Sneha Ravichandran(001096251)

Program Structures & Algorithms Fall 2021
Assignment 1

Task:

Random Walk

- Fill in missing code
- Deduce relationship between d and n
- Pass all unit test cases
- Show evidence

Relationship Conclusion:

We can say that the Euclidean distance(d) is almost close to the square root of the number of steps (n) taken by the drunk man plus a uncertain value(+ or -).

That is,

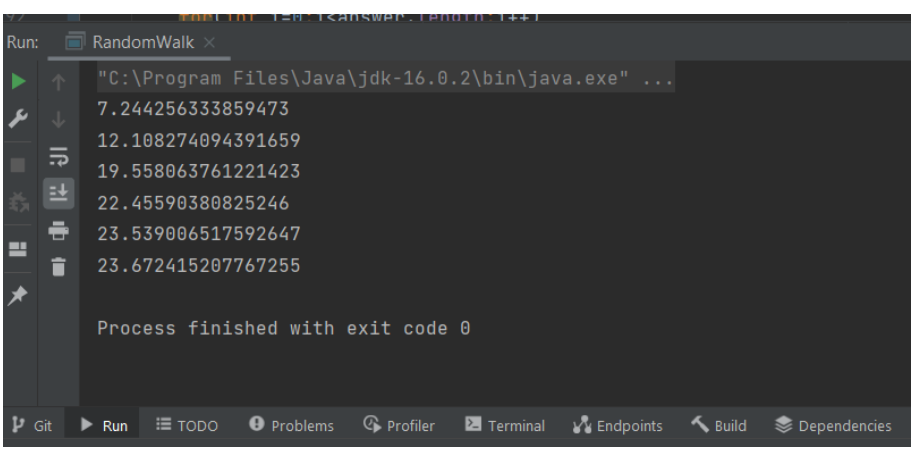
$$d \propto \sqrt{n} + \text{value(uncertain)}$$

Experiment was done with the following values:

int steps[]={100,200,400,600,800,1000}; Ran each experiment 30 times.

Evidence:

Output sample:

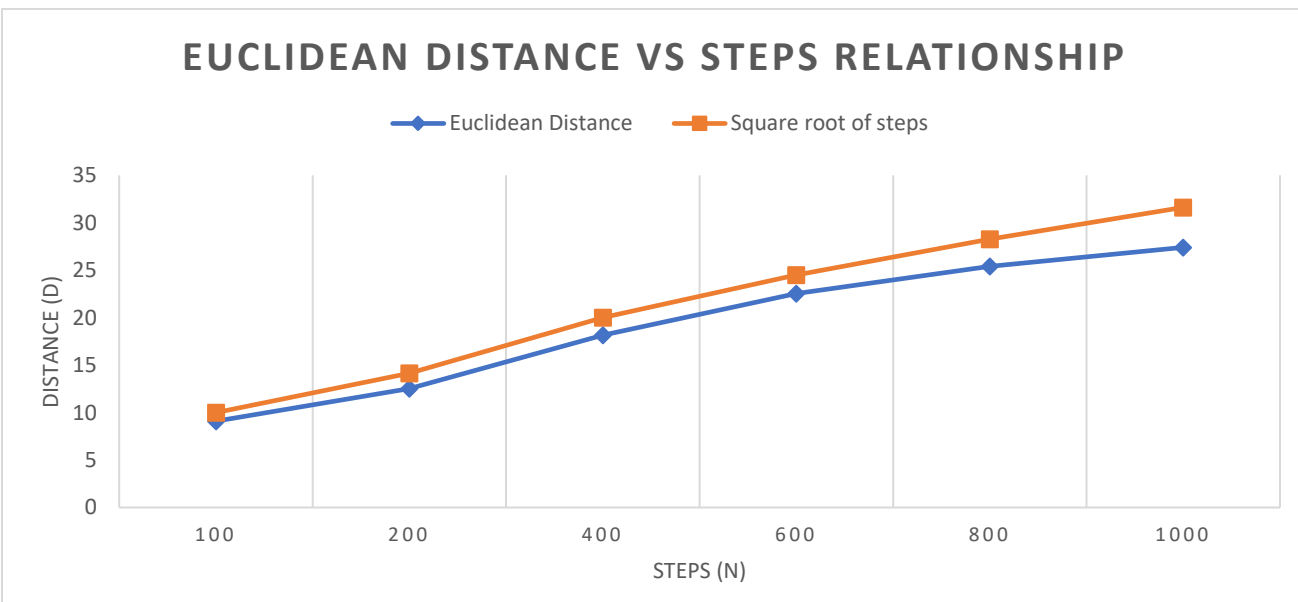


Excel Sheet calculations:

A	B	C	D	E	F	G
Steps	Cycles	Euclidean d 1	Euclidean d 2	Euclidean d 3	Euclidean d 4	Euclidean d 5
100	30	8.918310323	9.441978803	8.949566639	9.181482531	8.836942486
200	30	11.88109915	12.35238471	14.88003917	12.61527349	12.07817702
400	30	19.87346819	18.73325516	18.84011289	15.78470286	16.86023135
600	30	21.12688407	25.57088652	28.48382641	18.9816992	22.30344333
800	30	23.59718096	26.54697176	23.26549946	26.07943342	24.29937111
1000	30	28.12482647	29.05354474	27.42282362	26.01743991	28.58617632

H	I	J	K	L	M	N
Euclidean d 6	Euclidean d 7	Euclidean d 8	Euclidean d 9	Euclidean d 10	Average	Squareroot of steps
8.89682279	10.18220651	8.757969536	8.364381672	9.437338019	9.096699931	10
11.29711827	11.79442961	10.45880322	12.91152983	14.93856534	12.52074198	14.14213562
15.58283426	19.37096755	18.81898011	20.23891074	17.40153707	18.15050002	20
20.83786655	22.07702021	22.01515301	20.07596841	23.97668899	22.54494367	24.49489743
21.2617965	26.90145443	22.63405058	30.0476313	29.28825318	25.39216427	28.28427125
24.0761189	29.49715064	27.47156779	27.01341306	26.80983884	27.40729003	31.6227766

Graph Plot:



Code Changes:

Highlighted text are the code changes

```java
package edu.neu.coe.info6205.randomwalk;

import java.util.Random;

public class RandomWalk {

    private int x = 0;
    private int y = 0;

    private final Random random = new Random();

    /**
     * Private method to move the current position, that's to say the drunkard moves
     *
     * @param dx the distance he moves in the x direction
     * @param dy the distance he moves in the y direction
     */
    private void move(int dx, int dy) {
        // TO BE IMPLEMENTED
        x+=dx;
        y+=dy;
    }

    /**
     * Perform a random walk of m steps
     *
     * @param m the number of steps the drunkard takes
     */
```

```java
    */
    private void randomWalk(int m) {
        // TO BE IMPLEMENTED
        for(int i=0;i<m;i++)
        {
            randomMove();
        }
    }

    /**
     * Private method to generate a random move according to the rules of the situation.
     * That's to say, moves can be (+-1, 0) or (0, +-1).
     */
    private void randomMove() {
        boolean ns = random.nextBoolean();
        int step = random.nextBoolean() ? 1 : -1;
        move(ns ? step : 0, ns ? 0 : step);
    }

    /**
     * Method to compute the distance from the origin (the lamp-post where the drunkard starts) to his current position.
     *
     * @return the (Euclidean) distance from the origin to the current position.
     */
```

```java
public double distance() {
    // TO BE IMPLEMENTED
    double dist=Math.sqrt((x-0)*(x-0) + (y-0)*(y-0));
    return dist ;
}

/**
 * Perform multiple random walk experiments, returning the mean distance.
 *
 * @param m the number of steps for each experiment
 * @param n the number of experiments to run
 * @return the mean distance
 */
public static double randomWalkMulti(int m, int n) {
    double totalDistance = 0;
    for (int i = 0; i < n; i++) {
        RandomWalk walk = new RandomWalk();
        walk.randomWalk(m);
        totalDistance = totalDistance + walk.distance();
    }
    return totalDistance / n;
}
```

```java
public static void main(String[] args) {
    int steps[]={100,200,400,600,800,1000};
    double answer[]= new double[6];
    //if (args.length == 0)
        //throw new RuntimeException("Syntax: RandomWalk steps [experiments]");
    //int m = Integer.parseInt(args[0]);
    int n = 30;
    //if (args.length > 1) n = Integer.parseInt(args[1]);
    for(int i=0;i<steps.length;i++) {
        double meanDistance = randomWalkMulti(steps[i], n);
        //System.out.println(m + " steps: " + meanDistance + " over " + n + " experiments");
        answer[i]=meanDistance;
    }
    for(int i=0;i<answer.length;i++)
    {
        System.out.println(answer[i]);
    }
}

}
```

Unit Test Results Screenshot: