Program Structures and Algorithms
Spring 2023(SEC –8)

NAME: Mehul Natu
NUID: 002743870

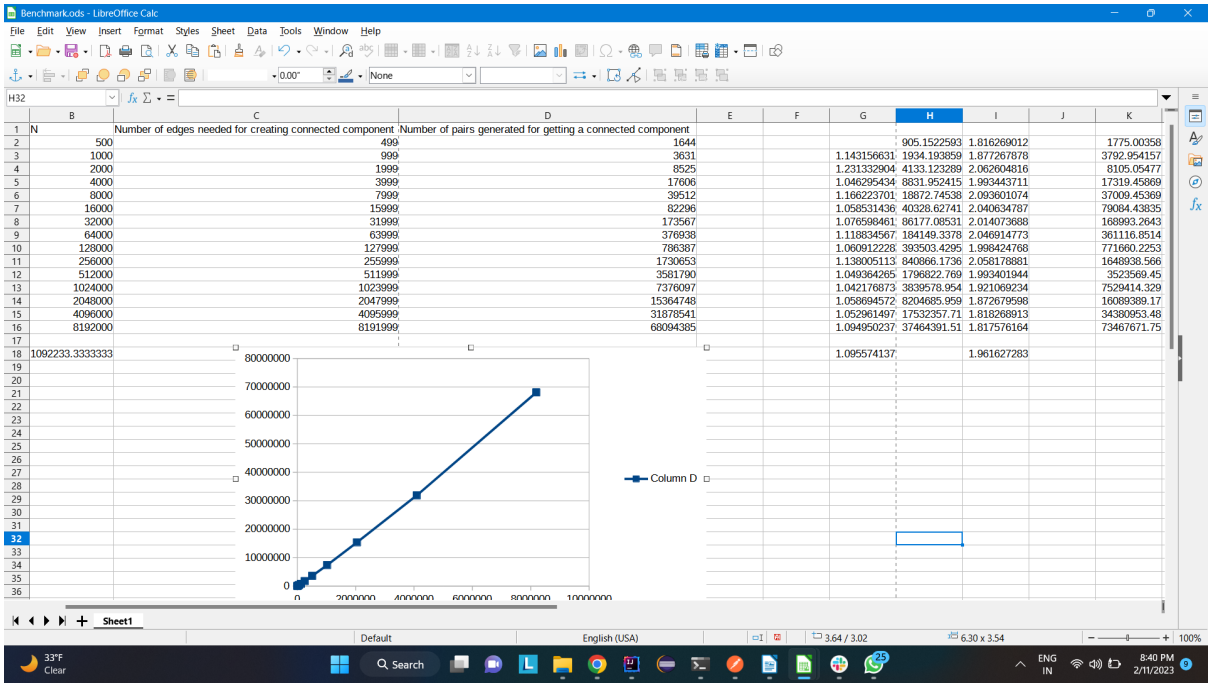
Task:

- Implement height-weighted Quick Union with Path Compression
- Using your implementation of UF_HWQUPC, develop a UF ("union-find") client that takes an integer value n from the command line to determine the number of "sites." Then generates random pairs of integers between 0 and n-1, calling connected() to determine if they are connected and union() if not. Loop until all sites are connected then print the number of connections generated.
- Determine the relationship between the number of objects (*n*) and the number of pairs (*m*) generated to accomplish this (i.e. to reduce the number of components from *n* to 1). Justify your conclusion in terms of your observations and what you think might be going on.

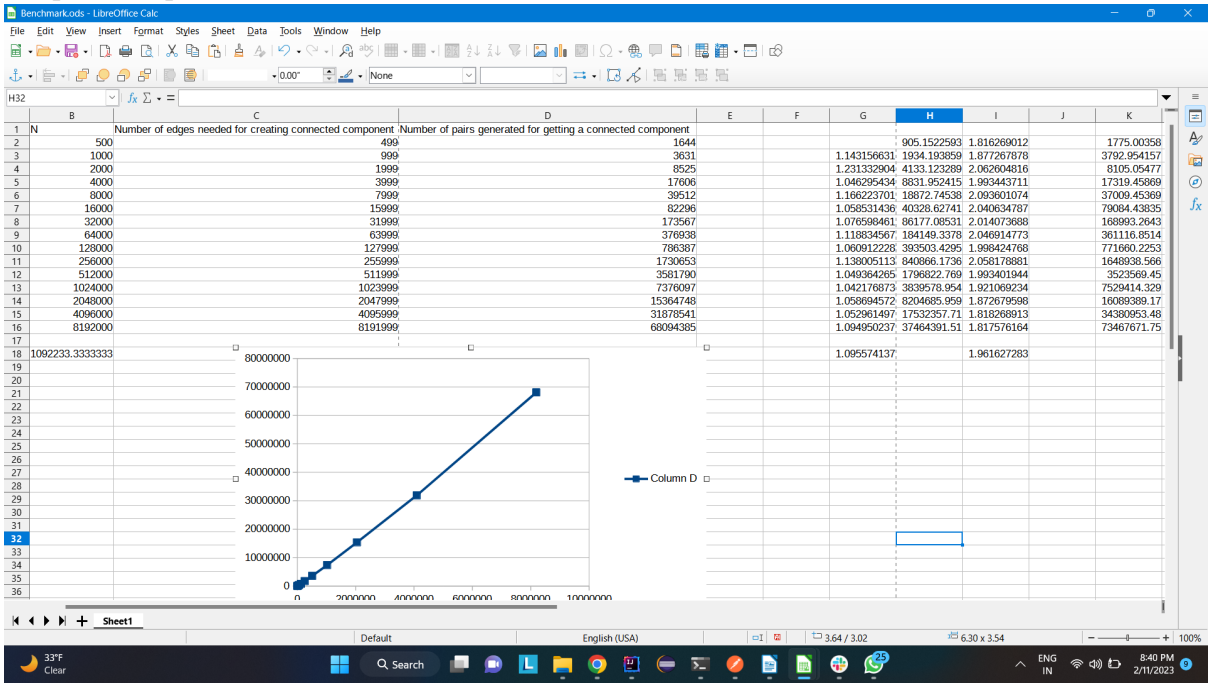
Relationship Conclusion: It should be kind of linear or little more than just linear. M = NC. Where C is some constant. First lets start with how many edges do we need to create a single component. It should be N – 1. This is the minimum number of edges or Unions we need to call, to get all sites connected. Proof - Suppose there are 3 nodes to connect all three of them the minimum we will require is 2. And similarly for 4 nodes the minimum is 3.
When we start generating two random numbers between 0 and n - 1 to union. then when there exist no edges or in the starting it will have a higher probability of generating edges in the starting the probability would be: $(n^2- n)/n^2$. This is because there could be n^2 possible type of pair which we could generate for example in case of 3 nodes we can randomly generate 9 edges like – (0, 0), (0, 1), (0, 2), (1, 1), (1, 0), (1, 2), (2, 2), (2, 0), (2, 1), and "-n" because we cannot create self edges and there could be n self edges. And probability is number of edges which we can create to get the edges which we need for connecting nodes – $(n^2- n)$ divided by total number of probable outcomes. But as the graphs gets connected the probability start decreasing – because now the number of edges increases and and edges which we want to connect decreases and probability becomes - $(n^2 – n – 2x)/n^2$, here x is the number of edges which we have already created. And 2 is because if we have already created an edge between 0 and 1 then if our random number generator generated (1, 0) or (0, 1) it will not create any edge as it is already present. So the probability start decreasing and since in the starting the probability is fairly high as compared to at the end. Because of this interplay it could be that the M is linearly proportional to N. Since more the Number of N the more edges we need to create. Thus increasing kind of linearly.
G

Evidence to support that conclusion:
In the below screenshot we can see by the slope of the log of M and log N values in the column G and averaging it out so we get slope – 1.0955. thus the relationship turn out to be like $M = N^{1.09}C$. And now C is calculated by $C = M/N^{1.09}$. Here we are inserting the actual values of M which we got from the experiment which is present in the column D, $N^{1.09}$ here we are inserting the original values of N present in the column B. The result of this ratio is present in I. And averaging it out we get C as 1.9. This the relationship turn out to ab around $M = 1.9N^{1.09}$. And plotting the graph between column B(experimental values of N) and column D(experimental values of M) we nearly get a straight line. Which means it should be

kind of linear. To get these values the code is written in the file named UF_BenchMark. Also this xlxs file is named Benchmark.xlxs.

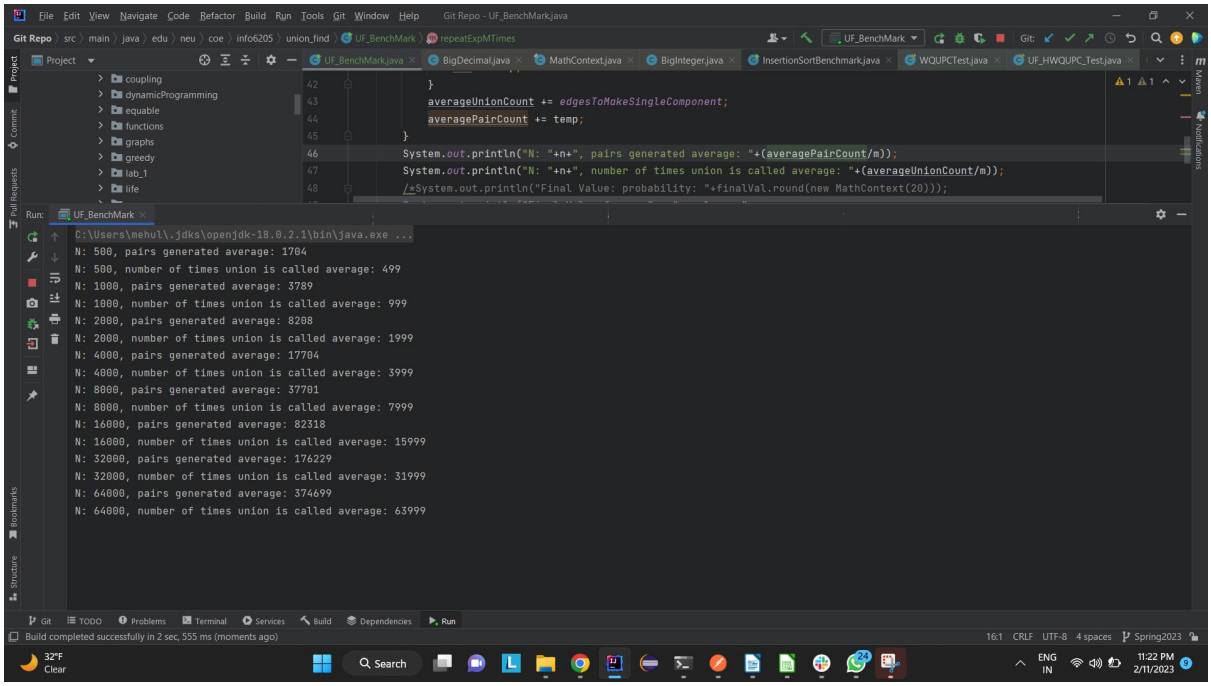


Graphical Representation:

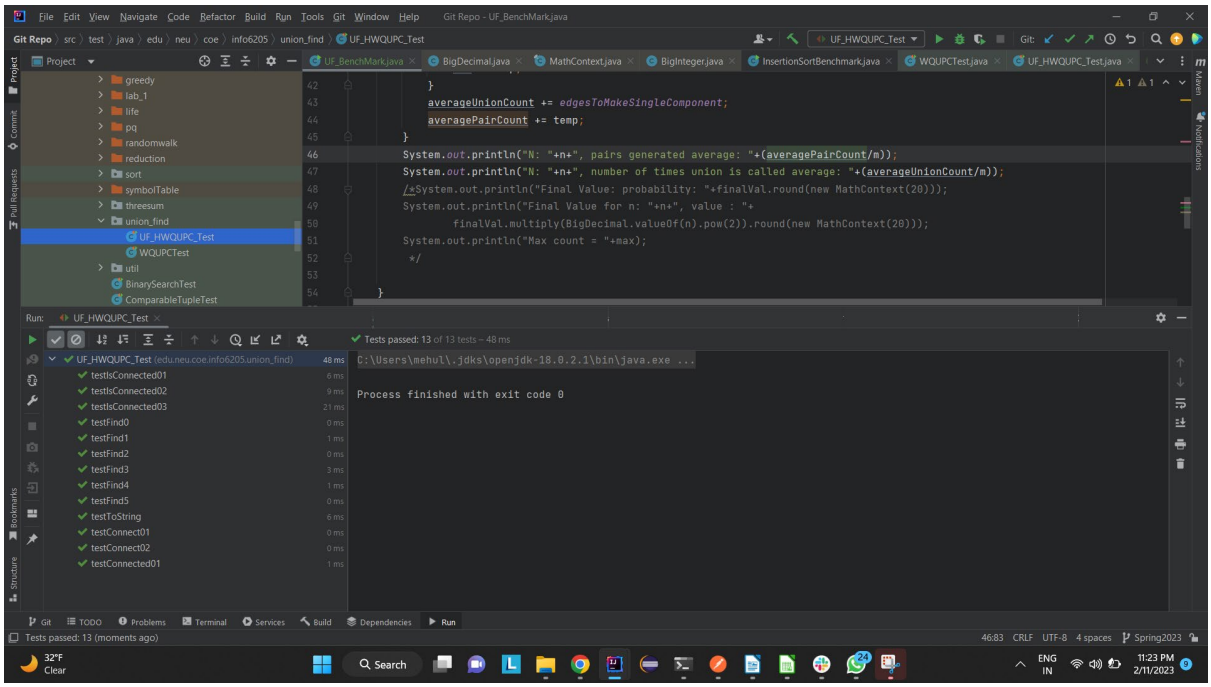


Unit Test Screenshots:

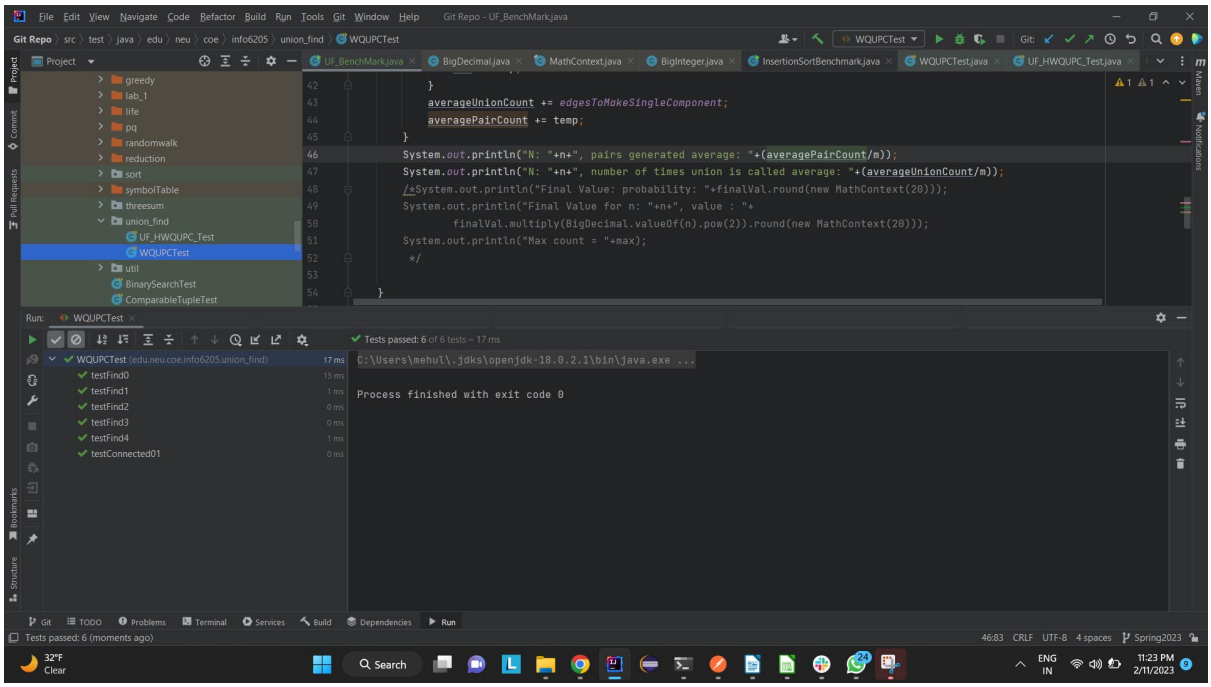
First Screenshot of the UF_BenchMark.java running.



Screenshot for UF_HWQUPC_Test running



ScreenShot of WQUPCTest running


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Git Repo - UF_BenchMark.java
Git Repo src test java edu neu coe info6205 union_find WQUPCTest
WQUPCTest
Project
greedy
lab_1
life
pq
randomwalk
reduction
sort
symbolTable
threesum
union_find
UF_HWQUPC_Test
WQUPCTest
util
BinarySearchTest
ComparableTupleTest
UF_BenchMark.java
BigDecimal.java
MathContext.java
BigInteger.java
InsertionSortBenchmark.java
WQUPCTest.java
UF_HWQUPC_Test.java
averageUnionCount += edgesToMakeSingleComponent;
averagePairCount += temp;
System.out.println("N: "+n+", pairs generated average: "+(averagePairCount/m));
System.out.println("N: "+n+", number of times union is called average: "+(averageUnionCount/m));
/*System.out.println("Final Value: probability: "+finalVal.round(new MathContext(20)));
System.out.println("Final Value for n: "+n+", value : "+
finalVal.multiply(BigDecimal.valueOf(n).pow(2)).round(new MathContext(20)));
System.out.println("Max count = "+max);
*/
}
Run: WQUPCTest
Tests passed: 6 of 6 tests – 17 ms
WQUPCTest (edu.neu.coe.info6205.union_find) 17 ms
testFind0 15 ms
testFind1 1 ms
testFind2 0 ms
testFind3 0 ms
testFind4 1 ms
testConnected01 0 ms
C:\Users\mehul\.jdks\openjdk-18.0.2.1\bin\java.exe ...
Process finished with exit code 0
Git TODO Problems Terminal Services Build Dependencies Run
Tests passed: 6 (moments ago)
46:83 CRLF UTF-8 4 spaces Spring2023
32°F Clear
Search
ENG IN
11:23 PM 2/11/2023