INFORME DEL PROYECTO ETAPA 2

Incluye la documentación correspondiente a la implementación del proyecto.

2248 Game.

PARTE 2

FUNCIONALIDADES:

Esta segunda etapa agrega nuevas ayudas para el jugador, contando con un botón “Ayuda movida máxima”, el cual al ser presionado calcula y muestra el camino que consiga el mayor puntaje en la grilla actual. Además, agrega otro botón “Ayuda máximos iguales adyacentes” que, al ser activado, calcula y muestra el camino que consiga generar el número más grande posible adyacente a otro igual (preexistente).

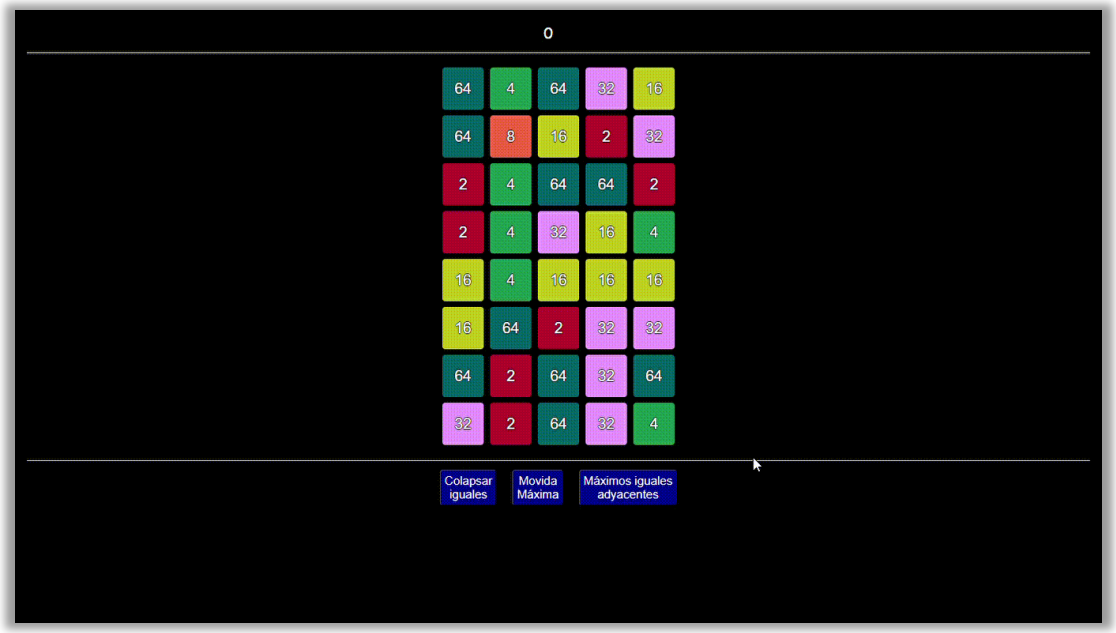
Si hay más de un camino que cumpla la condición, se toma cualquiera de ellos.

Al formar diferentes caminos, se mostrarán carteles felicitando y motivando al jugador, dependiendo de la longitud del camino realizado.

Para mejor visualización de los caminos, el último bloque de cada uno se encuentra destacado, remarcado con un contorno dorado; y su tamaño es sutilmente mayor al de los demás.

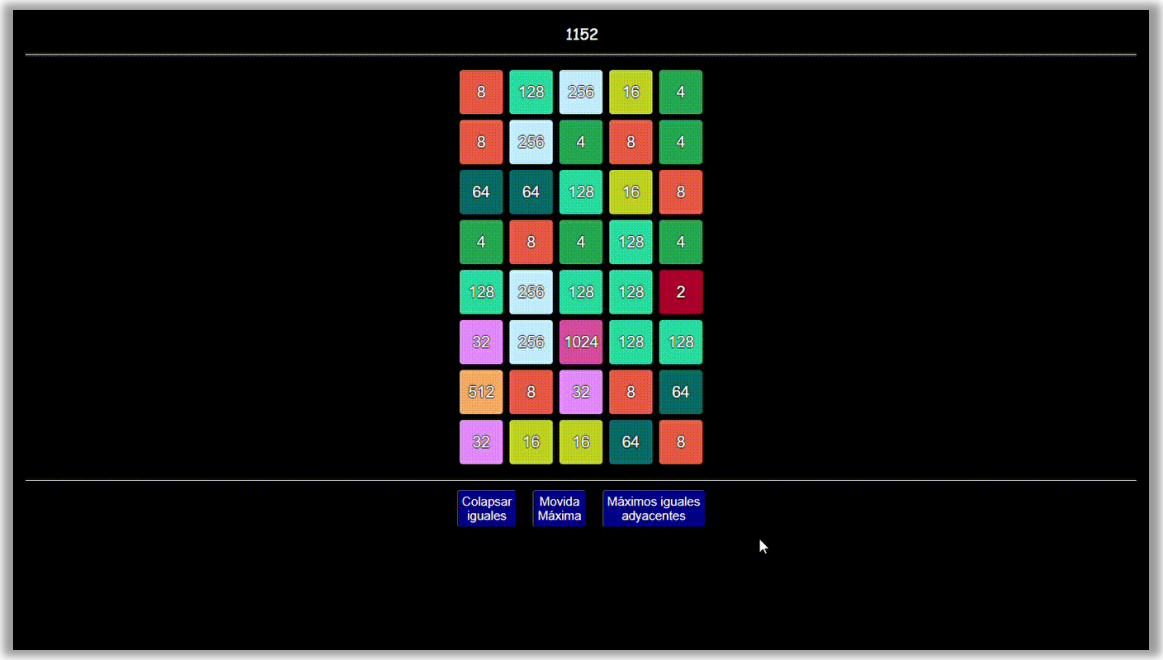
Ayuda Movida Máxima:

Al presionar el botón, el jugador podrá ver en pantalla el camino trazado según la funcionalidad descripta previamente. El último bloque de la grilla estará destacado entre los demás para poder ser presionado por el jugador. Al hacerlo, el proceso es el mismo que si se hubiese trazado el camino manualmente, brindando los puntos correspondientes a dicho camino.

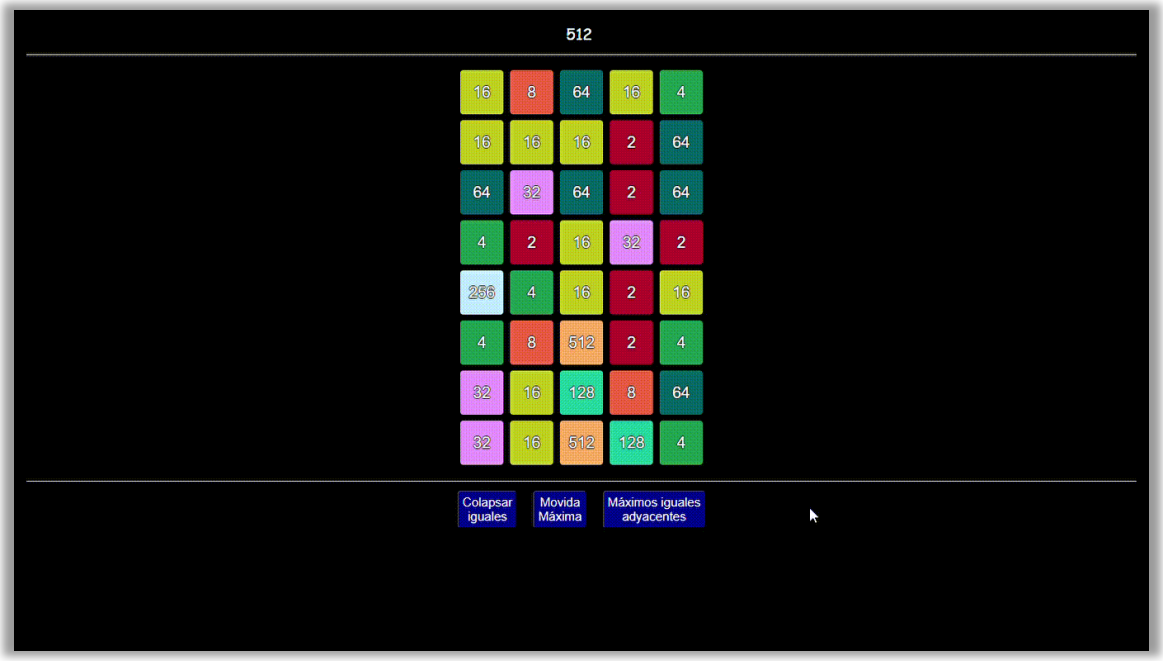


Ayuda Máximos Iguales:

Al presionar el botón, se mostrará sobre la grilla un camino que generará el mayor número posible, adyacente a otro igual ya existente. El último bloque del camino se mostrará destacado para poder ser clickeado por el jugador, eliminando el camino y sumando los puntos correspondientes.



Encuentra un camino que forma un 1024, el cual tiene un 1024 adyacente.



Encuentra un camino que forma un 256, que no tiene otro adyacente, pero al aplicar gravedad cae sobre un bloque 256.

IMPLEMENTACIÓN EN PROLOG:

Ayuda movida máxima:

Predicado maxMove/3:

maxMove(+Grid, +NumOfColumns, -Path).

Este predicado calcula y devuelve el camino que consiga el mayor número a partir de la grilla "Grid" actual.

Llama a ***shellMaxMove/6*** que se encarga de encontrar ese camino máximo para toda la grilla, recorriéndola por índice.

Predicado shellMaxMove/6:

shellMaxMove(+Grid, +NumOfColumns, +Index, +ActualPath, +ActualScore, -Path).

Este predicado recorre recursivamente la grilla encontrando todos los caminos posibles, terminando en el caso base, que se da cuando el índice supera los límites de la grilla.

Una vez que tiene todos los caminos (*AllPaths*), ***getMaxPathFromList/5*** se encarga de devolver el camino con el mayor puntaje, junto con su puntaje. El ***findall/*3** se encarga de buscar todos los pares [camino, puntaje] de todos los caminos obtenidos a partir de un índice, por el predicado ***recMaxMove/8.***

Luego, si el MaxScore es mayor al puntaje actual, entonces se reemplaza el camino actual con el nuevo mayor. Se incrementa el índice para volver a llamar al predicado con el bloque siguiente en la grilla y repetir el proceso.

Predicado recMaxMove/8:

Encuentra todos los posibles caminos a partir de un índice dado, similar a los recorridos ya hechos en funcionalidades anteriores.

Si el bloque adyacente al actual no fue recorrido, se agrega a la lista del camino actual. Luego, se actualiza el score, sumando el puntaje del bloque agregado y llamándose recursivamente para seguir armando el camino a partir de cada bloque adyacente agregado. Llega al caso base cuando no haya más adyacentes por recorrer. Una vez terminado, si es *ActualScore* es mayor al puntaje máximo obtenido hasta el momento, se actualiza el camino con el camino actual obtenido, y lo mismo se hace con el puntaje.

Para más detalle acerca de los predicados auxiliares utilizados en la implementación, consultar la documentación presente en los comentarios del código Prolog.

Aclaración: Grillas con muchos bloques del mismo valor generan que el predicado tarde mucho tiempo en generar el camino, por la propia complejidad temporal del problema.

Ayuda máximos iguales adyacentes:

Predicado maxEqual/3:

Se encarga de hacer la llamada general de la ayuda con la grilla. Se obtiene el mayor valor de la grilla ingresada la cual es usada en el predicado cáscara ***shellMaxEqual/7***.

Predicado shellMaxEqual/7:

shellMaxEqual(+Grid, +NumOfColumns, +Index, +ActualPath, +ActualScore, -Path).

Encuentra el camino máximo para toda la grilla que cumpla con la condición de la funcionalidad descripta. Tiene un caso base que se da cuando el índice está fuera de la grilla (se recorrió la grilla completa), y dos casos recursivos.

En el primero se obtiene el valor del bloque de dicho índice y se comprueba que no sea igual al mayor de la grilla. De serlo se sabe de antemano que no podrá ser posible encontrar un camino con ese bloque que cumpla la condición, ya que generaría un resultado que no se encuentra en la grilla, por lo que no habría adyacentes.

El segundo se ejecuta si no es posible armar caminos con el bloque actual, pasando directamente al siguiente.

Optimizamos el código colocando un cut(!) luego de dif(InitialValue,MaxValue) en el primer caso recursivo para evitar que se ejecute el segundo, ya que esa llamada se va a hacer en el mismo predicado luego de armar todos los caminos a partir del bloque actual.

Si el bloque es válido entonces se llama a ***recMaxEquals/8*** para armar recursivamente caminos a partir del bloque actual, el cual se agrega a la lista del camino.

La búsqueda del mayor camino con adyacente en la lista generada se hace con el predicado ***getMaxEqualPathFromList\9***, el cual se encuentra detallado en la documentación del código. Este predicado compara todos los caminos obtenidos y aplica una gravedad simulada a la grilla con dichos caminos para ver si existe uno mejor que cumpla la condición, pero posterior a la introducción del efecto gravedad. Para simular la gravedad, reemplaza el bloque generado al eliminar el Path de la grilla, por un valor -1, para optimizar la búsqueda del nuevo índice del bloque al aplicarle la gravedad. Luego, se busca ese -1 en la grilla actualizada, y se obtiene el índice para controlar los adyacentes al mismo.

Otra optimización agregada consiste en pasar el valor máximo obtenido por un camino por parámetro, para compararlo con los puntajes generados por los nuevos encontrados. Si el nuevo es menor o igual al máximo, no es necesario simular la gravedad ya que no va a ser posible que el camino genere un puntaje mayor al que ya se obtuvo previamente.

Predicado shellGravity\4:

Aplica gravedad a la grilla, pero sin generar bloques nuevos. Se ayuda con el predicado ***gravityOneColumn\4*** para aplicar gravedad por cada columna. Evitamos que se generen bloques nuevos, reemplazando los bloques a eliminar por valores 1, los cuales nunca serán tenidos en cuenta en la grilla. Esto se hace para evitar que se chequee la adyacencia con un bloque nuevo generado al azar, lo cual no cumpliría con la especificación de la función.

Predicado recMaxEquals\8:

recMaxEquals(+Grid, +NumOfColumns, +MaxValue, +Index, +ActualPath, +ActualScore, +Aux, -Results).

Arma una lista de adyacentes al bloque actual con ***findAllPosiblePaths\5*** y si no es vacía, llama a ***recMaxEqualsHelper\8*** para seguir armando los caminos a partir de los adyacentes y encontrar el máximo posible que cumpla la condición de adyacencia.

Devuelve todos los caminos posibles a partir de cada índice.

Predicado recMaxEqualsHelper\8:

recMaxEqualsHelper(+Grid, +NumOfColumns, +MaxValue, +[Adyacent|Rest], +ActualPath, +ActualScore, +Aux, -Results).

Si el bloque no fue agregado previamente al camino, se calcula el nuevo score y se agrega el nuevo camino a la lista de caminos junto con el score que genera.

Luego se llama recursivamente al ***recMaxEquals\8*** con dicho bloque para buscar recursivamente sus adyacentes, y se sigue llamando al ***Helper*** con el resto de adyacentes actuales.