Kang-hee Cho

CSCI-361: Project 2

Dr. Sateesh Mane

READ-ME File

This file you are about to read explains everything you need to know about my submission. I will describe each java file contained in my submission as well as how to use them. I have configured some nice tests for you to make things easier on your part. I will let you know how to run them correctly and what to expect. Just a disclaimer, I have coded, compiled, and ran all of my files on the mars server. This is because venus is unreliable sometimes.

**JAVA FILE INDEX:**

**quantum\_spin.java:**

This java file contains the quantum\_spin class. I have created this class to act as a node-class for our linked list. You may open the file in your terminal in venus using the vim editor if you wish to take a look at the code and my documentation.

**spin\_configuration.java:**

This java file contains the spin\_configuration class. I have created this class to act as a linked-list class for this project. You may open the file in your terminal in venus using the vim editor if you wish to take a look at the code and my documentation.

**spin\_test.java:**

This java file is the test class for our spin\_configuration class. You mentioned in the project documentation that:

A screenshot of a cell phone

Description automatically generated

I have figured out, using this test file I have created, that no matter where the change may reside in the new configuration, the resulting difference of the energy doesn’t change. However, being short on time, I manually created an energy difference function inside my main class.

**graphs.java:**

This file contains my tests for calculating both M and CP theory, as shown here:

A screenshot of a cell phone

Description automatically generated

I have made sure that my functions returned the right values

**proj2.java:**

This java file contains the main driver class for our project. I have made it so that we will start the metropolis algorithm from scratch and start with the nF value at 30. I have made it very easy for you to test my code. All you have to do is compile the program with

**javac proj2.java**

**java proj2**

Make sure all of the provided java files in the zip archive all reside in the same directory of proj2.java. Then after that, the rest should be self-explanatory. Just provide the values of B, C, and T at the prompts from the program and then you are set. My program will automatically fire up the metropolis algorithm 1000 times simultaneously, starting with 30 configuration changes within each iteration of the algorithm. The program will increment the value of nF by 1 after each 1000-thread iteration. Then, after extensive testing, I came upon the answer **52**, proven with the screenshot below:

A screenshot of a cell phone

Description automatically generated

The above screenshot shows the actual output of my code, and I am confident that every time you run my program, you will come across different values for relative errors and variances for each 1000-thread iteration and maybe even achieve a different answer for nF that is also correct.

From my end, after testing my program some number of times, there needs to be 52 configuration changes per iteration of the metropolis algorithm in order to get the relative error below 0.02. If you take a look at my code, I have provided extensive documentation alongside it to help guide you through my process.

**Challenges:**

Although it would have been nice to be able to solve all of the challenges, I was only able to get to the first one. If I had more time, I confidently believe that I would have been able to solve all of them. You can test my code for challenge 1 by calling it manually in the main function of proj2.java. **I have not yet tested this function so it might not work.**