Program Structures & Algorithms

Spring 2022

Assignment No. 3

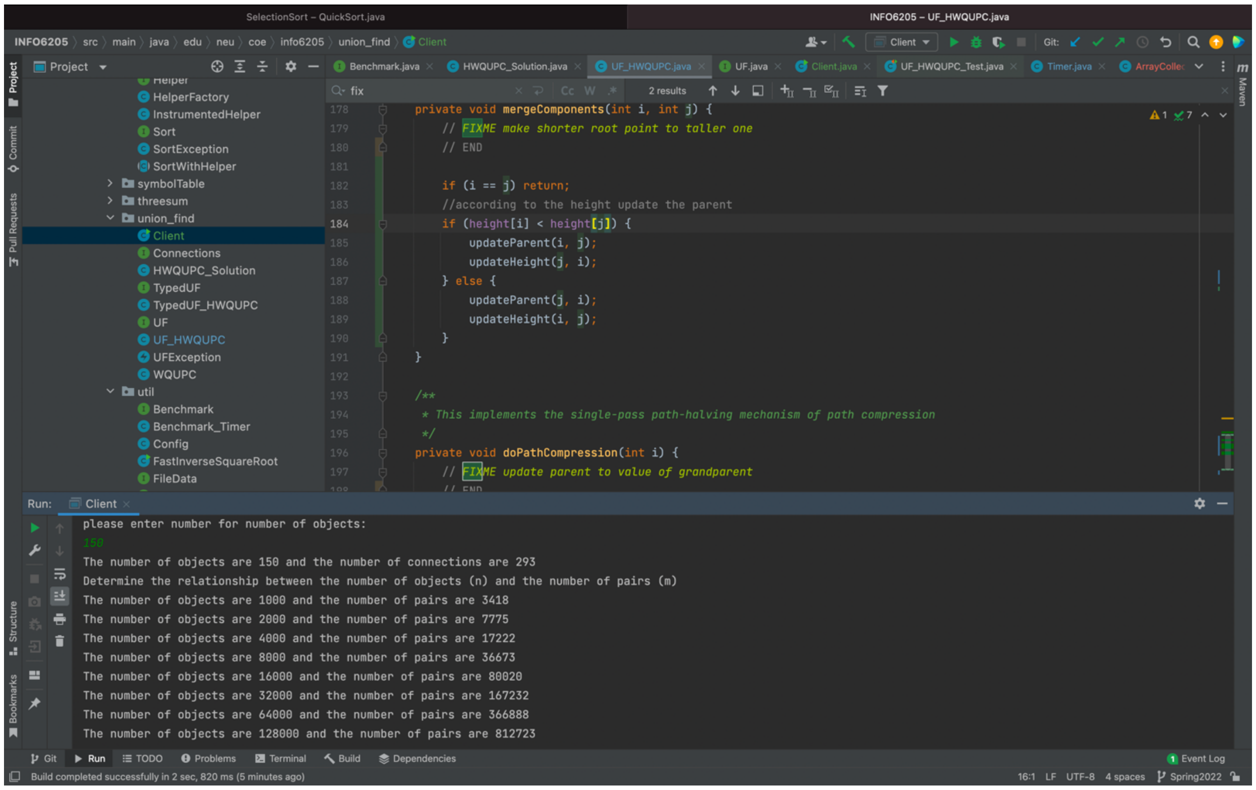
Name:Deepali Kasture

(NUID): 001586375

- **Task**

1. Implement height-weighted Quick Union with Path Compression.
2. 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."
3. 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).

- **Output screenshot**



```java
    private void mergeComponents(int i, int j) {
        // FIXME make shorter root point to taller one
        // END

        if (i == j) return;
        //according to the height update the parent
        if (height[i] < height[j]) {
            updateParent(i, j);
            updateHeight(j, i);
        } else {
            updateParent(j, i);
            updateHeight(i, j);
        }
    }

    /**
     * This implements the single-pass path-halving mechanism of path compression
     */
    private void doPathCompression(int i) {
        // FIXME update parent to value of grandparent
```

```
please enter number for number of objects:
150
The number of objects are 150 and the number of connections are 293
Determine the relationship between the number of objects (n) and the number of pairs (m)
The number of objects are 1000 and the number of pairs are 3418
The number of objects are 2000 and the number of pairs are 7775
The number of objects are 4000 and the number of pairs are 17222
The number of objects are 8000 and the number of pairs are 36673
The number of objects are 16000 and the number of pairs are 80020
The number of objects are 32000 and the number of pairs are 167232
The number of objects are 64000 and the number of pairs are 366888
The number of objects are 128000 and the number of pairs are 812723
```

Build completed successfully in 2 sec, 820 ms (5 minutes ago)

Code:

```java
/**
 * Returns the component identifier for the component containing site {@code
p}.
 *
 * @param p the integer representing one site
 * @return the component identifier for the component containing site {@code
p}
 * @throws IllegalArgumentException unless {@code 0 <= p < n}
 */
public int find(int p) {
    validate(p);
    int root = p;
    // TO BE IMPLEMENTED
    while (root != parent[root]) {
        if (pathCompression) {
            doPathCompression(root);
        }
        root = parent[root];
    }
    //
    return root;
    //END
}
```

```
private void mergeComponents(int i, int j) {
        // FIXME make shorter root point to taller one
        // END

        if (i == j) return;
        //according to the height update the parent
        if (height[j] > height[i]) {
            updateParent(i, j);
            updateHeight(j, i);
        } else {
            updateParent(j, i);
            updateHeight(i, j);
        }
    }

    /**
     * This implements the single-pass path-halving mechanism of path
compression
     */
    private void doPathCompression(int i) {
        // FIXME update parent to value of grandparent
        // END
        parent[i] = parent[parent[i]];
    }
```

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

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

public class Client {

    public static int count(int n) {
        //count the number of connections for n objects for random values of
a and b
        UF_HWQUPC uf = new UF_HWQUPC(n);
        Random r = new Random();
        int x = 0;
        while (uf.components() > 1) {
            int a = r.nextInt(n);
            int b = r.nextInt(n);

            uf.connect(a, b);
            x++;
        }
        return x;
    }
```

```java
    public static void main(String[] args) {
        //Take input n
        System.out.println("please enter number for number of objects:");
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        //get number of connections
        System.out.println("The number of objects are " + n + " and the
number of connections are " + count(n));
//step 3
        System.out.println("Determine the relationship between the number of
objects (n) and the number of pairs (m)");

        for (int i = 1000; i < 140000; i *= 2) {
            int total = 0;
            // test  count 5 times for an average number
            for (int j = 0; j < 5; j++) {
                total += count(i);
            }
            int avg = total / 5;
            System.out.println("The number of objects are " + i + " and the
number of pairs are " + avg);
        }
    }

}
```

- **Relationship Conclusion**

  m : The number of pairs
  n : The number of objects

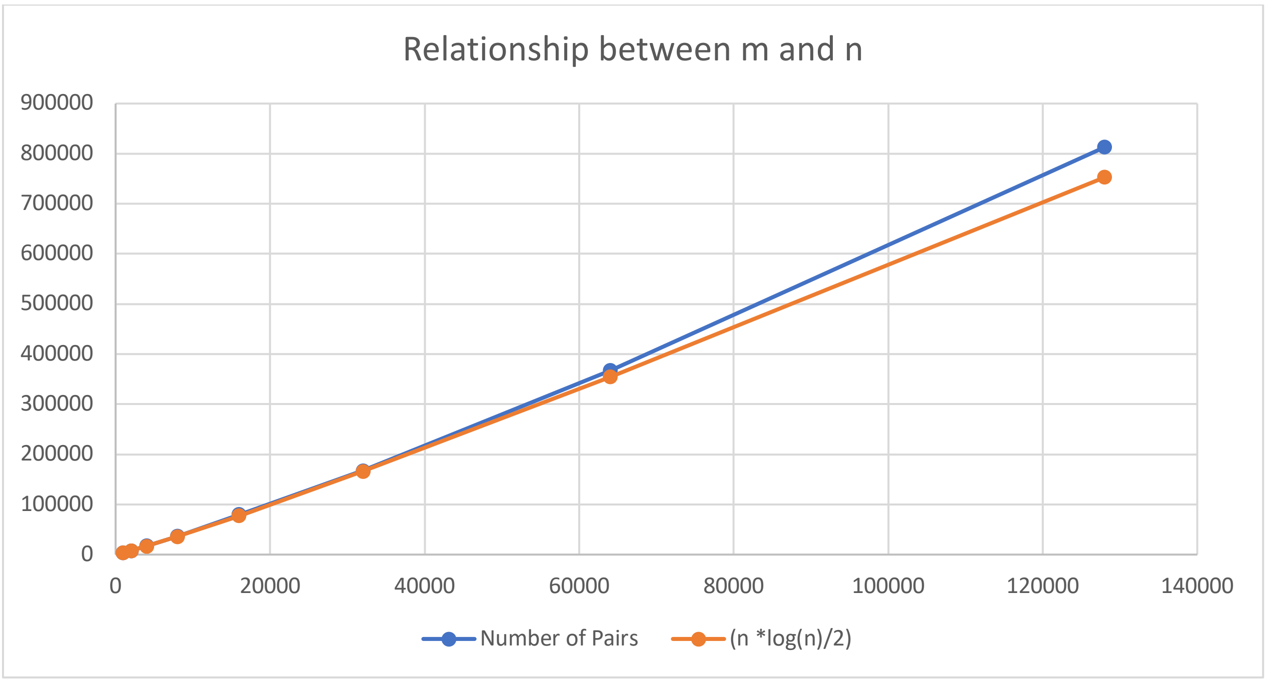
  Relationship between the number of objects (n) and the number of pairs (m)

  $m = \frac{1}{2} n \, \mathrm{Log}(n)$

- **Evidence / Graph**

Input:150

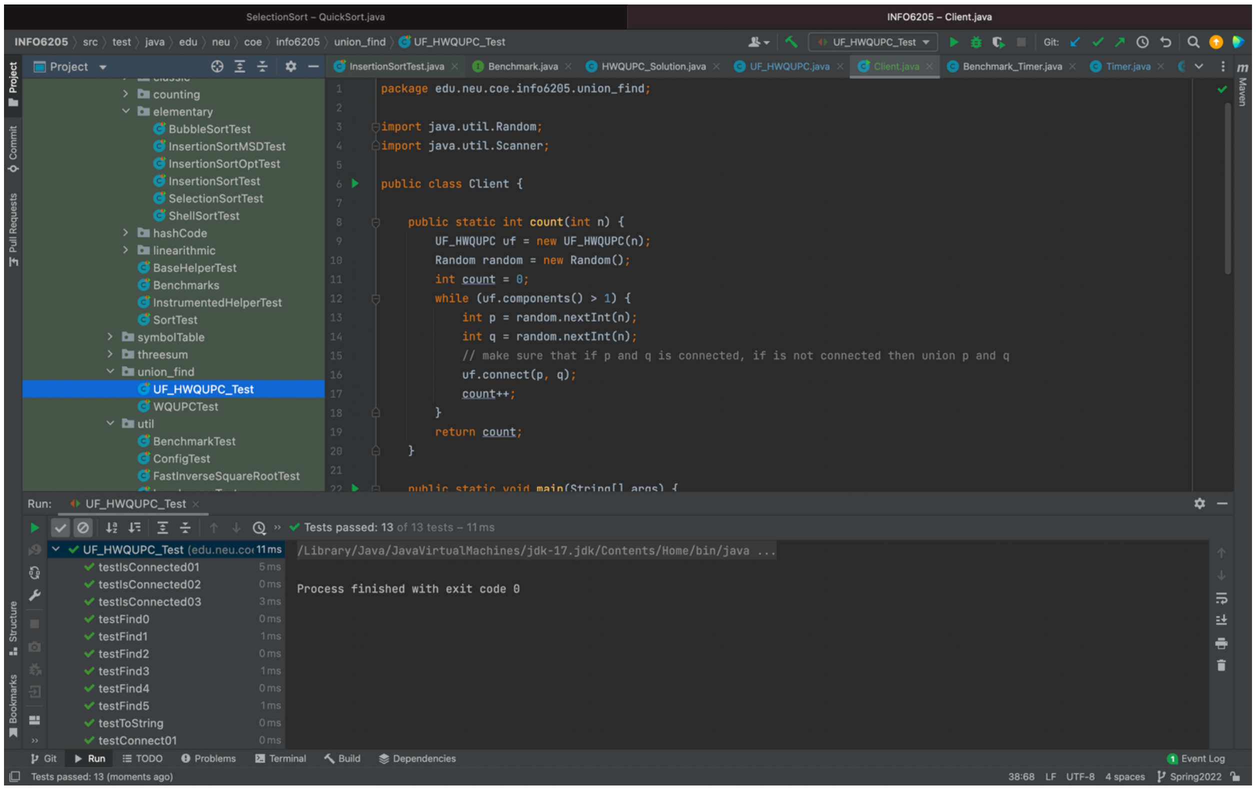
Number of Objects	Number of Pairs	(n *log(n)/2)
1000	3418	3454
2000	7775	7601
4000	17222	16588
8000	36673	35949
16000	80020	77443
32000	167232	165976
64000	366888	354132
128000	812723	752626



1. **Result:**

   -Run Client.java. Enter a number from command line as input.
   -To test the relationship between m and n we use higher values.
   -We passed n=150. Ran the count function 5 times to get average number of its pairs
   -X axis is number of objects.
   -Y axis is number of pairs
   -Blue line represents number of objects Vs number of pairs
   -Orange line represents calculated o/p Vs number of objects

- **Unit tests result**



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

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

public class Client {

    public static int count(int n) {
        UF_HWQUPC uf = new UF_HWQUPC(n);
        Random random = new Random();
        int count = 0;
        while (uf.components() > 1) {
            int p = random.nextInt(n);
            int q = random.nextInt(n);
            // make sure that if p and q is connected, if is not connected then union p and q
            uf.connect(p, q);
            count++;
        }
        return count;
    }

    public static void main(String[] args) {
```

```
Tests passed: 13 of 13 tests – 11 ms

UF_HWQUPC_Test (edu.neu.coe  11 ms
  testIsConnected01   5 ms
  testIsConnected02   0 ms
  testIsConnected03   3 ms
  testFind0           0 ms
  testFind1           1 ms
  testFind2           0 ms
  testFind3           1 ms
  testFind4           0 ms
  testFind5           1 ms
  testToString        0 ms
  testConnect01       0 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

```
Tests passed: 13 (moments ago)
```