**CONWAYS CAF/FUSE assignment**

By Ramesh

**Introduction:**

This assignment solution is to solve the Conway’s game of life problem.

Total time taken to analyze, code & test and document – 3hrs

Analyze – 1hr, Code & Test – 1hr and Document- 1hr

**Description:**

*Design Considerations:*

* I assume live cells as ‘1’ and dead cells as ‘0’
* Arrays are pretty fast, We choose array for this assignment, we can access element in O(1).
* User inputs only either ‘1’ or ‘0’ , no other characters while passing dynamic array.
* Assume no Negative numbers as well.

[We can add some if conditions to check they the numbers are not valid but not added in this solution]

*Solution:*

* For this solution , I taken an arbitrary array to just get the count of neighbor cells to that particular cell
* I traverse through each and every cell only once
* If I find live cell then I will update the arbitrary array by incrementing neighbor cells count
* If I find a dead cell , I will ignore it completely
* Final step, I will go through arbitrary array and check the neighbor count and I will update the output array according to rules

**Unit test cases:**

1. Null check , whether the pass parameters are valid or not
2. Array out of bound exception or any other exception handling checking
3. Corner cases like give all ‘0’s and all ‘1’s
4. Give random 1’s and 0’s and check the output and make sure the edges are checked as well
5. Performance test, how long it takes to compute the output

**Program Analysis:**

r – Number of Rows

c – Number of Colum’s

Number of neighbor = 8

***Performance complexity:***

As we iterate main matrix once and also we iterate arbitrary matrix once

* O(r x c) + O(r x c) = 2\*O(r X C)
* Performance = O(r x c)

Worst Case: when all the cells are ‘1’ , then we need to update the arbitrary array every time, which goanna be : O(r x c x 8)

***Space Complexity:***

We add additional arbitrary array to the program, which had r rows and c columns

* Space complexity =O(r x c)

Other ideas:

* We can use a flag, so that We can keep recalculate the next states until it stops
* We can use a recursion way doing this and update arbitrary array as we proceed but complexity will be same as ours
* And also its possible to divide the matrix and calculate it separately with thread, which will increase the performance.