In theoretically we saw the code time complexity is Big O(n2) . Here we can see from graph the code is near to O(n2). That’s mean the line goes high with the increasing of value.