**Program Structures and Algorithms**

**Assignment No: 3**

Name: Amar Nagargoje

NUID: 002273113

GitHub:<https://github.com/amarneu/INFO6205/blob/Spring2024/src/main/java/edu/neu/coe/info6205/union_find/UF_HWQUPC.java>

1. **Tasks:**

Step 1:  
(a) Implement height-weighted Quick Union with Path Compression.

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).

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.

1. **Relationship conclusion:**

The relationship between the number of objects(n) and the number of randomly generated pairs (m) in order to reduce the number of components/objects from n to 1, in other words in order for all the pairs to be connected can be given as follows: After taking the average of the value of m over: 50 runs of the program(for the same value of n),

we can say that m = c \* n \* log(n)

where c = m/n \* log(n)

which is approximately equal to 1.22 as observed over n ranging from 500 to 516000(doubling)

Therefore, we can summarize the relationship as m ∝ n \* log(n)

1. **Evidence that support conclusion:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of Objects (n)** | **No of pairs generated on avg of 50 runs** | **n log n** | **m/n log n** |
| 500 | 1677 | 1349.485002 | 1.242696286 |
| 1000 | 3745 | 3000 | 1.248333333 |
| 2000 | 7982 | 6602.059991 | 1.209016581 |
| 4000 | 17503 | 14408.23997 | 1.214790984 |
| 8000 | 37331 | 31224.7199 | 1.195559164 |
| 16000 | 81944 | 67265.91972 | 1.218209761 |
| 32000 | 174483 | 144164.7993 | 1.210302382 |
| 64000 | 369082 | 307595.5183 | 1.199893945 |
| 128000 | 786762 | 653722.8761 | 1.203509971 |
| 256000 | 1714288 | 1384509.431 | 1.238191638 |
| 516000 | 3506299 | 2947727.246 | 1.189492347 |

A graph with a line

Description automatically generated

Output:

A screenshot of a computer

Description automatically generated

1. Unit test demonstration

A screenshot of a computer

Description automatically generated