Erarica Mehra
NUID: 002113683

Assignment 3:
Height-weighted Quick Union with Path Compression

Tasks

- Implemented Height-weight Quick Union with Path Compression.
- Created a new class called UF_HWQUPC_Client that takes input, (number of sites) from the command line, connects the pairs if they are not connected and returns the number of connections/pairs.
- Determined the relationship between number of sites (N) and number of connections (M)

Relationship/Conclusion:

According to the data shown in the screenshot and spreadsheet, it can be clearly concluded that

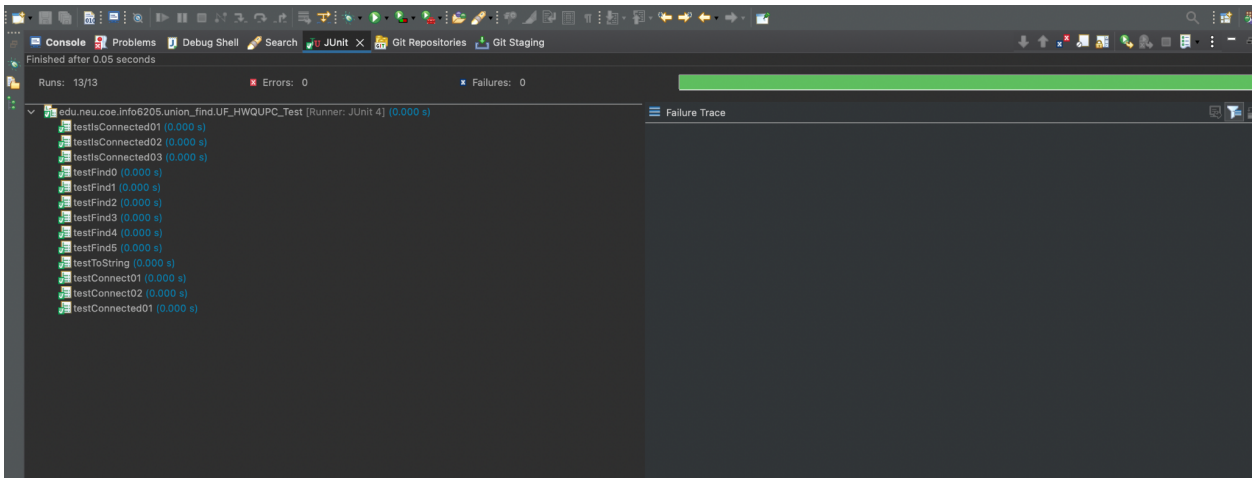
M = NLogN/2,
where M is the number of connections/pairs and N is the number of sites.

Worst case time complexity for weighted quick union is O(logN).
This relationship holds true for increasing values of the number of sites.
In case of weighted Quick Union with Path Compression, initial operations might be expensive because the tree would be of log height, but then it goes cheaper as the tree becomes flatter and flatter, with every element becoming the direct descendent of the root.
Operations in case of weighted Quick Union with Path Compression are very close to amortized constant time. Hence this algorithm is optimal.
M operations on N nodes is O (N + M Log N). This is practical for any realistic input.

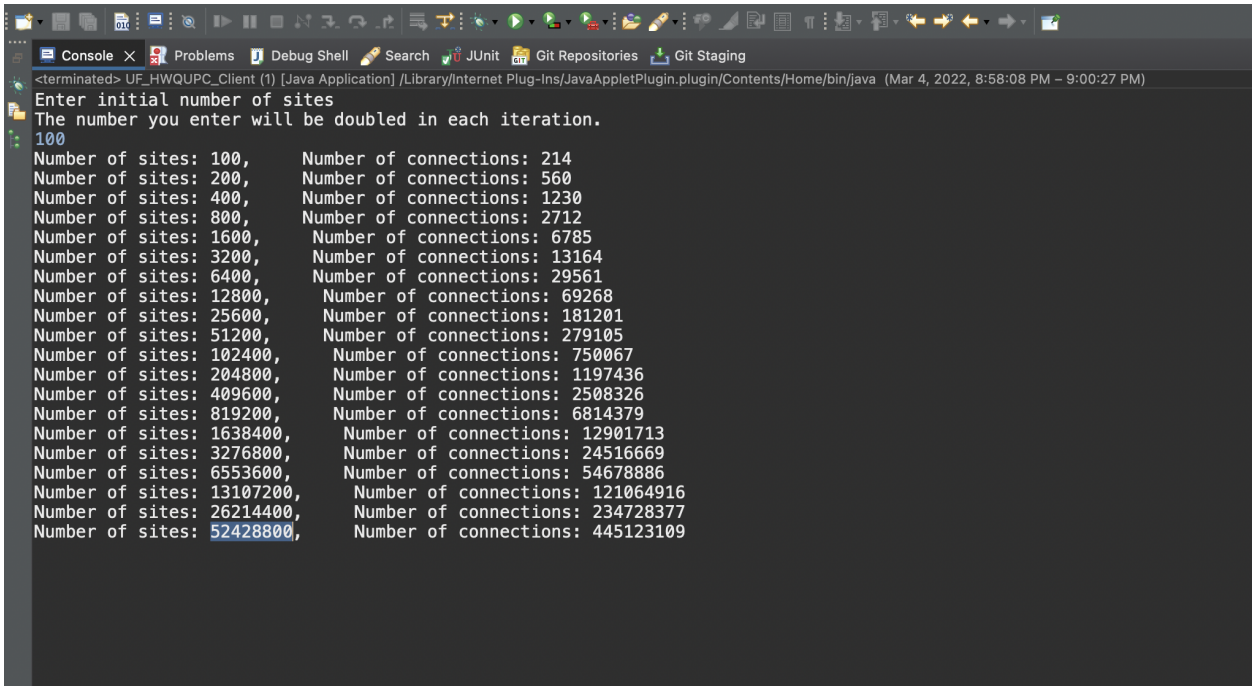
Evidence :

1. **Screenshot of the test cases:**

   The following is the snapshot of the unit test cases



2. **Console Output :**
   On executing the main method in the UF_HWQUPC_Client class, the following output is retrieved. This output is further analyzed and plotted in the graph to reach the conclusion derived in the above section.



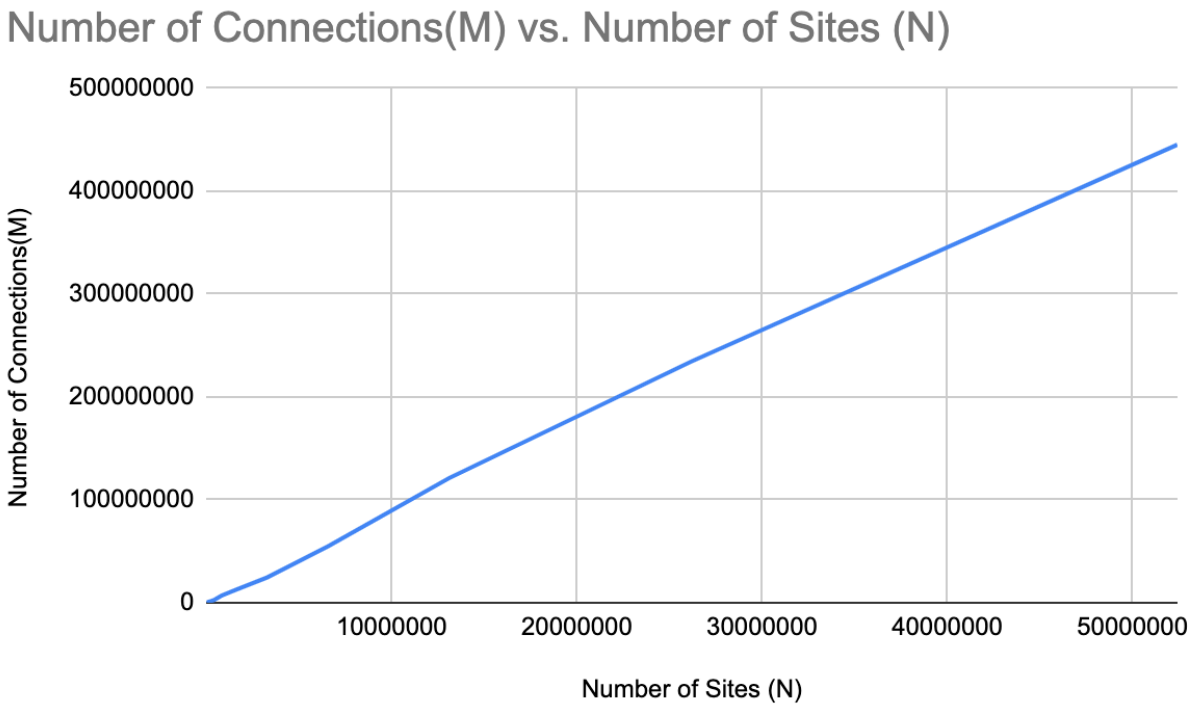
3. **Spreadsheet Data:**
   Here I have used sufficiently large different values of N

Number of Sites (N)	Number of Connections(M)
100	214
200	560
400	1230
800	2712
1600	6785
3200	13164
6400	29561
12800	69268
25600	181201
51200	279105
102400	750067
204800	1197436
409600	2508326
819200	6814379
1638400	12901713

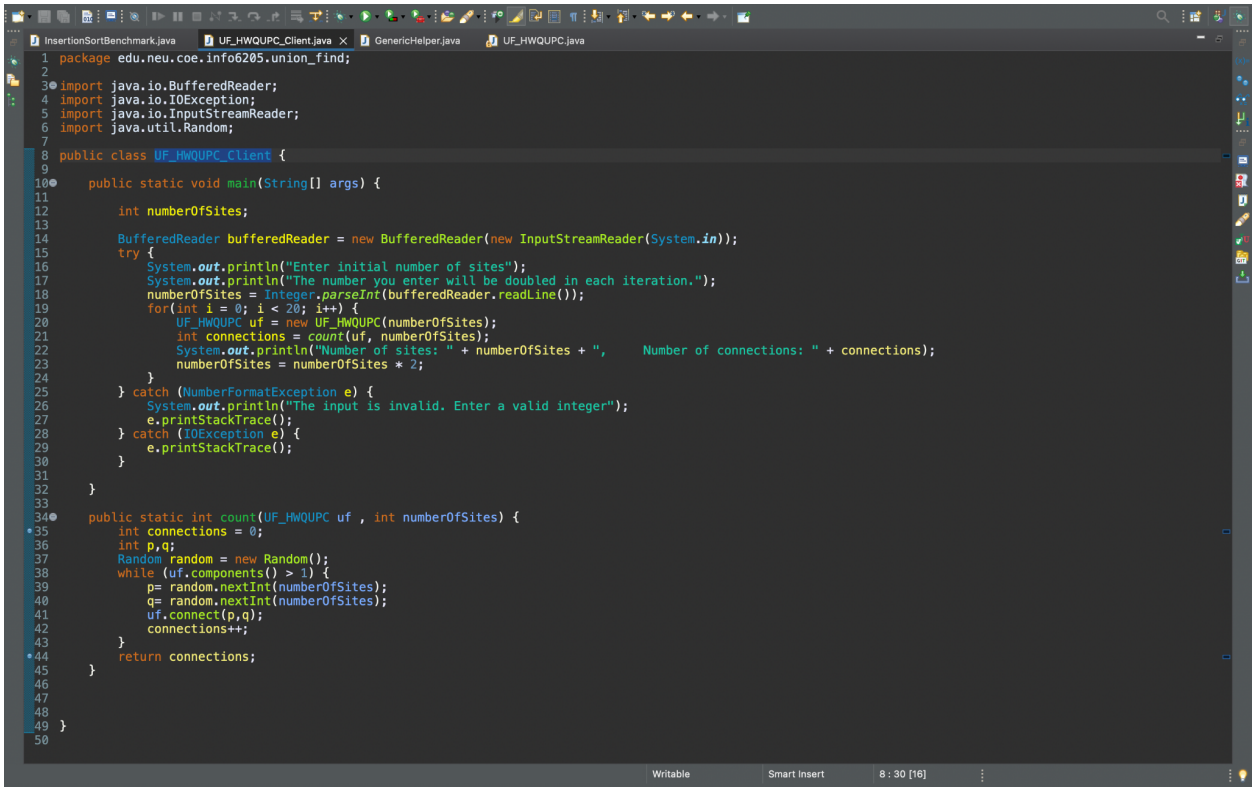
3276800	24516669
6553600	54678886
13107200	121064916
26214400	234728377
52428800	445123109

4. **Graphical Representation:**

   Observations from the experiments are analyzed and plotted on a graph. On close observations it can be clearly derived that the following graph is similar to M = NLogN/2.



Code Snippet:




```java
package edu.neu.coe.info6205.union_find;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Random;

public class UF_HWQUPC_Client {

    public static void main(String[] args) {

        int numberOfSites;

        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        try {
            System.out.println("Enter initial number of sites");
            System.out.println("The number you enter will be doubled in each iteration.");
            numberOfSites = Integer.parseInt(bufferedReader.readLine());
            for(int i = 0; i < 20; i++) {
                UF_HWQUPC uf = new UF_HWQUPC(numberOfSites);
                int connections = count(uf, numberOfSites);
                System.out.println("Number of sites: " + numberOfSites + ",     Number of connections: " + connections);
                numberOfSites = numberOfSites * 2;
            }
        } catch (NumberFormatException e) {
            System.out.println("The input is invalid. Enter a valid integer");
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    public static int count(UF_HWQUPC uf , int numberOfSites) {
        int connections = 0;
        int p,q;
        Random random = new Random();
        while (uf.components() > 1) {
            p= random.nextInt(numberOfSites);
            q= random.nextInt(numberOfSites);
            uf.connect(p,q);
            connections++;
        }
        return connections;
    }



}

```