

Qinyun Lin (NUID: 001582464)

Program Structures & Algorithms

Fall 2021

Assignment No. 1

Task :

- Run experiments for several values of n (steps), and run several times for each n
- Deduce the relationship between the steps and the random walking distance
- Demonstrate the relationship(expression) via graphs

Relationship Conclusion: $D = 0.88659\sqrt{n}$

Evidence to support the conclusion:

1. Output

Run experiments with n from 1 to 5000, step 10; run each n for 1000 times

```
RandomWalk x
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" ...
=====Create file writer success!=====

1 steps: distance = 1.0 over 1000 experiments
11 steps: distance = 2.9828919120788977 over 1000 experiments
21 steps: distance = 4.262821469795345 over 1000 experiments
31 steps: distance = 4.90312780388685 over 1000 experiments
41 steps: distance = 5.595536794213923 over 1000 experiments
51 steps: distance = 6.354461555060933 over 1000 experiments
61 steps: distance = 7.087162842615088 over 1000 experiments
71 steps: distance = 7.342113761672071 over 1000 experiments
81 steps: distance = 7.928003341685084 over 1000 experiments
91 steps: distance = 8.579301832880855 over 1000 experiments
101 steps: distance = 8.899839080542261 over 1000 experiments
111 steps: distance = 9.509461601139735 over 1000 experiments
121 steps: distance = 9.545450480416264 over 1000 experiments
131 steps: distance = 10.063213674557582 over 1000 experiments
141 steps: distance = 10.519912534524869 over 1000 experiments
151 steps: distance = 10.977425813274634 over 1000 experiments
161 steps: distance = 11.285038115478109 over 1000 experiments
```

```
RandomWalk x
4861 steps: distance = 60.08834973031614 over 1000 experiments
4871 steps: distance = 61.95867388724073 over 1000 experiments
4881 steps: distance = 63.27015055133456 over 1000 experiments
4891 steps: distance = 59.73614786784709 over 1000 experiments
4901 steps: distance = 59.43210224650577 over 1000 experiments
4911 steps: distance = 61.647913514896835 over 1000 experiments
4921 steps: distance = 62.52527516294052 over 1000 experiments
4931 steps: distance = 62.87162166507873 over 1000 experiments
4941 steps: distance = 63.51883185401669 over 1000 experiments
4951 steps: distance = 62.50860899403477 over 1000 experiments
4961 steps: distance = 62.97202219251272 over 1000 experiments
4971 steps: distance = 60.880743196768556 over 1000 experiments
4981 steps: distance = 62.09859522366502 over 1000 experiments
4991 steps: distance = 61.12769149230631 over 1000 experiments

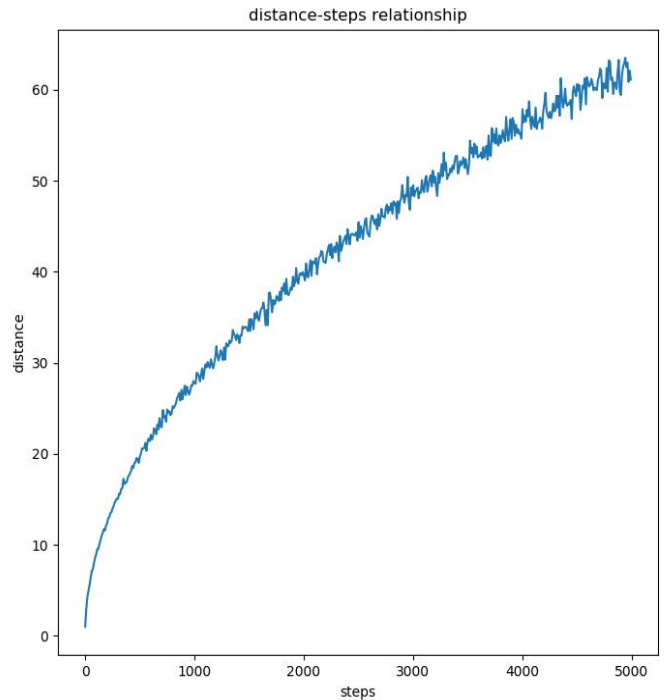
=====Results have been written to a file!=====

Process finished with exit code 0
```

2. Graphical Representation

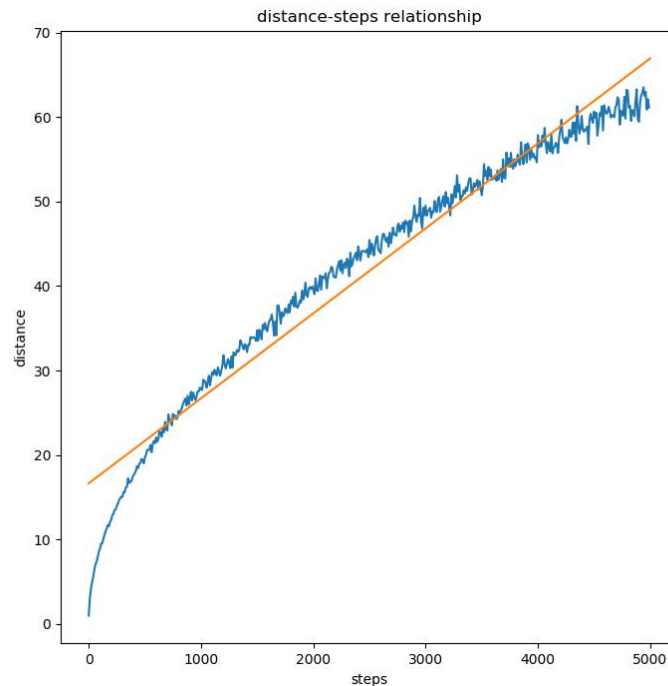
(1) Save the experiment results to .csv file, then use python to process these data

1	1	1.0	482	4811	61.122380379227785
2	11	2.9828919120788977	483	4821	61.335386732853244
3	21	4.262821469795345	484	4831	59.533136931887185
4	31	4.90312780388685	485	4841	60.808931435144856
5	41	5.595536794213923	486	4851	60.72724511557802
6	51	6.354461555060933	487	4861	60.08834973031614
7	61	7.087162842615088	488	4871	61.95867388724073
8	71	7.342113761672071	489	4881	63.27015055133456
9	81	7.928003341685084	490	4891	59.73614786784709
10	91	8.579301832880855	491	4901	59.43210224650577
11	101	8.899839080542261	492	4911	61.647913514896835
12	111	9.509461601139735	493	4921	62.52527516294052
13	121	9.545450480416264	494	4931	62.87162166507873
14	131	10.063213674557582	495	4941	63.51883185401669
15	141	10.519912534524869	496	4951	62.50860899403477
16	151	10.977425813274634	497	4961	62.97202219251272
17	161	11.285038115478109	498	4971	60.880743196768556
18	171	11.699461561800485			
19	181	11.570807453002029			

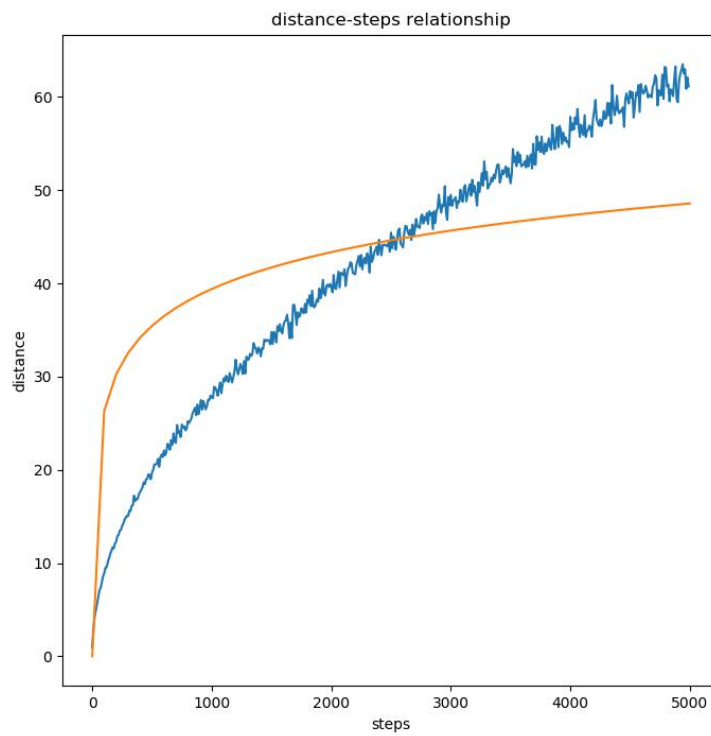


(2) I tried 3 kinds of functions and used LSE(least squares method) to fit it, the result is shown below:

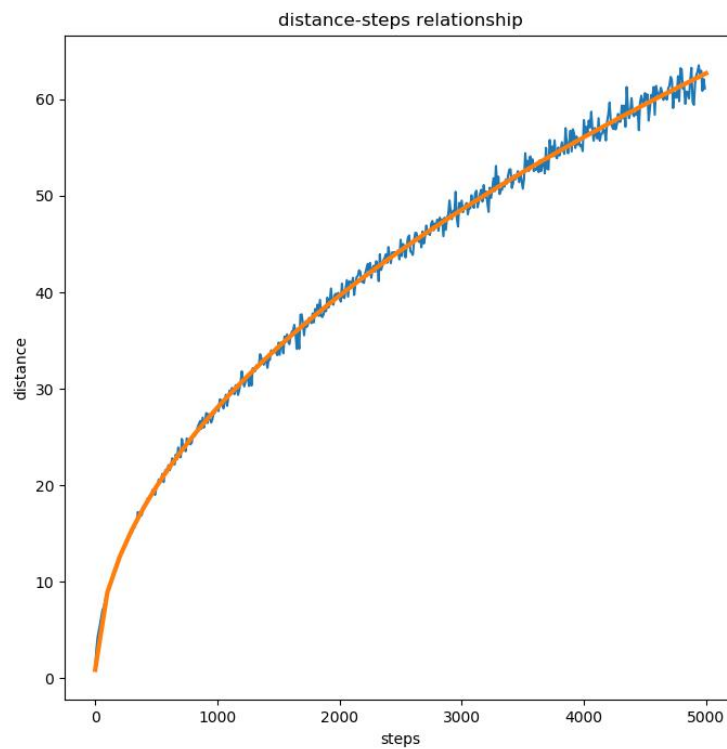
a. Linear function:



b. Log function:



c. Square root function:



(3) It is obvious that the Square root function fits it much better, so I deduce the relationship between $n(\text{steps})$ and $d(\text{distance})$ should be described by this kind of function. The LSE related code and result in python are shown below:

```
def cal_by_lse_2():  
    def func(p, x):  
        # guess the function format should be  $y = k * \sqrt{x}$  via the graph  
        k = p  
        return k * np.sqrt(x)  
  
    def error(p, x, y):  
        return func(p, x) - y  
  
    # calculate the expression via LSE(least squares method)  
    xi = np.asarray(steps)  
    yi = np.asarray(distance)  
    p0 = np.asarray([1 / 25.0]) # guess the init value of k via the graph  
  
    p = leastsq(error, p0, args=(xi, yi))  
    k = p[0][0]  
  
    return k
```

Run: visualization ×

▶ D:\Coding\tools\Anaconda3\python.exe D:/Coding/Java/INF06205-
result expression: $\text{distance} = 0.88659195681 * \sqrt{\text{steps}}$

Process finished with exit code -1

Unit tests result:

The screenshot displays an IDE interface with a project explorer on the left, a code editor in the center, and a run console at the bottom.

Project Explorer: A tree view on the left shows the project structure. The 'randomwalk' package is expanded, and 'RandomWalkTest' is selected.

Code Editor: The center pane shows the source code for 'RandomWalkTest.java'. The code is as follows:

```
5 package edu.neu.coe.info6205.randomwalk;
6
7 import ...
8
12
13 public class RandomWalkTest {
14
15     @Test
16     public void testMove0() {
17         RandomWalk rw = new RandomWalk();
18         PrivateMethodTester pmt = new PrivateMethodTester(rw);
19         pmt.invokePrivate( name: "move", ...parameters: 1, 0);
20         assertEquals( expected: 1.0, rw.distance(), delta: 1.0E-7);
21     }
22 }
```

Run Console: The bottom pane shows the execution results. It indicates that all tests passed.

Run: RandomWalkTest x

✓ Tests passed: 6 of 6 tests – 198 ms

✓ RandomWalkTest (edu.neu.coe.info6205.randomwalk) 198 ms

✓ testRandomWalk2 8 ms

✓ testMove0 2 ms

✓ testMove1 2 ms

✓ testMove2 2 ms

✓ testMove3 1 ms

✓ testRandomWalk 183 ms

Process finished with exit code 0