

Program Structures & Algorithms

Spring 2022

Assignment No. 3

WQUPC

Name: Aishwarya Surve

(NUID): 002123768

STEP 1

(a) Implement height-weighted Quick Union with Path Compression. For this, you will flesh out the class `UF_HWQUPC`. All you have to do is to fill in the sections marked with `// TO BE IMPLEMENTED ... // ...END IMPLEMENTATION`.

(b) Check that the unit tests for this class all work. You must show "green" test results in your submission (screenshot is OK).

- **Output Screenshot**
 1. `UF_HWQUPC_Test`

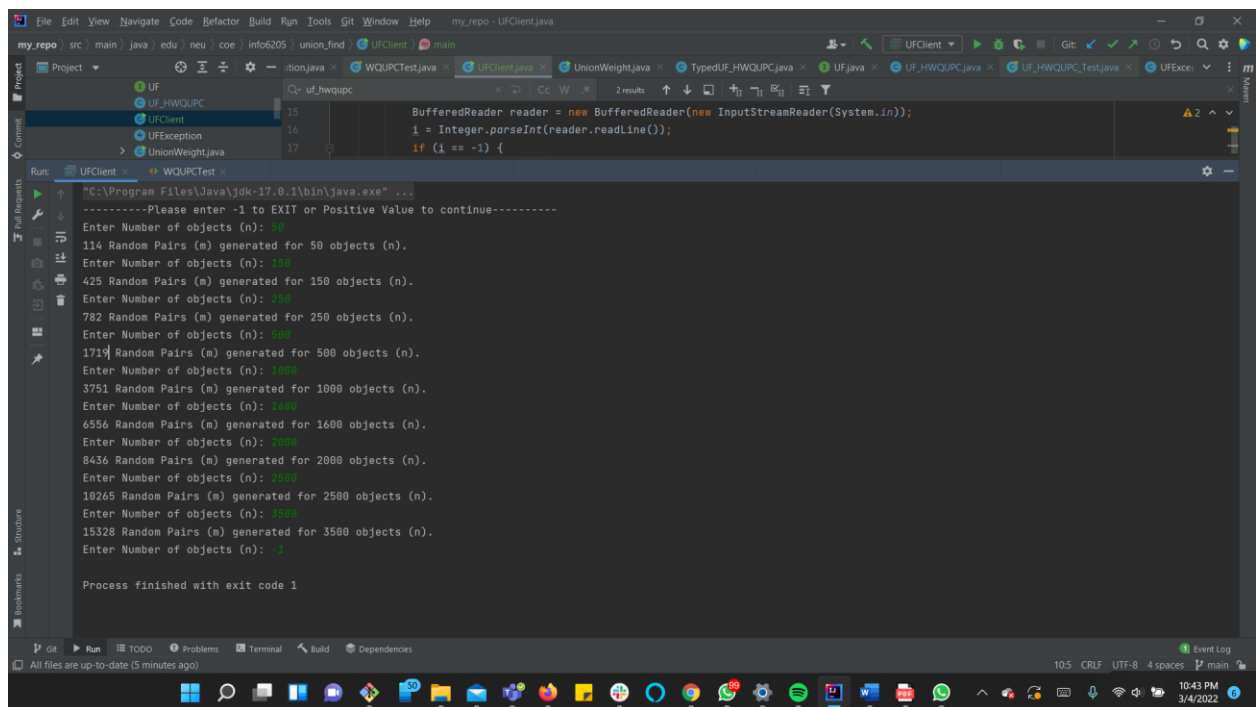
The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project structure with folders like `lab_1`, `life`, `pq`, `randomwalk`, `reduction`, `sort`, `symbolTable`, `threesum`, `union_find`, and `UF_HWQUPC_Test`.
- Code Editor:** Displays the `UF_HWQUPC_Test.java` file. The code includes a package declaration `package edu.neu.coe.info6205.union_find;`, an import statement `import ...`, and a class definition `public class UF_HWQUPC_Test {`. Inside the class, there is a `@Test` annotation and a `testToString()` method that creates a `UF_HWQUPC` object and asserts its `toString()` output.
- Run Console:** Shows the output of the test run. It indicates that 13 out of 13 tests passed in 7 ms. The tests listed are `testIsConnected01` through `testIsConnected05`, `testFind0` through `testFind5`, `testToString`, `testConnect01`, `testConnect02`, and `testConnect01`. All tests passed with a duration of 0 ms.
- Terminal:** Shows the command `"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...` and the message `Process finished with exit code 0`.

STEP 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).

- **Output Screenshot**
 1. UFClient



```
my_repo - my_repo - UFClient.java
my_repo - src - main - java - edu - neu - coe - info6205 - union_find - UFClient - main
Project - Project - UF - UF_HWQUPC - UFClient - UFException - UnionWeight.java
Run - UFClient - WQUPCTest
C:\uf_hwqupc
15
16
17
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
i = Integer.parseInt(reader.readLine());
if (i == -1) {
    "C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
    -----Please enter -1 to EXIT or Positive Value to continue-----
    Enter Number of objects (n): 50
    114 Random Pairs (m) generated for 50 objects (n).
    Enter Number of objects (n): 100
    425 Random Pairs (m) generated for 100 objects (n).
    Enter Number of objects (n): 200
    782 Random Pairs (m) generated for 200 objects (n).
    Enter Number of objects (n): 500
    1714 Random Pairs (m) generated for 500 objects (n).
    Enter Number of objects (n): 1000
    3751 Random Pairs (m) generated for 1000 objects (n).
    Enter Number of objects (n): 1500
    6556 Random Pairs (m) generated for 1500 objects (n).
    Enter Number of objects (n): 2000
    8436 Random Pairs (m) generated for 2000 objects (n).
    Enter Number of objects (n): 2500
    10265 Random Pairs (m) generated for 2500 objects (n).
    Enter Number of objects (n): 3000
    15328 Random Pairs (m) generated for 3000 objects (n).
    Enter Number of objects (n): -1
    Process finished with exit code 1
```

STEP 3

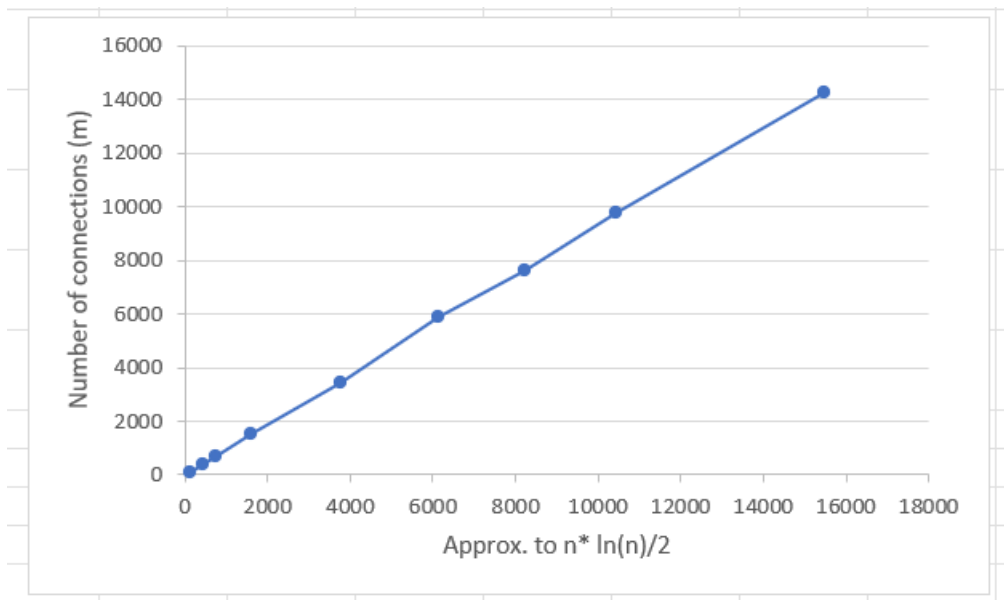
Determine the relationship between the number of objects (n) and the number of pairs (m) generated to accomplish this (i.e. to reduce the number of components from n to 1). Justify your conclusion in terms of your observations and what you think might be going on.

Evidence to support the conclusion

- **Table**

Number of Objects (n)	Number of connections (m)	Approx. to $n \cdot \ln(n)/2$
50	114	97
150	425	375
250	782	690
500	1719	1553
1000	3751	3453
1600	6556	5902
2000	8436	7600
2500	10265	9780
3500	15328	14280

- **Graphical Representation**



Conclusion

1. Based on the number of tests conducted, The relationship between the number of objects (n) and the number of pairs (m) is $m = n * \ln(n)/2$, as the observed Number of objects column and Number of connections column are definitely equivalent, as evidenced by the evidence supplied above.
2. Weight Quick-Union with Path Compression is a linear process in practice.
3. Weight Quick-Union with Path Compression isn't exactly linear in principle.