Varad Ratnakar Desai (001465732)

Program Structures & Algorithms

Spring 2021

Assignment No. 3

- **Task**
- Your task is
- 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).
- 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.
- Don't forget to follow the submission guidelines. And to use sufficient (and sufficiently large) different values of n.

- **Output**

```
Count = 606 for Size = 200
Count = 1328 for Size = 400
Count = 2823 for Size = 800
Count = 6270 for Size = 1600
Count = 13701 for Size = 3200
Count = 29909 for Size = 6400
Count = 64350 for Size = 12800
Count = 138843 for Size = 25600
Count = 289276 for Size = 51200
Count = 640725 for Size = 102400
Count = 1326585 for Size = 204800
Count = 2744495 for Size = 409600
Count = 5783197 for Size = 819200
Count = 12364102 for Size = 1638400
Count = -17404827 for Size = 3276800
Count = 10773364 for Size = 6553600
Count = -17416004 for Size = 13107200
Count = 18109574 for Size = 26214400
Count = 10838905 for Size = 52428800
Count = 13832695 for Size = 104857600
```

- **Conclusion:**

1. As we increase the number of objects for our algorithm the number of pairs connected also increases.
2. As justified in the tabular and graphical evidence below, the relationship between number of objects and number of pairs is:

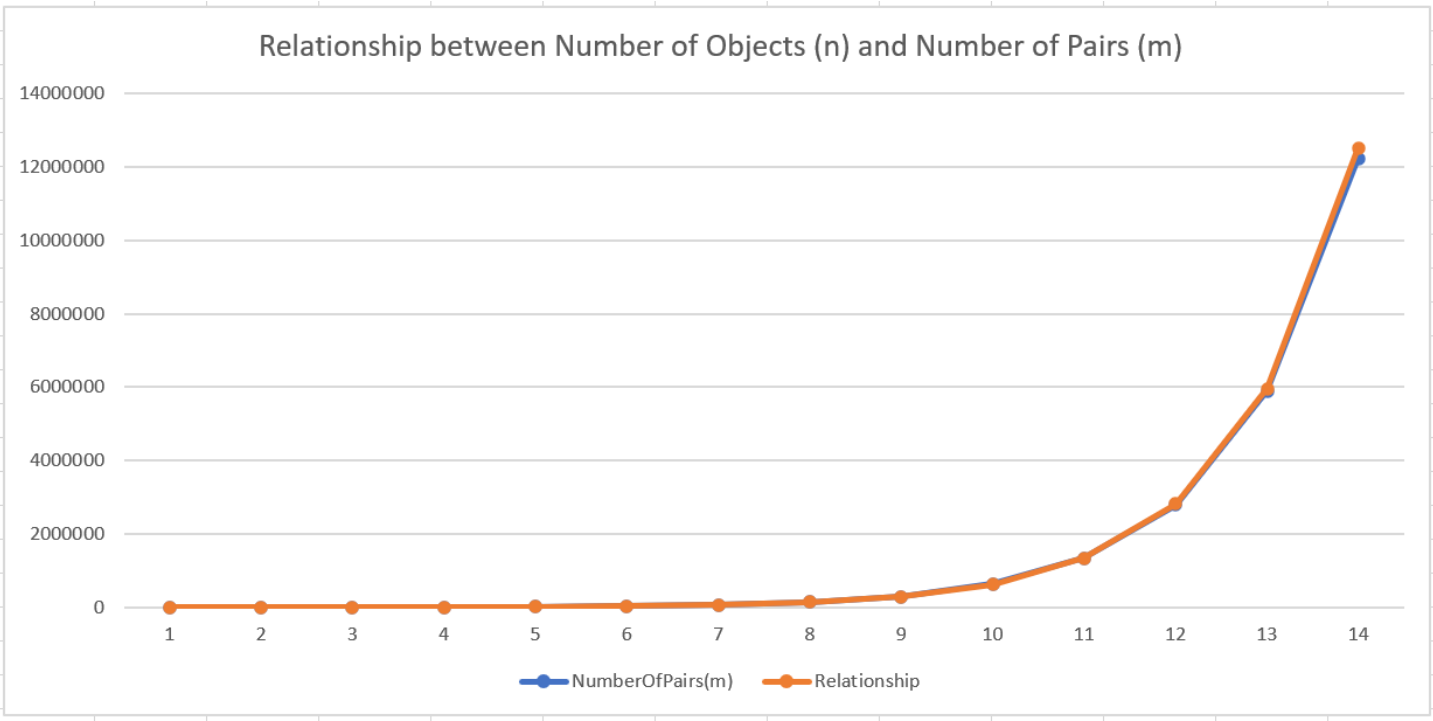
$$m = k * n \lg(n)$$

Here,
k = 0.37
n = Number of objects
m = Number of Pairs

- **<u>Evidence to support the conclusion:</u>**
- **Tabular representation:**

Number of Objects (n)	Number of Pairs (m)	Relationship = 0.37 * n * log(n)
200	592	565.645358
400	1351	1279.290716
800	2903	2854.581432
1600	6308	6301.162864
3200	13918	13786.32573
6400	29971	29940.65146
12800	62959	64617.30291
25600	141660	138706.6058
51200	289798	296357.2117
102400	641492	630602.4233
204800	1342031	1336980.847
409600	2781688	2825513.693
819200	5899729	5954131.387
1638400	12238434	12514470.77

- **Graphical representation:**



- **Unit tests result**