Alex Salazar 9/23

15042189

CSC 370 Cover Document for Project 1

**Students whom I’ve discussed ideas with**

Gurdev seepersaud “max”

Jorge Cruz

Ken hill

Adam binder

**1.6 Graphs and analysis (2D data and graphs)**

|  |  |  |
| --- | --- | --- |
| **N** | **RN^2** | **FSAW(N)** |
| **10** | 26.245401780820973 | 0.04205813 |
| **11** | 30.01340852217676 | 0.0286922 |
| **12** | 34.196551872002544 | 0.01935601 |
| **13** | 38.305601568034255 | 0.01314257 |
| **14** | 42.77017525413036 | 0.00884487 |
| **15** | 47.15800335635167 | 0.00596481 |
| **16** | 52.1084914216143 | 0.00402465 |
| **17** | 56.7217301446708 | 0.0027096 |
| **18** | 61.79938518965801 | 0.0018217 |
| **19** | 66.79623023485523 | 0.00122501 |
| **20** | 72.06611732632535 | 8.1431E-4 |
| **21** | 77.09762677262951 | 5.4862E-4 |
| **22** | 82.90044301905253 | 3.6793E-4 |
| **23** | 88.5685936151855 | 2.4339E-4 |
| **24** | 94.1182617247526 | 1.6269E-4 |
| **25** | 99.70915153760652 | 1.0796E-4 |
| **26** | 107.26827276439363 | 7.347E-5 |
| **27** | 111.93239611329336 | 4.837E-5 |
| **28** | 118.40290820963344 | 3.301E-5 |
| **29** | 130.23866239120477 | 2.183E-5 |
| **30** | 131.82940360610263 | 1.442E-5 |
| **31** | 138.70194384449243 | 9.26E-6 |
| **32** | 137.12140575079871 | 6.26E-6 |
| **33** | 151.52154195011337 | 4.41E-6 |
| **34** | 159.5 | 2.92E-6 |
| **35** | 158.86206896551724 | 1.74E-6 |
| **36** | 173.1904761904762 | 1.26E-6 |
| **37** | 167.76190476190476 | 8.4E-7 |
| **38** | 189.42857142857142 | 7.0E-7 |
| **39** | 171.4390243902439 | 4.1E-7 |
| **40** | 219.72727272727272 | 2.2E-7 |

(Value for Beta)β = 0.402x

(Value for Alpha) α = 1.4622

1.7 Software Design and Optimization

**Software Design:**

My project one has a total of 6 files titled “ArrayList.java”, “Main.java”, “myThreads.java”, “position.java”, “ResultObj.java”, “SAW.java”. The first file which is the foundation of the project is the position.java. It’s purpose is to build objects that represent coordinates in 2,3, and 4 dimensions. Its variables are x,y,z,t, each representing components that would describe a point on a cartesian plane. The Class contains three constructors each to create the working dimension, the class contains all necessary setters and getters.

The Second most important file from the bottom up is the ResultObj.java file. It’s a class whose purpose is to create an object that’ll store the number of successful self-avoiding walks (NSAW), along with Rn^2 (The end to end squared distance) . This class is the reason why I can return data from the SAW.java file. Instead of sending back a ton of separate variables called NSAW and RN^2, I package those values in an object and return them. This is safer and perhaps better software design than returning variables, this class comes with all proper setters, and getters.

The third file, called SAW.java is the center piece of my project. The class has 6 major functions, 3 of which are named generate2DSaw, generate3DSaw, and generate4DSaw. All three of those functions receive steps as a parameter to indicate the amount steps to be taken during the session. All of the generate functions have an accompanying move function (move2D, move3D, and move4D). In all my generate functions, I initialize all of the components of a positional object necessary to create the dimensional objects with their respective origins ((0,0),(0,0,0), and (0,0,0,0)). This is a way to keep the walk from starting anywhere except on the origin.

Two storage containers are created, a hashmap and an arrayList. The arraylist stores each positional object, the hashmap wraps and stores the arraylist as it stores an object which translates to an index position, and each index has a unique signature k-value pair which I store as a string. For example 0,1 is stored exactly as that but represented as a string. This way of storing data becomes paramount in my programs ability to check whether a step that is being potentially considered, already exists somewhere within memory.

Generate saw has several int variables (depending on the dimension) and one temp positional object that stores and keep track of the previous coordinate. For example, Current\_x, Current\_y of 2DGenerateSaw function is a way of keeping track of each step as they place on the plane and only if a step is valid does it update with the values from the positional object named temp. The current values are also important because they update the end\_to\_end variables which is necessary for later RN^2 calculations. The generate functions call their respective move functions which take a random number, created by the thread local random function. The move function also takes a temp positional object with the current position of the walk. The move function determines its step depending on the number generated randomly, based on that it moves x,y,z,t on the board.

Once a coordinate has been chosen, an object is returned with the contents to its respective generate function, I created a temporary string to convert the ints into a string and check to see if I have stored those values beforehand. If they haven’t been stored, they are accepted into the path that is taken by the walk, otherwise my program takes it as intersection and breaks out of the walk. Finally the last thing the generate functions do if the walk is successful, is to return an object with the End to end values squared and summed, and a plus one for Nsaw of the walk was a success.

The next file that I want address is the generic arrayList.java file. This file is program I made in my previous 313 class in the spring of 2019. Its just a generic arrayList with all the necessary functions to the class to take any generic form of storage. Although this maybe a bit overkill for my project, I wanted to implement a clear function to remove all the contents saved within the array list. The last two files are the myThreads.java and the Main.java file.

The myThreads class is responsible for what each individual thread will perform as it is being created. The class receives two parameters, one to indicate the degree that will be chosen in the run method so that the proper instructions may be carried out (i.e call the generate 2d, 3d, or 4d) the other value represents the amount of steps to be carried out through the session. The run method makes use of the ResultObj class to carry the information from the saw generate class. Since the generate function returns values as a resultobj class I thought it would be necessary to receive it as a result obj and then break it into the sum of its parts to so that the right values are placed in the correct variables. The class Returns N\_SAW and endtoend to the main class.

The main class is responsible for calling the mythreads class. It has 3 void functions for 2D, 3D, and 4D. Depending on which function to be called in the static void main method, a dimension will be chosen. In the main class I have chosen to do 1000 threads, 10^5 walks, and 30 steps (I =10….41). The main method is responsible for dumping all the data on the screen.

The program works like this, there is a for loop that starts at i = 10, the iterator will iterate until step 41 each iteration increments the amount of steps being sent to the threads. After 1000 threads \*10^5 walks each have processed, all the data will be dumped on screen.

**Optimizations**:

In my early versions of my program I had trouble with run time. I had too many variables that were being created sequentially. I would first declare a variable with a name. on the line beneath the creation of that variable I would fill it with the purposed code. Instead I attempted to do what people call daisy chaining.

A close up of a logo

Description automatically generated

Throughout my program this style is ubiquitous, it increased performance on my computer. When I was building this program, I thought that perhaps one of the most important things to think about was the way I should store and retrieve. So from the beginning I thought it would be best to implement a HashMap with an arraylist. When finding things in a container retrieving anything from a HashMap is 0(1) and when adding to an array list it’s the same, so these are two little things that helped to improve my program.

**Self-avoiding random walks in higher dimensions**

**3D DATA set and graphs**

|  |  |  |
| --- | --- | --- |
| **N** |  | **FSAW(N)** |
| **10** | 16.817813385107403 | 0.14572031 |
| **11** | 18.837364969394578 | 0.11554325 |
| **12** | 20.951601647445088 | 0.09133658 |
| **13** | 23.059777502077893 | 0.07228472 |
| **14** | 25.23060529090025 | 0.05701903 |
| **15** | 27.371199048298624 | 0.04505615 |
| **16** | 29.622667900865224 | 0.03553183 |
| **17** | 31.843203953585117 | 0.02798068 |
| **18** | 34.12127371950317 | 0.02202934 |
| **19** | 36.40306656788883 | 0.01735882 |
| **20** | 38.73996228182499 | 0.01364329 |
| **21** | 40.97256719529447 | 0.0107448 |
| **22** | 43.432978714585005 | 0.00844381 |
| **23** | 45.8398020834429 | 0.00665533 |
| **24** | 48.219977484431844 | 0.00521417 |
| **25** | 50.58887428350835 | 0.00409286 |
| **26** | 53.16495733674266 | 0.00321471 |
| **27** | 55.585050996626556 | 0.00253154 |
| **28** | 57.99029355465905 | 0.00199043 |
| **29** | 60.522024861079885 | 0.00155665 |
| **30** | 62.85375758366495 | 0.00122632 |
| **31** | 65.20661900870051 | 9.6661E-4 |
| **32** | 67.92540191215775 | 7.5203E-4 |
| **33** | 70.62756518794447 | 5.906E-4 |
| **34** | 72.84294465341215 | 4.6525E-4 |
| **35** | 75.71843860660462 | 3.6429E-4 |
| **36** | 78.33879093198992 | 2.8584E-4 |
| **37** | 81.12294143549097 | 2.271E-4 |
| **38** | 84.12634577043578 | 1.7555E-4 |
| **39** | 86.7052206518604 | 1.3868E-4 |
| **40** | 87.77367933271547 | 1.079E-4 |

(Value for Beta)β = 0.241x

(Value for Alpha) α = 1.2002

**4DGraphs and Datasets**

|  |  |  |
| --- | --- | --- |
| **N** | **Rn^2** | **FSAW(N)** |
| **10** | 26.64650386489166 | 0.20160591 |
| **11** | 31.214613112578206 | 0.1678602 |
| **12** | 36.16513901480431 | 0.13969915 |
| **13** | 41.456112193343664 | 0.11622579 |
| **14** | 47.11109163708101 | 0.09676705 |
| **15** | 53.107096972306515 | 0.08056661 |
| **16** | 59.46000595941151 | 0.06698648 |
| **17** | 66.1907959331034 | 0.05574275 |
| **18** | 73.26966688556898 | 0.04639817 |
| **19** | 80.7222559670002 | 0.03862326 |
| **20** | 88.50622116381273 | 0.03213627 |
| **21** | 96.68160870603006 | 0.02672906 |
| **22** | 105.19180589310824 | 0.0222599 |
| **23** | 113.93270430910403 | 0.01851545 |
| **24** | 123.25014770881097 | 0.01541885 |
| **25** | 132.7781977581897 | 0.01280483 |
| **26** | 142.61333898255833 | 0.01066813 |
| **27** | 152.93735539610483 | 0.0088777 |
| **28** | 163.57130750506988 | 0.00738674 |
| **29** | 174.6692037793783 | 0.00616186 |
| **30** | 186.09009196014537 | 0.00510656 |
| **31** | 197.5924166376315 | 0.00424772 |
| **32** | 209.70432561509554 | 0.00354701 |
| **33** | 222.4457465894858 | 0.0029541 |
| **34** | 234.9967833090385 | 0.00245594 |
| **35** | 247.89542298010716 | 0.00204194 |
| **36** | 262.23896116144317 | 0.00169651 |
| **37** | 275.7378241282679 | 0.00140955 |
| **38** | 289.1502243493868 | 0.00117005 |
| **39** | 304.14942129511394 | 9.7891E-4 |
| **40** | 319.50771498771496 | 8.14E-4 |

(Value for Alpha) α = 1.8035

(Value for Beta)β = 0.184x

Results for alpha and beta

|  |  |  |
| --- | --- | --- |
| Dimension |  |  |
| 2D | 1.4622 | 0.402x |
| 3D | 1.2002 | 0.241x |
| 4D | 1.8035 | 0.184x |