**Program Structures and Algorithms**

**Assignment-1**

**Summer-2022**

**Dimpleben Kanjibhai Patel – 002965372**

**Problem:** Random Walk

**Task**: To deduce the relationship between the distance of man from the lamp and the number of steps taken by the man from the original position.

**Output:**

100 Steps with 30 experiments

Graphical user interface, text, application, email

Description automatically generated

200 Steps with 30 experiments

Graphical user interface, text, application, email

Description automatically generated

300 Steps with 30 experiments

Graphical user interface, text, application, email

Description automatically generated

400 Steps with 30 experiments

Graphical user interface, text, application, email

Description automatically generated

500 Steps with 30 experiments

Graphical user interface, text, application, email

Description automatically generated

600 Steps with 30 experiments

Graphical user interface, text, application, email

Description automatically generated

700 Steps with 30 experiments

Graphical user interface, text, application, email

Description automatically generated

800 Steps with 30 experiments

Graphical user interface, text, application, email

Description automatically generated

900 Steps with 30 experiments

Graphical user interface, text, application, email

Description automatically generated

1000 Steps with 30 experiments



**Conclusion:**

The simulation is performed by keeping the number of experiments constant i.e., equal to 30 and varying the number of steps from 100 to 1000. The result of the simulation is shown in the screenshot. It is observed that the distance(D) is approximately equal to the square root of the steps taken(m).

**D =**

D => Mean Euclidean Distance of man from the lamp post.

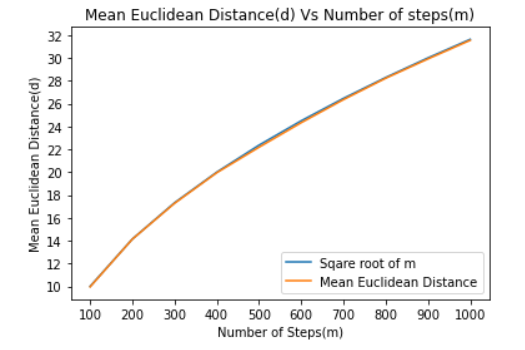
m => number of steps taken by man

**Evidence that supports the conclusion:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of Steps(m)** | **Number of Experiments (n)** | **Mean Euclidean Distance (D)** | **Actual Value of Square root of m ()** |
| 100 | 30 | 9.96 | 10 |
| 200 | 30 | 14.13 | 14.14 |
| 300 | 30 | 17.27 | 17.32 |
| 400 | 30 | 19.95 | 20 |
| 500 | 30 | 22.18 | 22.36 |
| 600 | 30 | 24.33 | 24.49 |
| 700 | 30 | 26.35 | 26.45 |
| 800 | 30 | 28.22 | 28.28 |
| 900 | 30 | 29.91 | 30 |
| 1000 | 30 | 31.54 | 31.62 |

**Graphical Representation:**

The graphical representation of the above table is as shown below. The x-axis represents the number of steps taken by man and y-axis represent the Mean Euclidean distance of man from original position. It is the comparison graph of actual value of square root of steps(m) and the Mean Euclidean Distance(D) that we calculated. It can be observed that the above relationship is approximately satisfied.



**Code:**

**RandomWalk.java**

|  |
| --- |
| */\*  \* 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) {  x += dx;  y += dy;  }   */\*\*  \* Perform a random walk of m steps  \*  \* @param m the number of steps the drunkard takes  \*/* private void randomWalk(int m) {  for(int i=1; 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.  \*/* public double distance() {  double distance= Math.*pow*(x,2)+Math.*pow*(y,2);  return Math.*sqrt*(distance);  }   */\*\*  \* 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 = Integer.*parseInt*(args[0]);  int n = 30;  if (args.length > 1) n = Integer.*parseInt*(args[1]);  double meanDistance = *randomWalkMulti*(m, n);  System.*out*.println(m + " steps: " + meanDistance + " over " + n + " experiments");  }  } |

**Unit test Result:**

Graphical user interface, text, application, email

Description automatically generated