Yiqing Huang (001525629)

**Program Structures & Algorithms**

**Fall 2021**

**Assignment No. 3**

* **Task (List down the tasks performed in the Assignment)**
* **Step 1:**
  + **Implement height-weighted Quick Union with Path Compression;**
  + **Check that the unit tests all work;**
* **Step 2:**
  + **Develop a UF (“union-find”) client that takes an integer value from the command line as the number of “sites”;**
  + **Repeatedly generate a pair of integers between to and connect the two sites if they are not connected until there is only one component;**
  + **Encapsulate the client and package the algorithm above as a static method *count()*;**
  + **Create a main function that takes from the command line;**
* **Step 3:**
  + **Use sufficient (and sufficiently large) different values of to determine the relationship between the number of objects/sites and the number of pairs/connections**
  + **Justify the conclusion in terms of the observations and give a reasonable explanation.**
* **Step 1**

***\* src/main/java/edu/neu/coe/info6205/union\_find/UF\_HWQUPC.java***

* + **Unit tests result: All PASS.**
  + ***UF\_HWQUPC***
  + ***Graphical user interface, text

    Description automatically generated***
* **Step 2**

***+ src/main/java/edu/neu/coe/info6205/union\_find/UFClient.java***

***+ src/main/java/edu/neu/coe/info6205/union\_find/Pair.java***

* ***When there is one and only one input argument from the command line, call count() once and print the result.***
* ***Text

  Description automatically generated***
* ***Here is the result when :***
* ***Graphical user interface, text, website

  Description automatically generated***
* **Step 3**

***\* src/main/java/edu/neu/coe/info6205/union\_find/UFClient.java***

* **Relationship Conclusion:**
* **Evidence to support the conclusion:** 
  + **Design a *multiCount(int startN, int times)* function to call *count(int n)* function for multiple times to satisfy the requirement of using sufficient (and sufficiently large) different values of .**
  + **Apply the Doubling Method to the *multiCount* function i.e., .**
  + **Modify the main function of *UFClient*. When there are 2 input arguments from the command line, parse the first one as the initial and the second one as the number of times to repeat.**
  + **Text

    Description automatically generated**
  + **Let and and here is the result:**
  + **A picture containing text

    Description automatically generatedChart, line chart

    Description automatically generated**
  + **From the result and its plot, it is obvious that .**
  + **Let’s prove the observation.**
    - **Initially, there are sites, components, and connections.**
    - **Since we will do a union operation when and only when the pair of sites are not connected, the remaining components will decrease 1 every time we union.**
    - **Also, the number of connections increases by when and only when we do a union operation.**
    - **Thus, let’s assume that we have done union operation for times where . The number of connections is . The remaining components is .**
    - **Let’s assume there are in total times of union operation. The total number of connections is . Because the loop in the *count* function will break only if there is only remaining component, we obtain an equation .**
    - **Thus,**
    - **Q.E.D**