Ganupa Vineeth Reddy (002194126)

Program Structures & Algorithms Fall 2021

Assignment No. 3 (WQUPC)

! **Task (List down the tasks performed in the Assignment)**

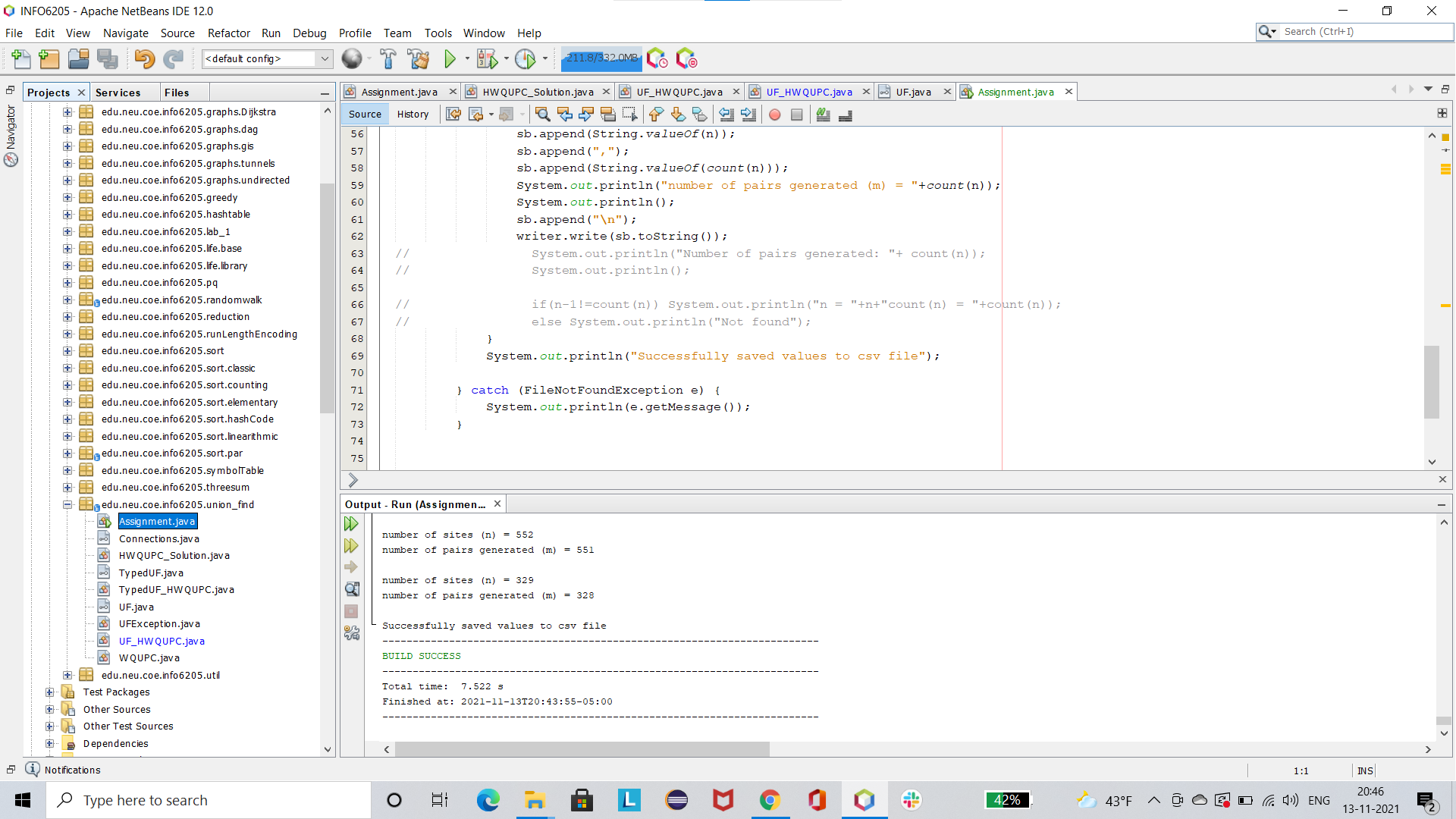
1. To start off, First I Implemented height-weighted Quick Union with Path Compression
2. And then I Developed Union Find using the above UF\_HWQUPC by taking random n (number of sites) values. And returned the value m (number of pairs generated).
3. Derived relaPonship between n (number of sites) and m (number of pairs generated).

! **RelaIonship Conclusion: (For ex : z = a \* b)**

m = number of pairs generated, n = number of sites RelaPonship: m = n-1

! **Evidence to support the conclusion:**

 **Output (Snapshot of Code output in the terminal)**

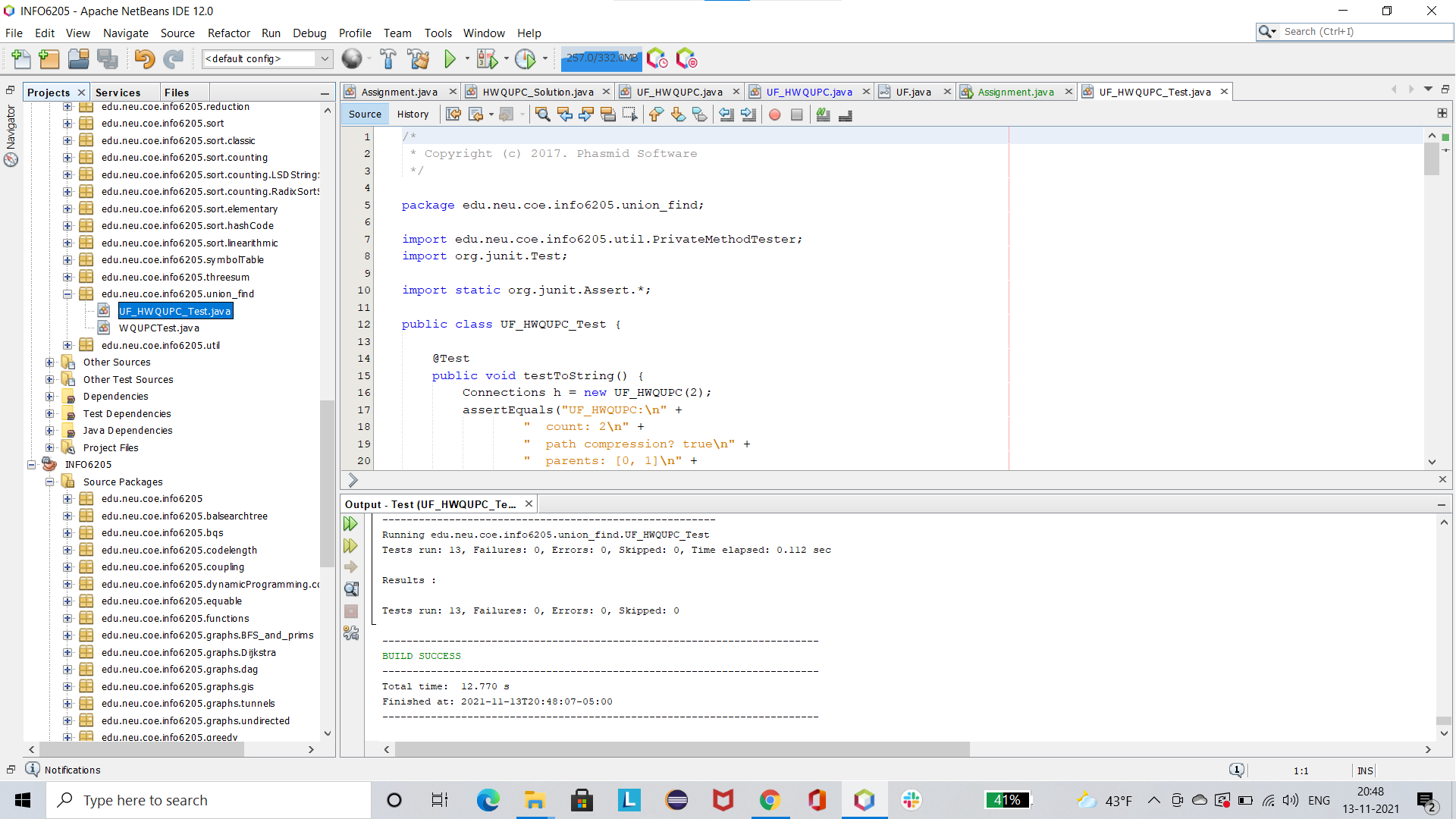
****

 **Graphical RepresentaIon(ObservaIons from experiments should be tabulated and analyzed by ploUng graphs(usually in excel) to arrive on the relaIonship conclusion)**

By generaPng random number of sites for 1000 Pmes, we can observe that every Pme m=n-1 because we generate pairs only when those two sites are not connected. Thus we generate only those pairs, whose connecPon has not been established yet.



! **Unit tests result:(Snapshot of successful unit test run)**

****

**Assignment.java**

package edu.neu.coe.info6205.union\_find;

import java.io.File;

import java.io.FileNotFoundException; import java.io.PrintWriter;

import java.util.Random; import java.util.Scanner;

import java.util.concurrent.ThreadLocalRandom;

public class Assignment {

static boolean isCompletelyConnected(int n,UF\_HWQUPC uf){ //this function checks whether all the nodes are completely connected

int temp=uf.find(0); int tempCount=n; for(int i=1;i<n;i++){

if(uf.find(i)!=temp){ break;

}else{

tempCount--;

}

}

return tempCount==1;

}

static int count(int n){ int count=0;

UF\_HWQUPC uf=new UF\_HWQUPC(n);

while(!*isCompletelyConnected*(uf.size(),uf)){ int p= (int) (Math.*random*()\*n);

int q= (int) (Math.*random*()\*n);

if(!uf.connected(p,q)){

//

System.out.println("p="+p+" q="+q);

uf.connect(p,q); count++;

}

}

// System.out.println("Root is: "+uf.find(0)); return count;

}

public static void main(String[] args) {

//

Scanner sc=new Scanner(System.in);

//

System.out.println("Enter number of sites: ");

// int n=sc.nextInt(); int n=0;

try(PrintWriter writer = new PrintWriter(new File("test.csv"))){

StringBuilder sbh = new StringBuilder(); sbh.append("i,n,m\n"); writer.write(sbh.toString());

for(int i=0;i<1000;i++){

n = ((int)(Math.*random*()\*1000))+1;

System.*out*.println("number of sites (n) =

"+n);

StringBuilder sb = new StringBuilder(); sb.append(String.*valueOf*(i)); sb.append(","); sb.append(String.*valueOf*(n)); sb.append(","); sb.append(String.*valueOf*(*count*(n)));

System.*out*.println("number of pairs generated (m) = "+*count*(n));

System.*out*.println(); sb.append("\n"); writer.write(sb.toString());

// System.out.println("Number of pairs generated: "+ count(n));

// System.out.println();

// if(n-1!=count(n)) System.out.println("n =

"+n+"count(n) = "+count(n));

//

else System.out.println("Not found");

}

System.*out*.println("Successfully saved values to csv file");

}

System.*out*.println(e.getMessage());

} catch (FileNotFoundException e) {

}

}