

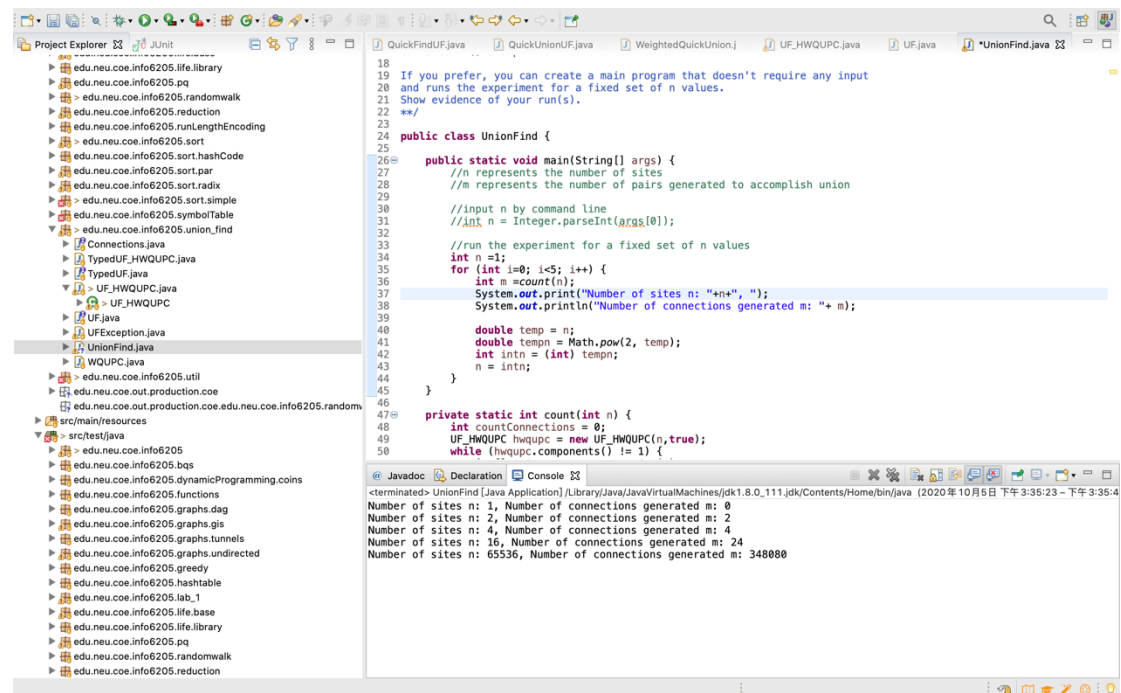
INFO 6205

Program Structures & Algorithms

Fall 2020

Assignment 3

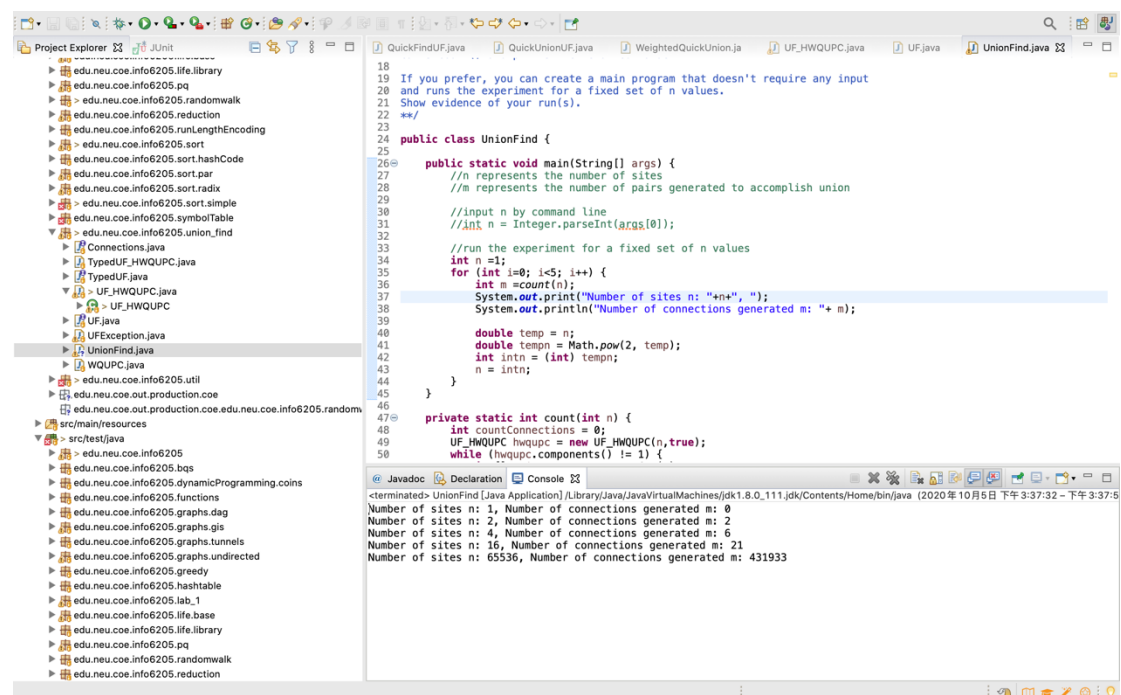
Output (only show part of the output):



```
18
19 If you prefer, you can create a main program that doesn't require any input
20 and runs the experiment for a fixed set of n values.
21 Show evidence of your run(s).
22 **/
23
24 public class UnionFind {
25
26     public static void main(String[] args) {
27         //n represents the number of sites
28         //m represents the number of pairs generated to accomplish union
29
30         //input n by command line
31         //\n\n n = Integer.parseInt(args[0]);
32
33         //run the experiment for a fixed set of n values
34         int n = 1;
35         for (int i=0; i<5; i++) {
36             int m = count(n);
37             System.out.println("Number of sites n: "+n);
38             System.out.println("Number of connections generated m: "+ m);
39
40             double temp = n;
41             double tempn = Math.pow(2, temp);
42             int intn = (int) tempn;
43             n = intn;
44         }
45     }
46
47     private static int count(int n) {
48         int countConnections = 0;
49         UF_HWQUPC hwqupc = new UF_HWQUPC(n, true);
50         while (hwqupc.components() != 1) {
```

<terminated> UnionFind [Java Application] [Library:Java\JavaVirtualMachines\jdk1.8.0_111\jdk\Contents\Home\bin\java (2020年10月5日 下午3:35:23 - 下午3:35:4

Number of sites n: 1, Number of connections generated m: 0
Number of sites n: 2, Number of connections generated m: 2
Number of sites n: 4, Number of connections generated m: 4
Number of sites n: 16, Number of connections generated m: 24
Number of sites n: 65536, Number of connections generated m: 348880



```
18
19 If you prefer, you can create a main program that doesn't require any input
20 and runs the experiment for a fixed set of n values.
21 Show evidence of your run(s).
22 **/
23
24 public class UnionFind {
25
26     public static void main(String[] args) {
27         //n represents the number of sites
28         //m represents the number of pairs generated to accomplish union
29
30         //input n by command line
31         //\n\n n = Integer.parseInt(args[0]);
32
33         //run the experiment for a fixed set of n values
34         int n = 1;
35         for (int i=0; i<5; i++) {
36             int m = count(n);
37             System.out.println("Number of sites n: "+n);
38             System.out.println("Number of connections generated m: "+ m);
39
40             double temp = n;
41             double tempn = Math.pow(2, temp);
42             int intn = (int) tempn;
43             n = intn;
44         }
45     }
46
47     private static int count(int n) {
48         int countConnections = 0;
49         UF_HWQUPC hwqupc = new UF_HWQUPC(n, true);
50         while (hwqupc.components() != 1) {
```

<terminated> UnionFind [Java Application] [Library:Java\JavaVirtualMachines\jdk1.8.0_111\jdk\Contents\Home\bin\java (2020年10月5日 下午3:37:32 - 下午3:37:5

Number of sites n: 1, Number of connections generated m: 0
Number of sites n: 2, Number of connections generated m: 2
Number of sites n: 4, Number of connections generated m: 6
Number of sites n: 16, Number of connections generated m: 21
Number of sites n: 65536, Number of connections generated m: 431933

```

18
19 If you prefer, you can create a main program that doesn't require any input
20 and runs the experiment for a fixed set of n values.
21 Show evidence of your run(s).
22 **/
23
24 public class UnionFind {
25
26     public static void main(String[] args) {
27         //n represents the number of sites
28         //m represents the number of pairs generated to accomplish union
29
30         //input n by command line
31         //int n = Integer.parseInt(args[0]);
32
33         //run the experiment for a fixed set of n values
34         int n = 1;
35         for (int i=0; i<5; i++) {
36             int m = count(n);
37             System.out.println("Number of sites n: " + n + ", ");
38             System.out.println("Number of connections generated m: " + m);
39
40             double temp = n;
41             double tempn = Math.pow(2, temp);
42             int intn = (int) tempn;
43             n = intn;
44         }
45     }
46
47     private static int count(int n) {
48         int countConnections = 0;
49         UF_HWQUPC hwqpc = new UF_HWQUPC(n, true);
50         while (hwqpc.components() != 1) {

```

Console Output:

```

<terminated> UnionFind [Java Application] \Library\Java\JavaVirtualMachines\jdk1.8.0_111_jdk\Contents\Home\bin\java (2020年10月5日 下午4:14:13 - 下午4:14:3
Number of sites n: 1, Number of connections generated m: 0
Number of sites n: 2, Number of connections generated m: 3
Number of sites n: 4, Number of connections generated m: 4
Number of sites n: 16, Number of connections generated m: 23
Number of sites n: 65536, Number of connections generated m: 404105

```

Conclusion:

n : the number of objects

m : the number of pairs generated to accomplish the union

$\lg n$ and $\lg m$ have a linear relationship, and the slope is approximately 1.253:

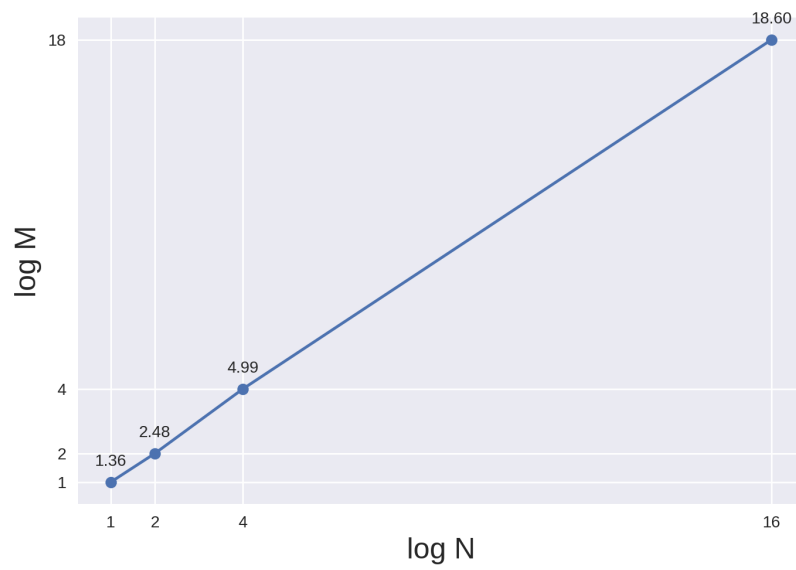
$$\lg m \approx 1.253 * \lg n$$

When $\lg n$ increases, $\lg m$ increases by a higher degree than $\lg n$.

Evidence:

$n \backslash m$	output 1	output 2	output 3	output 4	output 5	output 6	output 7
2	4	2	2	2	3	2	3
4	7	5	4	6	7	6	4
16	53	24	24	21	26	52	23
65536	317080	408000	348080	431933	396867	475866	404105

n	$\log_2 n$	Average m	$\log_2 m$
2	1	2.571	1.3623
4	2	5.571	2.4779
16	4	31.857	4.9935
65536	16	397418.714	18.6003



Respective slope:

$$1.36/1 = 1.36$$

$$2.48/2 = 1.24$$

$$4.99/4 = 1.2475$$

$$18.6/16 = 1.1625$$

Average slope:

$$1.36+1.24+1.2475+1.1625 \approx 1.253$$