Qinyun Lin (NUID: 001582464)

Program Structures & Algorithms

Fall 2021

Assignment No. 1

Task :

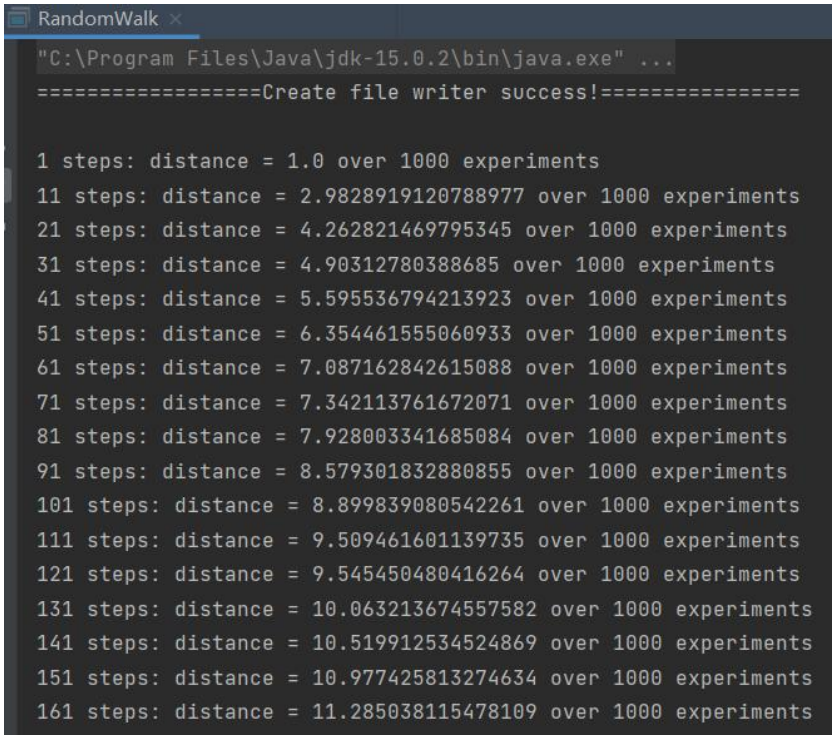
- Run experiments for several values of n(steps), and run several times for each n
- Deduce the relationship between the steps and the random walking distance
- Demonstrate the relationship(expression) via graphs

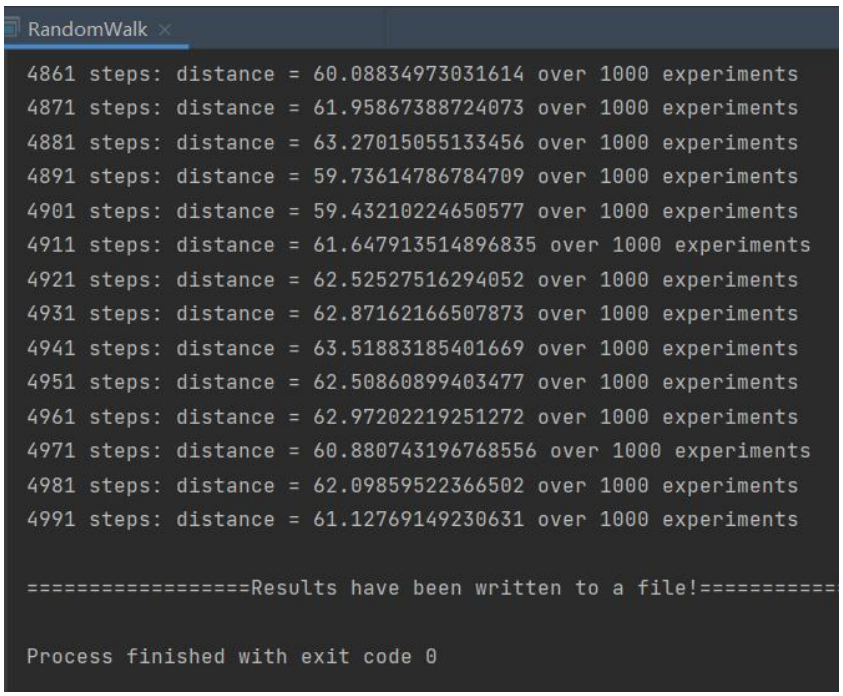
Relationship Conclusion: $\mathrm{D} = 0.88659\sqrt{n}$

Evidence to support the conclusion:

1. Output

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



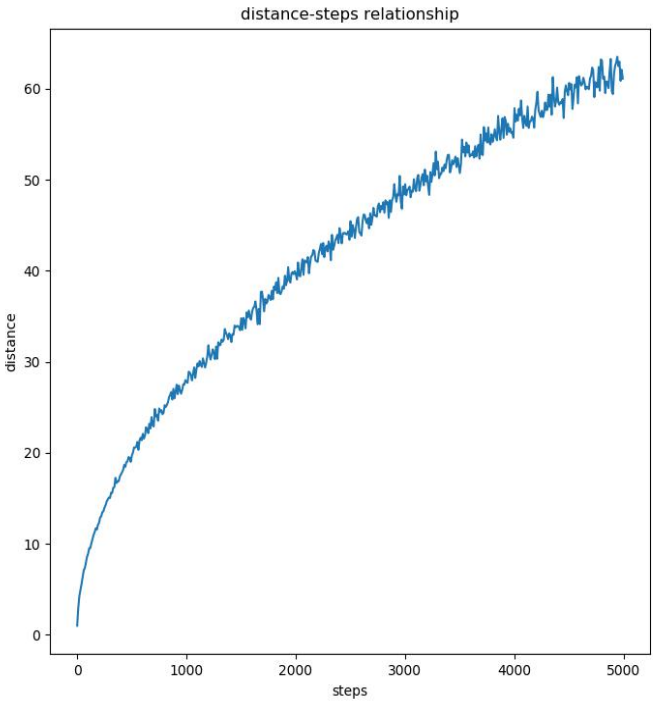


2. **Graphical Representation**

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

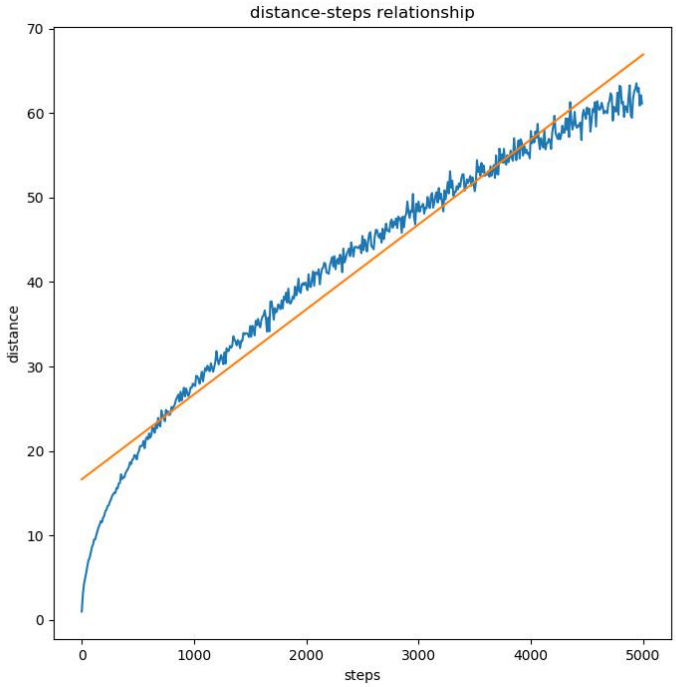
```
1 1.0
11 2.9828919120788977
21 4.262821469795345
31 4.90312780388685
41 5.595536794213923
51 6.354461555060933
61 7.087162842615088
71 7.342113761672071
81 7.928003341685084
91 8.579301832880855
101 8.899839080542261
111 9.509461601139735
121 9.545450480416264
131 10.063213674557582
141 10.519912534524869
151 10.977425813274634
161 11.285038115478109
171 11.699461561800485
181 11.570807453002029
```

```
4811 61.122380379227785
4821 61.335386732853244
4831 59.533136931887185
4841 60.808931435144856
4851 60.72724511557802
4861 60.08834973031614
4871 61.95867388724073
4881 63.27015055133456
4891 59.73614786784709
4901 59.43210224650577
4911 61.647913514896835
4921 62.52527516294052
4931 62.87162166507873
4941 63.51883185401669
4951 62.50860899403477
4961 62.97202219251272
4971 60.880743196768556
```

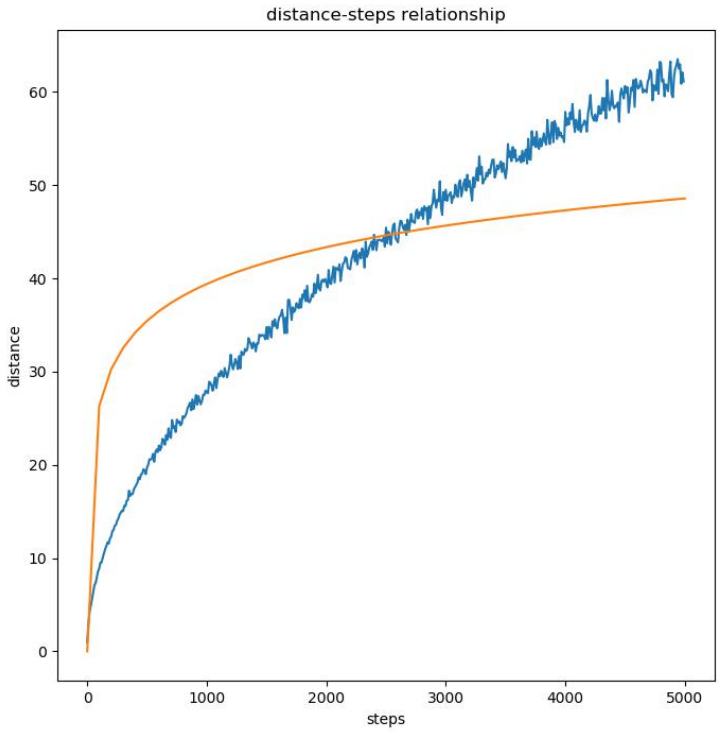


(2) I tried 3 kinds of functions and used LSE(least squares method) to fit it, the result is shown below:

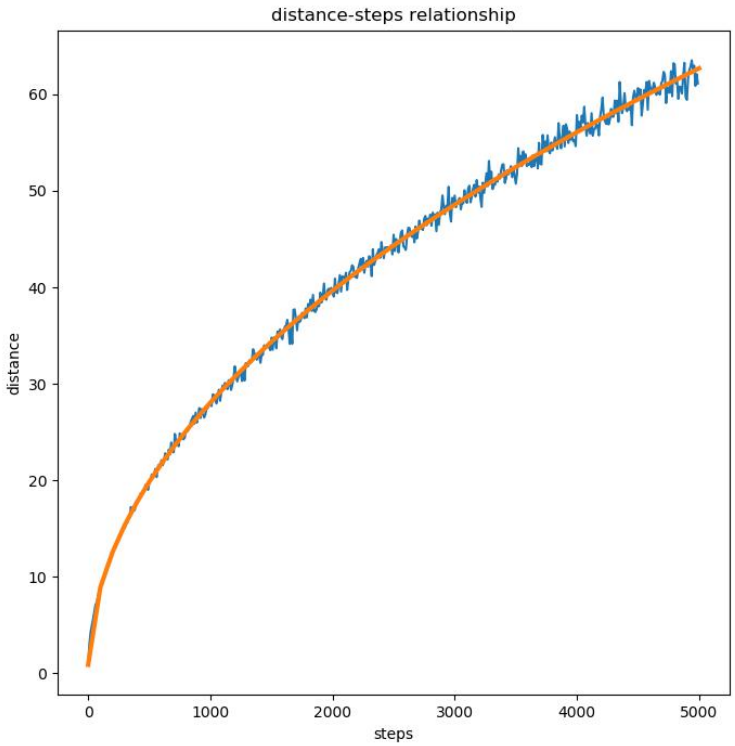
a. Linear function:



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(steps) and d(distance) should be described by this kind of function. The LSE related code and result in python are shown below:

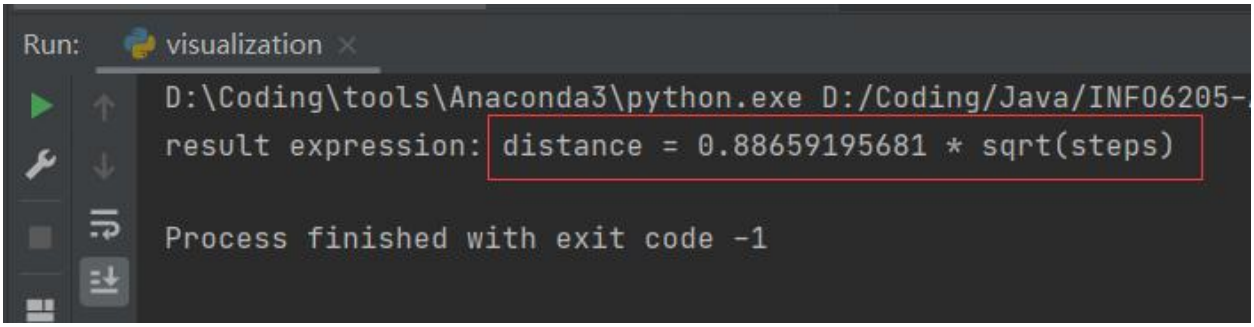
```python
def cal_by_lse_2():
    def func(p, x):
        # guess the function format should be y = k * sqrt(x) via the graph
        k = p
        return k * np.sqrt(x)

    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
```



Unit tests result: