Program Structures and Algorithms

Spring 2023(SEC 01)

NAME: Shivani Datar

NUID: 002772160

**Task:**

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).

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).

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.

**Relationship Conclusion:**

The relationship between the number of objects(n) and number of connections formed (Asked in step 2) is always n-1. As for all n nodes to be connected with each other with at least degree 1 is n-1. That is if we have n = 1000 we can form 999 connections until all n components/nodes are connected with each other.

We are generating random pairs(m) between 0 to n-1 and checking if they are connected with each other, until all the components are connected with each other. Since the generation of pairs is random in nature, I am generating (m) for same n 50 times and then taking the average of the pairs generated.   
  
The relationship between (n) and (m) can be said as,  
m = c \* n \* log n

Where c = 1.1 approximately in this calculation and can differ machine to machine.

Therefore, we can say that,  
m ∝ n log n

**Evidence to support that conclusion:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N(total number of objects) | M(total number of pairs generated) | log n | n log n | m/n | c = m/n log n |
| 1000 | 3228 | 3 | 3000 | 3.228 | 1.076 |
| 2000 | 7615 | 3.30103 | 6602.05999 | 3.8075 | 1.15342787 |
| 4000 | 16484 | 3.60205999 | 14408.24 | 4.121 | 1.14406756 |
| 8000 | 32687 | 3.90308999 | 31224.7199 | 4.085875 | 1.04683085 |
| 16000 | 70211 | 4.20411998 | 67265.9197 | 4.3881875 | 1.04378265 |
| 32000 | 164049 | 4.50514998 | 144164.799 | 5.12653125 | 1.13792688 |
| 64000 | 381878 | 4.80617997 | 307595.518 | 5.96684375 | 1.24149403 |
| 128000 | 784233 | 5.10720997 | 653722.876 | 6.12682031 | 1.19964136 |
| 256000 | 1535820 | 5.40823997 | 1384509.43 | 5.99929688 | 1.10928822 |

Graph :-

Chart, line chart

Description automatically generated

Code :-

Step 1.

1. In find method, we check if the given node is connected to the parent root or not. And until its not connected we do the compression so that we can have a path compressed structure.

Text

Description automatically generated

1. Under union we call mergeComponents method. Here we first check if they are same, else we try to find which node has a smaller connected set of nodes, we point the smaller one to the taller connected set of nodes.

Text

Description automatically generated

1. Under do path compression, we have to lower the size of the connected nodes, we do that by pointing a current node’s child with the parent root of that node.

Text

Description automatically generated

Step 2.  
a. To have a UF Client I have added a main method in the same class, which takes in the number of objects and then calls the method count, and later on gives us number of connections formed and number of random pair generated till all the nodes/components are connected with each other.  
  
Text

Description automatically generated

b. Under the static count method, I am instantiating uf object of UF\_HWQUPC. Then I randomly generate pairs p and q until the total count of unconnected nodes/components is zero. The if condition checks if they are already connected, if not I increment the connections variable which gives us total number of connections. Either way, in the end the code will increment the variable pairs, which shows total number of pairs generated. And it will return an int array with both the values connections and number of pairs.

Text

Description automatically generated

**Graphical Representation:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N(total number of objects) | M(total number of pairs generated) | log n | n log n | m/n | c = m/n log n |
| 1000 | 3228 | 3 | 3000 | 3.228 | 1.076 |
| 2000 | 7615 | 3.30103 | 6602.05999 | 3.8075 | 1.15342787 |
| 4000 | 16484 | 3.60205999 | 14408.24 | 4.121 | 1.14406756 |
| 8000 | 32687 | 3.90308999 | 31224.7199 | 4.085875 | 1.04683085 |
| 16000 | 70211 | 4.20411998 | 67265.9197 | 4.3881875 | 1.04378265 |
| 32000 | 164049 | 4.50514998 | 144164.799 | 5.12653125 | 1.13792688 |
| 64000 | 381878 | 4.80617997 | 307595.518 | 5.96684375 | 1.24149403 |
| 128000 | 784233 | 5.10720997 | 653722.876 | 6.12682031 | 1.19964136 |
| 256000 | 1535820 | 5.40823997 | 1384509.43 | 5.99929688 | 1.10928822 |

Chart, line chart

Description automatically generated

**Unit Test Screenshots:**

Text

Description automatically generated