Program Structures & Algorithms
Spring 2022
Assignment No. 1

Name: Shashank Siripragada
NUID: 002193773

Task:

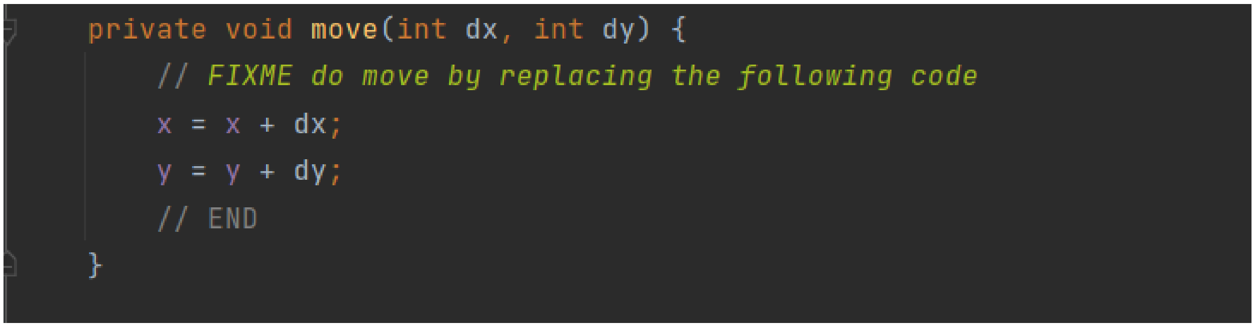
Imagine a drunken man who, starting out leaning against a lamp post in the middle of open space, takes a series of steps of the same length: 1 meter. The direction of these steps is randomly chosen from North, South, East, or West. **After m steps, how far (*d*), generally speaking, is the man from the lamp post?** Note that *d* is the Euclidean distance of the man from the lamp-post. It turns out that there is a relationship between *d* and *n* which is typically applicable to many different types of stochastic (randomized) experiments. Your task is to implement the code for the experiment and, most importantly, to **deduce the relationship**.

Experiment:

Let's run the Random Walk experiment with values of n = {50, 100, 500, 1000} times and for the number of steps ranging from 1<=m<=50.

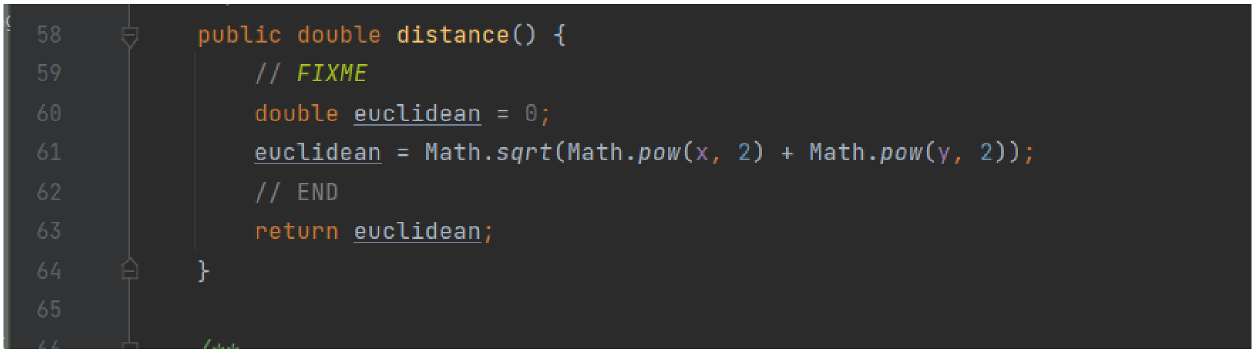
Code:
Following snippets indicate the filled code in RandomWalk.java. The code is modified to generate euclidean distance simultaneously for **n** = {50, 100, 500, 1000} experiments for **m steps** taken. The RandomWalk.java code is attached to the folder.

Move Code:



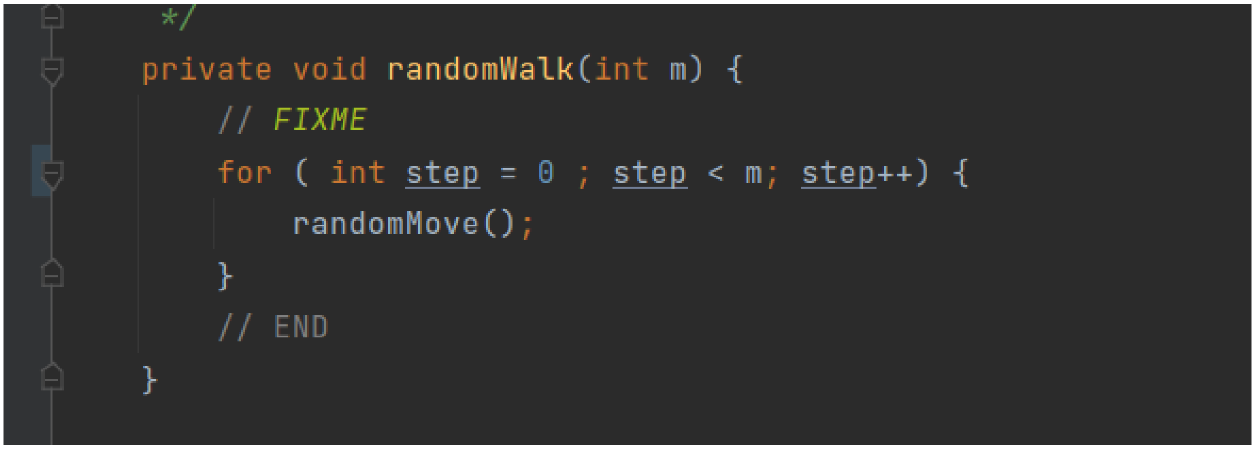
```java
    private void move(int dx, int dy) {
        // FIXME do move by replacing the following code
        x = x + dx;
        y = y + dy;
        // END
    }
```

Euclidean Distance:



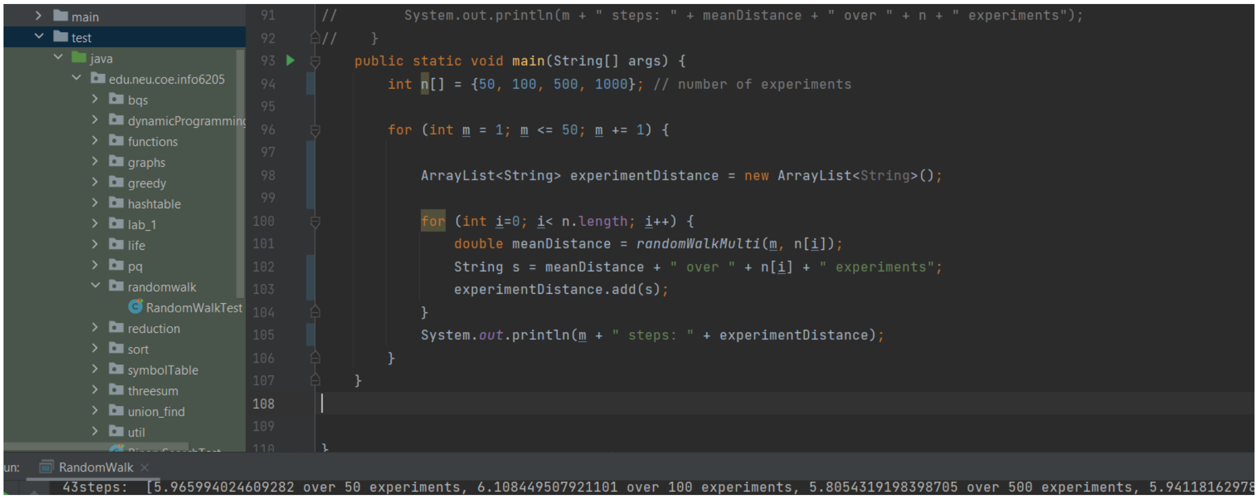
```java
    public double distance() {
        // FIXME
        double euclidean = 0;
        euclidean = Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
        // END
        return euclidean;
    }

```

Randomwalk:



```java
     */
    private void randomWalk(int m) {
        // FIXME
        for ( int step = 0 ; step < m; step++) {
            randomMove();
        }
        // END
    }
```

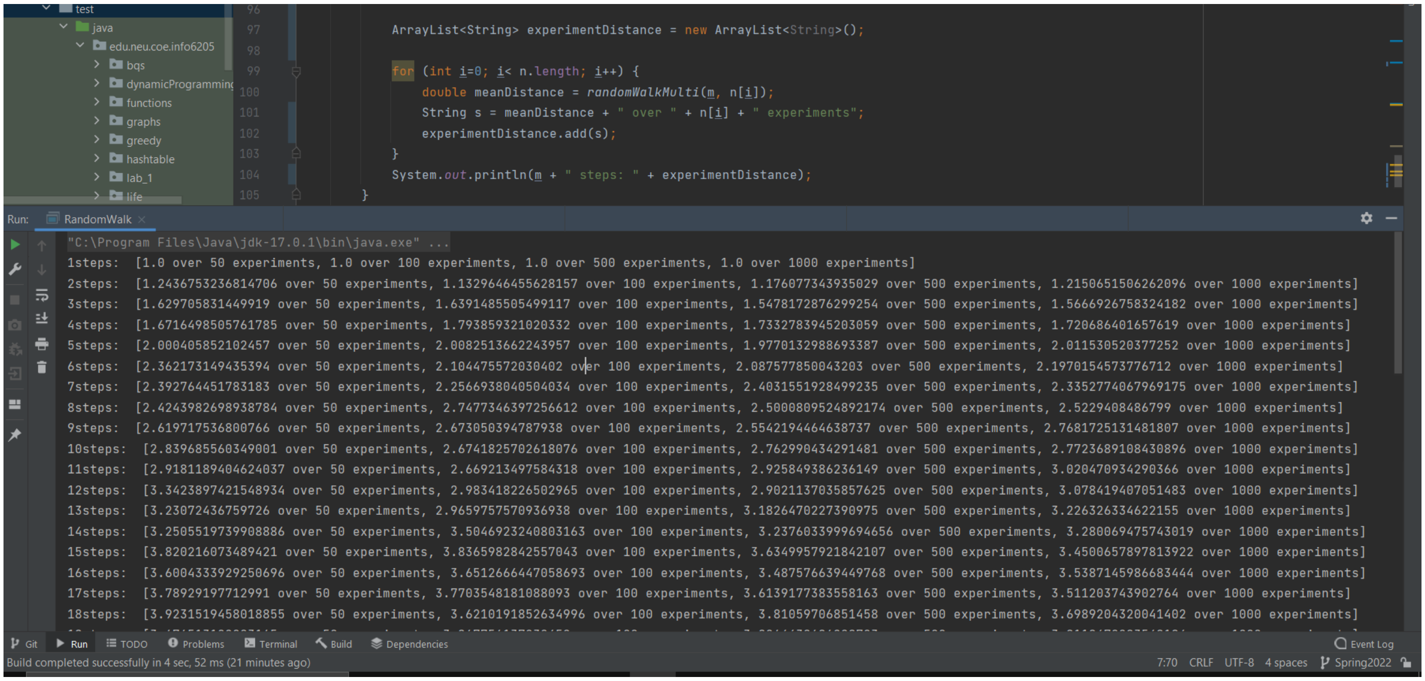
Modifying the main method for generation:



```java
//        System.out.println(m + " steps: " + meanDistance + " over " + n + " experiments");
//    }
    public static void main(String[] args) {
        int n[] = {50, 100, 500, 1000}; // number of experiments

        for (int m = 1; m <= 50; m += 1) {

            ArrayList<String> experimentDistance = new ArrayList<String>();

            for (int i=0; i< n.length; i++) {
                double meanDistance = randomWalkMulti(m, n[i]);
                String s = meanDistance + " over " + n[i] + " experiments";
                experimentDistance.add(s);
            }
            System.out.println(m + " steps: " + experimentDistance);
        }
    }


}
```

```
43steps:  [5.965994024609282 over 50 experiments, 6.108449507921101 over 100 experiments, 5.8054319198398705 over 500 experiments, 5.94118162978
```

Output Screenshot:



```java

        ArrayList<String> experimentDistance = new ArrayList<String>();

        for (int i=0; i< n.length; i++) {
            double meanDistance = randomWalkMulti(m, n[i]);
            String s = meanDistance + " over " + n[i] + " experiments";
            experimentDistance.add(s);
        }
        System.out.println(m + " steps: " + experimentDistance);
    }
```

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
1steps:  [1.0 over 50 experiments, 1.0 over 100 experiments, 1.0 over 500 experiments, 1.0 over 1000 experiments]
2steps:  [1.2436753236814706 over 50 experiments, 1.1329646455628157 over 100 experiments, 1.176077343935029 over 500 experiments, 1.2150651506262096 over 1000 experiments]
3steps:  [1.629705831449919 over 50 experiments, 1.6391485505499117 over 100 experiments, 1.5478172876299254 over 500 experiments, 1.5666926758324182 over 1000 experiments]
4steps:  [1.6716498505761785 over 50 experiments, 1.793859321020332 over 100 experiments, 1.7332783945203059 over 500 experiments, 1.720686401657619 over 1000 experiments]
5steps:  [2.000405852102457 over 50 experiments, 2.0082513662243957 over 100 experiments, 1.9770132988693387 over 500 experiments, 2.011530520377252 over 1000 experiments]
6steps:  [2.362173149435394 over 50 experiments, 2.104475572030402 over 100 experiments, 2.087577850043203 over 500 experiments, 2.1970154573776712 over 1000 experiments]
7steps:  [2.392764451783183 over 50 experiments, 2.2566938040504034 over 100 experiments, 2.4031551928499235 over 500 experiments, 2.3352774067969175 over 1000 experiments]
8steps:  [2.4243982698938784 over 50 experiments, 2.7477346397256612 over 100 experiments, 2.5000809524892174 over 500 experiments, 2.5229408486799 over 1000 experiments]
9steps:  [2.619717536800766 over 50 experiments, 2.673050394787938 over 100 experiments, 2.5542194464638737 over 500 experiments, 2.7681725131481807 over 1000 experiments]
10steps:  [2.839685560349001 over 50 experiments, 2.6741825702618076 over 100 experiments, 2.762990434291481 over 500 experiments, 2.7723689108430896 over 1000 experiments]
11steps:  [2.918118940462403 7 over 50 experiments, 2.669213497584318 over 100 experiments, 2.925849386236149 over 500 experiments, 3.020470934290366 over 1000 experiments]
12steps:  [3.3423897421548934 over 50 experiments, 2.983418226502965 over 100 experiments, 2.9021137035857625 over 500 experiments, 3.078419407051483 over 1000 experiments]
13steps:  [3.23072436759726 over 50 experiments, 2.9659757570936938 over 100 experiments, 3.182647022739097 5 over 500 experiments, 3.226326334622155 over 1000 experiments]
14steps:  [3.2505519739908886 over 50 experiments, 3.5046923240803163 over 100 experiments, 3.2376033999694656 over 500 experiments, 3.280069475743019 over 1000 experiments]
15steps:  [3.820216073489421 over 50 experiments, 3.8365982842557043 over 100 experiments, 3.6349957921842107 over 500 experiments, 3.4500657897813922 over 1000 experiments]
16steps:  [3.6004333929250696 over 50 experiments, 3.6512666447058693 over 100 experiments, 3.487576639449768 over 500 experiments, 3.5387145986683444 over 1000 experiments]
17steps:  [3.78929197712991 over 50 experiments, 3.7703548181088093 over 100 experiments, 3.6139177383558163 over 500 experiments, 3.511203743902764 over 1000 experiments]
18steps:  [3.9231519458018855 over 50 experiments, 3.6210191852634996 over 100 experiments, 3.81059706851458 over 500 experiments, 3.6989204320041402 over 1000 experiments]
```

Build completed successfully in 4 sec, 52 ms (21 minutes ago)

Relationship Conclusion:

From the below tabular data and plot we can conclude that mean distance (d) closely follows the square root of the number of steps (m).

$$d \approx \sqrt{m}$$

Where d = average distance & m = number of steps.

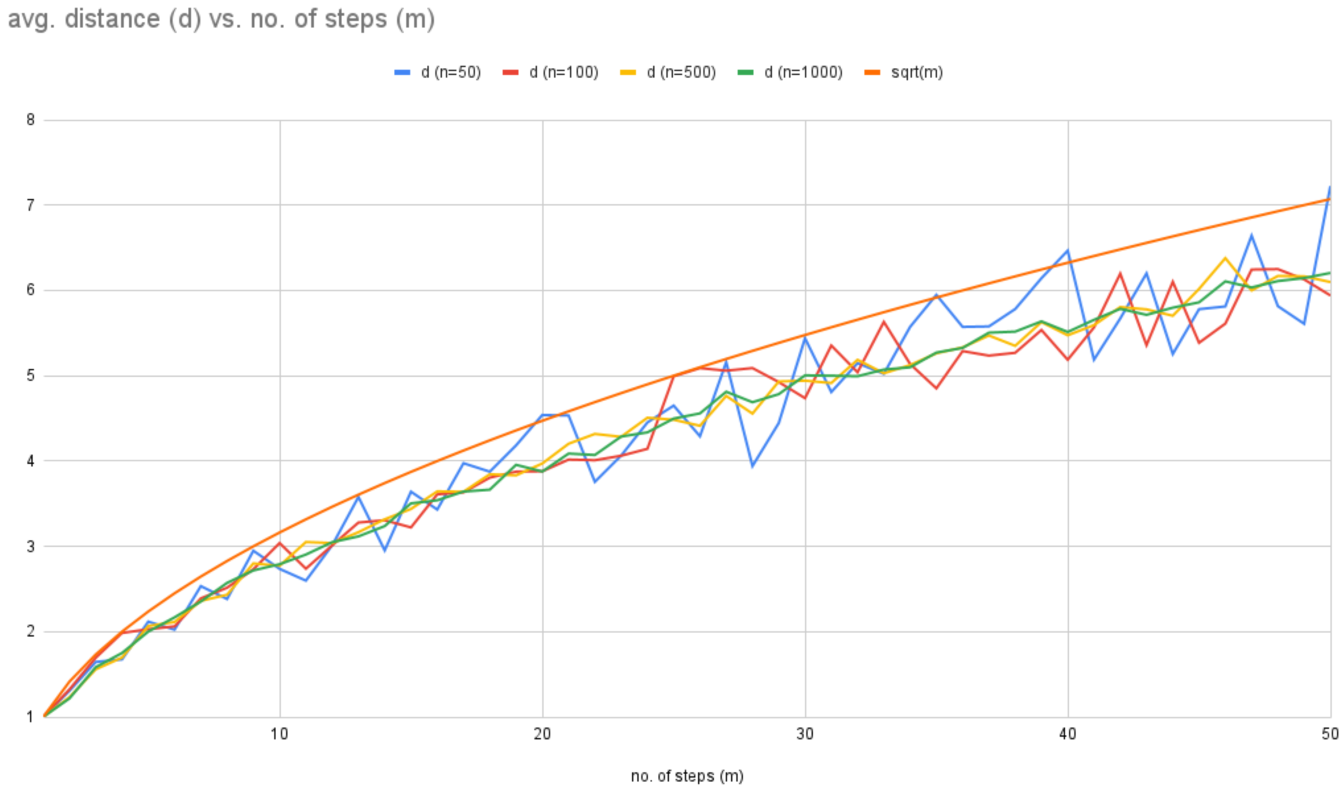
Evidence for Relationship b/w d & m:

Now to deduce the relationship between d and m we examine the graph of d vs. m for n number of experiments. The data can be found in attached Random Walk table csv file.

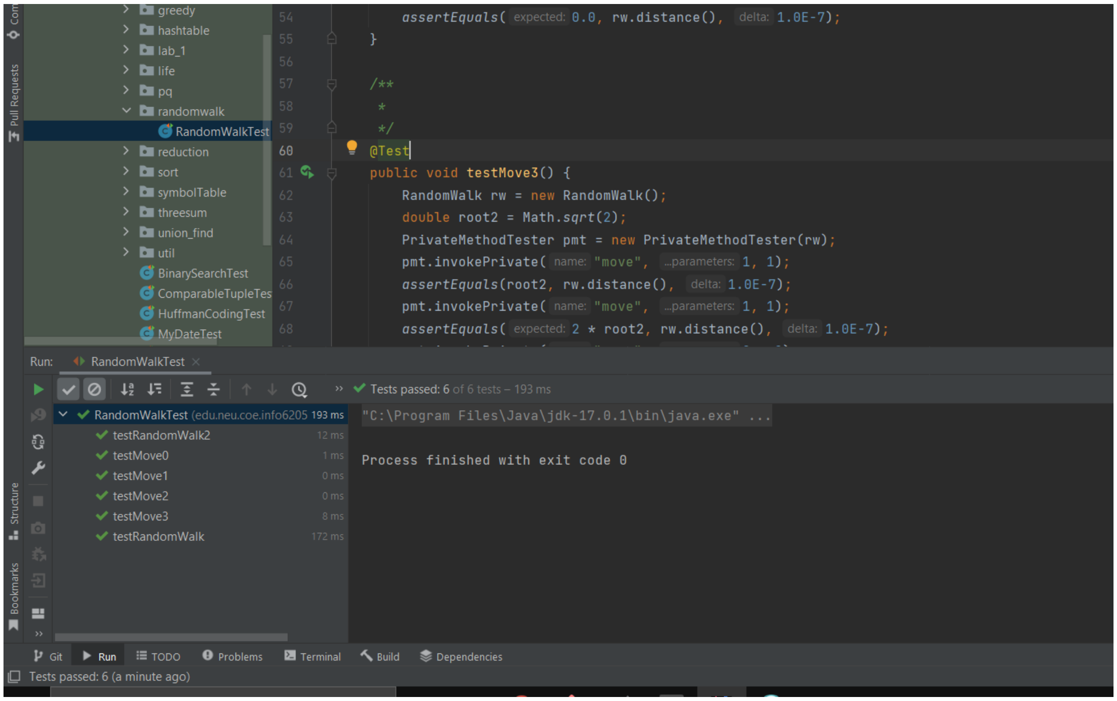
no. of steps (m)	d (n=50)	d (n=100)	d (n=500)	d (n=1000)	sqrt(m)
1	1	1	1	1	1
2	1.307106781	1.322670273	1.231592063	1.217248917	1.414213562
3	1.644984472	1.694427191	1.558289424	1.583525492	1.732050808
4	1.674678467	1.984569999	1.696459248	1.749476301	2
5	2.117147988	2.028294827	2.065991798	2.000726856	2.236067977
6	2.023641342	2.061440002	2.113142756	2.167333004	2.449489743
7	2.533463297	2.389735552	2.360438865	2.353335627	2.645751311
8	2.382608611	2.514734067	2.432400294	2.571554029	2.828427125
9	2.948457153	2.73141007	2.79990044	2.717703623	3
10	2.73570181	3.038444043	2.772273624	2.790401602	3.16227766
11	2.597960015	2.737949231	3.050621545	2.903541519	3.31662479
12	3.01210622	3.011198208	3.038283387	3.048981223	3.464101615
13	3.578701031	3.278863648	3.165854333	3.116417686	3.605551275
14	2.952347357	3.305935625	3.317095601	3.237516165	3.741657387
15	3.640617156	3.221627549	3.437953906	3.503655586	3.872983346
16	3.431298576	3.609532192	3.644013036	3.539901326	4
17	3.974249773	3.632748142	3.640092344	3.642083194	4.123105626
18	3.875170082	3.805292589	3.844858146	3.664354402	4.242640687
19	4.183424526	3.87530735	3.83225651	3.955719591	4.358898944
20	4.539826459	3.8799277	3.971406185	3.877007163	4.472135955
21	4.533570173	4.017009025	4.20254432	4.087338286	4.582575695
22	3.756154214	4.00940389	4.317765032	4.070931957	4.69041576
23	4.063678665	4.062066872	4.2841837	4.286642999	4.795831523
24	4.45213109	4.143078103	4.507286793	4.335986695	4.898979486
25	4.649789881	4.993137353	4.484304108	4.497846189	5
26	4.290205873	5.089260405	4.413281362	4.558293729	5.099019514
27	5.158955606	5.05915888	4.763046053	4.812262445	5.196152423
28	3.941183818	5.0895781	4.555847117	4.689369371	5.291502622
29	4.445218192	4.925727947	4.932624102	4.784011348	5.385164807
30	5.439629102	4.736769767	4.941556553	5.003860991	5.477225575

31	4.808882162	5.354074178	4.914804836	5.000210547	5.567764363
32	5.14737045	5.040886678	5.18631843	4.993865684	5.656854249
33	5.022718707	5.631096524	5.028759347	5.072179062	5.744562647
34	5.572552158	5.137596106	5.125027019	5.098517676	5.830951895
35	5.948097082	4.851903772	5.26239532	5.2713131	5.916079783
36	5.572303116	5.288634496	5.332589382	5.325929064	6
37	5.577770761	5.23554563	5.471507825	5.50475941	6.08276253
38	5.781438142	5.268633334	5.350428472	5.516945023	6.164414003
39	6.143074062	5.536040612	5.628807277	5.636957815	6.244997998
40	6.466588998	5.187183953	5.47401902	5.512837656	6.32455532
41	5.188360351	5.565993296	5.593071074	5.65518846	6.403124237
42	5.671463551	6.195953456	5.80492951	5.784029049	6.480740698
43	6.199788252	5.359020118	5.77772852	5.713653757	6.557438524
44	5.254389115	6.100540496	5.703016472	5.797630876	6.633249581
45	5.780147273	5.386310475	6.017927913	5.860381502	6.708203932
46	5.812127719	5.611069074	6.378874617	6.106182375	6.782329983
47	6.64135239	6.244605498	5.999477222	6.034924458	6.8556546
48	5.815549057	6.249604779	6.169666447	6.10966714	6.92820323
49	5.608962671	6.129122911	6.165258777	6.145046441	7
50	7.223032935	5.940582881	6.097792535	6.204889971	7.071067812

Relationship Plot:



Units test result: Let's make sure we pass all the test cases from RandomWalkTest.java. As we see all the test cases are successfully passed.



```java
        assertEquals( expected: 0.0, rw.distance(),  delta: 1.0E-7);
    }

    /**
     *
     */
    @Test
    public void testMove3() {
        RandomWalk rw = new RandomWalk();
        double root2 = Math.sqrt(2);
        PrivateMethodTester pmt = new PrivateMethodTester(rw);
        pmt.invokePrivate( name: "move",  ...parameters: 1, 1);
        assertEquals(root2, rw.distance(),  delta: 1.0E-7);
        pmt.invokePrivate( name: "move",  ...parameters: 1, 1);
        assertEquals( expected: 2 * root2, rw.distance(),  delta: 1.0E-7);
```

```
✓ Tests passed: 6 of 6 tests – 193 ms

✓ RandomWalkTest (edu.neu.coe.info6205 193 ms
    ✓ testRandomWalk2    12 ms
    ✓ testMove0           1 ms
    ✓ testMove1           0 ms
    ✓ testMove2           0 ms
    ✓ testMove3           8 ms
    ✓ testRandomWalk    172 ms

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...

Process finished with exit code 0

Tests passed: 6 (a minute ago)
```