Harshal Jaiswal (1097734)

Programming Structure & Algorithms

Fall 2021

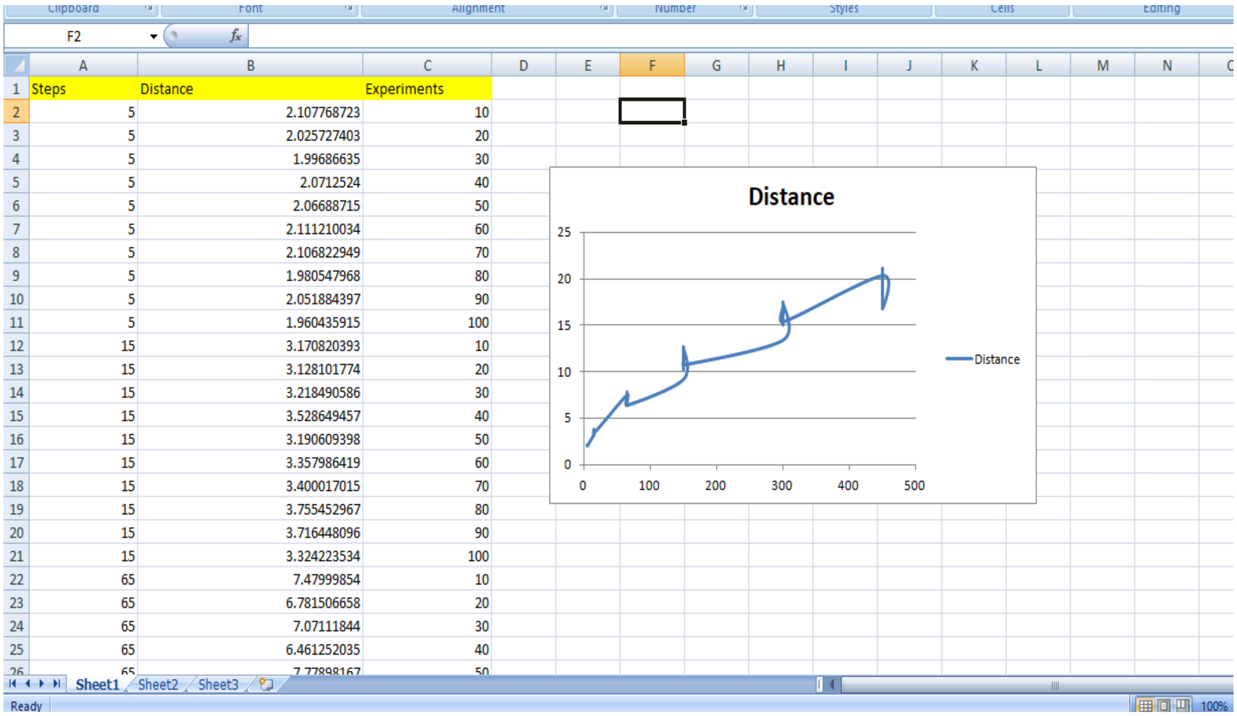
Assignment 1
Random Walk

- **Conclusion**: From the program experiment we have observed that the relationship between number of steps and distance is directly proportional.It can be seen that when we increase the value of steps there is a rise in the value of distance as well. This is observed under different numbers of experiments. Also, it is observed that the steps are closely equal to the square root of the distance.

$$n \propto d$$

$$d \cong \sqrt{n}$$

- **Evidence**

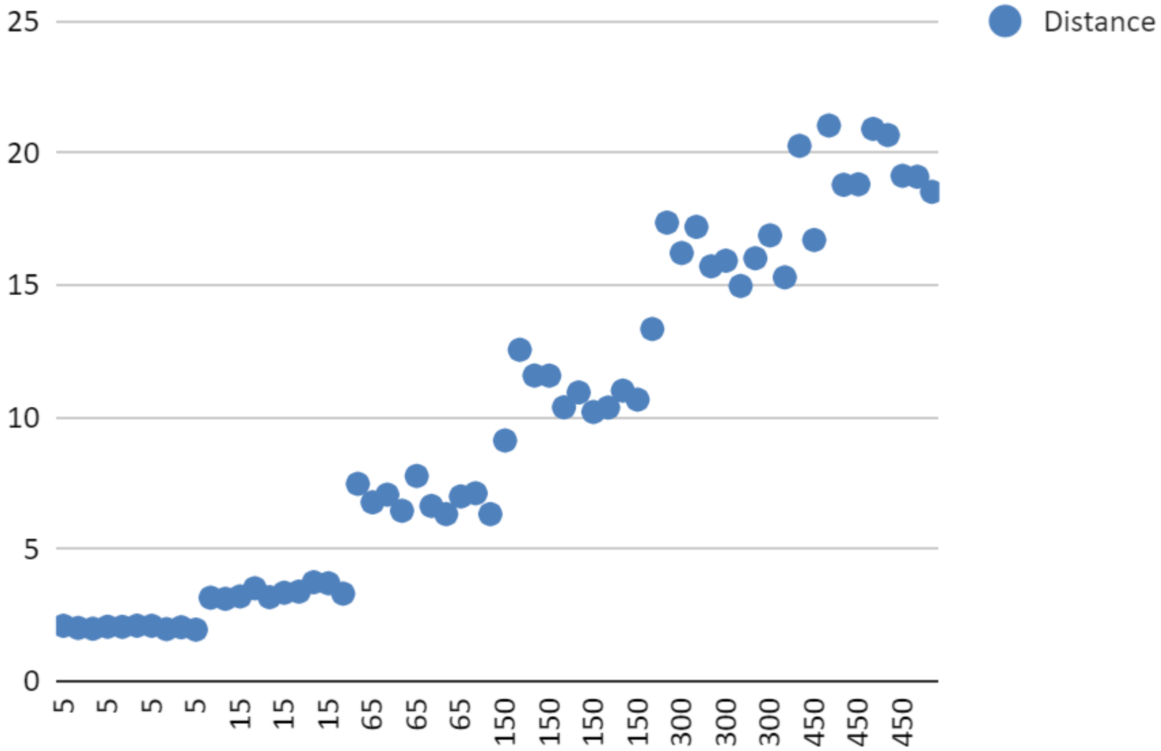


	A	B	C
1	Steps	Distance	Experiments
2	5	2.107768723	10
3	5	2.025727403	20
4	5	1.99686635	30
5	5	2.0712524	40
6	5	2.06688715	50
7	5	2.111210034	60
8	5	2.106822949	70
9	5	1.980547968	80
10	5	2.051884397	90
11	5	1.960435915	100
12	15	3.170820393	10
13	15	3.128101774	20
14	15	3.218490586	30
15	15	3.528649457	40
16	15	3.190609398	50
17	15	3.357986419	60
18	15	3.400017015	70
19	15	3.755452967	80
20	15	3.716448096	90
21	15	3.324223534	100
22	65	7.47999854	10
23	65	6.781506658	20
24	65	7.07111844	30
25	65	6.461252035	40
26	65	7.77898167	50

Observations

Steps	Distance	Experiments
5	2.107768723	10
5	2.025727403	20
5	1.99686635	30
5	2.0712524	40
5	2.06688715	50
5	2.111210034	60
5	2.106822949	70
5	1.980547968	80
5	2.051884397	90
5	1.960435915	100
15	3.170820393	10
15	3.128101774	20
15	3.218490586	30
15	3.528649457	40
15	3.190609398	50
15	3.357986419	60
15	3.400017015	70
15	3.755452967	80
15	3.716448096	90
15	3.324223534	100
65	7.47999854	10
65	6.781506658	20
65	7.07111844	30
65	6.461252035	40
65	7.77898167	50
65	6.643478206	60
65	6.334777023	70
65	7.002926162	80
65	7.12351745	90

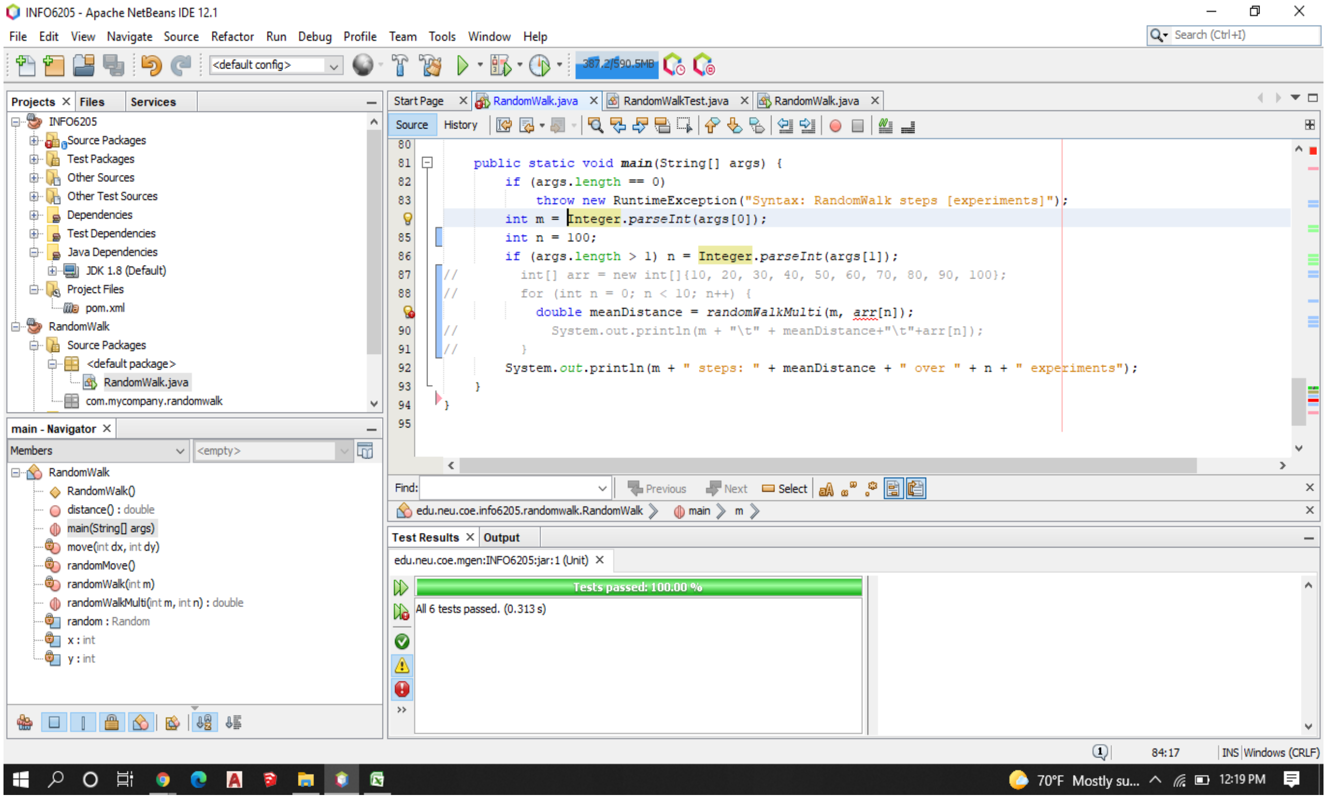
65	6.333883291	100
150	9.12132057	10
150	12.55422543	20
150	11.57892054	30
150	11.57590287	40
150	10.37817984	50
150	10.94007438	60
150	10.20087876	70
150	10.36833576	80
150	11.00974957	90
150	10.66822094	100
300	13.34151963	10
300	17.36778132	20
300	16.21820783	30
300	17.20783192	40
300	15.71476621	50
300	15.92688405	60
300	14.97038389	70
300	16.02055546	80
300	16.88512864	90
300	15.29731588	100
450	20.27448225	10
450	16.71013749	20
450	21.04407952	30
450	18.80006327	40
450	18.81750011	50
450	20.9148693	60
450	20.68171015	70
450	19.14035527	80
450	19.10487172	90
450	18.53440796	100



- **Code**

```
/*
 * Copyright (c) 2017. Phasmid Software
 */
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
//        System.out.println("dx: " +dx+", dy: "+dy);
        this.x = this.x + dx;
        this.y = this.y + dy;
    }

    /**
     * Perform a random walk of m steps
     *
     * @param m the number of steps the drunkard takes
     */
    private void randomWalk(int m) {
        // TO BE IMPLEMENTED
        for (int i = 0; i < m; i++) {
            this.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.
     */
    public double distance() {
        // TO BE IMPLEMENTED
//        System.out.println(" x: "+x+" y: "+y);
        return Math.sqrt((x * x) + (y * y));

    }

    /**
     * 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;
    }

    public static void main(String[] args) {
//        if (args.length == 0)
//            throw new RuntimeException("Syntax: RandomWalk steps
[experiments]");
        int m = 450;//Integer.parseInt(args[0]);
//        int n = 100;
//        if (args.length > 1) n = Integer.parseInt(args[1]);
        int[] arr = new int[]{10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
        for (int n = 0; n < 10; n++) {
            double meanDistance = randomWalkMulti(m, arr[n]);
            System.out.println(m + "\t" + meanDistance+"\t"+arr[n]);
        }
//        System.out.println(m + " steps: " + meanDistance + " over " + n + "
experiments");
    }
}
```

- ### ScreenShot of all test case passing



```

    public static void main(String[] args) {
        if (args.length == 0)
            throw new RuntimeException("Syntax: RandomWalk steps [experiments]");
        int m = Integer.parseInt(args[0]);
        int n = 100;
        if (args.length > 1) n = Integer.parseInt(args[1]);
//        int[] arr = new int[]{10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
//        for (int n = 0; n < 10; n++) {
            double meanDistance = randomWalkMulti(m, arr[n]);
//            System.out.println(m + "\t" + meanDistance+"\t"+arr[n]);
//        }
        System.out.println(m + " steps: " + meanDistance + " over " + n + " experiments");
    }
}

```

edu.neu.coe.mgen:INFO6205:jar:1 (Unit)

Tests passed: 100.00 %

All 6 tests passed. (0.313 s)