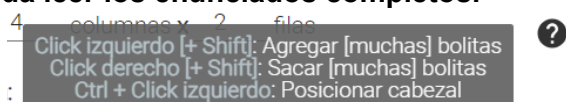


# Actividades extra en Gobstones

**Disclaimer:** Estos ejercicios pueden ser de una dificultad mayor a los trabajados en el Guía de Problemas y exceden a lo que se va a evaluar en la materia. El orden de los enunciados no se correlaciona con su nivel de dificultad. Pueden resolverlos en el orden que prefieran. Se recomienda leer los enunciados completos.

Para editar el tablero inicial revisar



Direcciones válidas: Este, Oeste, Norte, Sur.

Colores válidos: Rojo, Negro, Azul, Verde.

## Primitivas de Gobstones útiles

- Poner(<Color>): comando que pone en la celda actual una pelotita del color indicado.
- Sacar(<Color>): comando que saca de la celda actual una pelotita del color indicado.
- Mover(<Dirección>): comando que mueve el cabezal hacia la dirección indicada.
- nroBolitas(<Color>): función que retorna el valor de la cantidad de bolitas del color indicado que hay en la celda actual.
- puedeMover(<Dirección>): función que retorna un valor booleano que indica si se puede mover el cabezal hacia la dirección indicada.
- opuesto(<Dirección>): función que devuelve la dirección opuesta a la pasada por parámetro.

Operaciones aritméticas: +, -, \*, div, ^, mod<sup>1</sup>

Ejemplos:

- “4 div 2” es equivalente a escribir el número 2
- “3+2\*3” es equivalente a escribir el número 9
- “4 mod 2” es equivalente al número 0. Leer sobre la operación módulo

Operaciones booleanas: ||, &&, not

Ejemplos

- Negación: “not False” es equivalente a “True”
- Conjunción: “True && False” es equivalente a “False”
- Disyunción: “True || False || False” es equivalente a “True”

---

<sup>1</sup> En Gobstones la operación matemática módulo se escribe con la palabra clave *mod*. Esta operación devuelve el resto de dividir el primer número por el segundo. Por ejemplo, 10 mod 3 devuelve 1, 10 mod 5 devuelve 0 y 10 mod 7 devuelve 3.

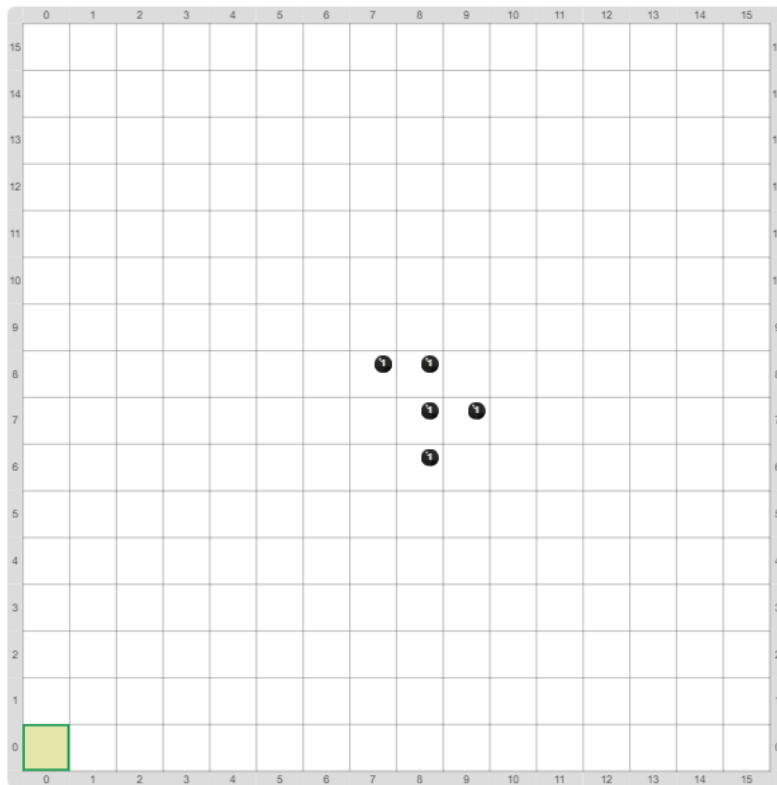
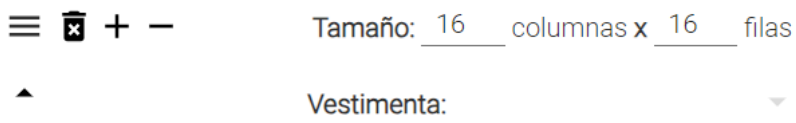
# Problema 1: El juego de la vida<sup>2</sup>

## Especificación

Pre:

- 1) Tablero es  $n \times m$  (filas x cols) con  $n \geq 16$  y  $m \geq 16$ .
- 2) En las posiciones  $(n/2, m/2-1)$ ,  $(n/2, m/2)$ ,  $(n/2-1, m/2)$ ,  $(n/2-1, m/2+1)$  y  $(n/2-2, m/2)$  hay una bolita negra.
- 3) El resto de las celdas del tablero están vacías.

A continuación un ejemplo de un posible tablero inicial:



Post:

- 1) El tablero no tiene bolitas que no sean negras.
- 2) El tablero puede tener todas sus celdas vacías o tener un patrón de celdas con bolitas negras que respete las reglas del juego.
- 3) Si una celda no está vacía entonces sólo tiene una bolita negra.

---

<sup>2</sup> [https://es.wikipedia.org/wiki/Juego\\_de\\_la\\_vida](https://es.wikipedia.org/wiki/Juego_de_la_vida)

## Enunciado

Se pide que implementen el juego de la vida, el cual consiste en lo siguiente:

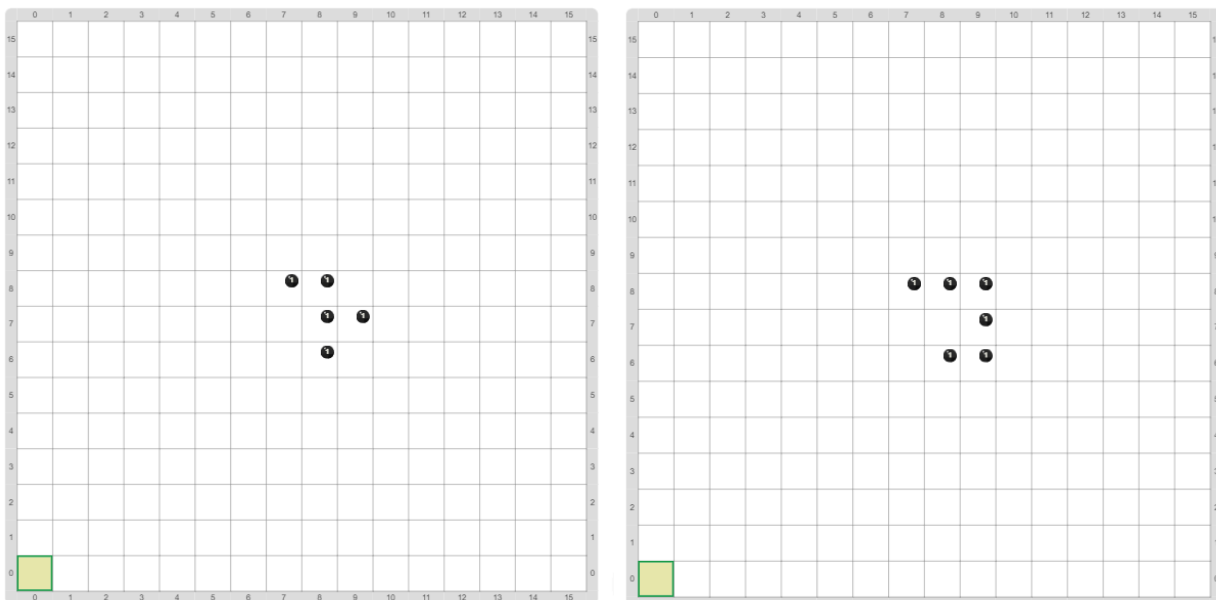
El juego de la vida es una simulación de la evolución de células que interactúan entre sí a lo largo del tiempo según unas reglas específicas. Las células tienen dos estados: vivas o muertas. En Gobstones vamos a modelar a una célula viva como una celda del tablero con una bolita negra, y a una célula muerta como una celda del tablero que esté vacía.

Dentro de la distribución del tablero vamos a referirnos a células vecinas de una célula viva a todas las células vivas que sean lindantes a ella en cualquiera de las siguientes 8 direcciones: N, NE, NO, S, SE, SO, E y O.

En este juego el estado de las células va a evolucionar por turnos. El estado de todas las células se tiene en cuenta para calcular el estado de las mismas al turno siguiente. Todas las células se actualizan simultáneamente en cada turno, siguiendo estas reglas:

- **Nace:** Si una célula muerta tiene exactamente 3 células vecinas vivas "nace" (es decir, al turno siguiente estará viva).
- **Muere:** una célula viva puede morir por uno de 2 casos:
  - **Sobrepoblación:** si tiene más de tres vecinos alrededor.
  - **Aislamiento:** si tiene solo un vecino alrededor o ninguno.
- **Vive:** una célula se mantiene viva si tiene 2 o 3 vecinos a su alrededor.

A continuación un ejemplo de evolución de un conjunto de células de un turno a otro:

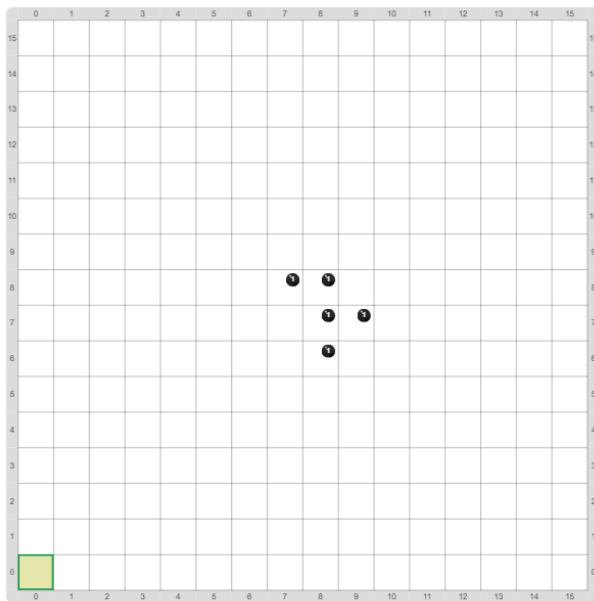


Estas reglas se deben aplicar simultáneamente a todas las células, entonces, para poder hacerlo, se sugiere utilizar un tablero de transición entre cada turno ya que para procesar un tablero y generar el tablero siguiente no se puede matar o dar vida a una célula hasta que no se hayan aplicado las reglas a todas las células del tablero. De lo contrario podrían

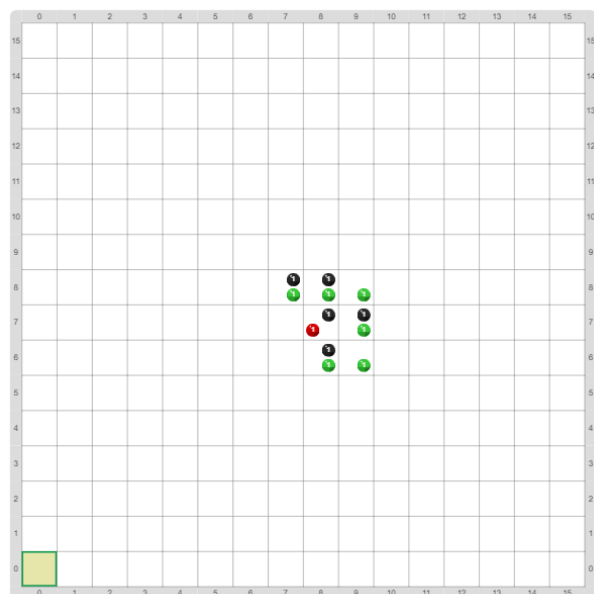
generarse coaliciones entre las células de la generación nueva con las células de la generación anterior. Por lo tanto, en lugar de eliminar o dar vida a una célula directamente, primero generaríamos un tablero de transición para marcar las células que van a vivir y las que van a morir, y luego, a partir del tablero de transición se puede generar el tablero de la próxima generación/turno.

A continuación un ejemplo de evolución utilizando un tablero de transición:

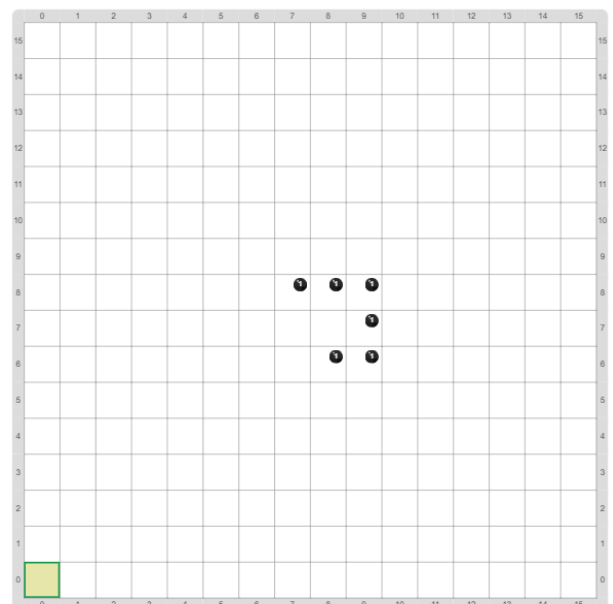
Tablero turno n



Tablero de transición



Tablero turno n+1



Se pide implementar y ejecutar en Gobstones una partida del juego de la vida por 25 turnos o más.

## Problema 2: Viborita

### Especificación

Pre:

- 1) Tablero es  $n \times m$  (filas x cols) con  $n \geq 8$  y  $m \geq 8$ .
- 2) Hay una cantidad aleatoria de celdas con sólo una bolita roja.
- 3) El cabezal está ubicado en el extremo SO del tablero.
- 4) En el extremo SO del tablero hay sólo una bolita verde.
- 5) El resto de las celdas del tablero están vacías.

Post:

- 1) El tablero no tiene bolitas que no sean verdes.
- 2) Si una celda no está vacía entonces sólo tiene una bolita verde.
- 3) La cantidad de bolitas verdes en el tablero es igual a 1 más la cantidad de bolitas rojas en el tablero inicial.
- 4) Las celdas con bolitas verdes deben tener al menos una celda lindante con una bolita verde.

### Enunciado

Se pide que implementen una versión simplificada del juego de la viborita, que consiste en lo siguiente:

La viborita va a estar representada por una serie de celdas consecutivas con una bolita verde. Las bolitas rojas representan su alimento y están en una posición fija del tablero. El tamaño inicial de la viborita es de una celda y se va a incrementar en uno por cada celda con bolita roja que visite en el tablero.

La viborita tiene que recorrer el tablero y “comer” todas las bolitas rojas. **Cuando no haya más bolitas rojas en el tablero se da por finalizado el juego.**

Una vez que la viborita visita una celda con alimento el alimento debe desaparecer de esa celda y su tamaño debe aumentar en uno.

La viborita no puede pasar por encima de ella misma, es decir no puede pasar por celdas que ya tengan una bolita verde.

La viborita puede moverse por el tablero sólo en sentido vertical y horizontal, no diagonal.

## Problema 2.1: Viborita

### Especificación

Pre:

- 1) Tablero es 16 x 16 (filas x cols).
- 2) Hay una celda con sólo una bolita roja.
- 3) El cabezal está ubicado en el extremo SO del tablero.
- 4) En el extremo SO del tablero hay sólo una bolita verde.
- 5) El resto de las celdas del tablero están vacías.

Post:

- 1) El tablero no tiene bolitas que no sean verdes.
- 2) Si una celda no está vacía entonces sólo tiene una bolita verde.
- 3) La cantidad de bolitas verdes en el tablero es igual a 17.
- 4) Las celdas con bolitas verdes deben tener al menos una celda lindante con una bolita verde.

### Enunciado

Se pide que implementen una simulación del juego de la viborita, que consiste en lo siguiente:

La viborita va a estar representada por una serie de celdas consecutivas con una bolita verde. Las bolitas rojas representan su alimento. El tamaño inicial de la viborita es de una celda y se va a incrementar en uno por cada celda con bolita roja que visite en el tablero. La viborita tiene que recorrer el tablero y “comer” la bolita roja. **Inmediatamente después de que la viborita coma la bolita roja una bolita roja nueva aparecerá en otra posición del tablero. Cuando la viborita haya comido 16 bolitas rojas se da por finalizado el juego.**

Una vez que la viborita visita una celda con alimento el alimento debe desaparecer de esa celda y su tamaño debe aumentar en uno.

La viborita no puede pasar por encima de ella misma, es decir no puede pasar por celdas que ya tengan una bolita verde.

La viborita puede moverse por el tablero sólo en sentido vertical y horizontal, no diagonal.

La secuencia de 16 bolitas rojas que irán apareciendo en el tablero se ubicarán en las siguientes posiciones:

(8,8)	(14,0)	(8,8)	(14,0)
(15,7)	(12,12)	(15,7)	(12,12)
(5,0)	(7,3)	(5,0)	(7,3)
(7,5)	(2,15)	(7,5)	(2,15)

## Problema 2.3: Viborita

### Especificación

Pre:

- 1) Tablero es 16 x 16 (filas x cols).
- 2) Hay una celda con sólo una bolita roja.
- 3) El cabezal está ubicado en el extremo SO del tablero.
- 4) En el extremo SO del tablero hay sólo una bolita verde.
- 5) Hay cuatro celdas con bolitas negras.
- 6) El resto de las celdas del tablero están vacías.

Post:

- 1) El tablero no tiene bolitas que no sean verdes o negras.
- 2) Si una celda no está vacía entonces sólo tiene una bolita verde o una bolita negra.
- 3) La cantidad de bolitas verdes en el tablero es igual a 17.
- 4) Las celdas con bolitas verdes deben tener al menos una celda lindante con una bolita verde.
- 5) Las bolitas negras están ubicadas en la misma posición que en el tablero inicial y con las mismas cantidades.

### Enunciado

Sobre la implementación anterior agregar las siguientes funciones:

Las celdas con bolitas negras representan obstáculos, la viborita puede atravesarlos sólo si su tamaño es mayor o igual a la cantidad de bolitas negras en la celda. Si el tamaño de la viborita es menor al tamaño del obstáculo la viborita debe esquivarlo.

La viborita puede salir por un extremo del tablero y entrar por el extremo opuesto.

## Problema 3: Ordenamiento

### Especificación

Pre:

- 1) Tablero es 2 x 4 (filas x cols).
- 2) Todas las celdas del tablero tienen bolitas negras en distintas cantidades, no hay celdas vacías.
- 3) El cabezal está ubicado en el extremo NO del tablero.

Post:

- 1) El tablero no tiene bolitas que no sean negras y no hay ninguna celda vacía.
- 2) Por cada celda del tablero podemos encontrar una celda con la misma cantidad de bolitas en el tablero inicial.
- 3) Observando el tablero de arriba hacia abajo, de izquierda a derecha, la cantidad de bolitas en cada celda sigue un orden ascendente.

Ejemplo:

Tablero inicial

	0	1	2	3	
1	90	40	70	35	1
0	34	3	12	16	0
	0	1	2	3	

Tablero final

	0	1	2	3	
1	3	12	16	34	1
0	35	40	70	90	0
	0	1	2	3	



## Enunciado

Implementar un programa que ordene de forma ascendente la cantidad de bolitas por celda.

Desafío: El programa debe ordenar todas las celdas en 64 intercambios como máximo.

## Problema 3.1: Ordenamiento

### Especificación

Pre:









- 1) Tablero es 2 x 4 (filas x cols).
- 2) Todas las celdas del tablero tienen bolitas de distintos colores en distintas cantidades, no hay celdas vacías.
- 3) Cada celda tiene bolitas de un único color.
- 4) El cabezal está ubicado en el extremo NO del tablero.

Post:

- 1) El tablero no tiene ninguna celda vacía.
- 2) Por cada celda del tablero podemos encontrar una celda con la misma cantidad de bolitas y del mismo color en el tablero inicial.
- 3) Observando el tablero de arriba hacia abajo, de izquierda a derecha, la cantidad de bolitas en cada celda sigue un orden ascendente. Y respeta el orden Azul, Negro, Rojo, Verde.

Ejemplo:

Tablero inicial

	0	1	2	3	
1					1
0					0
	0	1	2	3	

Tablero final

	0	1	2	3	
1	3	3	16	18	1
0	35	35	41	50	0
	0	1	2	3	

## Enunciado

Implementar un programa que orden a las celdas según el orden numérico ascendente y según el orden de colores Azul, Negro, Rojo, Verde.