Assignment 4: Union Find  
  
Name: Yash Pravin Pawar  
Nu Id: 002747371  
Email: [pawar.ya@northeastern.edu](mailto:pawar.ya@northeastern.edu)Github: github.com/NeuYash/PSAAssignments  
  
Part 1:   
Here we have written code for doPathCompression, mergeCompression and find functions in UF\_HWQUPC.java (enclosed in zip file).

public int find(int p) {

validate(p);

int root = p;

// **FIXME**

while (p != parent[p])

p = parent[p];

if (this.pathCompression) {

doPathCompression(root);

}

// END

return p;

}

private void mergeComponents(int i, int j) {

// **FIXME** make shorter root point to taller one

if (i == j)

return;

if (height[i] < height[j]) {

parent[i] = j;

height[j] = Math.*max*(height[i], height[j] + 1);

} else {

parent[j] = i;

height[i] = Math.*max*(height[i], height[j] + 1);

}

// END

}

private void doPathCompression(int i) {

// **FIXME** update parent to value of grandparent

while (i != parent[i]) {

parent[i] = parent[parent[i]];

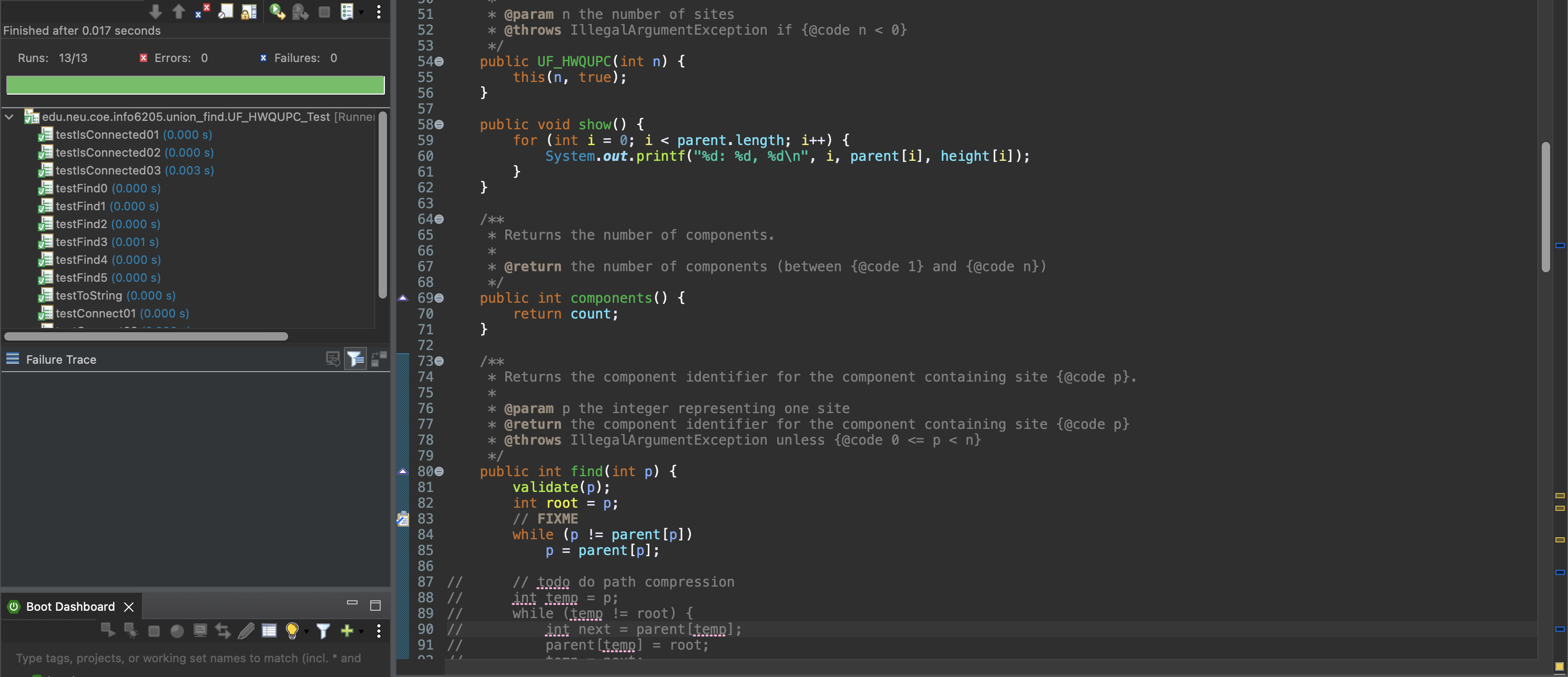
i = parent[i];

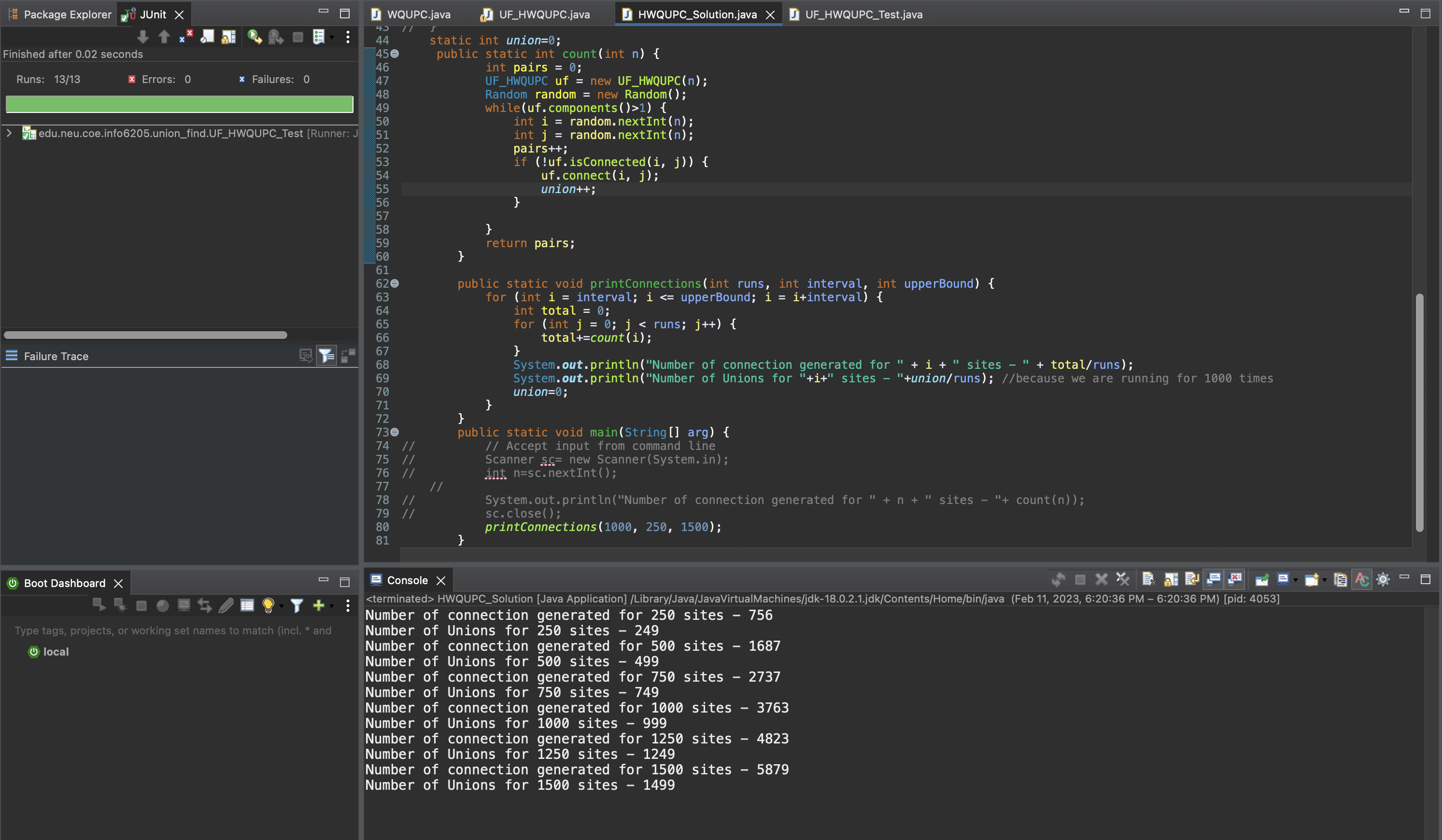
}

// END

}

We have tested this code using the provided test cases and all have passed.



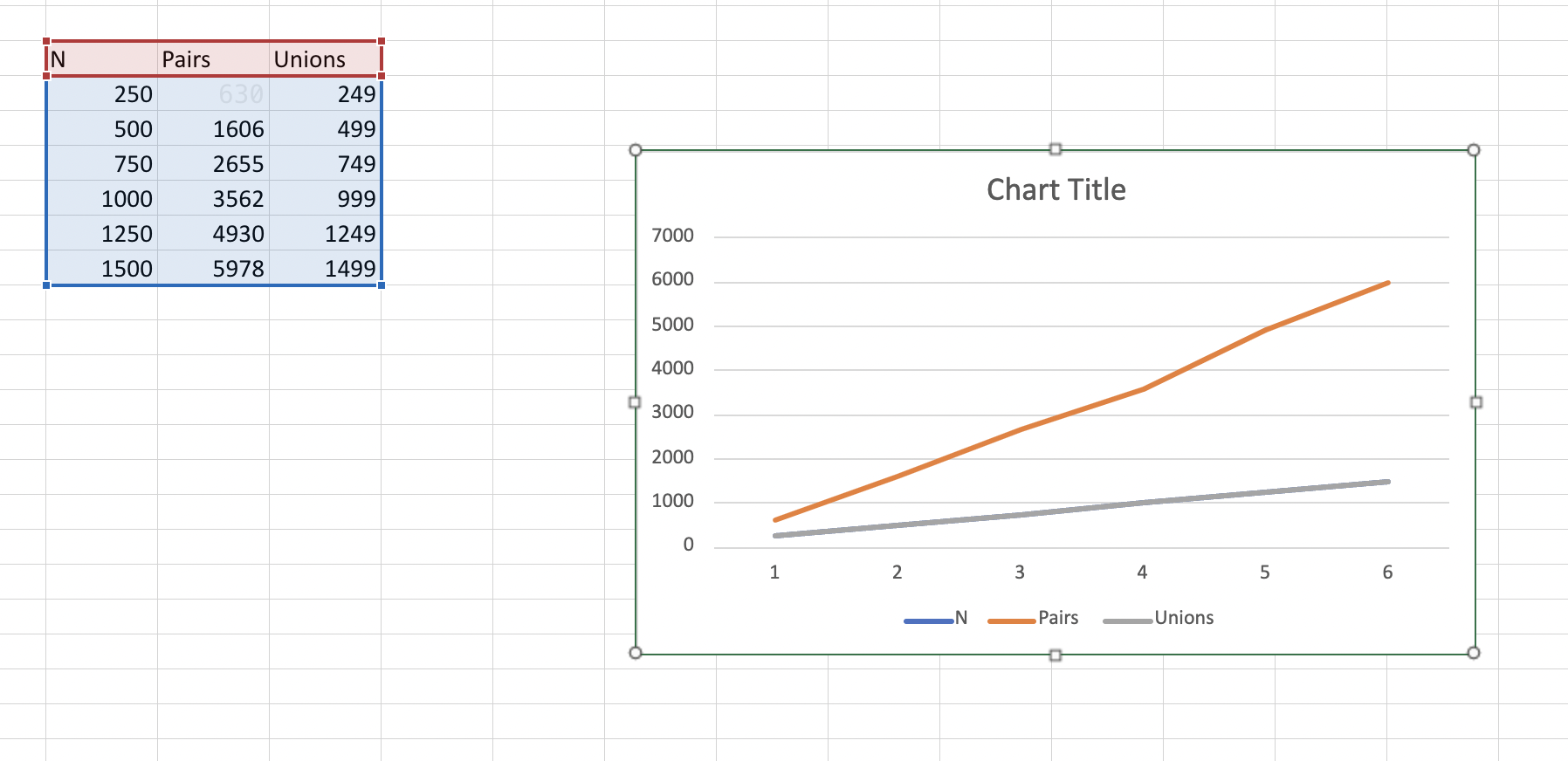
Part 2:   
Here We tried to implement a class having Main function which calls the union function for a given set of array if the array is not already connected. We are selecting 2 random numbers and checking if they are connected, if not we are connecting them using Union function. Number of random pairs generated can be more than number of unions. We have found out that, to connect a linear array, we need N-1 unions to connect a given array because we are not connecting any element while initiating the array.   
  


Function printConnection takes 3 arguments :   
1] runs: number of times our count function will run. We take this parameter to average out the value of total number of pairs generated.   
2] Interval : start value and inteval between two N values.  
3] upperBound: A threshold where a given N value should stop.  
  
This function calls Count function that creates an array of multiple N values and tries to find if two random elements in that array and check if they are connected.

**Number of Unions for N sized array = N-1**

Part 3:

Find a relationship between Number of Unions and Number of Pairs generated to get an N sized array connected.   
  
Here we tried to average the number of pairs generated value by taking an average of 1000 runs for any given N value and we have plotted a graph for the same.



By these values, using Math.log() function. We get a relation that for N-1 Unions we have to generate N/2\* logN pairs to get the array connected.   
  
**N-1 (Unions) α**

**Time complexity of Union Find is O(log n) without Union by Rank and PathCompression.**   
**But time complexity is enhanced to O(kn) where k is constant using PathCompression.**  
**It takes no Extra Time to update Each node to point directly to the root, thereby Compressing the path.**