**Project 2**

**CIS 677**

**Dhynasah Çakir**

**10/3/2018**

**OpenMP-enabled Simulation of Biological Crystal Growth**

**Analysis**

**Outline of program**

* In this program I imported the iostream, cstdlib, omp.h, chrono, and matrix.h (header file)
* The constructor class is used to populate a 2d matrix of size N\*N with all 0’s. I then set the center ([N/2] [N/2]) to 1. This is a simple step, which I then parallelize.
* The next function walkOneParticle is where I randomly generate 2 numbers from 0-(N-1) and set it to the indexes, all the while checking to make sure that if it landed on an already filled index it would generate another one.
* I sent a bool to false. And I use a while loop to confirm check that index and index 2 are greater than or equal to zero.
* I use an if statement inside of the while loop to check if any of the adjacent indexes is equal to 1. If so crystal is equal to true and I break out of the loop and that index to 1. If not, then the crystal will move.
* To parallelize this, I added pragma omp parallel to the original initialization of the 2d matrix and set lattice[i][j] to private.
* The only other thing I used pragma omp for was the for loop in the main file. Then set walkOneParticle to critical.
* Used chrono or portable timing to time the sequential and multi-threaded version of the program

**sequential vs. parallel**

My multi-threaded program did not perform as quickly as I would have expected. There was a considerable slow down with each added thread. As you can see in my graph, as the number of threads increased so did the time. It may very well be that many of my variables needed to be set to private as well as some parts of my walkOneParticle function should have been set to critical as well. I also believe that there was too much overhead, which inevitably caused the program to slow down significantly.

**Cout visualization A picture containing outdoor

Description generated with high confidence**