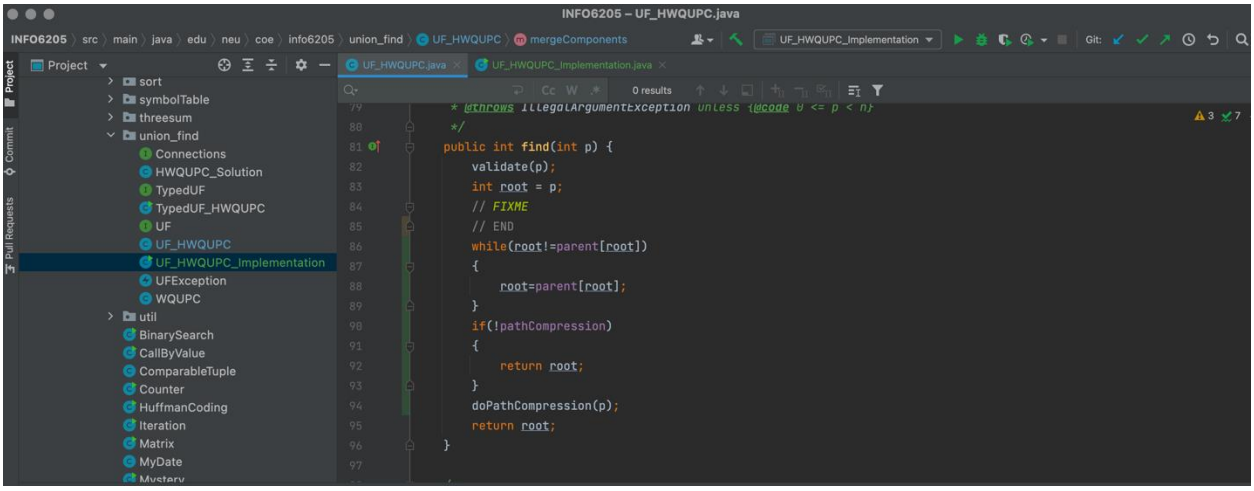
Program Structures & Algorithms
Spring 2022
Assignment No. 3
Sravya Ganda (002103774)

Task:

Expectation from the assignment is to implement a quick union with path compression by implementing three methods (find, Merge and doCompression) in UF_HWQUPC.java. There is a set of testcases that are to be passed. Secondly, we need to create random pairs of integers(m) between 0 to n-1 until our component becomes 1. Finally, a relationship must be defined between N and M.

Part 1:

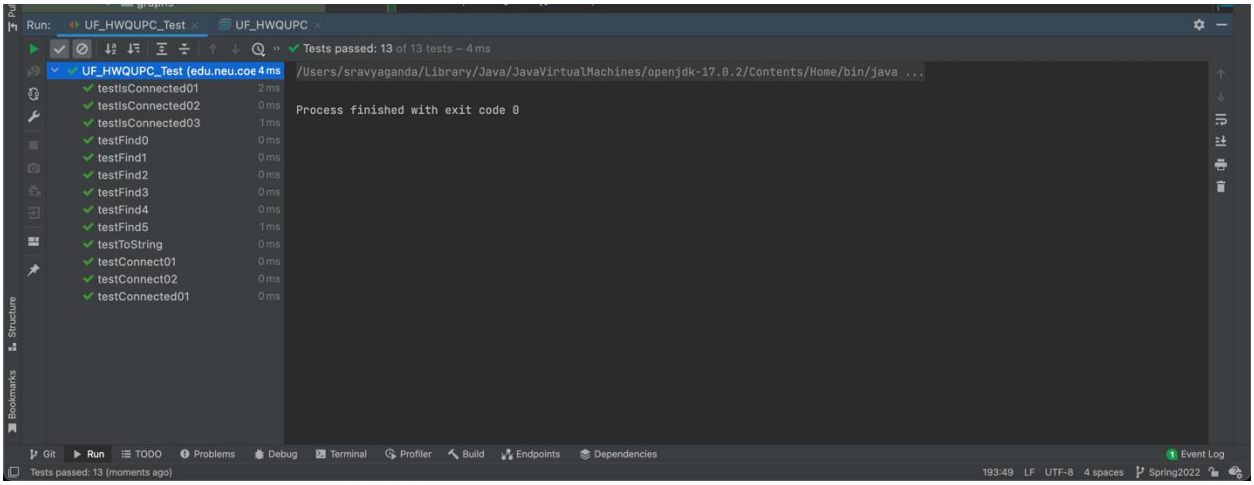
Included the below logic and test cases passed.



```java
    /**
     * This implements the single-pass path-halving mechanism of path compression
     */
    private void doPathCompression(int i) {
        // FIXME update parent to value of grandparent
        // END
        parent[i]=parent[parent[i]];
    }

}
```

```java
private void mergeComponents(int i, int j) {
    // FIXME make shorter root point to taller one
    // END
    int rootP=find(i);
    int rootQ=find(j);
    if(rootP == rootQ)
    {
        return;
    }
    if(height[rootP] < height[rootQ]) {
        parent[rootP] = rootQ;
        //height[rootP]++;
    }
    else if(height[rootP] == height[rootQ]){
     parent[rootQ]=rootP;
     height[rootP]++;
    }
    else
    {
        parent[rootQ] = rootP;
    }

}
```

Test Cases Screenshot



Task 2:

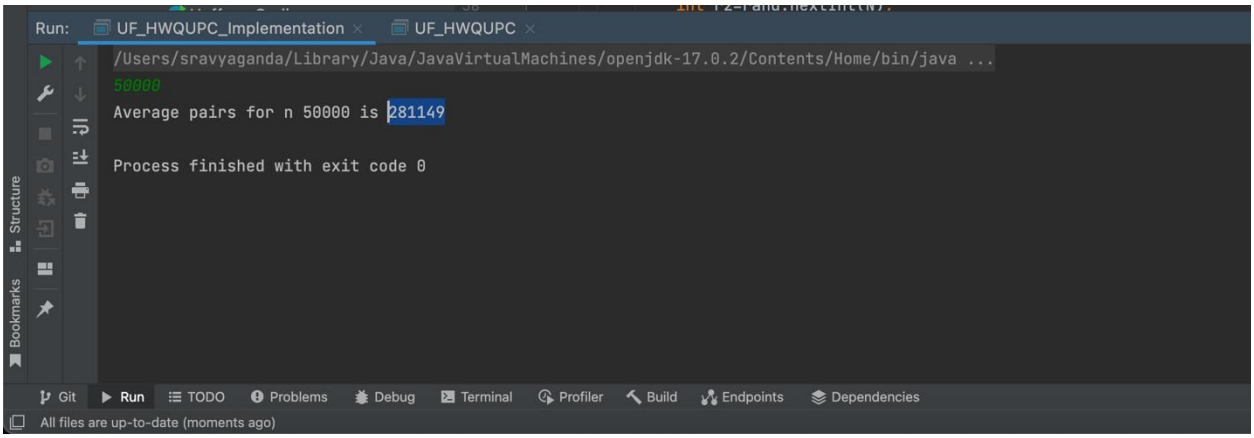
Created a new java class UF_HWQUPC_Implementation.java and implemented three methods (main, count, createPairs) the code screenshot is available below. Executed the random pairs method for 10 times and calculated the average M value for the accuracy.

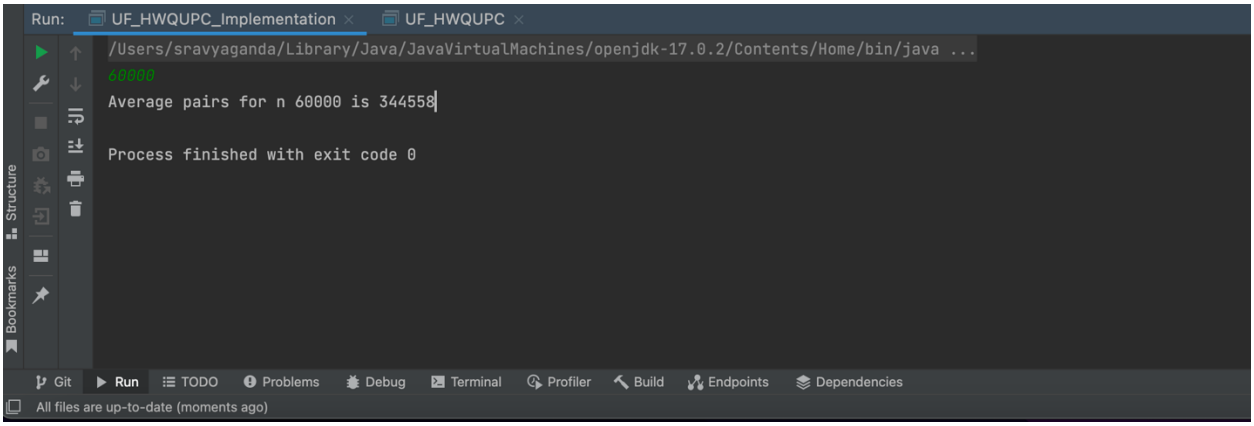
1. For N=50000 then M is approximately 281149
2. For N=60000 then M is approximately 344558
3. For N=70000 then M is approximately 412777
4. For N=80000 then M is approximately 462611

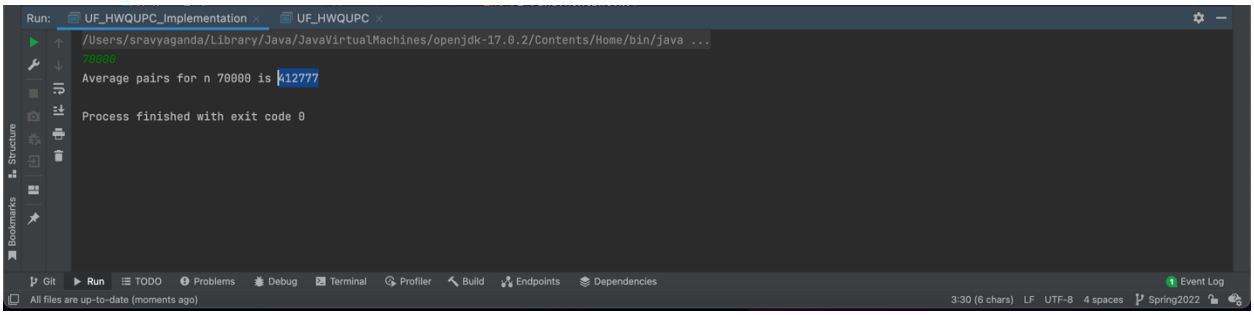
```java
public class UF_HWQUPC_Implementation {

    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();

        int avg=count(n);
        System.out.println("Average pairs for n "+ n + " is "+ avg );

    }

     public static int count(int n)
     {
         Queue<Integer> q=new LinkedList<Integer>();
         for(int i=1;i<=10;i++) {
             q.add(createPairs(n));
         }
         int avg=0;
         while(!q.isEmpty())
         {
             avg+=q.poll();
         }
         return avg/10;
     }
```

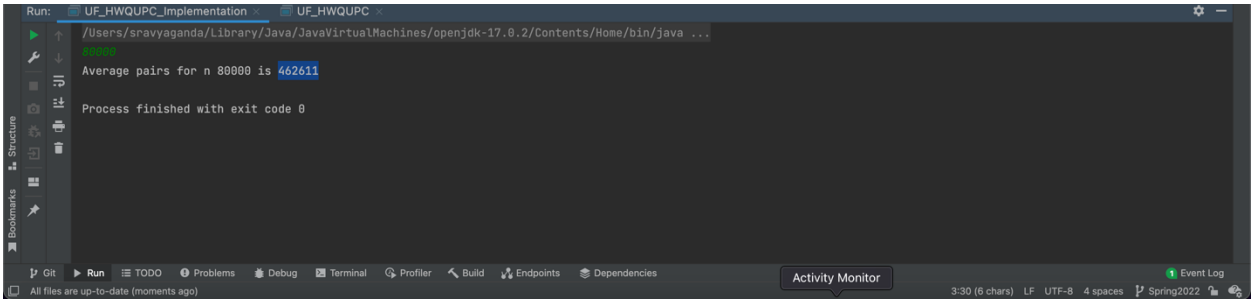
```java
    public static int createPairs(int N)
    {
        UF_HWQUPC uf=new UF_HWQUPC(N, pathCompression: true);
        Random rand=new Random();
        int m=0;
        while(uf.components()!=1)
        {
            int r1=rand.nextInt(N);
            int r2=rand.nextInt(N);
            uf.connect(r1,r2);
            m++;
        }
        return m;
    }
}
```

OUTPUT Screenshots









Task 3:

With different large values of N and from observations from the below table M is approximately N to the power of 1.16

N	N ^ 1.16	M
50000	282361	281149
60000	348863	344558
70000	417170	412777
80000	487062	462611

$$M \sim N \wedge 1.16$$

We can deduce this relationship to

$$M = (N * \ln(N))/2$$