SP Programming hw4

Conway’s Game of Life

Problem description

The Game of Life is a cellular automaton devised by the British mathematician John Horton Conway in 1970. It is the best-known example of a cellular automaton. The "game" is actually a zero-player game, meaning that its evolution is determined by its initial state, needing no input from human players. One interacts with the Game of Life by creating an initial configuration and observing how it evolves.

Goal

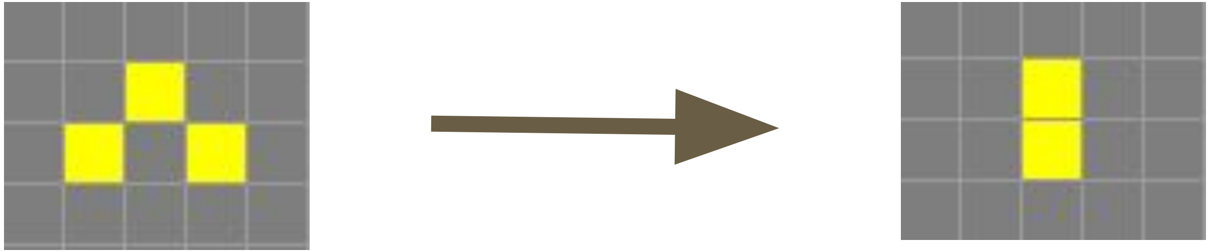
1. Implement Conway's game of life with multithreading to accelerate computation.
2. Understand the impact of the number of threads on system performance.
3. **Understand the impact of threads and processes on system performance.**

Online version

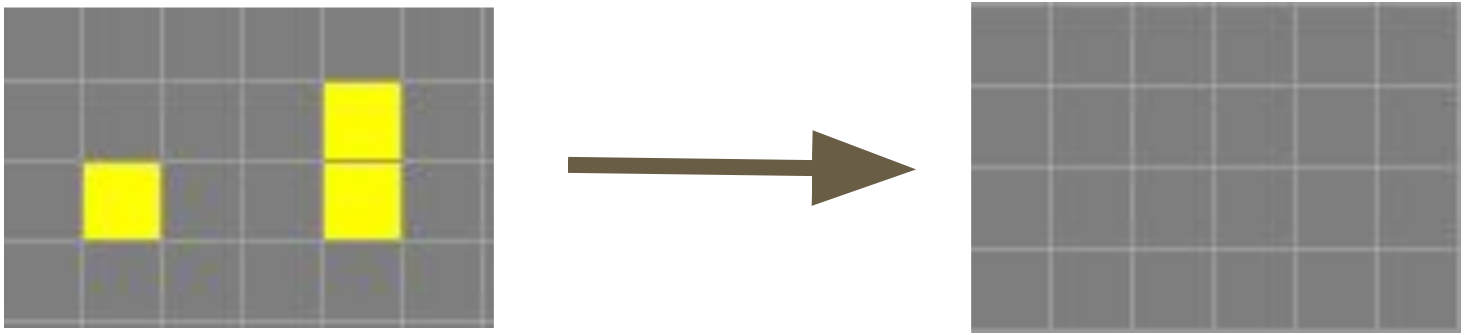
link

Rules

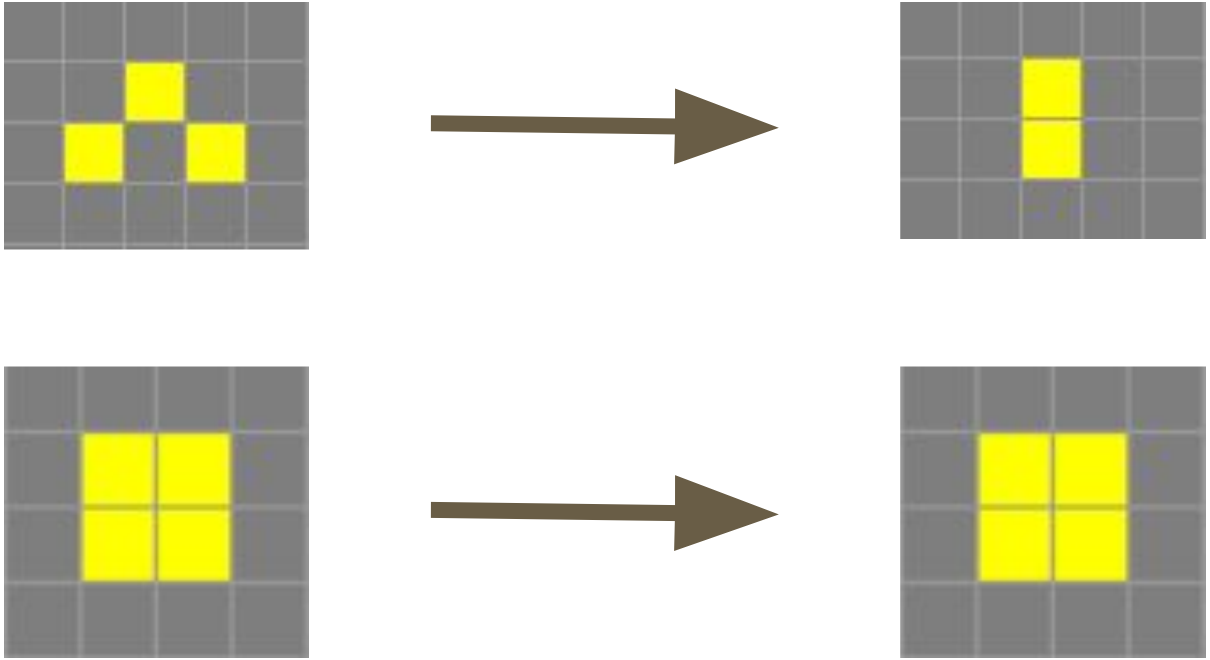
Any dead cell with exactly three live neighbours becomes a live cell



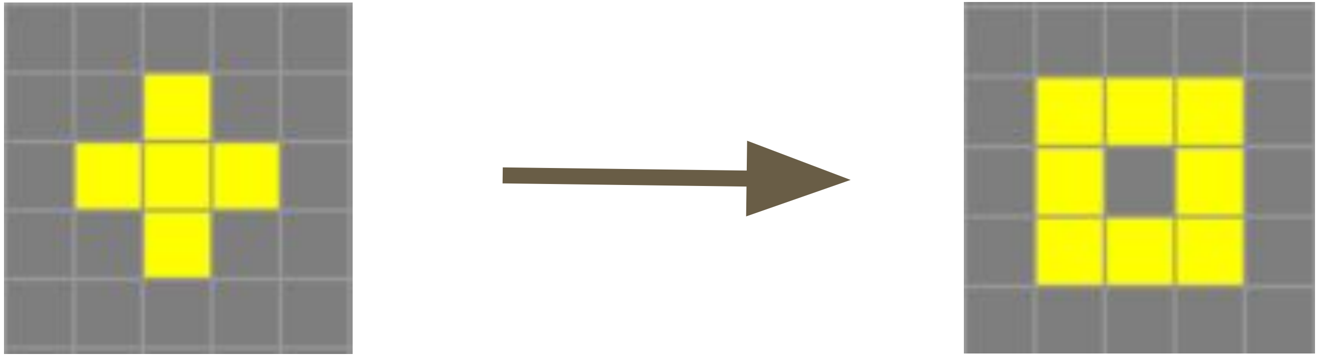
Any live cell with fewer than two live neighbours dies



Any live cell with two or three live neighbours lives on to the next generation



Any live cell with more than three live neighbours dies

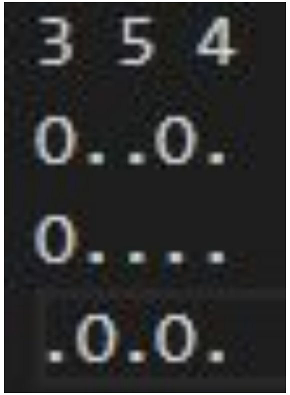


Sample input and output

1. The input and output file will be a .txt file.

Sample input and output

input.txt



```
3 5 4
O..O.
O....
.O.O.
```

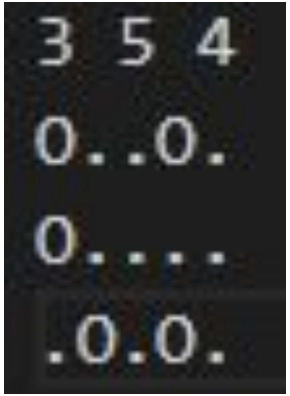
first line: **[row] [col] [epoch]**

O (big-O, not zero) : live cell

. : dead cell

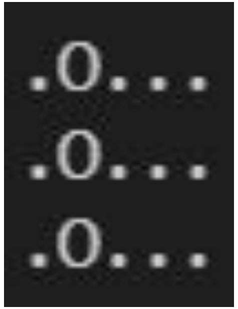
Sample input and output

input.txt



```
3 5 4
O..O.
O....
.O.O.
```

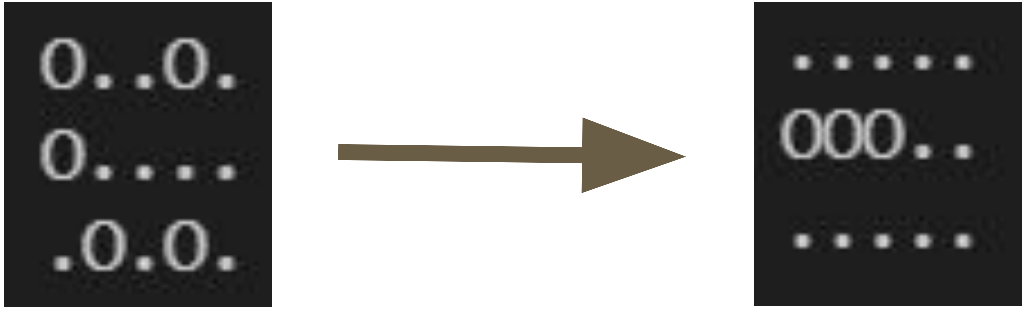
output.txt



```
.O...
.O...
.O...
```

Step by step

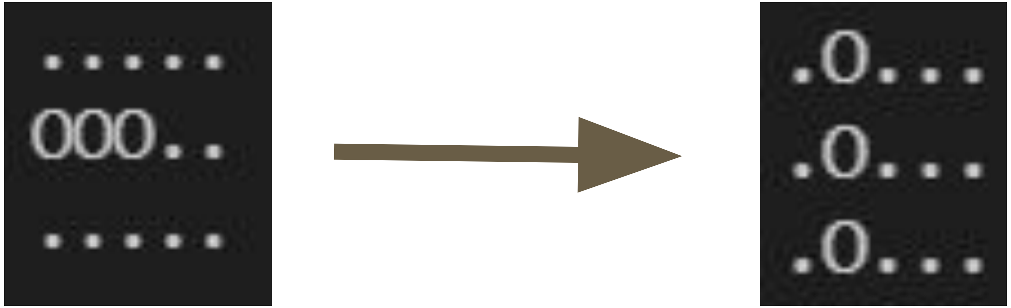
Epoch 1



```
O..O.
O....
.O.O.
```

```
.....
OOO..
.....
```

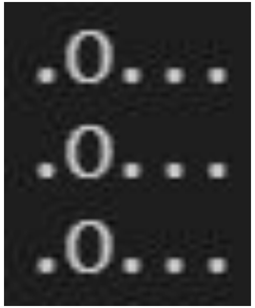
Epoch 2



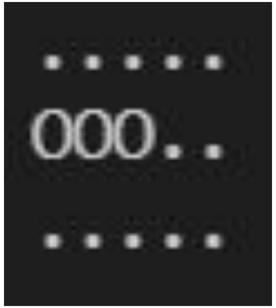
```
.....
OOO..
.....
```

```
.O...
.O...
.O...
```

Epoch 3

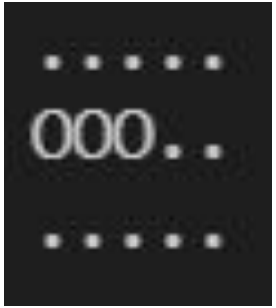


```
.O...
.O...
.O...
```



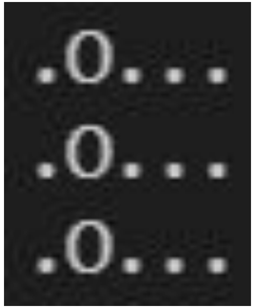
```
.....
OOO..
.....
```

Epoch 4



```
.....
OOO..
.....
```

Output



```
.O...
.O...
.O...
```

Execution

1. The following commands will be run when testing.
   - **make**
   - **./main [multithread or multiprocess] [quantity] [input file] [output file]**
     - ex.
     - **./main -p 2 ./infile1.txt ./output.txt**
     - **./main -t 20 ./infile2.txt ./output.txt**
2. Note:
   - Your output file should be in the same directory as the main executable file.
   - Your executable file name should be **main**.
   - About the second parameter, -p means using the multiprocessing to do this task, and -t means using the multithreading.

Grading

1. **(0.5%)** Running $ make generates executable files successfully, your executable file name needs to be main.
2. **(3%, 5 test cases, each 0.6%)** Finish test case in 3s (each) in multithreading (# of threads = 2).
3. **(3%, 5 test cases, each 0.6%)** Finish test case in 1s (each) in multithreading (# of threads = 20).
4. **(0.5%)** Compare (2)、(3) execution time in the large test case, giving your comment by this result on the report.
5. **(0.5%)** Use large test case to calculate execution time with different # of thread (= 1, 5, 10, ..., 100), drawing the line graph and giving your comment.

Grading

6. **(Bonus, 1%)** Use the multiprocessing which # of processes = 2 to finish the test case in 5s (each). **(5 test cases, each 0.1%)**, compare the result and (2), giving your comment by this result on the report. **(0.5%)**

    - Note: The number of processes to do this task should be 2.

7. **(Report, 0.5%)** Your report should contain (4)、(5) and (6) (if any) comments and critical parts of your program in (2)、(3)、(6) (if any), notice that the files must be less than or equal to 3 pages and save as PDF.

Submission

- Your assignment should be submitted to GitHub before the deadline. The submission should include **Makefile, report.pdf** and other **.c files**, as long as running **make** can generate all executables needed.

Reminders

1. When you do multiprocessing tasks, if you create multiple threads for each process, you will get 0 points.
2. Plagiarism is **STRICTLY** prohibited.
3. This assignment is not allowed to be submitted late, so please finish it as soon as possible.
4. If you have any questions, feel free to contact us via NTU COOL forum or email ntucsiesp@gmail.com, note that the beginning of your email should be **[Programming hw4]**.
5. Your files are compiled on the CSIE workstation, so make sure it can work smoothly on the workstation.

Reminders

6. Please note that we’re not obligated to answer the questions 24/7. While we try our best to help you with your problem, please give TA’s some reasonable time to answer your questions. Thanks a lot.