# Program Structures & Algorithms
# Spring 2023
# Assignment No. 4

Name: Anurag Nandre
(NUID): 002785735

## Task

- (Part 1) Implement height-weighted Quick Union with Path Compression
- (Part 2) Develop a UF 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
- (Part 3) Determine the relationship between the number of objects *"n"* and the number of pairs *"m"*

## Relationship Conclusion

The relationship between the number of objects (n) and the number of pairs (m) generated to reduce the number of components from n to 1 is:

$$m = f(n) = 0.5 \times n * ln(n)$$

where,

m = number of pairs generated to reduce the number of components to 1
n = number of objects

## Evidence to the Conclusion

Let f(n) be the number of pairs (m) generated to reduce the number of components from n to 1.

Taking initial value of n as 100 and using the doubling method, we can calculate the number of pairs (m) generated to reduce the number of components from n to 1, and compute the average number of pairs generated to accomplish this for each value of n.

For larger values of n, although not equal, the average number of pairs needed to reduce the components to 1 is close to $0.5 \times n * ln(n)$.

In this union-find operation, we check if the pairs are connected or disconnected (n ln(n)). There are only two possibilities for each pair. Hence, the relationship between m and n is almost identical to *0.5 x n * ln(n).*

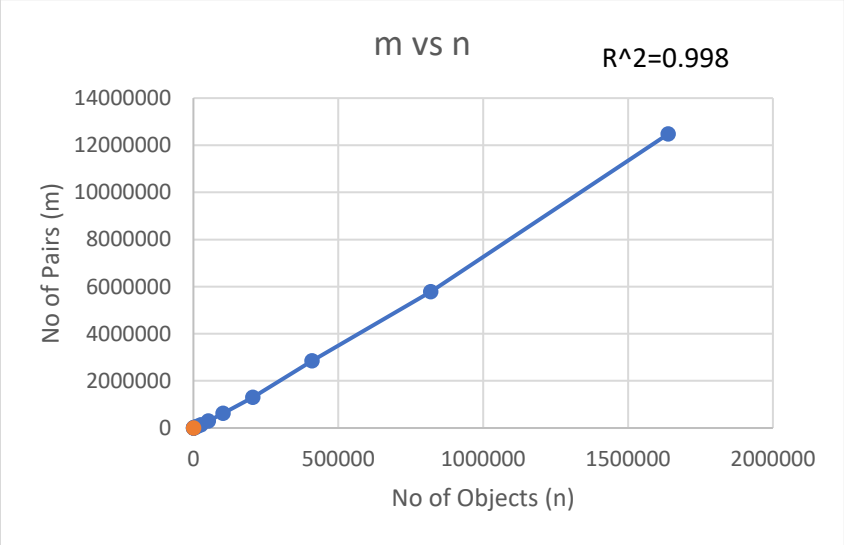
Below are the results for the performed simulations:

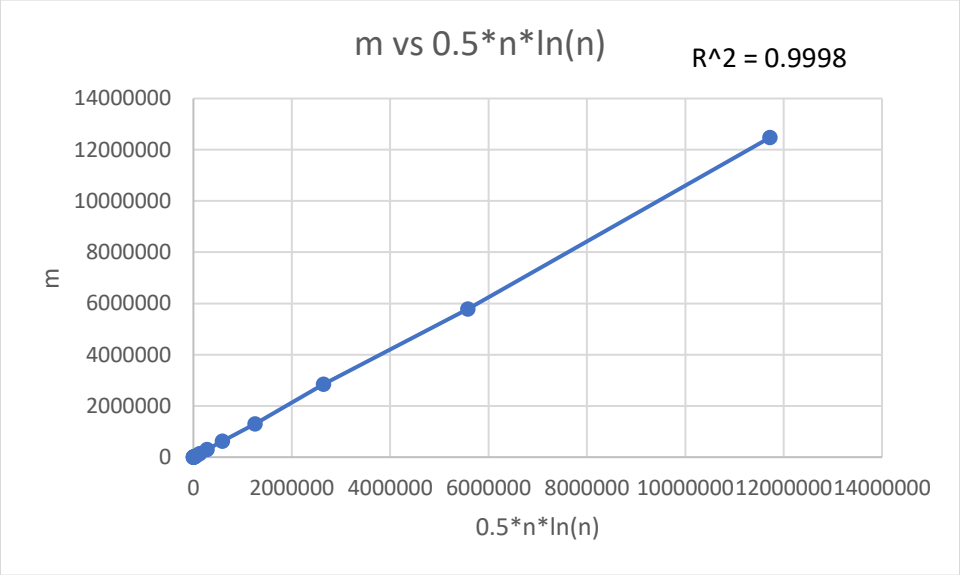
| Number of Objects (n) | 0.5 * n * ln(n) | Number of pairs (m) |
|---|---|---|
| 100 | 230 | 260 |
| 200 | 530 | 599 |
| 400 | 1198 | 1277 |
| 800 | 2674 | 2956 |
| 1600 | 5902 | 6545 |
| 3200 | 12913 | 13721 |
| 6400 | 28045 | 30380 |
| 12800 | 60526 | 61360 |
| 25600 | 129924 | 135060 |
| 51200 | 277593 | 297872 |
| 102400 | 590676 | 618964 |
| 204800 | 1252330 | 1293324 |
| 409600 | 2646617 | 2852519 |
| 819200 | 5577148 | 5780332 |
| 1638400 | 11722122 | 12476295 |

I have checked two plots to test the relationship between “n” and “m”. They are as follows

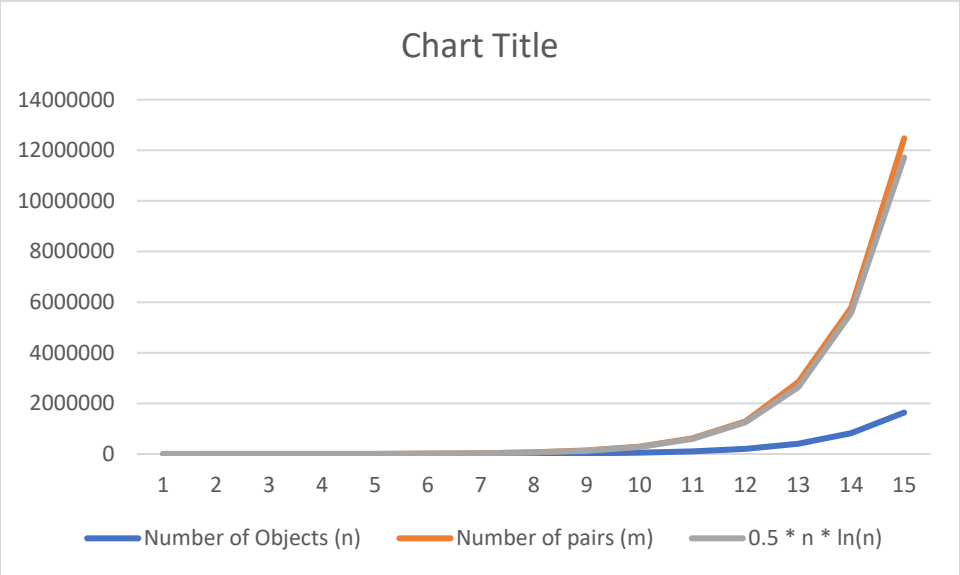
1) m vs n
2) m vs 0.5*n*ln(n)

Coefficient of determination ($R^2$) has been leveraged to identify the best fit among the below plots. But turns out that both the plots have similar $R^2$ value.

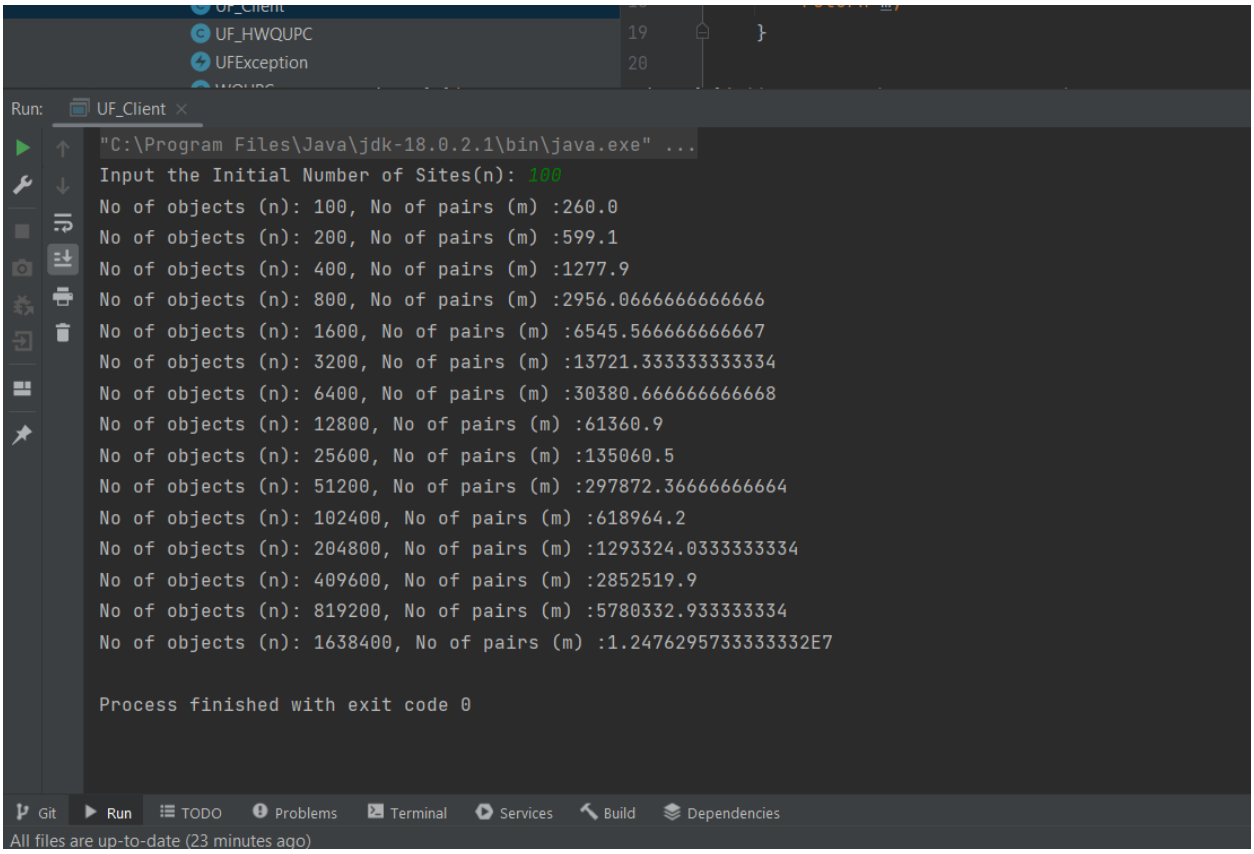





As $R^2$ value is not helping much here, I have plotted all the three parameters (m, n, 0.5*n*ln(n)) in a single plot for various observation points. From the plot below, it is clearly evident that “m” and “0.5*n*ln(n)” are strongly correlated and would be the best fit for our data points.



## Output Screenshot



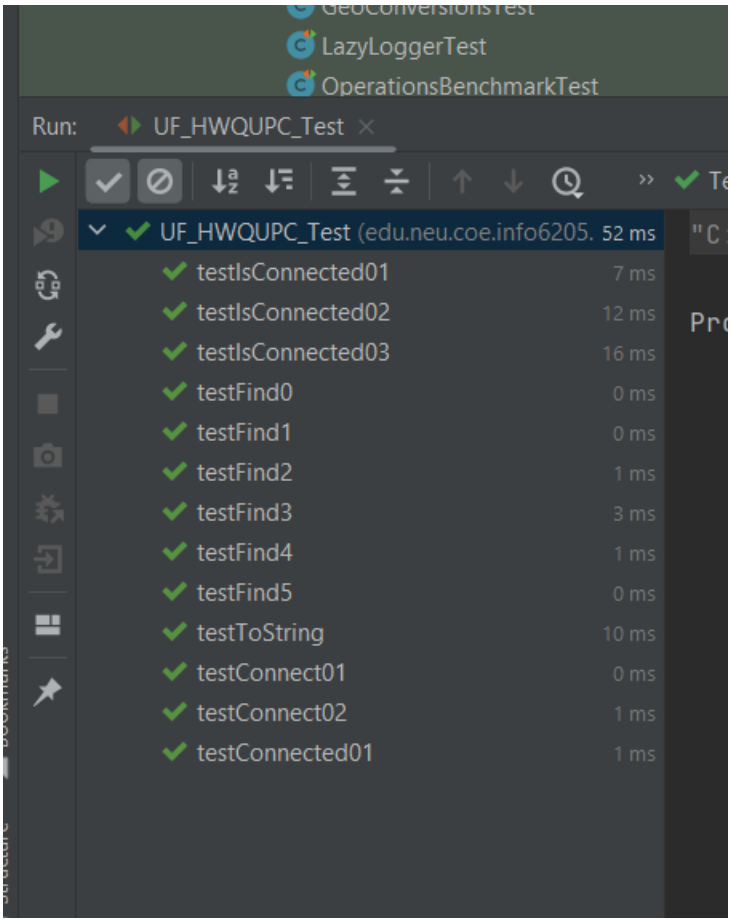

## Output

Input the Initial Number of Sites(n): 100
No of objects (n): 100, No of pairs (m) :260.0
No of objects (n): 200, No of pairs (m) :599.1
No of objects (n): 400, No of pairs (m) :1277.9
No of objects (n): 800, No of pairs (m) :2956.0666666666666
No of objects (n): 1600, No of pairs (m) :6545.566666666667
No of objects (n): 3200, No of pairs (m) :13721.333333333334
No of objects (n): 6400, No of pairs (m) :30380.666666666668
No of objects (n): 12800, No of pairs (m) :61360.9
No of objects (n): 25600, No of pairs (m) :135060.5
No of objects (n): 51200, No of pairs (m) :297872.36666666664

No of objects (n): 102400, No of pairs (m) :618964.2
No of objects (n): 204800, No of pairs (m) :1293324.0333333334
No of objects (n): 409600, No of pairs (m) :2852519.9
No of objects (n): 819200, No of pairs (m) :5780332.933333334
No of objects (n): 1638400, No of pairs (m) :1.2476295733333332E7

## Unit Tests

***UF_HWQUPC_Test.java***



***WQUPCTest.java***