Chem274B - Software Engineering Fundamentals for Molecular Sciences

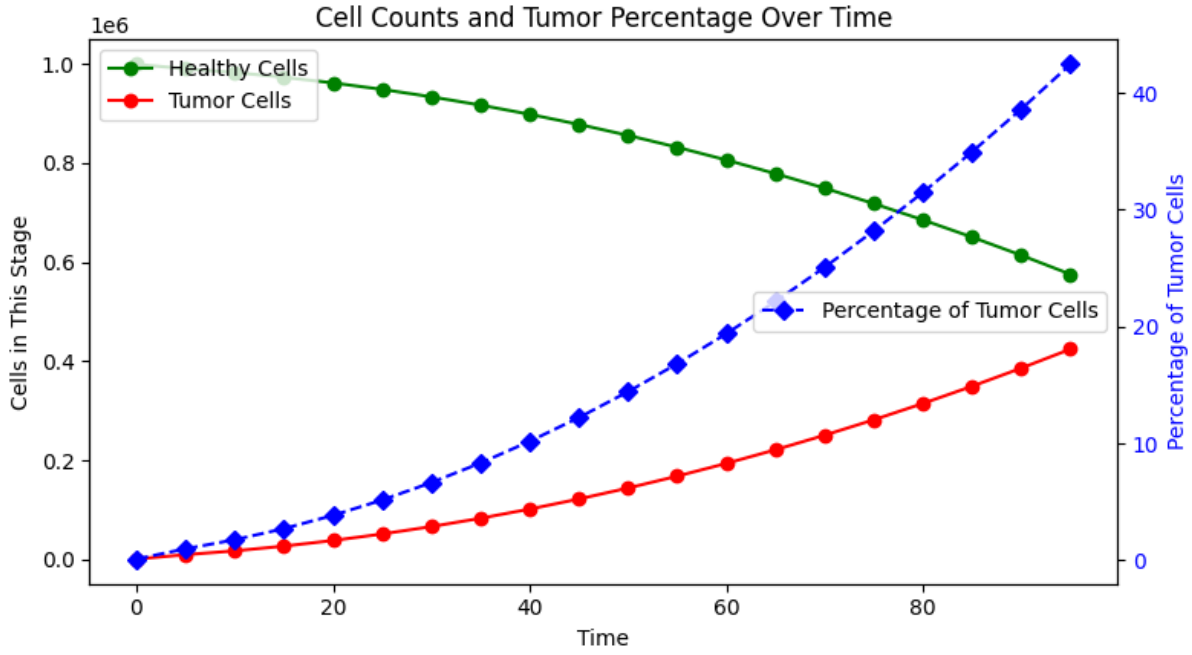
Final Project: Design and C++ Implementation of General Purpose Cellular Automata Library Responses for Part 2 and 3

Team members: Sheyda Nazarian, Yingyin Yu, Jedrick Regala Zablan

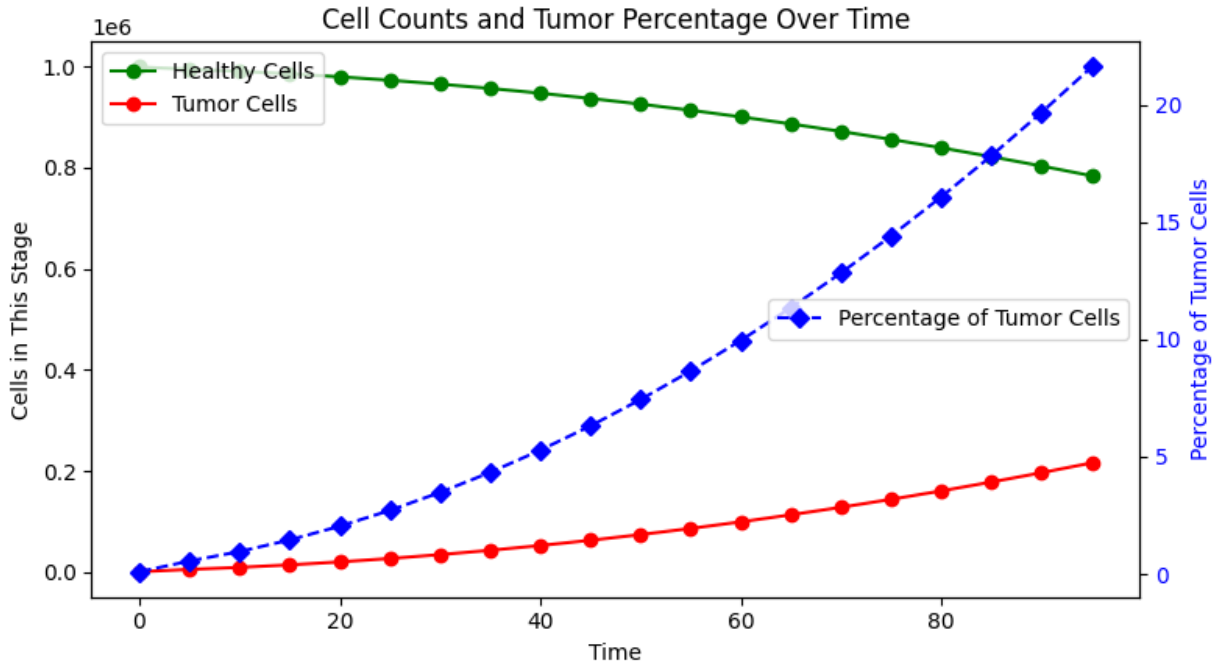
Part 2)

Our group was able to create a successful application that displayed the growth of cancer cells in tissue. The cancer grows from the center of the matrix and then spreads out to other cells. Although we were able to model this growth over time, we are unsure if the growth rate is scientifically accurate. We developed a simple model using the Von Neumann neighborhood, so we did not include other factors that could have impacted the cancer growth such as tissue density, properties of those cells, and real-life cell-cell interactions.

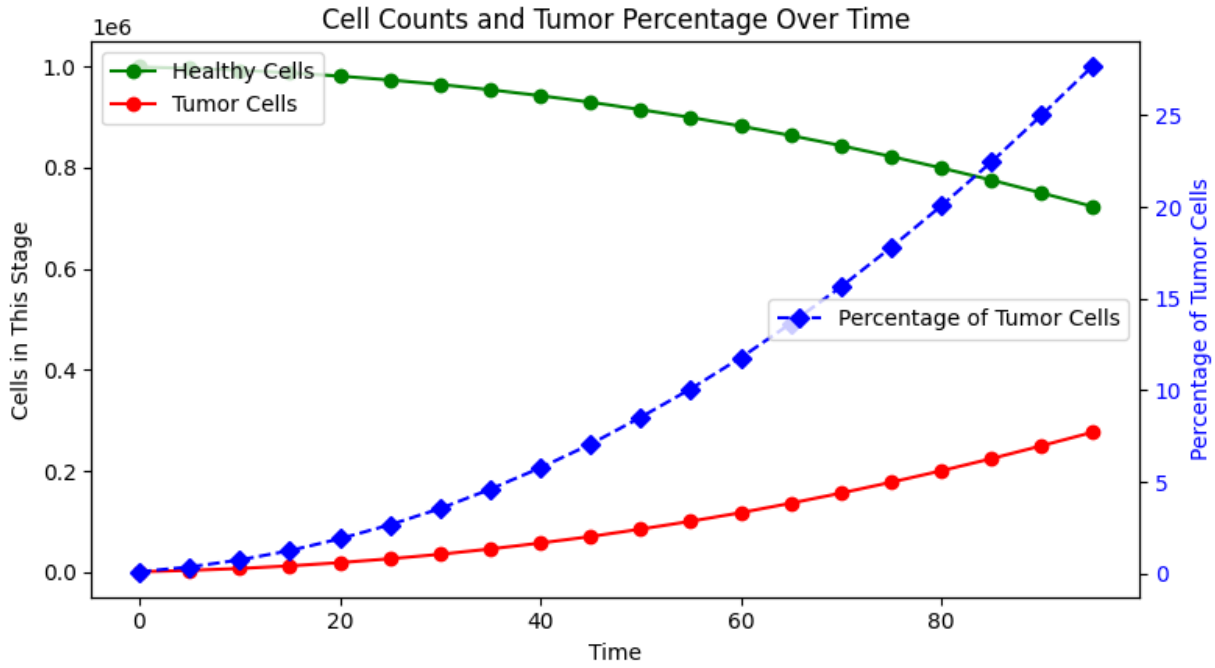
Cancer growth rate over time using the Moore Neighborhood model:



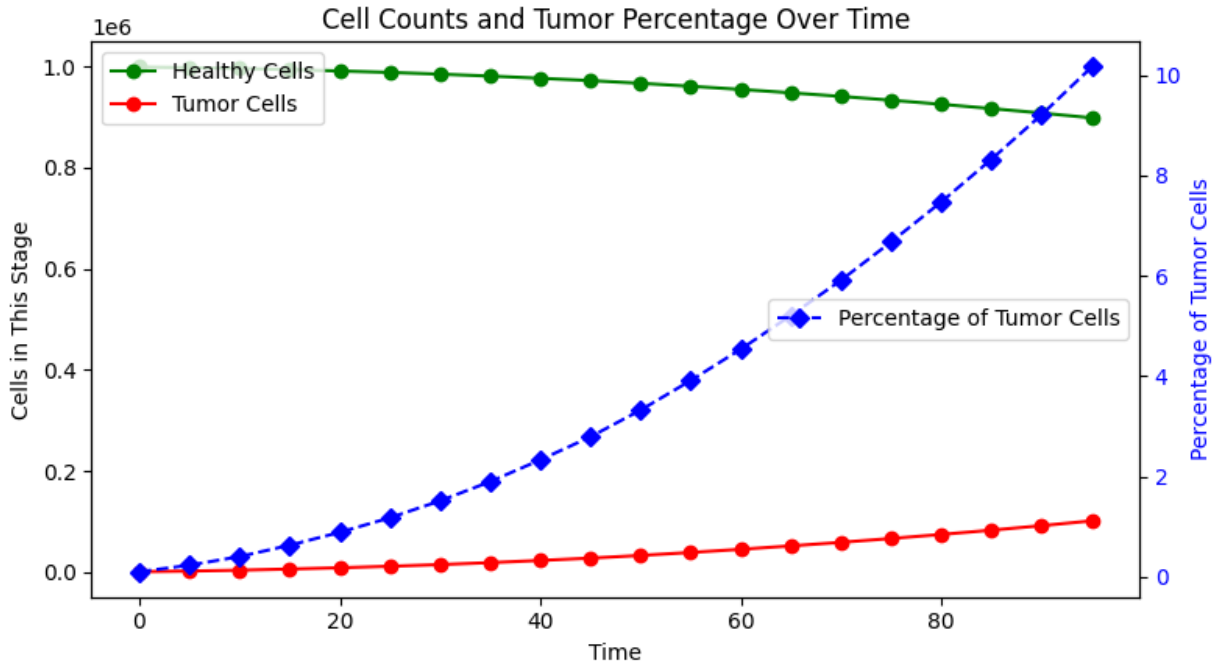
Cancer growth rate over time using the Von Neumann Neighborhood model:



Cancer growth rate over time using the Moore Neighborhood Probability model:



Cancer growth rate over time using the Von Neumann Neighborhood Probability model:



Part 3)

3.1) My role in the project was to do code reviews, participate in group meetings, and write different functions throughout the program. I wrote the functions for printing: “compute_with_report” and “compute_with_prob_report”. These functions created the .txt files that stored the matrix data that would be generated into a matrix grid and plots of the cell states for the cellular automaton. In “analyze.py”, I wrote the “cell_analyze_grid”, “sort_output” and ”plot_states” functions and I edited the main function. The first function analyzes the grid to determine the total number of cells, healthy cells, tumor cells, and the percent of tumor cells to total cells then generates a report in the command line. The “sort_output” function sorts the command line reports so that they are in order of time step rather than what report is generated in the least amount of time. The “plot_states” function used the data from “cell_analyze_grid” to plot a graph that showed the number of healthy cells and tumor cells on one y-axis and the percentage of tumor cells on a second y-axis over time steps.

3.2) My functions were vital to the overall functionality of the code since they were integral in creating the reports and graphical representations for the cellular automaton code. By performing code reviews and discussing with my time members in members and through text messages, I was able to help work out bugs and develop a fully

functioning project. During team meetings, we would discuss the plan for how we would approach different elements of the project and address issues in the current version of our project. Outside of meetings, I made sure to make myself available to my other team members and respond promptly to promote good communication and team workflow.

3.3) A challenge that my group first encountered was that our custom Game of Life rule was not functioning. The tumor got smaller and smaller as the time steps increased. To resolve this issue, we decided to use the Condition transition rule based on neighbor which allowed our cellular automaton to function properly. Another issue we had was our Git repository not updating properly when we attempted to push. We were able to resolve this issue by properly communicating with one another and troubleshooting. Personally, when developing the functions analyze.py, I was having issues with the imported libraries within the msse-python environment, so I had to build a new environment.

3.4) To maximize the performance of our library, specific design choices were made. First, nested loops were made to minimize time complexity and memory overhead. Vectors were used throughout our program to provide effective memory management and minimization of memory leaks. Vectors allow for resizing which could be helpful in the future if we were to improve our program such as creating dynamic matrix boundaries. Arrays could have been used since they have a fixed memory allocation which could have lessened the memory overhead. Our implementations also checked for and handled errors with informative messages included in the API which can help with overhead. The time complexity of our Condition transition rule based on neighbor rule would be linear and can be modeled by O(n). The space complexity is dependent on the size of the 2D matrix. Overall, the true effectiveness of our CA implementation can be tested by utilizing different grid sizes and comparing performance.

3.5) My group was able to develop a fully functioning cellular automaton application that showcased cancer cell growth in tissue. We were able to display how this cancer grows over time however we are unsure if this growth rate is scientifically accurate. We created a simple CA implementation, so we did not consider real-life Biological factors that could have impacted cancer cell growth. If we were to create a more accurate implementation and application in the future, we would implement functions that considered tissue density, properties of specific cell types, and real-life cell-cell interactions.

3.6)

3.6.1) Our library development followed the good software engineering practices that we have learned throughout this course. I do not believe that we should have done this differently since it provides an organized way of showcasing our implementation and our application.

3.6.2) In terms of software project management, we could have divided the project in a different way rather than initialization, computations, and plotting. We could have split up these three parts into subproblems and had a group member work on a subproblem in each part. This would help us work in parallel rather than sequentially as a group because part two needed part one and part three needed part two. We could have also had a higher frequency of meetings such as daily or every other day meetings, but this would not be possible since we are all working students. However, this could have improved the workflow.

3.6.3) Our product could be improved by changing our implementation to create a more accurate representation of cancer cell growth. In our simple model, we did not consider factors that would impact cancer cell growth in real-life biological models. Implementing these factors would help create a more accurate model that would be more informative for the user. For example, different tissues have different cell densities so that would be an important consideration when modelling different types of cancers. Another improvement would be to have dynamic boundaries which could help the memory overhead especially if the cells are sparse.

-

3.6.3. Product improvements

- Other factors that could have impacted cancer growth in reality
- Different cell types behave differently in reality
- Dynamic boundaries
    - Sparse can be inefficient
    - More memory usage when the boundary is large