

Program Structures and Algorithms Sec -8

Name: Uday Kiran Reddy Mulpuri NUID: 002781063

Assignment-4(WQUPC)

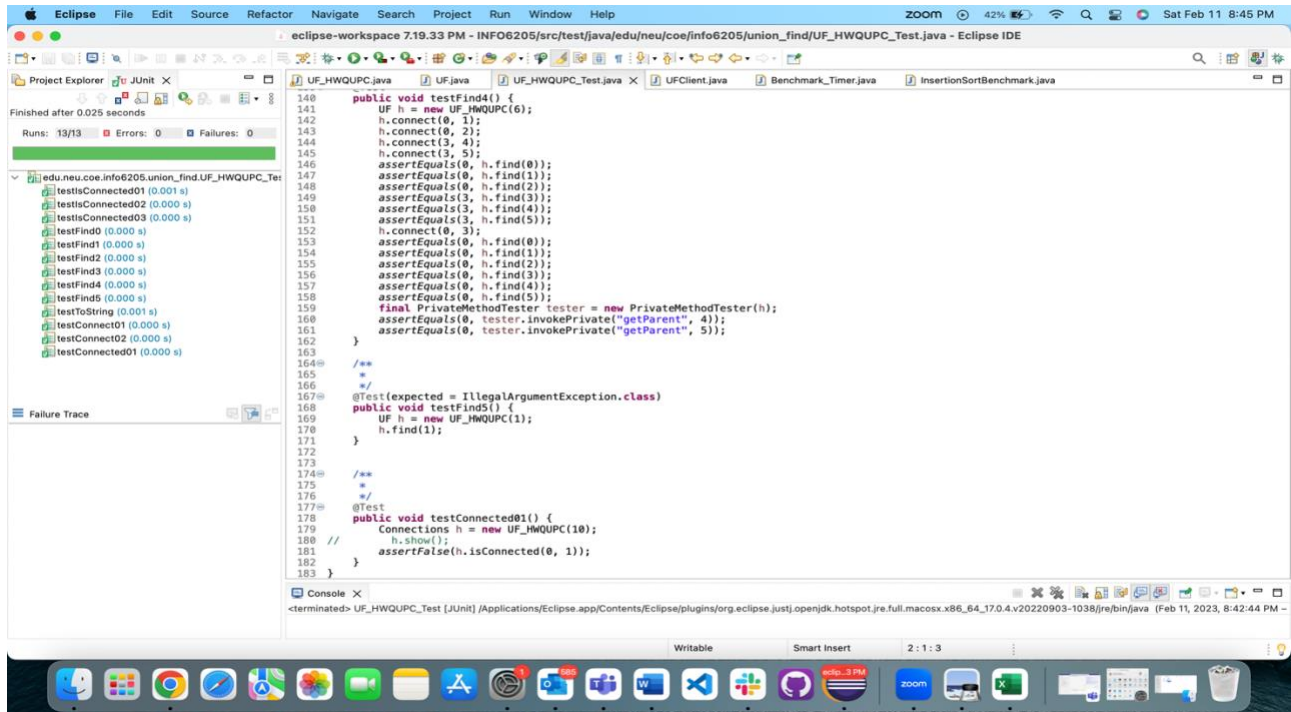
Step 1:

a) Implementing height-weighted Quick Union with Path Compression.

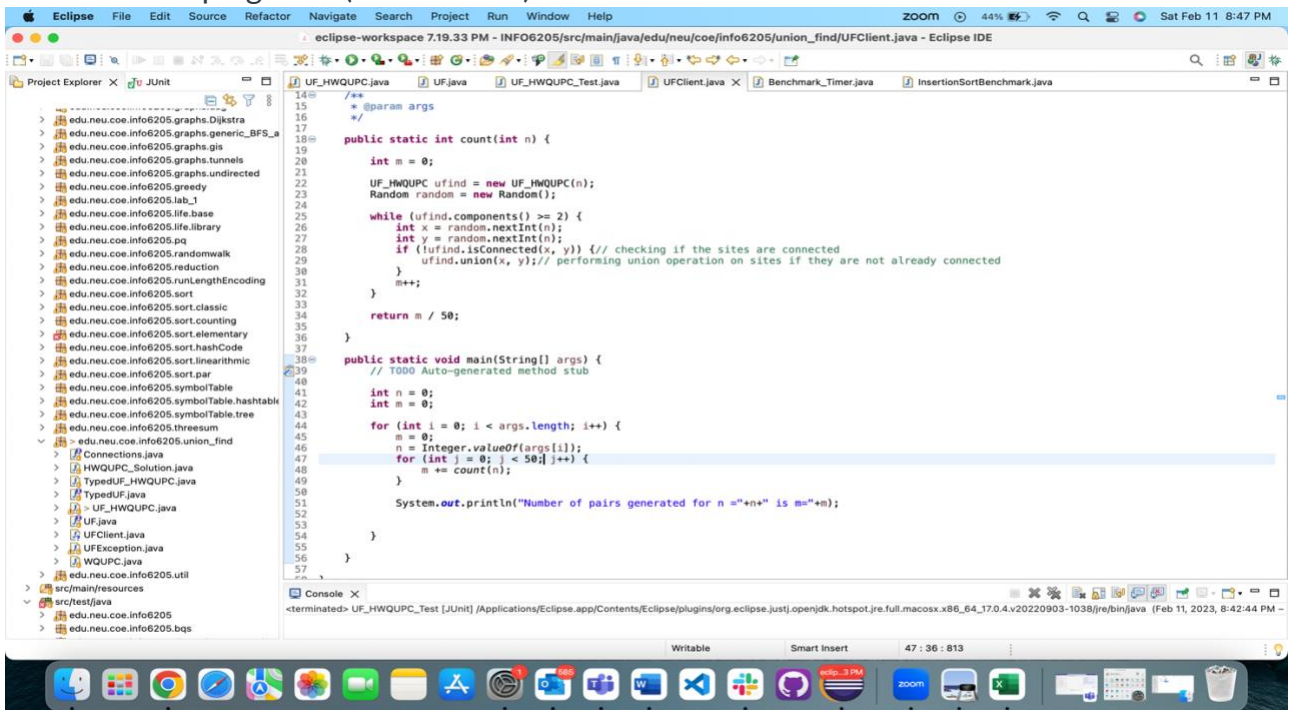
The screenshot shows the Eclipse IDE interface. The Project Explorer on the left lists a package structure under 'edu.neu.coe.info6205' containing various graph-related classes. The main editor displays the code for 'UF_HWQUPC.java'. The code implements a height-weighted Quick Union with Path Compression (HWQUPC). It includes a 'mergeComponents' method that updates parent pointers and heights, and a 'doPathCompression' method that compresses the path from a node to its root. The console at the bottom shows a successful execution of the 'UF_HWQUPC_Test [JUnit]'.

```
171 private final int[] height; // height[i] = height of subtree rooted at i
172 private int count; // number of components
173 private boolean pathCompression;
174
175 private void mergeComponents(int i, int j) {
176     // FIXME make shorter root point to taller one
177     if (i == j)
178         return;
179     if (height[i] < height[j]) {
180         updateParent(i, j);
181         updateHeight(j, i);
182     } else {
183         updateParent(j, i);
184         updateHeight(i, j);
185     }
186 }
187 // END
188
189 /**
190  * This implements the single-pass path-halving mechanism of path compression
191  */
192 private void doPathCompression(int i) {
193     // FIXME update parent to value of grandparent
194     while (i != parent[i]) {
195         parent[i] = parent[parent[i]];
196         i = parent[i];
197     }
198 }
199
200 }
201
202
```

b) Test Cases:



STEP 2: developing a UF ("union-find") client



Step 3: Relationship between the number of objects (n) and the number of pairs (m)

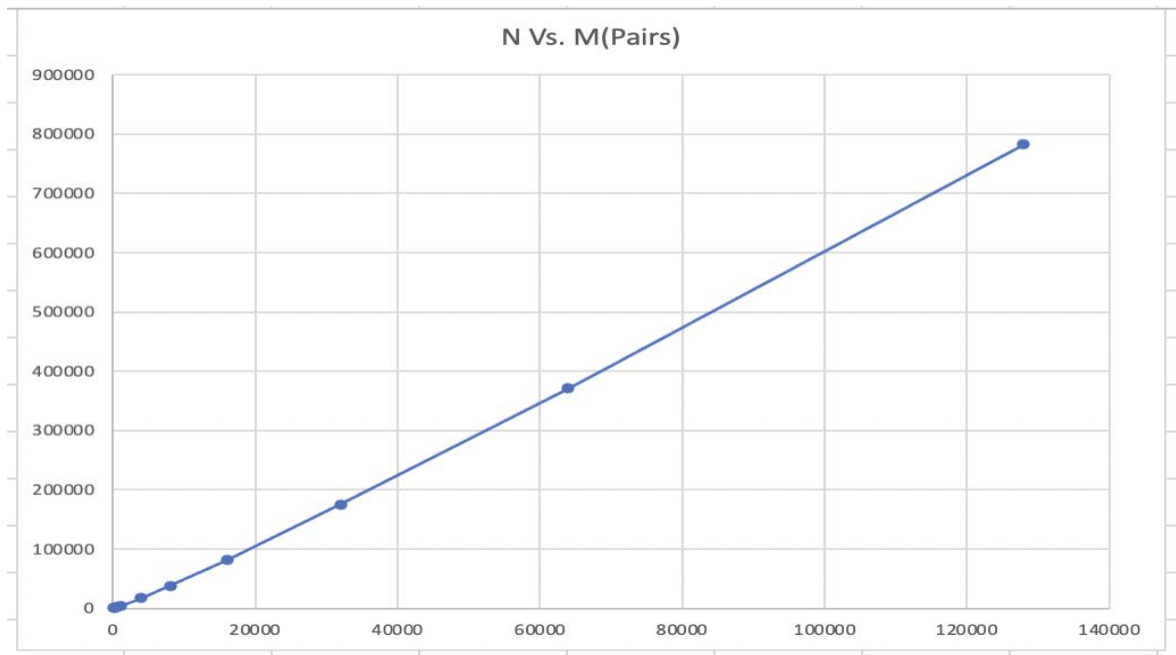
$$M=0.5*N*\ln(N)$$

The screenshot shows the Eclipse IDE with the following components:

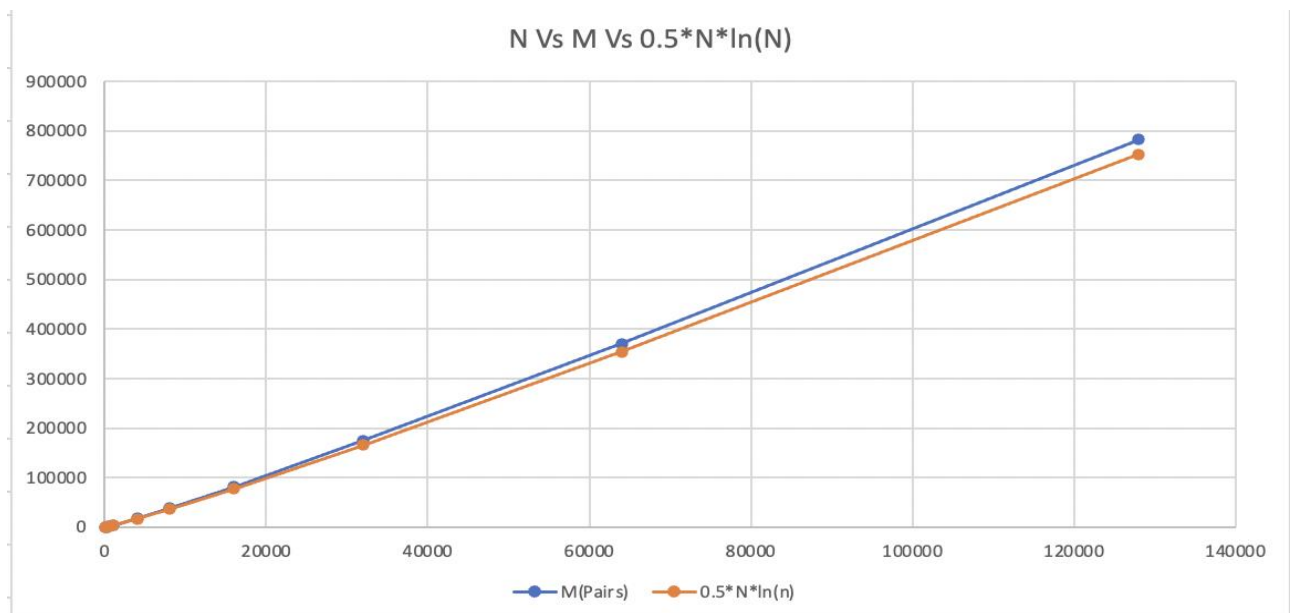
- Project Explorer:** A tree view on the left showing a project structure with various Java packages and classes, including `edu.neu.coe.info6205.union_find`.
- Editor:** The main window displays the source code of `UF_HWQUPC.java`. The code includes a `while` loop for finding components and a `main` method for testing.
- Console:** The bottom panel shows the output of the program, displaying the number of pairs generated for various values of `n` (100, 250, 500, 1000, 4000, 8000, 16000, 32000, 64000, 128000).

	A	B	C	D
1	N	M(Pairs)	N*ln(n)	0.5*N*ln(n)
2	100	240	460.5170186	230.2585093
3	250	750	1380.365229	690.1826147
4	500	1720	3107.304049	1553.652025
5	1000	3666	6907.755279	3453.877639
6	4000	17311	33176.19856	16588.09928
7	8000	37997	71897.57457	35948.78728
8	16000	81646	154885.504	77442.75201
9	32000	175295	331951.7178	165975.8589
10	64000	370732	708264.8552	354132.4276
11	128000	782193	1505252.549	752626.2747

Graph: (N vs M)



N Vs M vs $0.5 * N * \ln(N)$:



Relationship & Conclusion:

From the Above two graphs and values we can clearly see that M is almost equal to 0.5 times of $N \cdot \ln(N)$.

$$M = 0.5 \cdot N \cdot \ln(N)$$