To Design and implement the “Game of Life” program

Homework #1

Ahmed Tanvir Mahdad

CS 732

4th February, 2021

Problem Specification:

The "Game of Life" is a well-known cellular automaton devised by John Horton Conway, a Cambridge mathematician. The Game of Life is a board game that consists of a two-dimensional array of cells. Each cell can hold an organism and has eight neighboring cells (left, right, top, bottom, top-left, bottom-right, top-right, and bottom-left). Each cell can be in two states: alive or dead. The game starts with an initial state and cells either live, die or multiply in the next iteration (generation) according to the following rules:

1. If a cell is “alive” in the current generation, then depending on its neighbor’s state, in the next generation the cell will either live or die based on the following conditions:
   a. Each cell with one or no neighbor dies, as if by loneliness.
   b. Each cell with four or more neighbors dies, as if by overpopulation.
   c. Each cell with two or three neighbors survives.
2. If a cell is “dead” in the current generation, then if there are exactly three neighbors "alive" then it will change to the "alive" state in the next generation, as if the neighboring cells gave birth to a new organism.

The above rules apply at each iteration (generation) so that the cells evolve or change state from generation to generation. Also, all cells are affected simultaneously in a generation (i.e., for each cell you need to use the value of the neighbors in the current iteration to compute the values for the next generation).

I have to create a program that will take the problem size and the maximum number of iterations as input and total execution time as output. I have to report the following combination:

Test Case #	Problem size	Max. Generations	Time Taken
1	1000X1000	1000	
2	5000X5000	1000	
3	5000X5000	5000	
4	10000X10000	1000	
5	10000X10000	10000	

Program Design:

The design of the program is explained below:

1. The program takes 2 inputs, first input is the problem size (N) and problem size 'max'.
2. I dynamically allocate 2 arrays of (N+2)X(N+2) size (present, nextArr) for calculation purpose. In 'present', I store the problem is in present state and in 'nextArr', I calculate the next stage of the problem.
3. In initial stage I generate first state of the problem by generating random number in different position in 2 dimensional array.
4. I pass both 'present' and 'nextArr' array with 'gameOfLife' function. It returns the next state in 'nextArr' array.
5. In 'gameOfLife' function, in present function, for each cell, neighbor number is checked, if neighbor number satisfies the condition, then the cell value has been changed in nextArr.
6. In main function, I have implemented an infinite while loop. Break condition of while loop implemented if either of these 2 conditions met.
    a. Iteration reaches maximum limit
    b. If the array generated as next state is exactly same as previous array. For comparison I maintain a global variable and it will set to 1 when present and next state are same in 'gameOfLife' function.
7. Finally, I free the allocated memory.

Test Plan:

For testing the program, I need to check some primary modules are working as expected:

1. If array initialization is working as it is, I will print the initialized array to check if the cells are generated randomly with value 0 and 1 and ghost cells are initiated as zero.
2. I will calculate first 3 steps of the game of life program according to the initialized array in pen and paper. I will then print the next 3 steps by turning on 'INTPRINT' debug flag on program to check if the steps are calculated correctly.
3. I have tracked how many iterations, the program run and print it when 'TOTALSTEP' debug flag is on, to check if the program can run until maximum iterations.
4. I have also checked 'comparison_flag' to check if the program halted before reaching maximum iterations and will print the iteration number when debug flag is on.

Test Cases:

I have designed some test cases to check if the program is working perfectly. I have focused on primary functionality of the program to check if they are running properly

1. **Test Case 1:** Generate 4*4 array for maximum 6 Steps and print all the generated steps on console to check if the steps are generated perfectly. Checked these steps with pre-calculated result.

2. **Test Case 2**: Run the program for generating 10*10 array for maximum 10 steps to check how many iterations are done.
3. **Test Case 3**: Run the program for generating 4*4 array for maximum 100 steps to check if the program halted after some time. Noted down the step where it has halted.
4. **Test Case 4**: Run the program by generating 1000*1000 array to check the maximum iteration it can reach. We test this to check if maximum value is worked in the program properly.

After running these test cases, It returns convincing result that the maximum values are properly maintained and the transition from present state to next state is correct. Also, It has been checked that, the program can halt when present state and next states are same.

How to Run the Program:

Compile : *gcc -O5 mahdad_hw1.c*
Run: *./a.out (problem size) (max_genrations*)

For example *./a.out 4* 4 will create problem size of (4X4) with maximum generation 4

Debug:

To print all intermediate states: *gcc -O5 mahdad_hw1.c -DINTPRINT*
To print Where the game is halted (previous and next state is same): *gcc -O5 mahdad_hw1.c -DHALT_CHECKER*

To print Total number of steps: *gcc -O5 mahdad_hw1.c -DTOTALSTEP*

System Specification:

CPU: 2 x Intel(R) Xeon(TM) CPU 3.20GHz
Memory: 3.87 GB
OS name and version: Centos 7 amd64
Compiler Name and Version : gcc (GCC) 4.8.5 20150623

Result:

Test Case #	Problem size	Max. Generations	Time Taken
1	1000X1000	1000	20.328 s
2	5000X5000	1000	499.87 s
3	5000X5000	5000	2379.54 s
4	10000X10000	1000	1991.16 s
5	10000X10000	10000	

For result collection, I run my program on one of the Vulcan machine 3 times for each Test Case except testcase #5 and calculated average for each test case. For testcase#5, I have tried to run the program on several Vulcan machine multiple times. The connection has been lost every time before the program execution completed (Broken pipe -- disconnected), so, program execution also halted.

Analysis and Conclusion:

From the result, we have noticed 2 interesting trends:

1. The execution time increase with Max generation increase in linear time (O(n)). That means, if we increase max generation 2 times, the execution time will also increase 2 times.
2. The execution time increase with problem size in quadratic time(O(n^2)). That means, if we increase the problem size 2 times (1000X1000 to 2000X2000), the execution time will increase 4 times.

The goal of this assignment is to make us familiar with dynamic memory allocation and implement game of life program with it. Also, another goal was to find relation between the execution time and problem size and maximum iteration.

I have implemented the ‘game of life’ program and find the relation between the execution time and problem size and maximum iteration. So, goal of the assignment has been achieved.

Reference:

1. Example hw.c program provided in assignment by Dr. Bangalore.