

Random Walk

1. Intro

In this question, I was asked to use a program to imitate the behavior of the **Random Walk** (just walk like a drunken man). I need to start at (0,0), and move randomly towards four direction. Finally, calculate the Euclidean distance and deduce the relationship between it and steps.

```
private void move(int dx, int dy) {
    x += dx;
    y += dy;
    // END
}

public double distance() {
    return Math.sqrt(x * x + y * y);
    // END
}

private void randomWalk(int m) {
    //Random walk m steps
    while (m-- > 0) {
        randomMove();
    }
    // END
}
```

Code 1 Some methods which are required us to implement

2. Relationship

When **m** (the number of steps) is fixed, calculate **d** (Euclidean distance) and find the relationship between **m** and **d** as follows:

$$d = 3^{\log_{10} m} \quad (1)$$

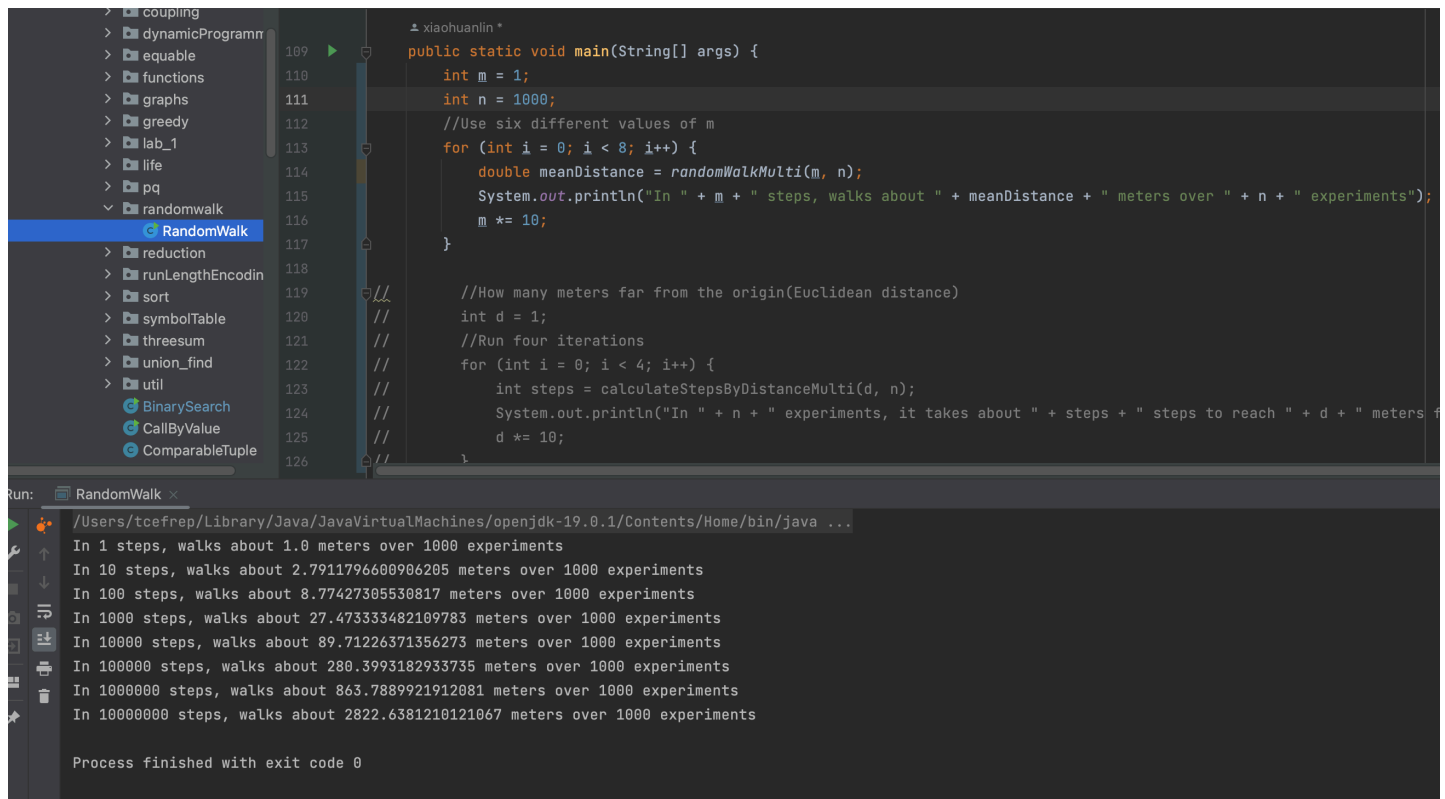
Although this assignment require us to find out after **m** steps, how far (**d**), and deduce the relationship between them.

However, I observed that when **d** is fixed, I got a **different relationship** between **d** (Euclidean distance) and **m**(steps):

$$m = d^2 \quad (2)$$

3. Evidence

3.1 After m steps, how far is this man from the origin



The screenshot shows an IDE with a project explorer on the left containing folders like 'coupling', 'dynamicProgram', 'equable', 'functions', 'graphs', 'greedy', 'lab_1', 'life', 'pq', 'randomwalk', 'reduction', 'runLengthEncodin', 'sort', 'symbolTable', 'threesum', 'union_find', 'util', 'BinarySearch', 'CallByValue', and 'ComparableTuple'. The 'RandomWalk' folder is selected. The main editor displays the following Java code:

```
public static void main(String[] args) {
    int m = 1;
    int n = 1000;
    //Use six different values of m
    for (int i = 0; i < 8; i++) {
        double meanDistance = randomWalkMulti(m, n);
        System.out.println("In " + m + " steps, walks about " + meanDistance + " meters over " + n + " experiments");
        m *= 10;
    }

    //How many meters far from the origin(Euclidean distance)
    int d = 1;
    //Run four iterations
    for (int i = 0; i < 4; i++) {
        int steps = calculateStepsByDistanceMulti(d, n);
        System.out.println("In " + n + " experiments, it takes about " + steps + " steps to reach " + d + " meters");
        d *= 10;
    }
}
```

The output window at the bottom shows the results of the program execution:

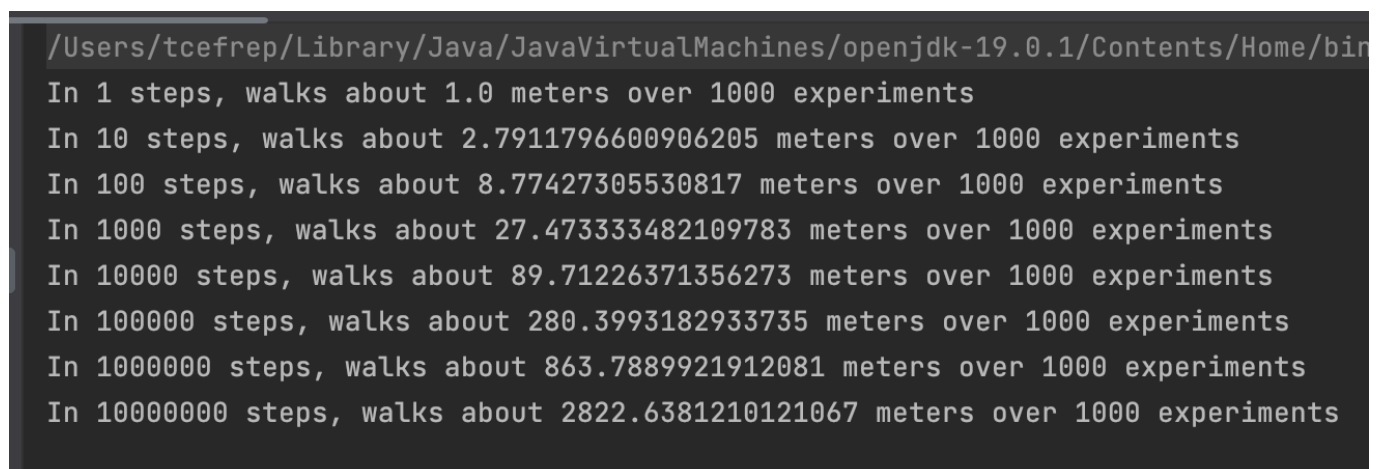
```
/Users/tcefrepp/Library/Java/JavaVirtualMachines/openjdk-19.0.1/Contents/Home/bin/java ...
In 1 steps, walks about 1.0 meters over 1000 experiments
In 10 steps, walks about 2.7911796600906205 meters over 1000 experiments
In 100 steps, walks about 8.77427305530817 meters over 1000 experiments
In 1000 steps, walks about 27.473333482109783 meters over 1000 experiments
In 10000 steps, walks about 89.71226371356273 meters over 1000 experiments
In 100000 steps, walks about 280.3993182933735 meters over 1000 experiments
In 1000000 steps, walks about 863.7889921912081 meters over 1000 experiments
In 10000000 steps, walks about 2822.6381210121067 meters over 1000 experiments

Process finished with exit code 0
```

Figure 1 Screenshot of operation results (m is fixed, calculate d from m)

I used eight different values of m (steps) to finish the test. My program run 1000 times in each value of m .

According to the main function, the initial value of the number of steps is 1. The number of steps per iteration will be multiplied by 10, and a total of 8 iterations have been made. The final value of the number of steps is 10,000,000. Since m is multiplied by 10 times each time, we can easily analyze the relationship between the m and the d according to the running results of the program



```
/Users/tcefrepp/Library/Java/JavaVirtualMachines/openjdk-19.0.1/Contents/Home/bin
In 1 steps, walks about 1.0 meters over 1000 experiments
In 10 steps, walks about 2.7911796600906205 meters over 1000 experiments
In 100 steps, walks about 8.77427305530817 meters over 1000 experiments
In 1000 steps, walks about 27.473333482109783 meters over 1000 experiments
In 10000 steps, walks about 89.71226371356273 meters over 1000 experiments
In 100000 steps, walks about 280.3993182933735 meters over 1000 experiments
In 1000000 steps, walks about 863.7889921912081 meters over 1000 experiments
In 10000000 steps, walks about 2822.6381210121067 meters over 1000 experiments
```

Figure 2 Screenshot of operation results (m is fixed, calculate d from m)

In the screenshot we can see, when the number of steps is multiplied by 10 times, the distance from the origin will also become about three times. Therefore, we can **deduce the relationship between d and m**.

$$d = 3^{\log_{10} m} \quad (3)$$

3.2 When this man is d meters from the origin, how many steps did he take

```

109 public static void main(String[] args) {
110     int m = 1;
111     int n = 1000;
112     //Use six different values of m
113     for (int i = 0; i < 8; i++) {
114         double meanDistance = randomWalkMulti(m, n);
115         System.out.println("In " + m + " steps, walks about " + meanDistance + " meters over " + n + " experiments");
116         m *= 10;
117     }
118
119     //How many meters far from the origin(Euclidean distance)
120     int d = 1;
121     //Run four iterations
122     for (int i = 0; i < 4; i++) {
123         int steps = calculateStepsByDistanceMulti(d, n);
124         System.out.println("In " + n + " experiments, it takes about " + steps + " steps to reach " + d + " meters far from the origin of coordinates");
125         d *= 10;
126     }
127 }

```

```

Run: RandomWalk
/Users/tcefre/Library/Java/JavaVirtualMachines/openjdk-19.0.1/Contents/Home/bin/java ...
In 1000 experiments, it takes about 1 steps to reach 1 meters far from the origin of coordinates
In 1000 experiments, it takes about 103 steps to reach 10 meters far from the origin of coordinates
In 1000 experiments, it takes about 9728 steps to reach 100 meters far from the origin of coordinates
In 1000 experiments, it takes about 950364 steps to reach 1000 meters far from the origin of coordinates

```

Figure 3 Screenshot of operation results (d is fixed, calculate m from d)

In the second test, I used four various values of **d** (Euclidean distance far from the origin).

The initial distance between the man and the origin is one meter. Also, distance will increase ten times after each iteration. Four iterations have been made.

```

/Users/tcefre/Library/Java/JavaVirtualMachines/openjdk-19.0.1/Contents/Home/bin/java ...
In 1000 experiments, it takes about 1 steps to reach 1 meters far from the origin of coordinates
In 1000 experiments, it takes about 103 steps to reach 10 meters far from the origin of coordinates
In 1000 experiments, it takes about 9728 steps to reach 100 meters far from the origin of coordinates
In 1000 experiments, it takes about 950364 steps to reach 1000 meters far from the origin of coordinates

```

Figure 4 Screenshot of operation results (d is fixed, calculate m from d)

In the screenshot you can see, the steps is almost the square of the distance.

$$m = d^2 \quad (4)$$

```

/**
 * Calculate how many steps it takes to walk about k meters(Euclidean distance)
 *
 * @param d how many meters should walk

```

```

    * @return how many steps it should take
    */
private int calculateStepsByDistance(int d) {
    int steps = 0;
    while (distance() < d) {
        randomMove();
        steps++;
    }
    return steps;
}

/**
 * Perform multiple random walk experiments, returning the mean steps when the man is
 * k meters far from the origin(Euclidean distance)
 *
 * @param d how many meters far from the origin
 * @param n the number of experiments to run
 * @return the mean steps to walk when the man is k meters far from the origin
 */
private static int calculateStepsByDistanceMulti(int d, int n) {
    int steps = 0;
    for (int i = 0; i < n; i++) {
        RandomWalk randomWalk = new RandomWalk();
        steps += randomWalk.calculateStepsByDistance(d);
    }
    return steps / n;
}

```

Code 2 Two new methods for testing

4. Unit Tests

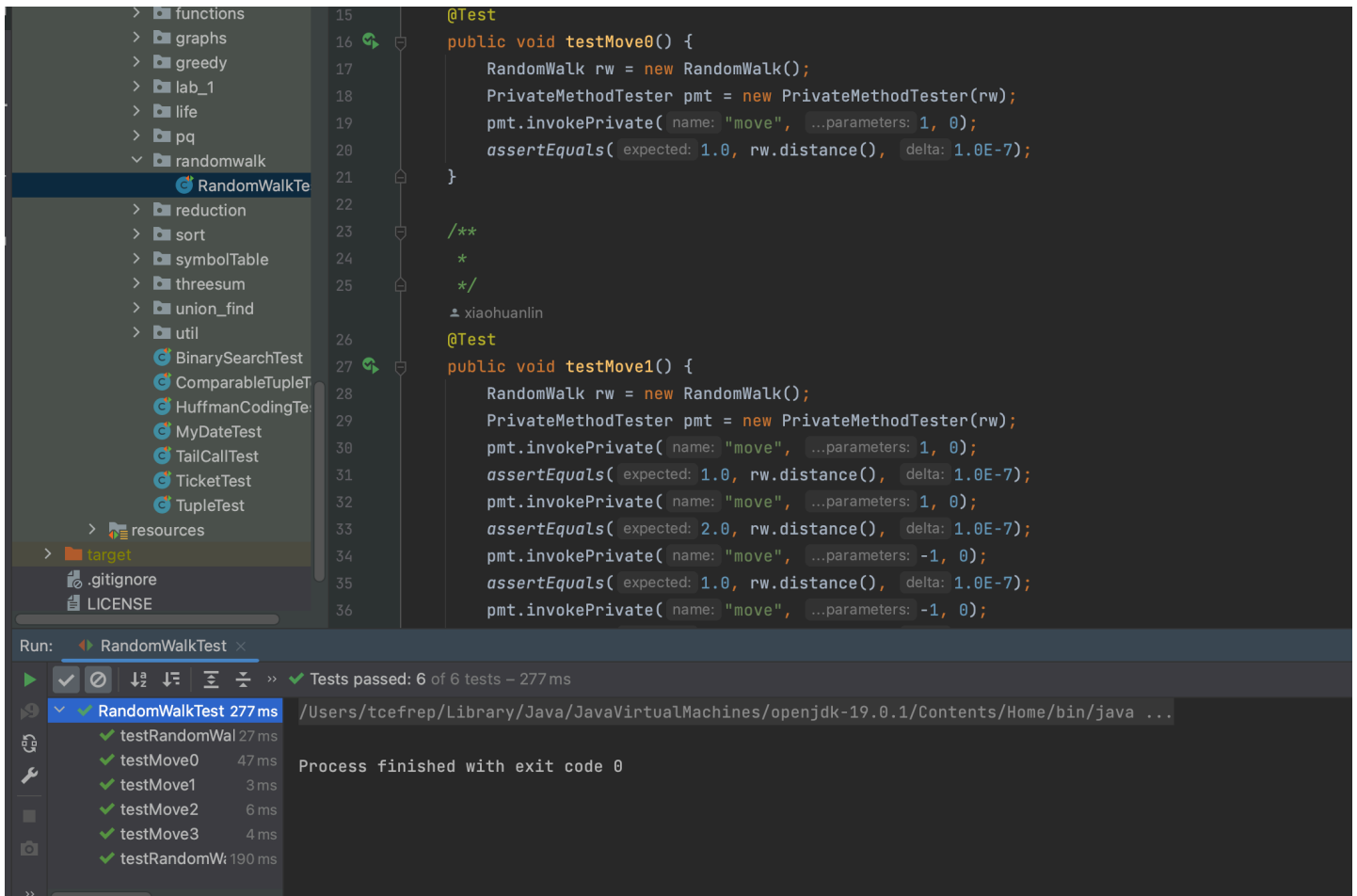


Figure 4 Screenshot of operation results (Unit Tests)

My program has successfully passed all test cases, which means the distance calculated by program operation is within the error range (diff smaller than $1.0E-7$).

5. Conclusion

Before the experiment, I take it for granted that drunk people walk randomly in four directions, and it is estimated that it will finally walk not far from the origin. I even think he may return to the original point.

However, after this experiment, I learned that the distance is proportional to the number of steps. After thinking for a while, the results of the experiment can also be explained. With each step taken by the drunken man, the previous base point will change.

- When the drunk take **the first step**, there is a **100%** possibility that he will far away from the origin
- When the drunk take **the second step**, there is a **75%** possibility that he will far away from the origin. Which means he only has **25%** of the probability will return to the origin. This is because when the drunk takes the first step, the base point is the origin. However, when the drunk takes the second step, the base point becomes the point which after the first step.