Juego de la Vida

- Cuadricula de NxN.
- En cada cuadro existe una celula que puede estar viva o muerta.
- Cada celula tiene 8 vecinos a su alrededor.
- Todas las celulas cambian de estado a la vez.

Condiciones para que este viva

- La celula debe tener 2 o 3 celulas vivas a su alrededor.
- Si una celula tiene 3 celulas vivas a su alrededor, esta revivirá (nace).

Condiciones para que muera

- Si la celula posee sobre población esta muere.
- Si esta no posee celulas a su alrededor, muere.

Caso Secuencial

La implementacion secuencial se da en la forma en la cual se implementa la busqueda de los vecinos.

```
PROGRAMA (){

    MATRIZ [N+1][N+1]; // MATRIZ CON BORDE
    VECINOS_VIVOS;
    VECINOS_MUERTOS;
    CELULA;
    FOR(I=1;I<N; I++){
        FOR(J=1;J<N; J++){
           CELULA = MATRIZ[I][J];

            VECINOS_VIVOS = CONTAR_VECINOS_VIVOS(CELULA,I,J, MATRIZ);

            VECINOS_MUERTOS = CONTAR_VECINOS_MUERTOS(CELULA, MATRIZ);

            IF(VECINOS_VIVOS < 2){

                ACTUALIZAR_ESTADO(CELULA, "MUERTA"); // NO HAY POBLACION.

            }ELSE IF(CELULA.ESTADO == "VIVA" && VECINOS_VIVOS > 4){

                ACTUALIZAR_ESTADO(CELULA, "MUERTA"); // SOBREPOBLACION.

            }ELSE IF(CELULA.ESTADO == "MUERTA" && VECINOS_VIVOS == 3){

                ACTUALIZAR_ESTADO(CELULA, "VIVA"); // NACE.

            }ELSE{
```

```
                // MANTIENE SU ESTADO.
            }

        }

    }

}
```

```
FUNCION CONTAR_VECINOS_VIVOS (CELULA,pos_fila,pos_columns, MATRIZ)
{

    vecinos;
    // Controlar estado del vecino y agregarlo a la lista.

    vecinos[0] = matriz[pos_fila+1][pos_columnas] // Vecino Derecha
    vecinos[1] = matriz[pos_fila+1][pos_columnas-1] // Vecino Superior
Derecha
    vecinos[2] = matriz[pos_fila+1][pos_columnas+1] // Vecino inferior
Derecha

    // Repetir para los de la izquierda y arriba y abajo.

    //Controlar si los vecinos pertenecen a la frontera, a traves de la
posicion de la filas y las columnas.

    // Retornar la cantidad de vecinos vivos

}

FUNCION CONTAR_VECINOS_MUERTOS(CELULA,pos_fila,pos_columns, MATRIZ){

    // IGUAL QUE FUNCION ANTERIOR.

    // RETORNAR LA CANTIAD DE VECINOS MUERTOS.

}



FUNCION ACTUALIZAR_ESTADO(CELULA, ESTADO){
    CELULA.ESTADO = ESTADO;
}
```

Caso MPI

```
PROGRAMA (){

    MATRIZ [N+1][N+1]; // MATRIZ CON BORDE
    MESSEGE [];
    VECINOS_VIVOS;
    VECINOS_MUERTOS;
    CELULA;


    IF(NUM_PROCESO = 0){ // PROCESO 0 ENCARGADO DE DISTRIBUIR LAS COLUMNAS
DE LA MATRIZ EN LOS PROCESOS.
        MESSEGE = DIVIDIR_COLUMNAS(MATRIZ); // OBTENER LAS COLUMNAS DE LA
MATRIZ A ENVIAR A LOS PROCESOS.
    }
    MPI_SCATTER(MESSEGE, CANTIDAD_FILAS) // COMPARTIR LAS COLUMNAS DE LA
MATRIZ.

    VECINOS_VIVOS = CONTAR_VECINOS_VIVOS(CELULA,I,J, MATRIZ);

    VECINOS_MUERTOS = CONTAR_VECINOS_MUERTOS(CELULA,I,J, MATRIZ);

    IF(VECINOS_VIVOS < 2){

        ACTUALIZAR_ESTADO(CELULA, "MUERTA"); // NO HAY POBLACION.

    }ELSE IF(CELULA.ESTADO == "VIVA" && VECINOS_VIVOS > 4){

        ACTUALIZAR_ESTADO(CELULA, "MUERTA"); // SOBREPOBLACION.

    }ELSE IF(CELULA.ESTADO == "MUERTA" && VECINOS_VIVOS == 3){

        ACTUALIZAR_ESTADO(CELULA, "VIVA"); // NACE.

    }ELSE{

        // MANTIENE SU ESTADO.
    }
}
```

Una opcion con MPI seria que cada columna es un proceso y para obtener los vecinos debe pedir las columnas adyacentes. Tener en cuenta los casos de la primera y ultima fila y columna. Se debe pedir solamente una columna adyacente.

Se haría con send y recieve y distinguiendo los casos de la primara y ultima fila y columna.