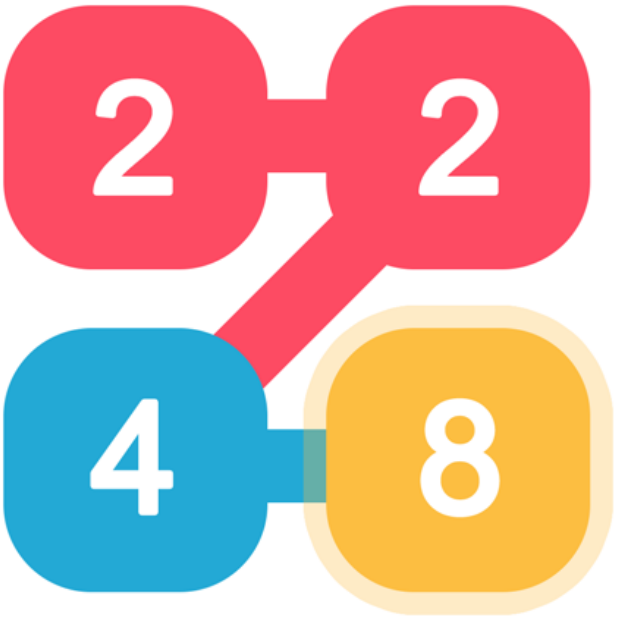
BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila

Proyecto 1
2248

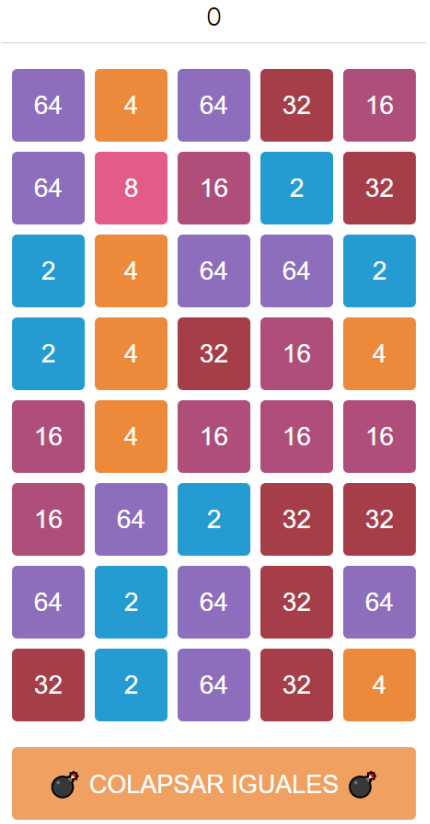


BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila

El 2248 es un juego de puzzle y estrategia cuyo objetivo es combinar bloques numerados para crear bloques de mayor valor y ganar puntos. Para tal fin se cuenta con la siguiente interfaz de juego:

La interfaz cuenta con:

- Una grilla de 5x8 con celdas que contienen elementos
- Un botón llamado **COLAPSAR IGUALES**: Al apretar este botón los cuadrados adyacentes con el mismo número se eliminan, pero no suma puntos, hace que entren nuevos cuadrados.
- Un botón llamado **CAMINO MAXIMO**: Al apretarlo se muestra el camino que resulta en la mayor potencia de dos posible con la configuración actual.
- Un botón llamado **ADYACENTE MAXIMO**: Al apretar el mismo se muestra el camino que resulta en la mayor potencia de dos posible tal que queda adyacente a un elemento del mismo valor.
- Un espacio destinado a la sumatoria de puntos del juego.
- Cuadrados: La interfaz del juego cuenta con cuadrados que contienen números que son dos o potencia de dos, estos pueden combinarse para formar números más grandes.
- Movimientos: Se puede interactuar con el juego mediante movimientos en las cuatro direcciones, abajo , arriba, izquierda, derecha y en diagonal, lo que unirá los cuadrados correspondientes y se sumarán los números que lo conforman.
- Puntuación: Cuando colapsen 2 o más cuadrados, hay 2 resultados visibles, el primero te mostrará el resultado antes de hacer la jugada que desees, y otra que es la puntuación total, esta última se hará invisible mientras tengas cuadrados unidos. La puntuación a obtener es 2248 pero podes seguir jugando para tener una mayor puntuación.



BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila

Para la movida básica del juego, que consiste en trazar un camino conectando dos o más números, donde cada número del camino se conecta con uno siguiente que es adyacente (horizontal, vertical o diagonal) en la grilla; donde el camino arranca conectando dos números iguales, y luego cada número del camino es igual al anterior o es la potencia de dos siguiente a dicho número anterior, se propuso el siguiente algoritmo:

1. calcular la suma del camino y computar la potencia de dos correspondiente a la suma,
2. guardar el valor de la suma en la grilla en la celda que contiene al último elemento del camino,
3. borrar el resto de las celdas del camino,
4. hacer que caigan las celdas no vacías de la grilla de elementos hacia la parte inferior del tablero,
5. se computan potencias de dos aleatorias para completar las celdas vacías,
6. se completa el tablero.

El siguiente código (resumido) representa la implementación del algoritmo en Prolog y mapea, línea a línea, cada uno de los incisos de la lista previa a este párrafo:

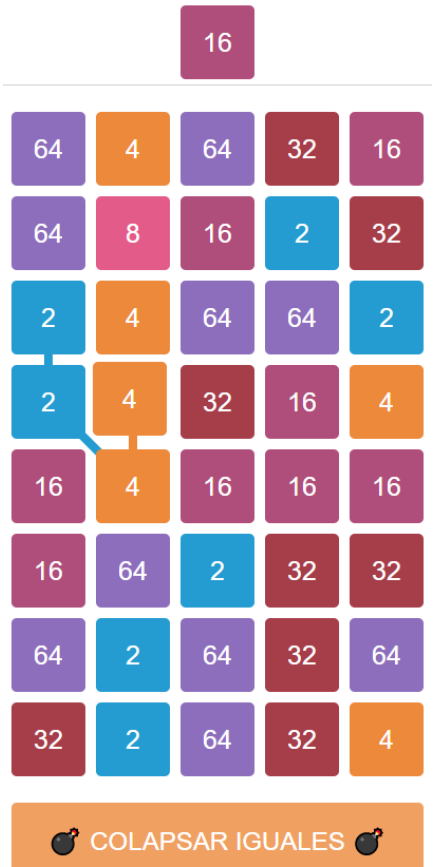
```
/**
 * join(+Grid, +NumOfColumns, +Path, -RGrids)
 *
  * RGrids es la lista de grillas representando el efecto, en etapas, de
combinar las celdas del camino Path
  * en la grilla Grid, con número de columnas NumOfColumns. El número 0
representa que la celda está vacía.
 */
join(Grid, NumOfColumns, Path, RGrids):-
    suma_camino_pot_dos(GrillaAgrupada, Path, Suma),
    set_suma_grilla(GrillaAgrupada, Path, Suma, GrillaSuma),
    set_ceros_grilla(GrillaSuma, PathSinUltimo, GrillaCeros),
    burbujear_ceros(GrillaCeros, GrillaBurbujeada),
    generar_rango(GrillaSuma, LimInferior, LimSuperior),
    reemplazar_ceros(GrillaBurbujeada, LimInferior, LimSuperior,
GrillaCompleta),
    RGrids = [GrillaSumaAplanada, GrillaCerosAplanada,
GrillaBurbujeadaAplanada, GrillaCompletaAplanada].
```

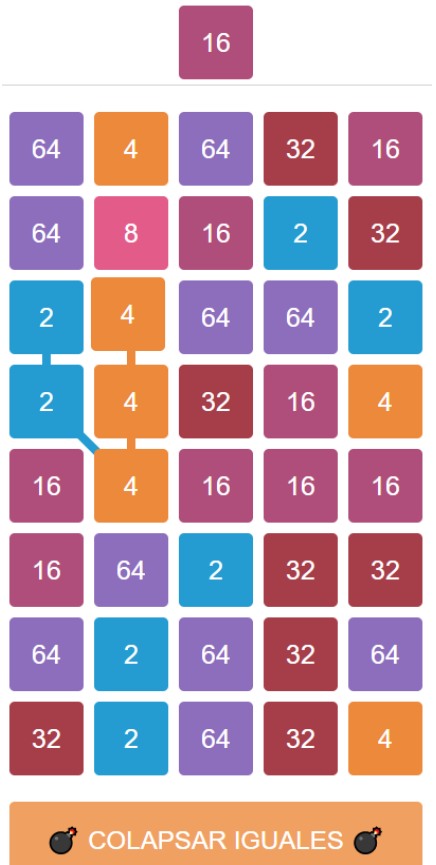
BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila

El predicado `join`, como expresa el comentario, recibe una `Grid` (una lista que contiene a los elementos de la grilla de juego), `NumOfColumns` y un `Path` (una lista de coordenadas que expresan las ubicaciones que conforman el camino que se desea poner en juego). Con esta información, y el cuerpo del predicado, se computa y unifica en `RGrids`, una lista de Grids (contienen también los elementos de la grilla) que denotan la evolución del juego.

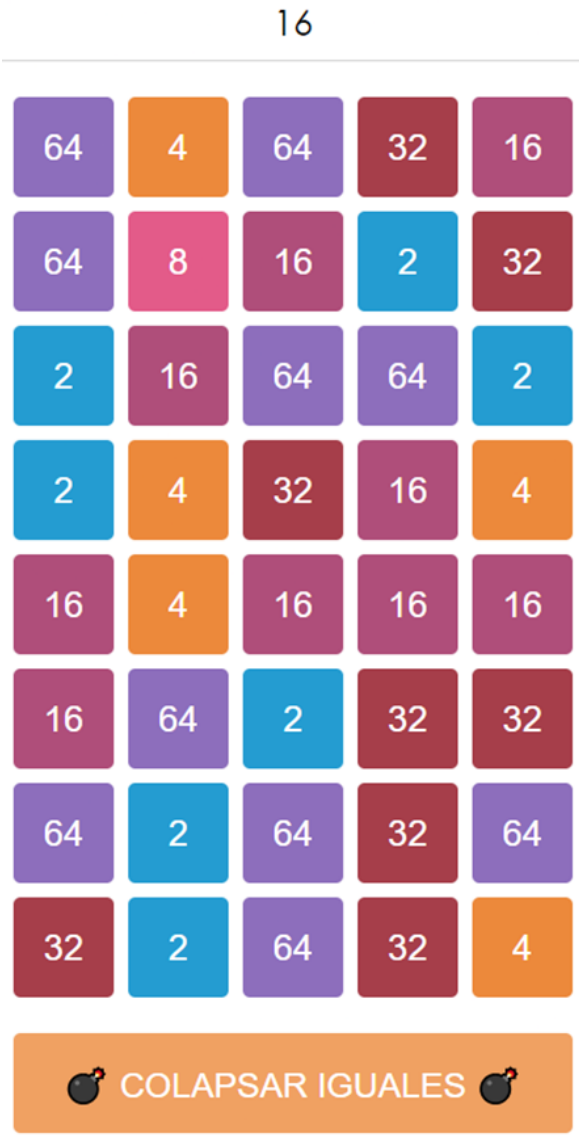
- Dado que `Grid` es una lista de elementos, se utiliza el predicado `agrupar/3` que busca unificar en `GrillaAgrupada` los elementos de `Grid` "en forma de matriz";
    - Sea `Grid = [a, a, a, b, b, b, c, c, c]` entonces `GrillaAgrupada = [[a, a, a], [b, b, b], [c, c, c]]`.

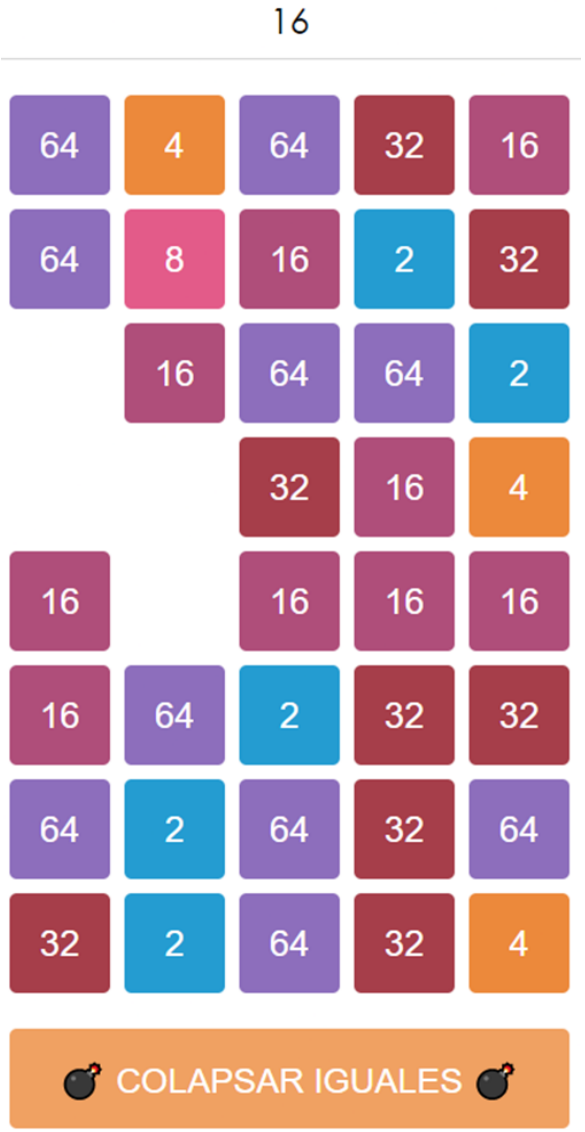
- `suma_camino_pot_dos/3` computa, a partir de las coordenadas de las celdas que conforman un camino, la suma que surge a partir de los elementos en cuestión. Dado que, como fue mencionado, en la grilla únicamente hay potencias de dos, en caso de que la suma no de una potencia de dos, se computa la menor potencia de dos mayor o igual que la suma que resultó de sumar los elementos.
    - En la imagen de la izquierda se observa un camino conteniendo a los elementos [2, 2, 4, 4] que, al sumarlos, se obtiene el número 12. Dado que este no es una potencia de dos, la suma parcial al momento, corresponde con 16, que es la potencia de dos más próxima y mayor que 12.
    - En la imagen de la derecha, por el contrario, se observa un camino conteniendo a los elementos [2, 2, 4, 4, 4] que, si se suman, se obtiene el número 16. Como 16 sí es una potencia de dos, la suma parcial conserva ese resultado.





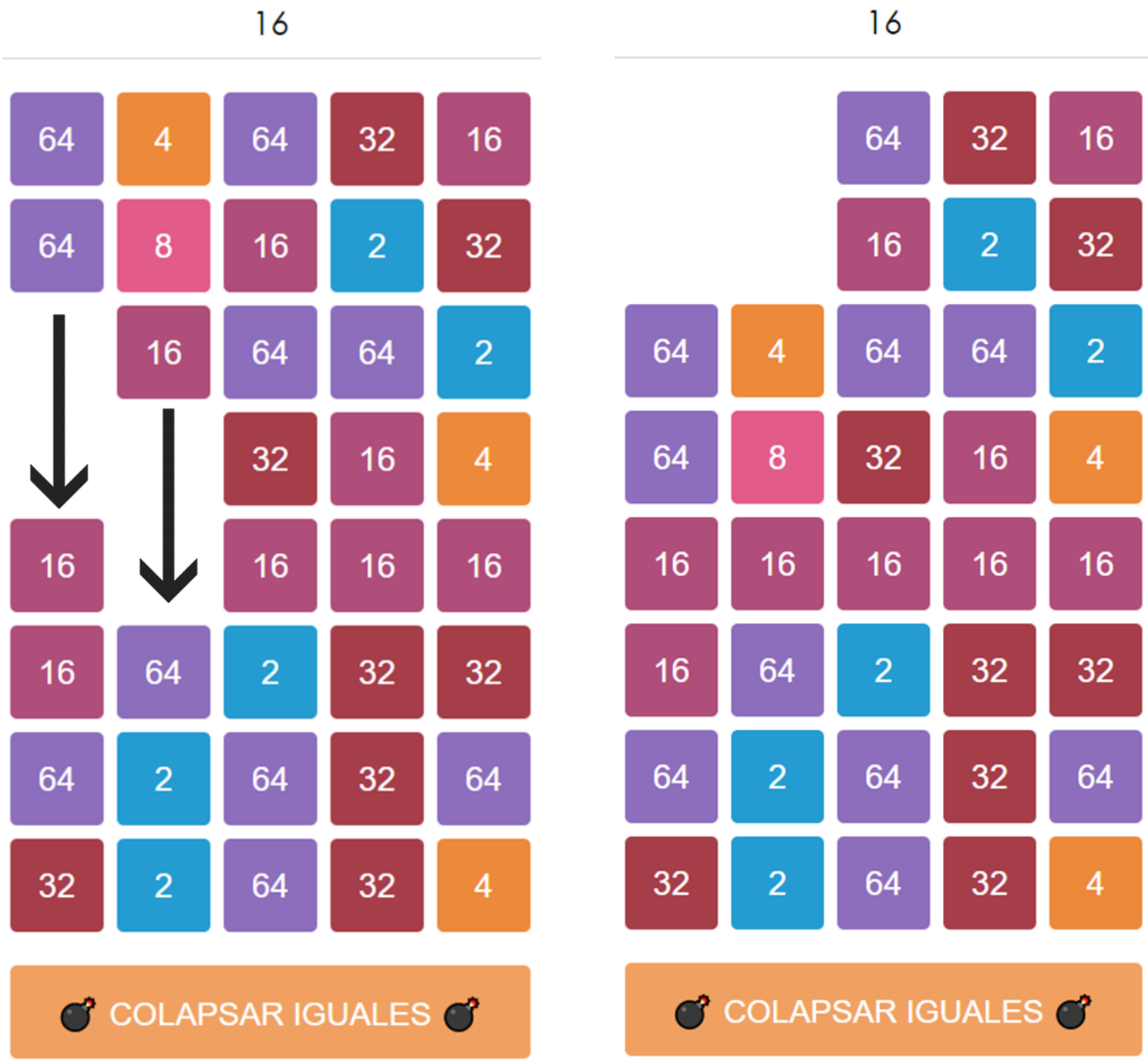
- `set_suma_grilla/4` setea, en la celda que contiene al último elemento del camino, el resultado de la suma de los elementos del camino elegido computada en `suma_camino_pot_dos/3`. En la siguiente imagen puede observarse como, si se decidiera jugar el camino de la imagen de la derecha del punto anterior, en la posición del 4, se sobre escribe un 16, ya que es el resultado de la suma del camino.
  [Imagen de la izquierda]
- `set_ceros_grilla/3` setea en cada elemento referido por las posiciones del camino, un cero. En la lógica del juego, un cero representa las celdas vacías por lo que desaparecen de la vista. Siguiendo el mismo camino de ejemplo, la vista del caso es la siguiente: **[Imagen de la derecha]**



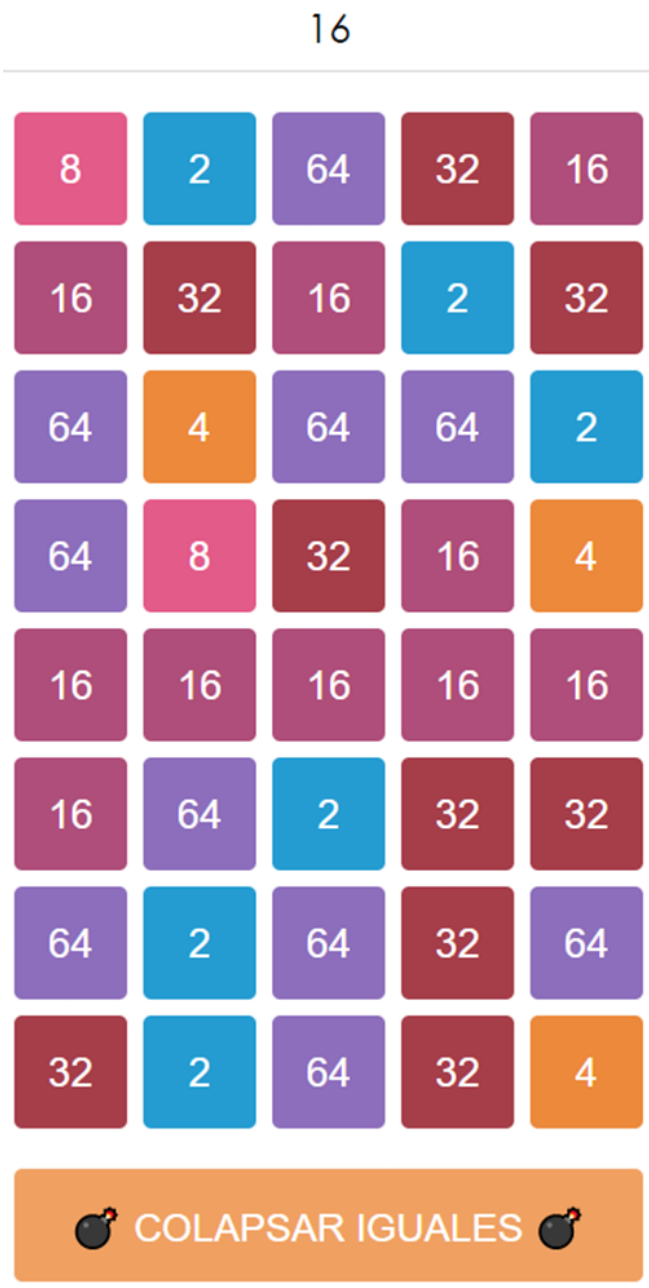


BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila

- `burbujear_ceros/2` busca hacer que la grilla tenga todos sus elementos no vacíos (es decir, distintos de cero), sobre la base del tablero, pero sin alterar el orden. Se busca, simplemente, “aplicar gravedad” sobre las celdas de la matriz; se “burbujean” los ceros hacia el extremo superior.



- `reemplazar_ceros/4` permite completar la grilla. En cada posición de la grilla donde haya un elemento vacío (o cero), se ubica una potencia de dos generada aleatoriamente a partir de los valores que se encuentran en juego.

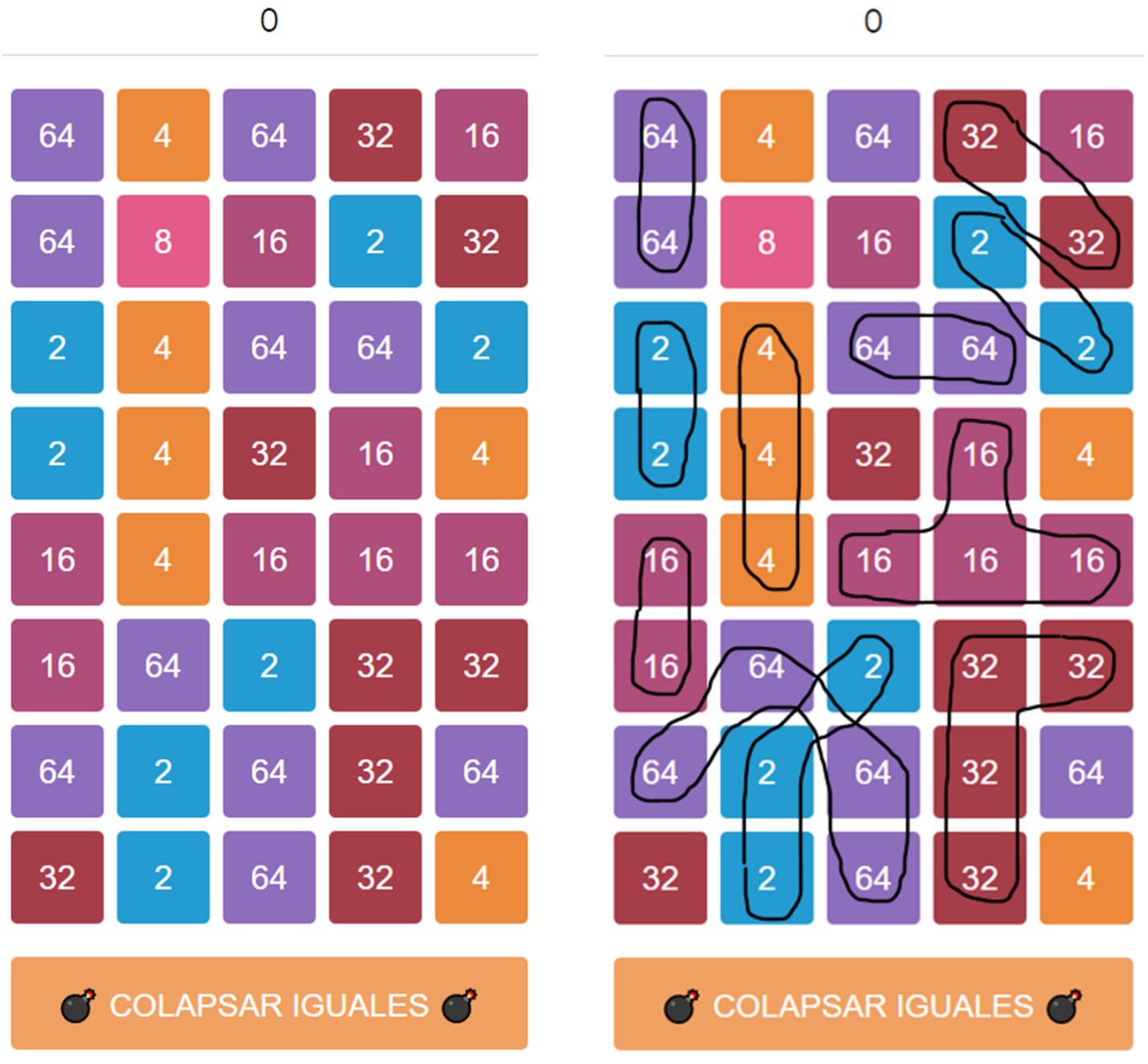


Los predicados explicados hasta el momento, son utilizados para la realización y compleción de una movida básica del juego. Sin embargo, como se mencionó anteriormente, hay una funcionalidad que agrega otro tipo de movimiento: 💣**COLAPSAR IGUALES**💣.

BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila

Esta funcionalidad, se ejecuta después de la interacción con el botón ubicado en la parte inferior de la interfaz del juego. Dada una grilla de elementos, luego de interactuar con el botón, se colapsan todos los elementos que son adyacentes a celdas que contienen el mismo elemento; la adyacencia, como fue mencionado, se contempla en vertical, horizontal y diagonal.

Dada la siguiente situación de juego (izquierda), se delimitan con negro (derecha) aquellos elementos que tienen al menos una celda adyacente conteniendo el mismo valor.



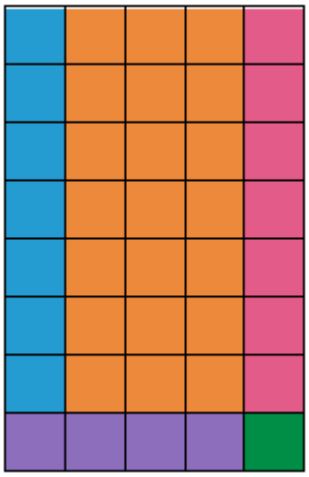
Luego, una vez clickeado el botón, para cada camino representado con negro, se aplica la misma lógica que la movida básica:

1. se realiza el cómputo de la suma del camino,
2. se setea esa suma en uno de los elementos (este paso no se ve reflejado en la animación del juego por decisiones de diseño ya que se vuelve algo confuso siendo tantos grupos, en ocasiones, para colapsar),
3. el resto de las celdas del camino se setean con cero,
4. se aplica el efecto de gravedad,
5. se completa la grilla con las nuevas potencias obtenidas aleatoriamente.

```
/*
 * booster(+Grilla, +CantColumnas, -GrillaEfectoResultante)
 *
 * GrillaEfectoResultante es la lista de grillas representando el efecto, en etapas, de
combinar las celdas de elementos iguales adyacentes
 * en la grilla Grid, con número de columnas NumOfColumns. El número 0 representa que
la celda está vacía.
 */
booster(Grid, NumOfColumns, RGrids):-
    grupos_iguales(Grid, NumOfColumns, GrillaMatriz, GruposCaminos),
    aplicar_efecto(GrillaMatriz, NumOfColumns, GruposCaminos, GrillasEvolucion),
    ultimo(GrillasEvolucion, GrillaCeros),
    burbujear_ceros(GrillaCeros, GrillaBurbujeada),
    generar_rango(GrillaBurbujeada, LimInferior, LimSuperior),
    reemplazar_ceros(GrillaBurbujeada, LimInferior, LimSuperior, GrillaCompleta),
    aplanar(GrillaCeros, GrillaCerosAplanada),
    aplanar(GrillaBurbujeada, GrillaBurbujeadaAplanada),
    aplanar(GrillaCompleta, GrillaCompletaAplanada),
    RGrids = [GrillaCerosAplanada, GrillaBurbujeadaAplanada, GrillaCompletaAplanada].
```

- `buscar_caminos_boostear/6` busca todos los elementos que tienen al menos un adyacente igual a él. La búsqueda siempre se realiza “hacia adelante” ya que, buscar adyacente hacia atrás no cobra ningún sentido porque solo resultaría en caminos redundantes. Para ello se encontró un patrón y una consecuente división en cinco casos de los elementos de la grilla:
    1. el elemento se encuentra en la primer columna pero no en la última fila; se deben observar las posiciones adyacentes a la derecha, hacia abajo, la diagonal inferior hacia la derecha,
    2. el elemento se encuentra en una columna no extrema y tampoco en la última fila; se deben observar las posiciones adyacentes a la derecha, la diagonal inferior izquierda, hacia abajo y la diagonal inferior derecha,

3. el elemento se encuentra en la última columna pero no en la última fila; los únicos elementos que deben chequearse se encuentran en las posiciones diagonal inferior izquierda y hacia abajo,
4. el elemento se encuentra en la última fila, pero no la última columna; sólo debe chequearse el elemento ubicado a su derecha,
5. el elemento está en la última fila y última columna (último elemento); no corresponde controlar ninguna posición.



Dado este código esquema respetando el tamaño de la grilla de juego, se observa un mapeo de la siguiente manera:

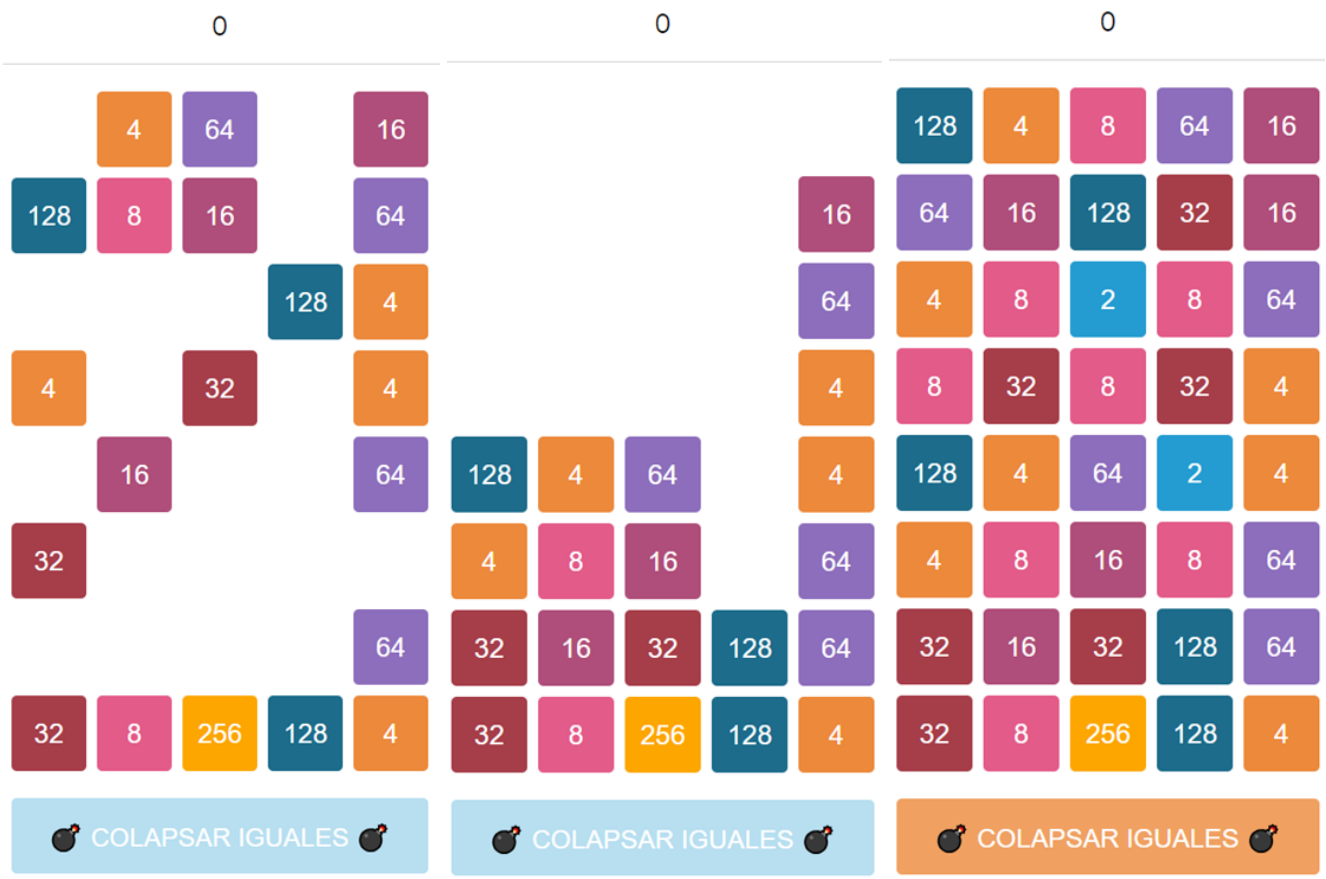
- caso 1,
- caso 2,
- caso 3,
- caso 4,
- caso 5.

- `grupos_iguales/4` se compone de:
  - `eliminar_listas_un_elemento/2` elimina todas las apariciones de caminos que están compuestos por un solo elemento; cobra sentido dado que `buscar_caminos_boostear/6` puede crear caminos de un solo elemento (la posición actual, cuando no tiene adyacentes iguales).
  - `buscar_grupos_booster/3` unifica grupos de posiciones. Dada una lista de caminos y una posicion, determina en cuáles de esos caminos está presenta la posición y para conocer el grupo de elementos en el que está contenida la misma.
  - `concatenar_caminos/2` concatena en una sola lista, todos los caminos computados en `buscar_caminos_booster/6` que compartan al menos un elemento.
- `aplicar_efecto/2` aplica la movida básica, camino a camino, de los grupos de elementos iguales y adyacentes. La movida básica, en esta instancia, no es completa sino que simplemente contempla `set_suma_grilla/4` y `set_ceros_grilla/3`. El efecto de gravedad, se aplica fuera de este predicado y, en consecuencia, también la compleción de la grilla con las potencias de dos aleatorias. Retorna una lista conteniendo las evoluciones por las que pasa la grilla luego de clickeado el botón y hasta el momento.
- Dado que `aplicar_efecto/2` computa una lista de listas de evoluciones, con `último/2` se rescata la última modificación a la que la grilla fue sometida.

BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila

- Los pasos restantes (`burbujear_ceros/2`, `generar_rango/3`, `reempleanzar_ceros/4`, `aplanar/2`) respetan la lógica de resolución de una movida básica.

A continuación, la evolución de la grilla luego de aplicar el efecto del botón 💣**COLAPSAR IGUALES**💣



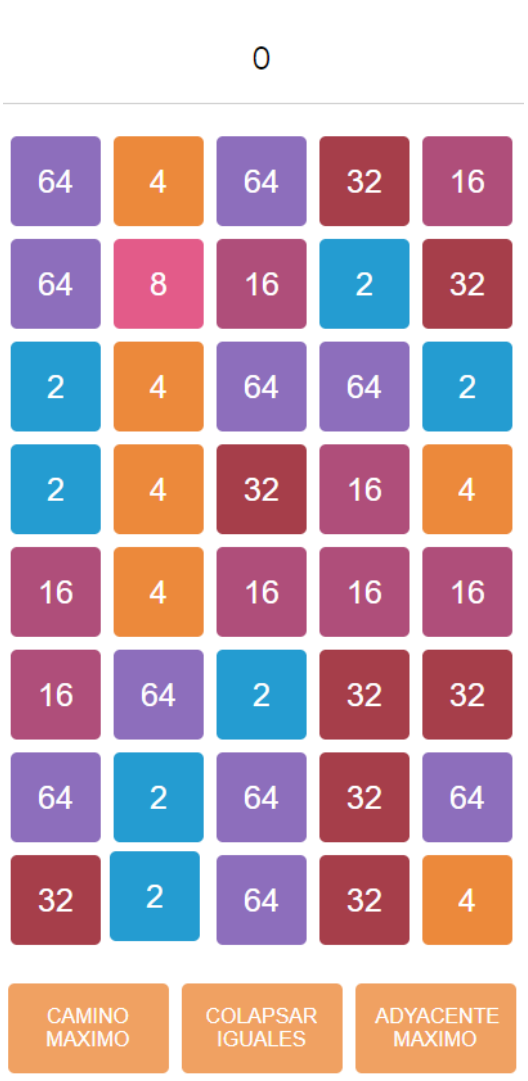
BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila

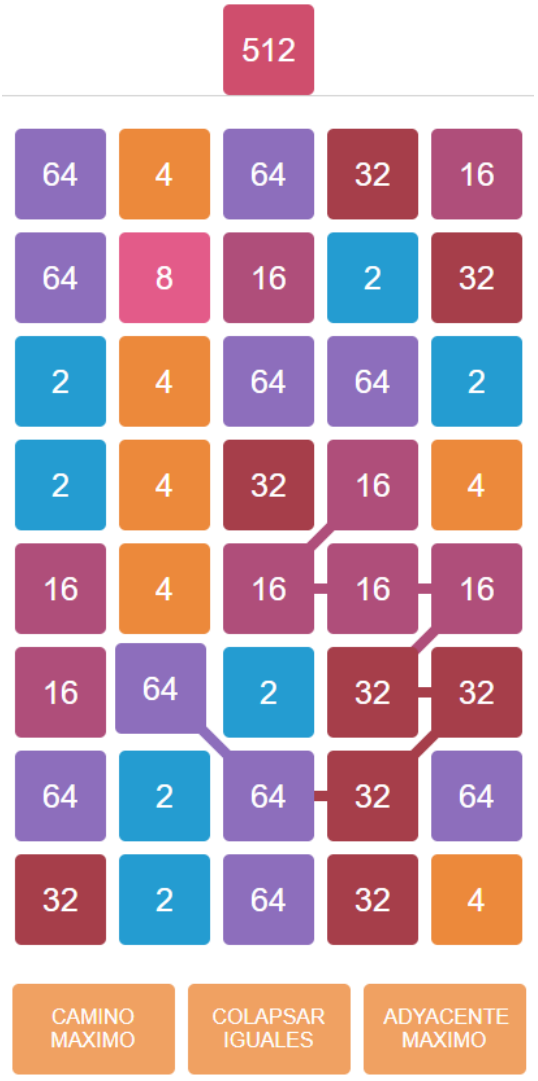
Esta funcionalidad, se ejecuta después de la interacción con el botón ubicado en la parte inferior de la interfaz del juego. Dada una grilla de elementos, luego de interactuar con el botón, se muestra el camino que une elementos de forma tal que genere la mayor potencia de dos posible.

```
/*
 * movida_maxima(+Grilla, +CantColumnas, -SumaCaminoMaximo, -CaminoMaximo)
 *
 * Calcula el camino (CaminoMaximo) que consigue el mayor número (SumaCaminoMaximo) a
partir de la configuración actual
 */
movida_maxima(Grilla, CantColumnas, SumaCaminoMaximo, CaminoMaximo):-
    agrupar(Grilla, CantColumnas, GrillaAgrupada),
    buscar_caminos_grilla(Grilla, Grilla, GrillaAgrupada, CantColumnas, 0, Caminos),
    aplanar(Caminos, CaminosAplanados),
    tamanio(CaminosAplanados, Tamanio),
    Tamanio > 0,
    !,
    max(Grilla, GrillaAgrupada, CantColumnas, CaminosAplanados, 0, Maximos),
    ultimo(Maximos, CaminoMaximo),
    suma_camino_pot_dos(GrillaAgrupada, CaminoMaximo, SumaCaminoMaximo).
```

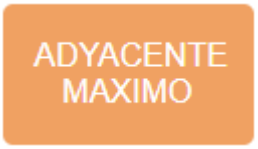
- `buscar_caminos_grilla/6` buscar todos los caminos que se pueden formar con los bloques de la grilla. Recorre celda por celda y para cada una:
    - `buscar_caminos_celda/8` busca todos los caminos que comienzan con la posición que alberga a la potencia de dos pasada. Utiliza la lógica de "visitados" para que no pase de nuevo por los mismos que ya se visitaron.
    - `crear_caminos/9` crea los caminos que pasan por un bloque específico contemplando que, de todos los adyacentes que tiene, el adyacente en particular sea válido para la funcionalidad (que el número allí alojado sea el mismo que el parametrizado o una potencia de dos superior inmediata).
    - `coordenadas_adyacentes/4` a partir de la coordenada pasada y el tamaño y la cantidad de columnas como restricciones dimensionales, va a unificar todas las coordenadas adyacentes a la coordenada antes nombrada.

    - `adyacentes_validas/5` a partir de una grilla representada como una lista de elementos de la grilla, lo que se hará es computar y unificar aquellas coordenadas que sean adyacentes a las coordenadas pasadas, que cumplan con la restricción de que tengan una potencia de dos igual a la que contiene la coordenada pasada, o una potencia inmediatamente superior a ella.
- `check_positions/4` predicado auxiliar que unifica las coordenadas chequeadas, que recibe y no se caen de la matriz siguiendo las restricciones del tamaño.
- `max/6` a partir de una grilla recibida, con su respectiva configuración y una lista de máximos la cual contiene los caminos, compara el valor de cada uno de los caminos con máximo actual, luego va a unificar en máximo todos los caminos que tengan o produzcan una suma mayor que el máximo actual de la lista.
- Los predicados que se usan y no están mencionados en este apartado respetan la lógica de resolución antes hecha y explicada, son predicados que son nativos del lenguaje, o son predicados de autoría propia que no se incluyen para no entorpecer la lectura del documento.
- Si se quisiera realizar la jugada mostrada debe hacerse click sobre el último elemento del camino; caso contrario presionar la tecla de escape.





BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila



Esta funcionalidad, se ejecuta después de la interacción con el botón ubicado en la parte inferior de la interfaz del juego. Dada una grilla de elementos, luego de interactuar con el botón, se muestra el camino que une elementos de forma tal que genere el bloque con la mayor potencia de dos posible, tal que quede adyacente a otro con el mismo valor que la potencia computada.

```
/*
 * maximo_adyacente(+Grilla, +CantColumnas, -SumaMaximoAdyacente, -CaminoAdyacente)
 *
 *  Calcule y y unifica en CaminoAdyacente el camino que consiga generar el número más grande posible adyacente a otro igual (preexistente)
 *  SumaMaximoAdyacente unifica con la menor potencia de dos mayor o igual a la suma del camino CaminoAdyacente
 */
maximo_adyacente(Grilla, CantColumnas, SumaMaximoAdyacente, CaminoAdyacente):-
    agrupar(Grilla, CantColumnas, GrillaAgrupada),
    buscar_caminos_grilla(Grilla, Grilla, GrillaAgrupada, CantColumnas, 0, Caminos),
    aplanar(Caminos, CaminosAplanados),
    length(CaminosAplanados, Tamanio),
    Tamanio > 0, !,
    max_list(Grilla, MaxPotencia),
    %Elimino los caminos de un solo elemento
    findall(C,
            (member(C, CaminosAplanados), length(C, T), T > 1),
            CaminosAplanadosAux),
    %Elimino los caminos que computan sumas superiores al mayor elemento de la grilla
    findall(C,
            (member(C, CaminosAplanadosAux),
             suma_camino_pot_dos(GrillaAgrupada, C, SumaC),
             (SumaC < MaxPotencia; SumaC = MaxPotencia)),
            CaminosSumaMenoresMaximoRepetidos),
    elimina_repetidos(CaminosSumaMenoresMaximoRepetidos, CaminosSumaMenoresMaximo),
    encontrar_camino_retornar(GrillaAgrupada,  CantColumnas,  CaminosSumaMenoresMaximo, CaminoAdyacente),
    length(CaminoAdyacente, TamanioCaminoRetorno),
    TamanioCaminoRetorno > 0,
    !,
    suma_camino_pot_dos(GrillaAgrupada, CaminoAdyacente, SumaMaximoAdyacente).
```

BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila

La lógica de este predicado se describe, sintéticamente, de la siguiente manera:

- Se buscan todos los caminos existentes de la grilla.
- Se aplanan porque `buscar_caminos_grilla/6` unifica en `Caminos` una lista de lista de caminos.
- Se busca la mayor potencia de dos posible ya que no son de interés aquellos caminos que generen una potencia de dos mayor a la que esté en juego.
- Se buscan todos los caminos que tengan longitud mayor que uno, ya que para que se juegue un camino debe tener al menos dos elementos.
- Se buscan, de todos los caminos posibles con longitud mayor o igual que dos, aquellos que no generen un bloque con una potencia de dos mayor que la que esté actualmente en juego.
- Se busca de forma efectiva el camino que satisfaga la funcionalidad y la suma de su consecuente bloque resultante. Esta búsqueda se funda en el predicado `encontrar_camino_retornar/4`.

- `encontrar_camino_retornar/4` dada una grilla de elementos, la cual está en forma de matriz y una lista cuya suma es igual al valor de alguno de los elementos adyacentes del último bloque, busca el camino que sume el mayor valor. Busca, para tal fin, todas las potencias que se encuentren en juego en la configuración actual.
    - `camino_retornar/4` dadas las potencias de dos que se encuentran en el juego y a partir de los caminos candidatos que son los caminos que pueden ser el camino con el máximo adyacente, lo que va a hacer es unificar el camino máximo adyacente en el camino que se retorna como resultado; de todas las potencias de dos en juego, conociendo cual es la mayor corriente para satisfacer la funcionalidad, se consulta cuáles de los caminos candidatos computan una suma que sea igual a la potencia de dos en cuestión.
    - `validar_resolucion_camino/3` determina si el camino que recibe genera un resultado válido, teniendo en cuenta que el bloque máximo adyacente puede ser adyacente como resultado de la aplicación de gravedad de la grilla. Para ello:
        - `coordenada_final_resolucion/2` unifica en coordenada final, la coordenada donde se encuentra el bloque final del camino que recibe el predicado, luego de jugado el camino y el efecto de gravedad aplicado.
        - `tiene_adyacente_valido/4` determina si, adyacente a un bloque con una coordenada particular dada, se encuentra otro bloque con valor igual al inicialmente mencionado.
        - `virtualizar_jugada/3` unifica en grilla virtual como quedaría la grilla a retornar después de que se juegue el camino que recibe.

BACA, Nahuel
PROLYGIN GONZÁLEZ, Mayra Ludmila