Ebenezer Ajay Williams Vincent Sankey Raj (002166250)

Program Structures & Algorithms

Fall 2021

Assignment No. 1

- **Task (List of tasks performed in the Assignment)**
  1) **Code modified in the main() method to perform multiple experiments.**
  2) **move(), randomWalk() and distance() methods are implemented.**
  3) **Input modified to process multiple values.**
  4) **Console output is modified to print a 3 lists with the values of d,**$\sqrt{M}$ **and M.**
  5) **Graph is plotted to find the relation** $d \propto \sqrt{M}$**.**
  6) **Mathematical proof and derivation is shown.**
  7) **Unit tests have been run and are successful.**

- **Relationship Conclusion:** $d \propto \sqrt{M}$

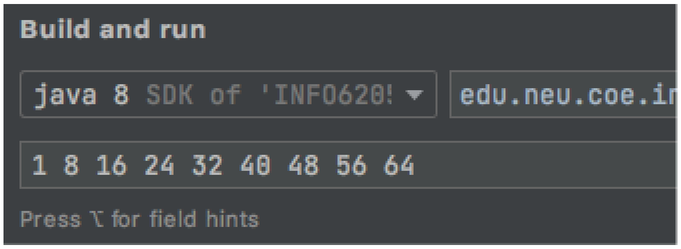
  Hence $d = k\sqrt{M}$ **,**

  Where k is a constant

- **Evidence to support the conclusion:**

1. **Output (Snapshot of Code output in the terminal)**

   Modified Input:



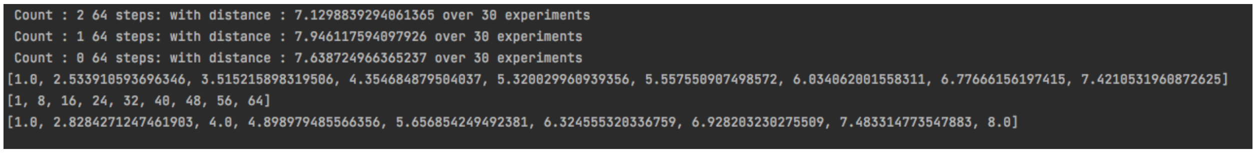
Build and run

java 8 SDK of 'INFO620! ▾ | edu.neu.coe.i

```
1 8 16 24 32 40 48 56 64
```

Press ⌥ for field hints

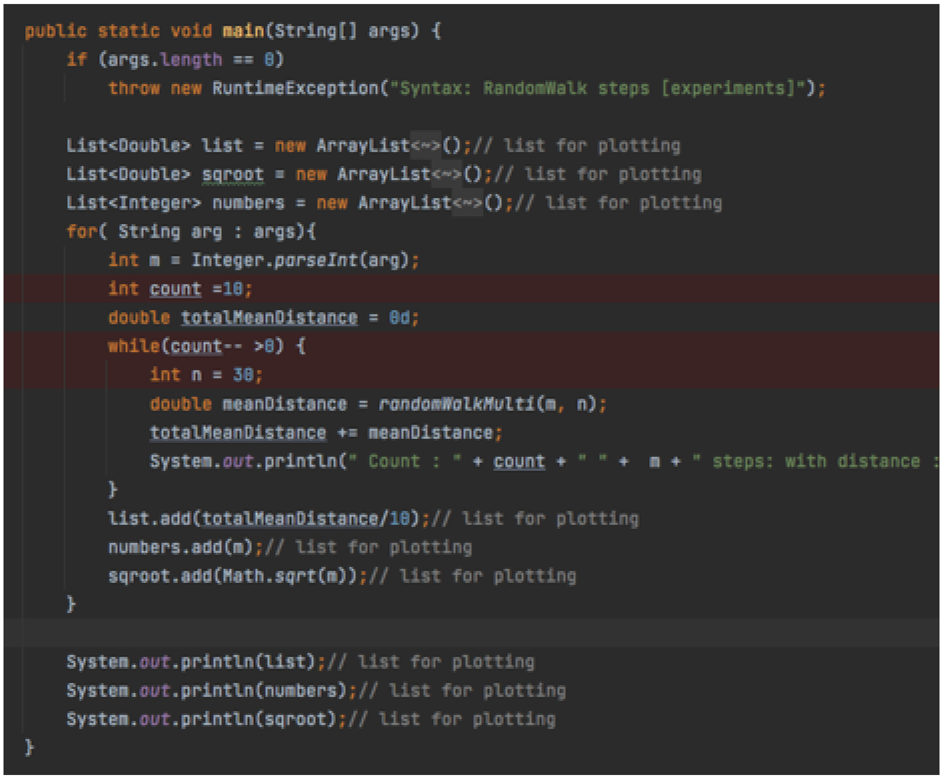
Console Output printing a list containing the average distance (d) , the number of steps(m) and the square root of the number of steps ($\sqrt{m}$)



```
 Count : 2 64 steps: with distance : 7.1298839294061365 over 30 experiments
 Count : 1 64 steps: with distance : 7.946117594097926 over 30 experiments
 Count : 0 64 steps: with distance : 7.638724966365237 over 30 experiments
[1.0, 2.533910593696346, 3.515215898319506, 4.354684879504037, 5.320029960939356, 5.557550907498572, 6.034062001558311, 6.77666156197415, 7.4210531960872625]
[1, 8, 16, 24, 32, 40, 48, 56, 64]
[1.0, 2.8284271247461903, 4.0, 4.898979485566356, 5.656854249492381, 6.324555320336759, 6.928203230275509, 7.483314773547883, 8.0]
```

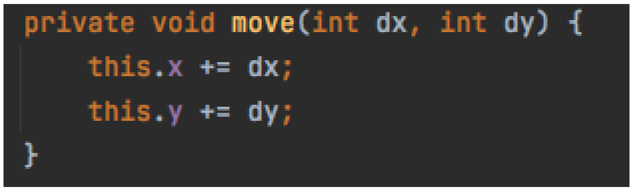
Code modified to get the required output on the terminal

- Main method



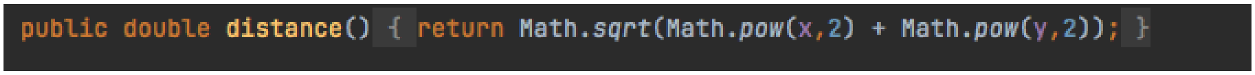
```
public static void main(String[] args) {
    if (args.length == 0)
        throw new RuntimeException("Syntax: RandomWalk steps [experiments]");

    List<Double> list = new ArrayList<~>();// list for plotting
    List<Double> sqroot = new ArrayList<~>();// list for plotting
    List<Integer> numbers = new ArrayList<~>();// list for plotting
    for( String arg : args){
        int m = Integer.parseInt(arg);
        int count =10;
        double totalMeanDistance = 0d;
        while(count-- >0) {
            int n = 30;
            double meanDistance = randomWalkMulti(m, n);
            totalMeanDistance += meanDistance;
            System.out.println(" Count : " + count + " " +  m + " steps: with distance :
        }
        list.add(totalMeanDistance/10);// list for plotting
        numbers.add(m);// list for plotting
        sqroot.add(Math.sqrt(m));// list for plotting
    }

    System.out.println(list);// list for plotting
    System.out.println(numbers);// list for plotting
    System.out.println(sqroot);// list for plotting
}
```

- move() method



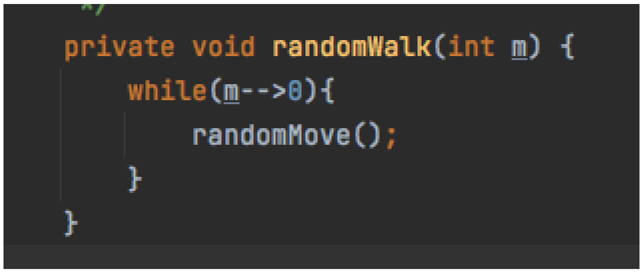
```
private void move(int dx, int dy) {
    this.x += dx;
    this.y += dy;
}
```

- **distance() method (computing euclidean distance)**



```
public double distance() { return Math.sqrt(Math.pow(x,2) + Math.pow(y,2)); }
```

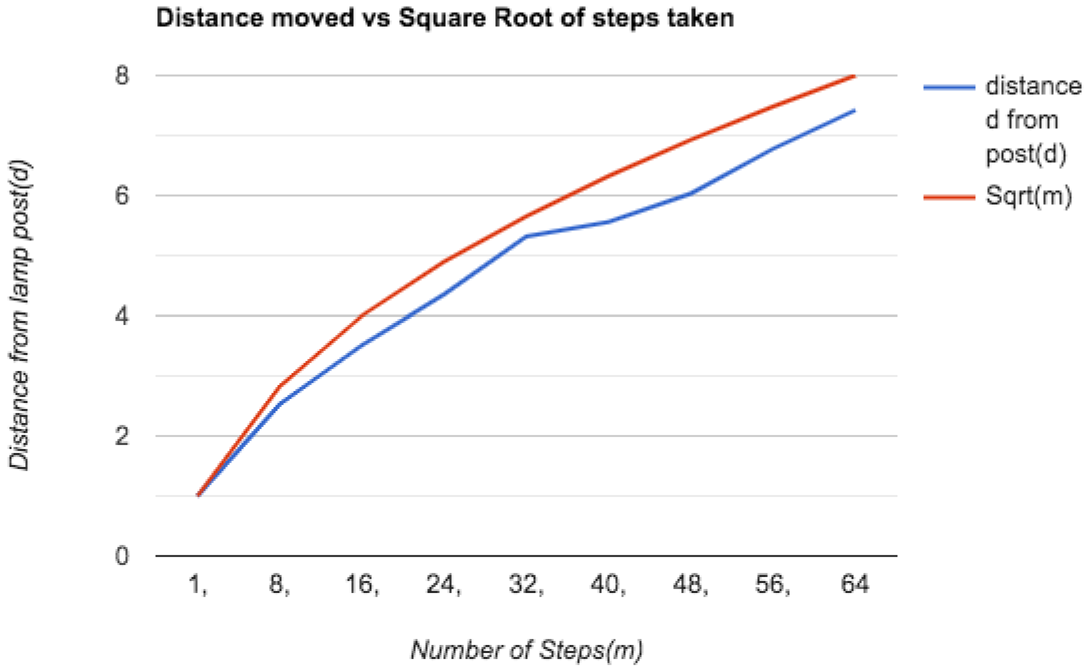
- **randomWalk() method (calls the randomMove() method m number of times)**



```
  */
    private void randomWalk(int m) {
        while(m-->0){
            randomMove();
        }
    }
```

2. **Graphical Representation**

Line graph showing the relationship between the root of m and the distance moved.



We can see that $d \propto \sqrt{m}$

where d -> average distance traveled over 30 experiments

m -> number of steps taken

Tabulated values:

M (steps taken)	Sqrt(M)	d(distance)
1	1	1
8	2.828427125	2.533910594
16	4	3.515215898
24	4.898979486	4.35468488
32	5.656854249	5.320029961
40	6.32455532	5.557550907
48	6.92820323	6.034062002
56	7.483314774	6.776661562
64	8	7.421053196

Mathematical proof.

Considering the movement of the man in only one dimension (can move only east (+1) or west(-1)), we see that the probability of having a distance of +1 or -1 for each step is ½.

$P(d = 1 \text{ or } d = -1) = ½$ --- (1)

Now, $d = d_1 + d_2 + d_3 + d_4 \ldots\ldots d_n$

Where d_n is the distance traveled in the n^{th} step.

Upon squaring we get

$d^2 = (d_1 + d_2 + d_3 + d_4 \ldots\ldots d_n) * (d_1 + d_2 + d_3 + d_4 \ldots\ldots d_n)$

Solving this we get

$d^2 = (d_1^2 + d_2^2 + d_3^2 + d_4^2 \ldots\ldots d_4^2) + 2\,(d_1d_2 + d_1d_3 + d_1d_4 \ldots\ldots + d_2d_3 + d_2d_4 \ldots\ldots d_{n-1}d_n)$

From (1) we see that the probability of moving +1 distance and -1 distance is the same.

Hence considering only d1 and d2 , we can conclude that

For all values of d1 and d2 (+1 & -1) , d1 *d2 $\in \{1, -1\}$**, hence d1 * d2 =0 (same probability and both events cannot happen at once)**

$d_n^2 = 1$ **(square of -1 and 1 is equal to 1)**

Thus,

$d^2 = (1+1+1+1+1+\ldots\ldots) + 2\,(0+0+0+0+0)$

$d^2 = M + 2 * 0$

$d^2 = M$

$d = \sqrt{M}$ **--- (2)**

where M is the number of steps taken

Considering the y axis as well (2 dimension), we can conclude that if x steps are taken along the x axis (east or west) then M-x steps will be taken along the y axis. Then the distance on the x axis and y axis will be $\sqrt{x}$ **and** $\sqrt{M - x}$ **respectively.**

Explanation: This is because if we move along one axis , we cannot move along the other axis.

Now we know distance from the origin (0,0) is:

$d = \sqrt{x^2 + y^2}$ **--- (3)**

Substituting x = $\sqrt{x}$ and y = $\sqrt{M - x}$

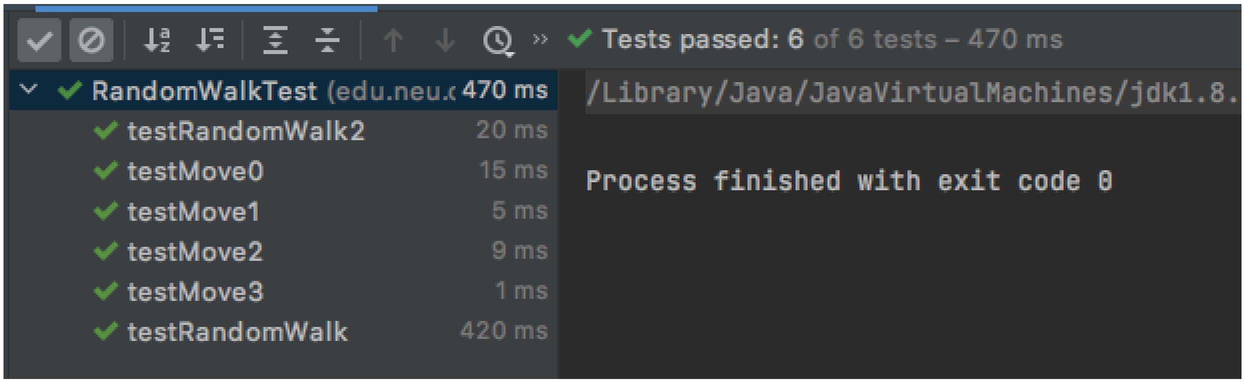
$$d = \sqrt{(\sqrt{x})^2 + (\sqrt{M - x})^2}$$

$$d = \sqrt{x + M - x}$$

$$d = \sqrt{M} \quad \text{---} \ \textbf{(4)}$$

Hence Proved in (4).

- **Unit tests result:** **(Snapshot of successful unit test run)**



```
✓ Tests passed: 6 of 6 tests – 470 ms

✓ RandomWalkTest (edu.neu.c 470 ms
    ✓ testRandomWalk2   20 ms
    ✓ testMove0         15 ms
    ✓ testMove1          5 ms
    ✓ testMove2          9 ms
    ✓ testMove3          1 ms
    ✓ testRandomWalk   420 ms

/Library/Java/JavaVirtualMachines/jdk1.8.

Process finished with exit code 0
```

Final code of RandomWalk.java

```
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) {
        this.x += dx;
        this.y += dy;
    }

    /**
     * Perform a random walk of m steps
     *
     * @param m the number of steps the drunkard takes
     */
    private void randomWalk(int m) {
        while(m-->0){
            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() {
        return Math.sqrt(Math.pow(x,2) + Math.pow(y,2));
    }

    /**
     * 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]");

        List<Double> list = new ArrayList<Double>();// list for
plotting
        List<Double> sqroot = new ArrayList<Double>();// list for
plotting
        List<Integer> numbers = new ArrayList<Integer>();// list
for plotting
        for( String arg : args){
            int m = Integer.parseInt(arg);
            int count =10;
            double totalMeanDistance = 0d;
            while(count-- >0) {
                int n = 30;
                double meanDistance = randomWalkMulti(m, n);
                totalMeanDistance += meanDistance;
                System.out.println(" Count : " + count + " " +  m
+ " steps: with distance : " + meanDistance + " over " + n + "
experiments");
            }
            list.add(totalMeanDistance/10);// list for plotting
```

```
            numbers.add(m);// list for plotting
            sqroot.add(Math.sqrt(m));// list for plotting
        }

        System.out.println(list);// list for plotting
        System.out.println(numbers);// list for plotting
        System.out.println(sqroot);// list for plotting
    }

}
```