Project Description: Game of Life in Sparse Format

Objective: the goal of this project is to apply the concepts you learned during the course to the design and development of a sparse formulation of the game of life. The project requires the development of code, possibly including convenient HPC libraries, building system visualization, testing using a unit testing framework, performance measurement, and profiling.

Background

The **Game of Life**, also known as **Life**, is a cellular automaton (CA) invented by John Conway (recently deceased for Covid :(( ) in 1970. It is a zero-player game: evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves.

The universe of the Game of Life is an infinite, two-dimensional grid of square *cells*, each of which is in one of two possible states, *live* or *dead*, (or *populated* and *unpopulated*, respectively). Every cell interacts with its eight neighbors which are the cells that are horizontally, vertically, or diagonally adjacent.

In the standard formulation of the game, at each step in time, the following transitions occur:

1. Any live cell with fewer than two live neighbors dies, as if by underpopulation.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

These rules emphasize the role of both live and dead cells and lead to the traditional implementation of the game of life where the state of the system is represented by a matrix of boolean values where each *(i, j)* entry represents a cell’s state: `true` for live, `false` for dead.

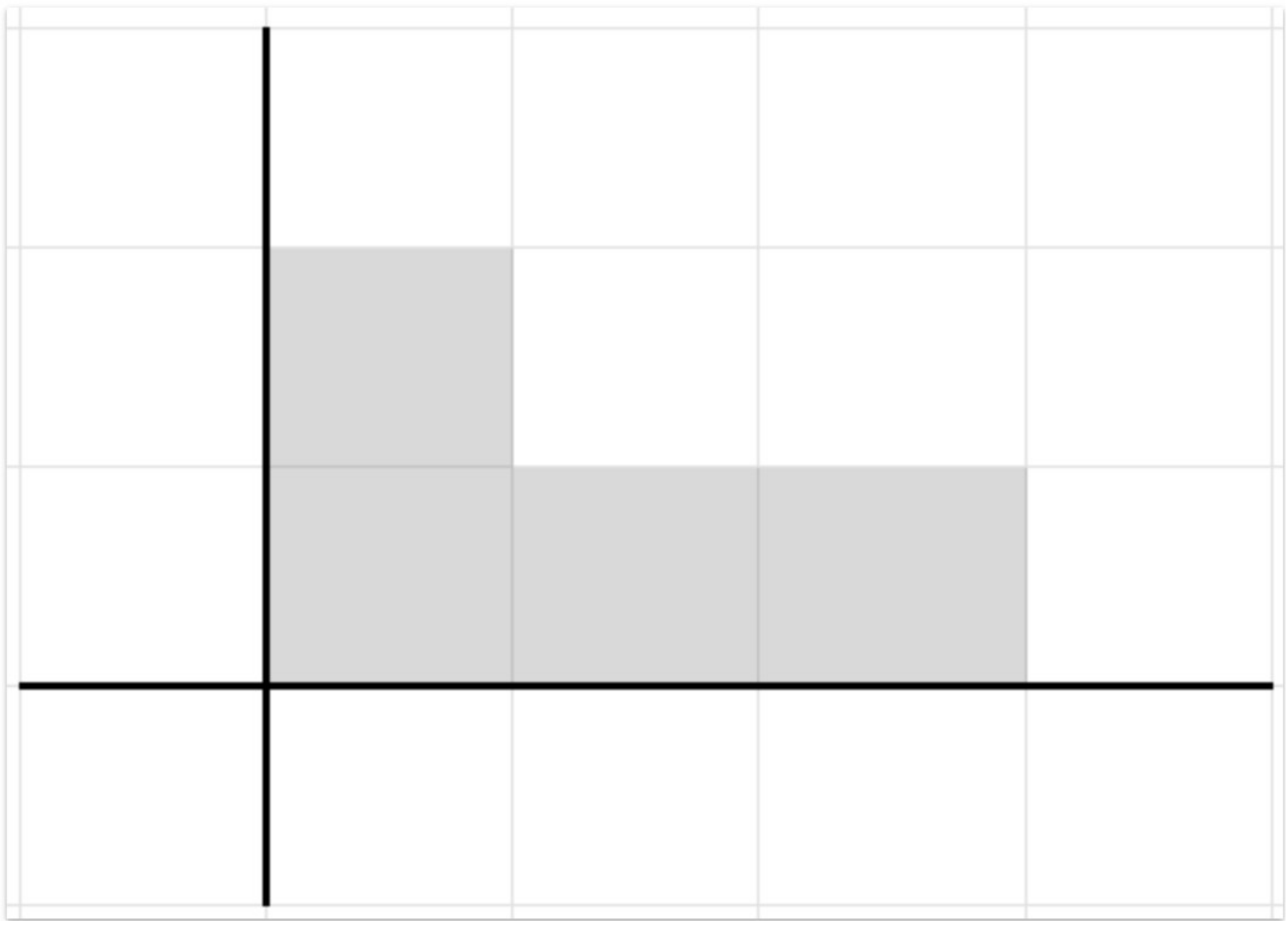
In this project, **we design and implement an alternative formulation of the game of life,** called sparse formulation. In this formulation, consider an infinite grid of cells defined as in the traditional formulation. For each live cell, we collect the coordinates *(i, j)* of all the cell’s neighbors and the coordinates of the live cell itself. We then count the number of times each cell appears in our collection. In the sparse formulation of the game of life, **only the cells obeying the following rules will be part of the next generation:**

- **Rule 1:** Any cell that occurs 3 times
- **Rule 2:** Any cell that occurs 4 times *and* is currently live.

This leads to a very natural implementation based on a **sparse representation of the grid**, **where we only keep track of live cells.**

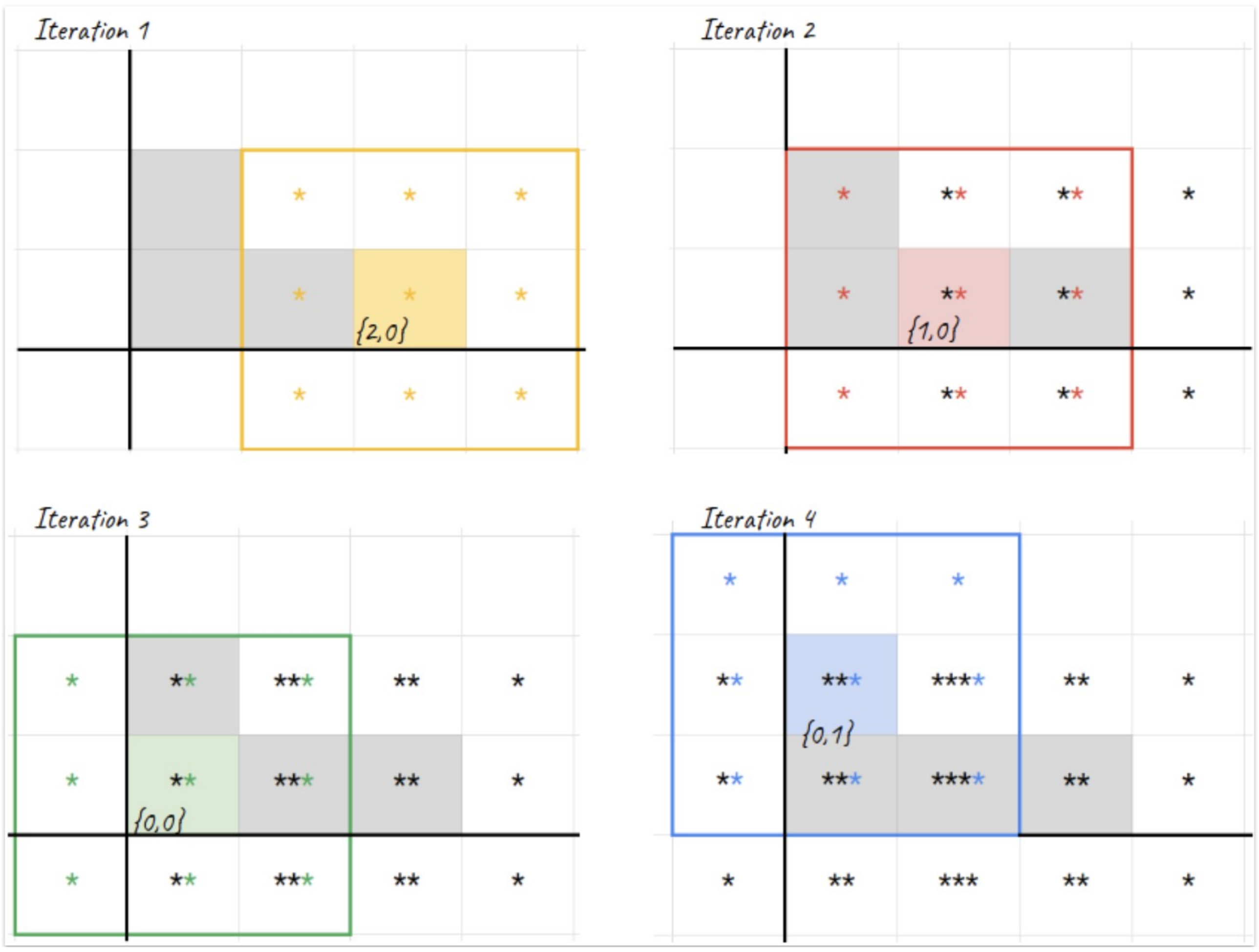
Example

Let's look at an example. For instance, to represent the population of living cells:



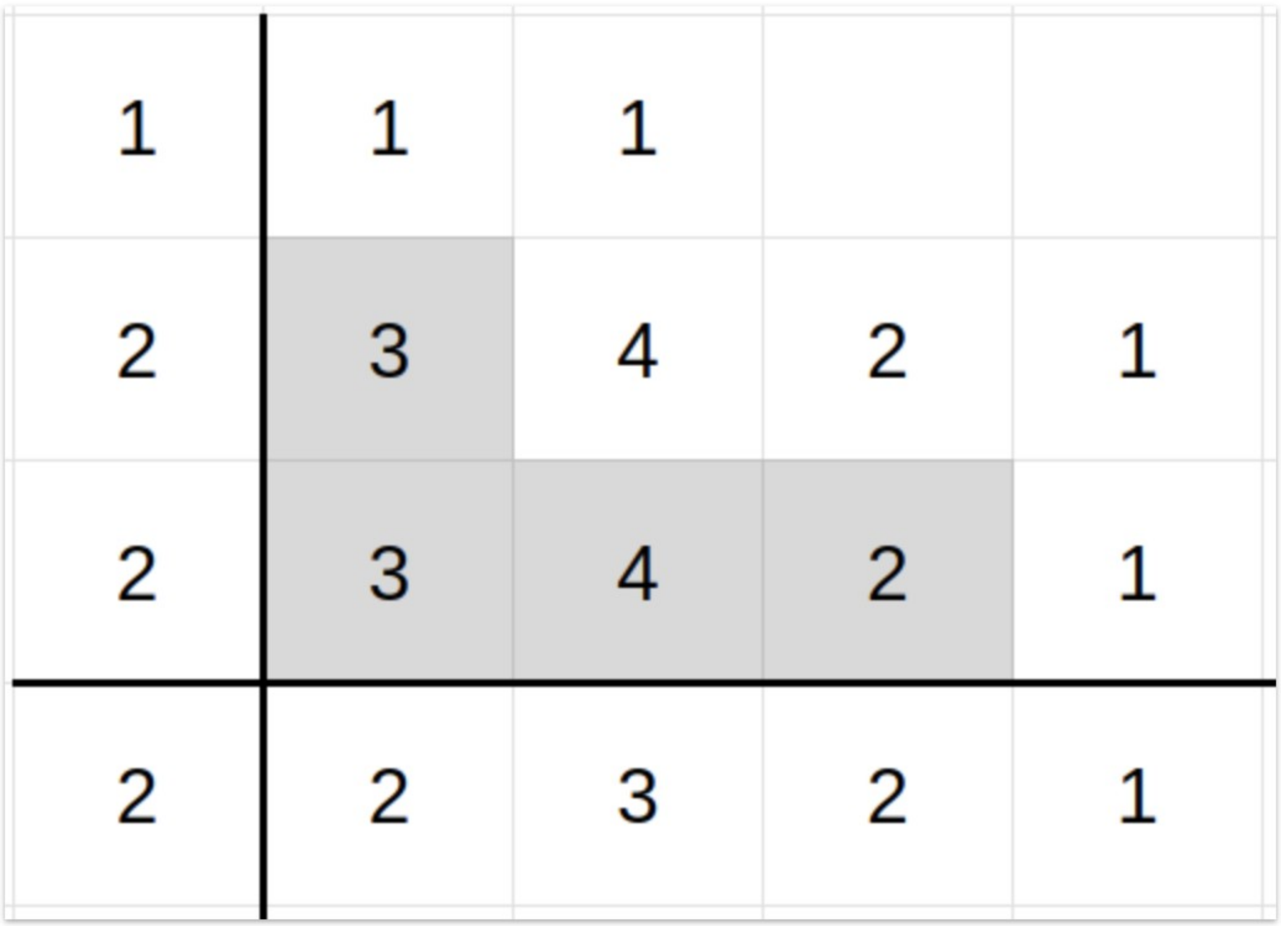
we can use an array population = [{2, 0}, {1, 0}, {0, 0}, {0, 1}] where we can think of the grid as a Cartesian plane where a live cell {x, y} is represented by a grey square with the bottom left corner in {x, y}.

To calculate the evolution of the system, we iterate over the live cells and *expand* them into the neighboring cells plus the live-cell itself like in the Figure below. This process is described by the following diagram, where the expansion of each live cell into the corresponding 3x3 square.



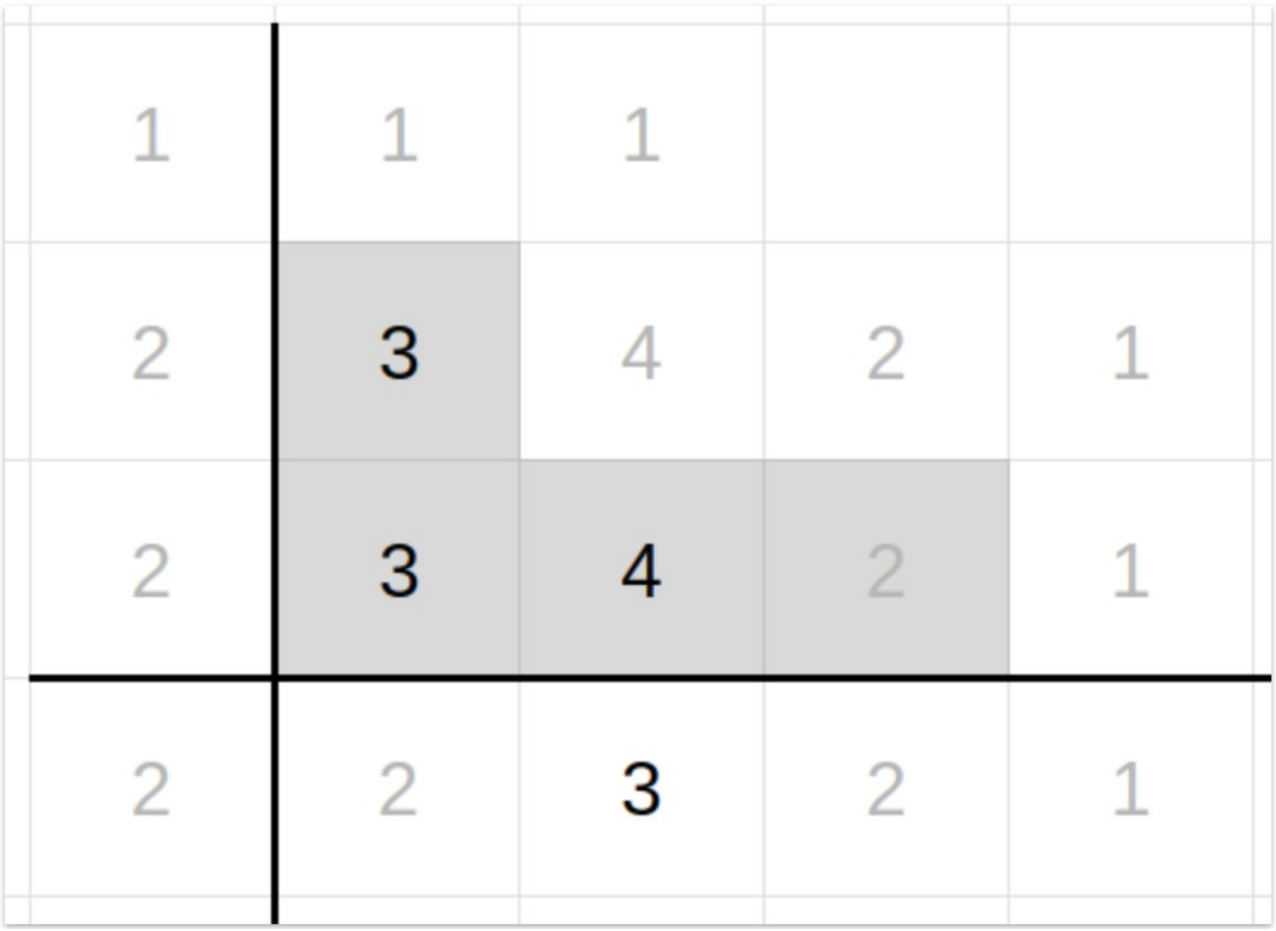
Cells in such squares are accumulated into a collection, with repetition. **Notice that the number of *stars* in each cell at the end of iteration 4 - bottom right picture - reflects the number of times a cell appears in any expanded square.**

We now count the number of times a cell appears in the computed collection as below



1	1	1		
2	3	4	2	1
2	3	4	2	1
2	2	3	2	1

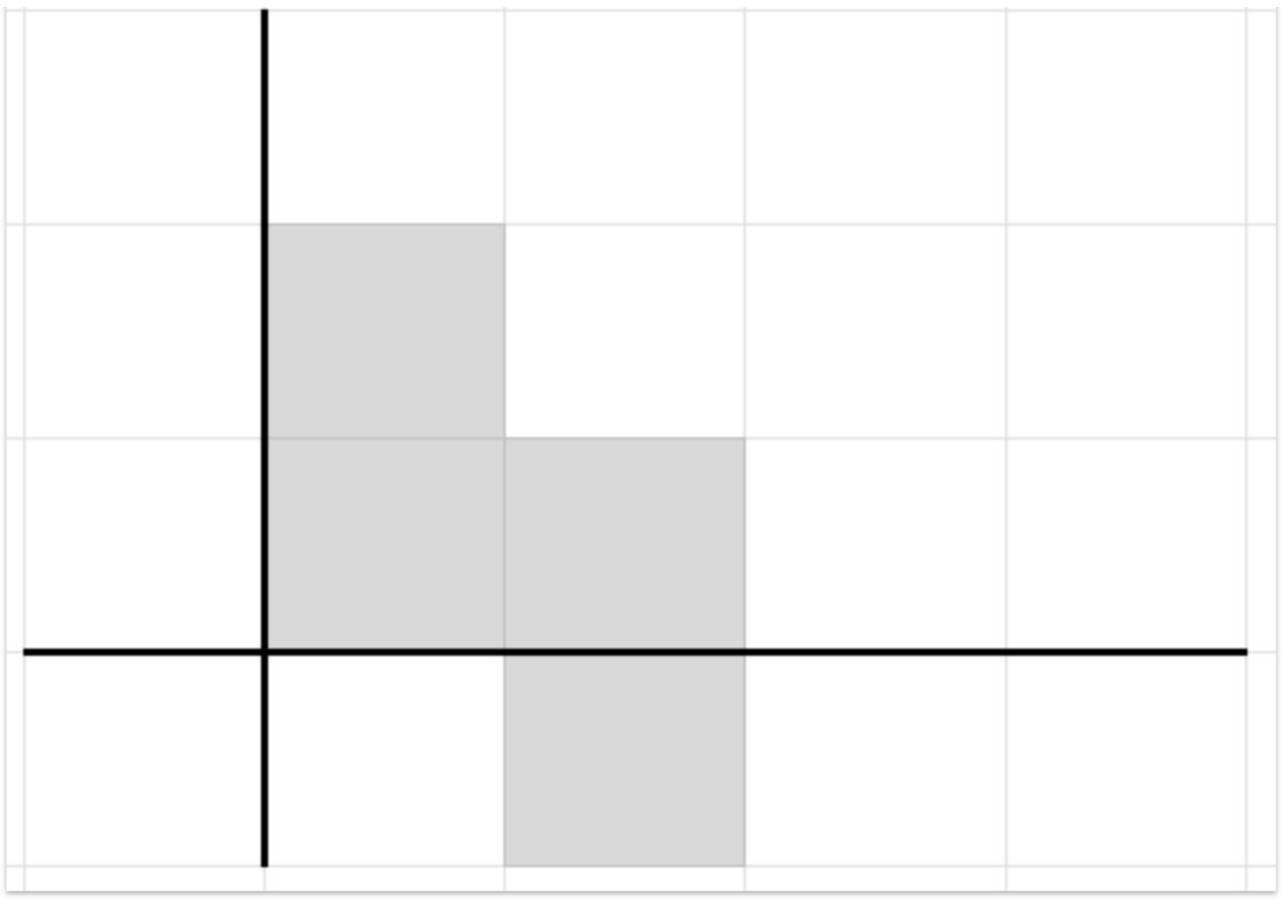
We then filter all the cells that don’t match either **Rule 1** or **Rule 2** defined above.



1	1	1		
2	**3**	4	2	1
2	**3**	**4**	2	1
2	2	**3**	2	1

The numbers in bold show the new generation of live cells. The light grey squares where the count is 4 are kept alive. Any cell with a count of 3 will be live in the next generation.

As the final step, we discard the number of occurrences and keep the cells' coordinates.



Tasks to be Performed

The goal of the project is the design and implementation of the game of life using a sparse implementation in **C/C++, Fortran, or Python calling C/C++/ Fortran** for compute-intensive routines. As part of this project, together with the design and implementation of the code, you need to complete the following tasks:

1. Prepare all the source code in C/C++, or Fortran, or Python calling C/Fortran for compute-intensive computations.
2. Test the implementation with three test cases using a unit testing framework, e.g. Google Test, pFUnit.
3. Prepare a building system either with GNU Autotools (not simple `Makefile`) or CMake to build the application and test suites.
4. Visualize the evolution of the system using ParaView or Visit. For this, you can write the position of the living cells to a file in VTK format (point data) and then use Paraview or Visit to visualize it.

5. Consider the usage of HPC libraries/data formats to improve the performance or implement additional features. Which HPC libraries can be used (e.g. BLAS, HDF5, VTK, etc)?
6. Analyze the performance of your code (execution time, cache usage, ...) and hotspots (functions that take most of the time) by varying the size of the initial population. For this, you should use the profiler in gperftools and the Linux perf tools (measuring hardware counters).
7. Analyze the advantage with respect to dense representation of the game in terms of memory and computation.
8. A Git repository containing all the code for the above. At the root level of the repository, include a `README.md` file that outlines:
    1. How to install the dependencies?
    2. How to build your code?
    3. How to run the test suite, what do they test for, and why?
    4. How to run the code, and adjust the initial conditions?
    5. How to visualize the output data?
    6. How to reproduce the experiments you presented in the presentation?

Grading Criteria and Submission Instructions

Refer to Final Project Submission & Presentation for full grading criteria and submission instructions. When in doubt, refer back to the lectures and tutorials presented during the course. Write in the Discussion: Final Project thread if you need help.

References

- The original article where the game of life was introduced: https://web.stanford.edu/class/sts145/Library/life.pdf
- Wikipedia page of the game of life: https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
- Sparse formulation of Game of Life (This exercise is adapted from this webpage): https://lbarasti.com/post/game_of_life/