

Program Structures and Algorithms
Spring 2023(SEC 03)

NAME: Vipul Rajderkar
NUID: 002700991

Task:

Part-1: Implement height-weighted Quick Union with Path Compression. Test all unit test cases.

Part-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." 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. Package your program as a static method `count()` that takes n as the argument and returns the number of connections; and a `main()` that takes n from the command line, calls `count()`, and prints the returned value. If you prefer, you can create a main program that doesn't require any input and runs the experiment for a fixed set of n values. Show evidence of your run(s).

Part-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).

Relationship Conclusion:

The relationship between the number of objects(n) and the number of connections formed is $n-1$. As all n nodes can be connected with $n-1$ links.

In task 3, ' m ' random pairs of integers between 0 and $n-1$ were formed, and it was checked if those 2 nodes are connected, and if not then connect them until all nodes are connected. The task was performed for 50 runs and then the average was taken.

The relationship between (n) and (m) can be derived as,

$m = c \times n \times \log n$, Where $c \approx 1.1$

Thus,

$m \propto n \log n$

Evidence to support that conclusion:

n (total number of sites)	m (total number of pairs generated)	log n	n log n	m/n	m/n log n
1000	3802	3	3000	3.802	1.26733333
2000	8076	3.301029996	6602.059991	4.038	1.22325456
4000	18150	3.602059991	14408.23997	4.5375	1.25969584
8000	38694	3.903089987	31224.7199	4.83675	1.23921048
16000	80904	4.204119983	67265.91972	5.0565	1.20274874
32000	183202	4.505149978	144164.7993	5.7250625	1.27078178
64000	373664	4.806179974	307595.5183	5.8385	1.21479013

Code Snippet:

Part 1: Implement *find*, *mergeComponents*, and *doPathCompression* methods

- find()*: In this method, we check if the given node is connected to the parent root or not. Path compression is done until it is connected.
- mergeComponents()*: If both nodes are not the same, we map the node with a lesser number of connections to a larger number of connections.
- doPathCompression()*: The size of the connected nodes has to be reduced hence we map the current node's child with the parent root of that node.

```
/**
 * 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}
 */
 Vipul Rajderkar
public int find(int p) {
    validate(p);
    int root = p;
    // FIXME
    if (pathCompression == true) {
        doPathCompression(p);
        while (root != parent[root]) {
            root = parent[root];
        }
    }
    else {
        while (root != parent[root])
            root = parent[root];
    }
    return root;
}
```

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

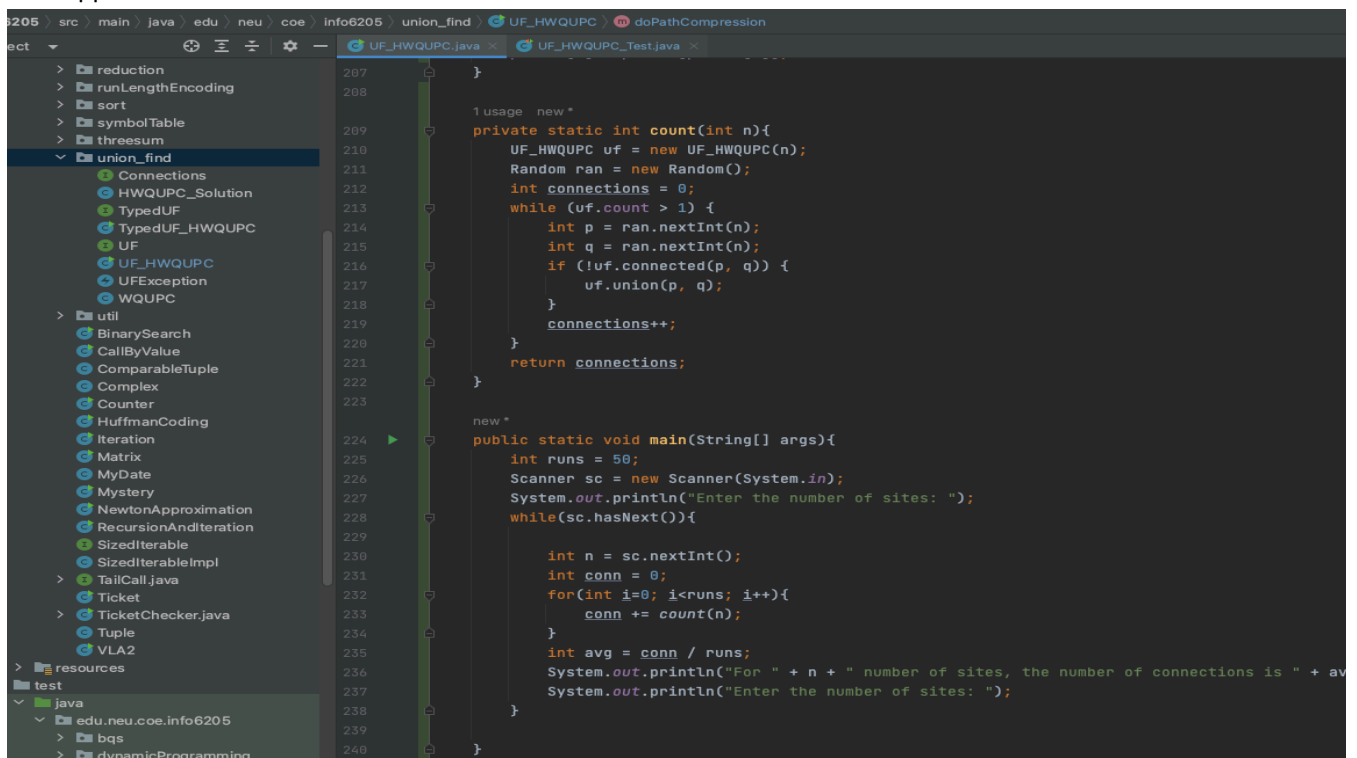
    if (i == j) {
        return;
    }
    if (height[i] < height[j]) {
        parent[i] = j;
        height[j] += height[i];
    } else {
        parent[j] = i;
        height[i] += height[j];
    }

    // END
}
```

```
1 usage Vipul Rajderkar
private void doPathCompression(int i) {
    // FIXME update parent to value of grandparent
    // END
    parent[i] = parent[parent[i]];
}
```

Part 2: Implement Client

Code Snippet:



```
205 src main java edu neu coe info6205 union_find UF_HWQUPC doPathCompression
206
207
208
209 1 usage new
210 private static int count(int n){
211     UF_HWQUPC uf = new UF_HWQUPC(n);
212     Random ran = new Random();
213     int connections = 0;
214     while (uf.count > 1) {
215         int p = ran.nextInt(n);
216         int q = ran.nextInt(n);
217         if (!uf.connected(p, q)) {
218             uf.union(p, q);
219             connections++;
220         }
221     }
222     return connections;
223 }
224
225 new
226 public static void main(String[] args){
227     int runs = 50;
228     Scanner sc = new Scanner(System.in);
229     System.out.println("Enter the number of sites: ");
230     while (sc.hasNext()){
231         int n = sc.nextInt();
232         int conn = 0;
233         for (int i=0; i<runs; i++){
234             conn += count(n);
235         }
236         int avg = conn / runs;
237         System.out.println("For " + n + " number of sites, the number of connections is " + avg);
238         System.out.println("Enter the number of sites: ");
239     }
240 }
```

Output:

For UF Client implementation main and count methods have been added in the same class. The count method returns the number of pairs generated. This method has been called for 50 times and then the average has been taken. The main method takes the input from the user for the value of n and then calls the count method.

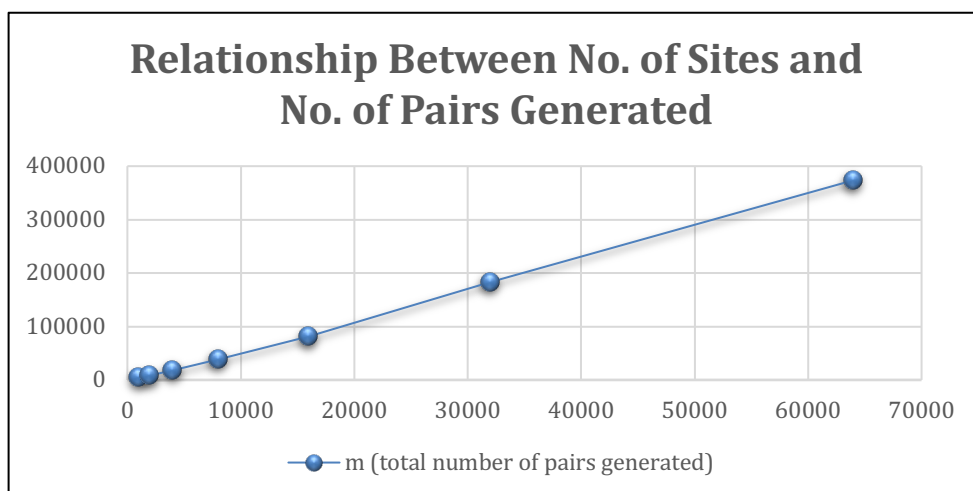
```
6205 > src > main > java > edu > neu > coe > info6205 > union_find > UF_HWQUPC > doPathCompression
UF_HWQUPC.java
UF_HWQUPC_Test.java

new *
public static void main(String[] args){
    int runs = 50;
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the number of sites: ");
    while(sc.hasNext()){
        int n = sc.nextInt();
        int conn = 0;
        for(int i=0; i<runs; i++){
            conn += count(n);
        }
        int avg = conn / runs;
        System.out.println("For " + n + " number of sites, the number of connections is " + avg);
        System.out.println("Enter the number of sites: ");
    }
}

/Users/Vipulrajdenkar/Library/Java/JavaVirtualMachines/openjdk-19.0.2/Contents/Home/bin/java ...
Enter the number of sites:
1000
For 1000 number of sites, the number of connections is 3802
Enter the number of sites:
2000
For 2000 number of sites, the number of connections is 8076
Enter the number of sites:
4000
For 4000 number of sites, the number of connections is 18150
Enter the number of sites:
8000
For 8000 number of sites, the number of connections is 38694
Enter the number of sites:
16000
For 16000 number of sites, the number of connections is 80904
Enter the number of sites:
32000
For 32000 number of sites, the number of connections is 183202
Enter the number of sites:
```

Graphical Representation:

n (total number of sites)	m (total number of pairs generated)	log n	n log n	m/n	m/n log n
1000	3802	3	3000	3.802	1.26733333
2000	8076	3.301029996	6602.059991	4.038	1.22325456
4000	18150	3.602059991	14408.23997	4.5375	1.25969584
8000	38694	3.903089987	31224.7199	4.83675	1.23921048
16000	80904	4.204119983	67265.91972	5.0565	1.20274874
32000	183202	4.505149978	144164.7993	5.7250625	1.27078178
64000	373664	4.806179974	307595.5183	5.8385	1.21479013



Unit Test Screenshots:

PSA-INFO6205 / src / test / java / edu / neu / coe / info6205 / union_find / UF_HWQUPC_Test / testToString

Project

PSA-INFO6205 [INFO6205] ~/MIS/PSA/PSA-IT

src

main

java

edu.neu.coe.info6205

balsearchtree

bqs

codelength

coupling

dynamicProgramming

equable

functions

graphs

greedy

UF_HWQUPC.java

UF_HWQUPC_Test.java

Comparison Failure

Comparison Failure

UF.java

Connections.java

InsertionSort.java

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

package edu.neu.coe.info6205.union_find;

import ...

no usages Vipul Rajderkar *

public class UF_HWQUPC_Test {

Vipul Rajderkar

@Test

public void testToString() {

Connections h = new UF_HWQUPC(2);

assertEquals(expected: "UF_HWQUPC:\n" +

" count: 2\n" +

" path compression? true\n" +

" parents: [0, 1]\n" +

Run: UF_HWQUPC_Test

Tests passed: 13 of 13 tests - 64 ms

UF_HWQUPC_Test (edu.neu.coe.info6205) 64 ms

testIsConnected01 8 ms

testIsConnected02 10 ms

testIsConnected03 25 ms

testFind0 0 ms

testFind1 1 ms

testFind2 0 ms

testFind3 2 ms

testFind4 1 ms

testFind5 1 ms

testToString 15 ms

testConnect01 0 ms

testConnect02 0 ms

testConnected01 1 ms

Process finished with exit code 0

Git

TODO

Problems

Terminal

Services

Build

Dependencies

Run

Tests passed: 13 (moments ago)

15:17 LF UTF-8 4 spaces main