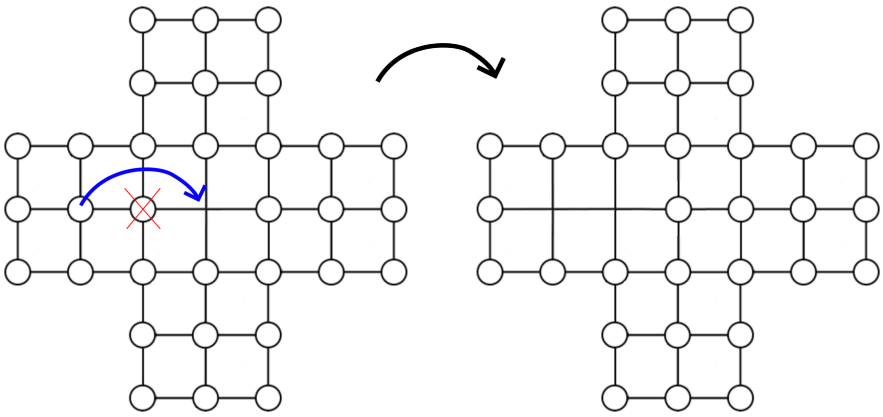
Programación II. Curso 2024-2025
Trabajo Obligatorio
GSenku (Senku Generalizado)

Descripción

En este trabajo se propone diseñar e implementar una versión particular del juego `Senku`[1] a la que denominaremos `GSenku`. El algoritmo principal estará basado en la estrategia de búsqueda con retroceso (backtracking).

La mecánica del juego base es muy sencilla: tenemos un tablero en el que todas las casillas menos una están ocupadas por piezas y el objetivo es conseguir eliminar todas las piezas menos una. Un movimiento permite mover una pieza a una posición del tablero vacía saltando sobre otra, la cual queda eliminada y se quita del tablero:



En la versión original, sólo se permiten movimientos en vertical y horizontal a distancia uno como en el ejemplo. Por otro lado, hay muchas versiones de los tableros como podemos ver en la Figuras 1 y 2. En este trabajo, vamos a generalizar la forma del tablero y los movimientos que podemos hacer.

Tablero: Para definir la forma de nuestro tablero, leeremos de un fichero de texto la matriz que lo define. El formato es como sigue:

Todas las imágenes tienen licencias abiertas y están tomadas de la Wikipedia.

[1] *Peg Solitaire* o *Solo Noble* en inglés, `https://en.wikipedia.org/wiki/Peg_solitaire`

Figura 1: Versiones europea/francesa (a la izquierda) e inglesa (a la derecha) del tablero.

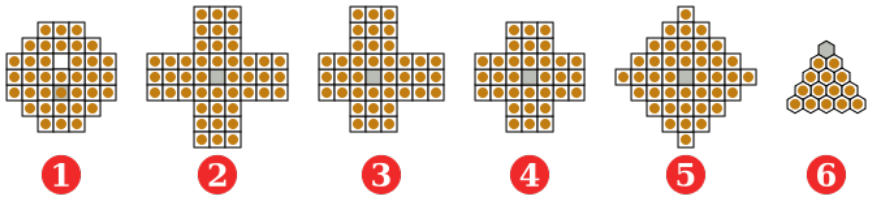


Figura 2: Diposiciones alternativas de los tableros (la 1 sería la europea y la 4 la inglesa, el resto son otras opciones que se pueden encontrar del solitario).

- La primera fila tiene las dimensiones de la matriz, $dimX$ y $dimY$.
- El resto del fichero representa la matriz de celdas del tablero, fila por fila, con los estados de cada celda separados por espacios. Cada celda puede tener uno de los siguientes valores (son caracteres):
  - '-': representa que esa celda no se usa, así podremos definir tableros de distintas formas (no se puede mover una pieza a esa celda).
  - 'o': representa que hay una pieza en esa celda (es una o minúscula).
  - 'x': representa una celda del tablero que está vacía (es parte del tablero y se puede mover una pieza a ella si se dan las condiciones).

En la Figura 3, podemos ver un ejemplo de especificación con el tablero inglés. Conviene destacar que aparte de la forma del tablero, esta especificación permite definir el estado de un tablero en un momento dado, pudiendo comenzar la búsqueda de la solución desde distintas configuraciones del mismo.

Movimientos permitidos: Los movimientos permitidos los vamos a leer de otro fichero que contendrá una matriz 3x3 que representa en qué direcciones se permite mover una

```
7 7
- - o o o - -
- - o o o - -
o o o o o o o
o o o x o o o
o o o o o o o
- - o o o - -
- - o o o - -
```

Figura 3: Ejemplo de especificación de tablero (tablero inglés).

pieza (condicionado a que se pueda mover capturando otra pieza). En este caso (también separados por espacios), la matriz representa una plantilla donde la celda central sería la pieza a mover y el resto de posiciones pueden contener un '+' cuando se nos permita mover en esa dirección o un '-' cuando no. Podemos ver en la Figura 4 un ejemplo con la especificación clásica de los movimientos, es decir: arriba, abajo, izquierda y derecha.

```
- + -
+ o +
- + -
```

Figura 4: Ejemplo de especificación de movimientos (configuración clásica).

Se describen a continuación la entrada, el objetivo y la salida esperada de `GSenku`.

Entrada: El programa leerá de la *línea de comandos* los siguientes parámetros:
`ficheroTablero ficheroMovimientos retardo [ficheroSalida]`
donde:

- `ficheroTablero` es el nombre del fichero que contiene la definición del tablero a utilizar así como el estado del mismo.
- `ficheroMovimientos` es el nombre del fichero que contiene los movimientos permitidos.
- `retardo` es el número de milisegundos que se debe esperar tras mostrar el estado del tablero, antes de mostrarlo en su estado siguiente.
- `ficheroSalida` es un parámetro opcional para almacenar los movimientos de la solución, o un -1 si no ha conseguido encontrar ninguna. En dicho fichero se almacenará un movimiento por línea, y cada movimiento simplemente se especificará con la posición inicial y final de la pieza movida. En la Figura 5 se muestra un

ejemplo de posibles movimientos partiendo del estado y movimientos permitidos de los ejemplos anteriores. Las posiciones son índices de vector (empiezan en 0). Si no se da un nombre de fichero de salida, se tendrá que utilizar por defecto `resultado.txt`.

```
3,1:3,3
1,2:3,2
(y así hasta el posible final)
```

Figura 5: Ejemplo de especificación de movimientos resultado. En caso de que no se alcance una solución, el fichero únicamente contendrá un -1.

Objetivo: El objetivo es solucionar un tablero que nos dan en un estado determinado utilizando únicamente los movimientos permitidos. Como se ha comentado anteriormente, este solitario se soluciona cuando se deja una sola pieza en el tablero. La solución (la lista de movimientos válidos que se han seguido) se almacenará en el fichero de resultados (`resultado.txt` por defecto, pero puede ser definido por el parámetro opcional de la línea de comandos).

Si `retardo`> 0 entonces se deberá mostrar por la pantalla el estado del tablero tras cada movimiento. Después de mostrar el estado del tablero, el programa deberá hacer una pausa de `retardo` milisegundos. Para acelerar la ejecución y pruebas, si `retardo`≤ 0, no se mostrará el tablero, almacenando únicamente la solución en el fichero.

El programa acabará cuando se alcance la solución o cuando se determine que no hay solución.

Material de apoyo

Como material de apoyo se proporciona el fichero `GSenku.hpp`. Dicho fichero define constantes y tipos de datos que serán necesarios para realizar el trabajo. El fichero también contiene las cabeceras y la descripción informal de las funciones que necesariamente habrán de implementarse. Lee con atención el contenido de este fichero antes de empezar a diseñar el resto de tu código.

Al final del fichero, aparecen comentados códigos de colores que se pueden utilizar como atributos para mostrar el tablero y las fichas con sus colores. Ten en cuenta que para *"dibujar"* una pieza o hueco del tablero, bastará representarlo escrito en pantalla como un caracter espacio en blanco, siempre y cuando el color de fondo con el que se escriba dicho espacio sea el adecuado en cada caso.

Por otro lado, para que podáis comprobar que la solución es válida, se os proporciona un programa que simula una lista de movimientos comprobando que tanto la solución como los formatos que uséis son válidos.

Tareas

Se deben realizar las siguientes tareas:

- Implementar en el fichero `GSenku.cpp` las funciones descritas en `GSenku.hpp`.
- Implementar en el fichero `mainGSenku.cpp` el programa principal que lea los parámetros de entrada y llame a las funciones descritas en `GSenku.hpp` para resolver el juego propuesto en este trabajo. Es decir, dados los parámetros leídos de la *línea de comandos*, deberá almacenar la solución en el fichero especificado y mostrar (si se define un retardo mayor que cero) la evolución del estado del tablero en pantalla, de acuerdo a los datos recibidos y a lo descrito en la sección anterior.
- Escribir un fichero `Makefile` que genere el ejecutable `mainGSenku` que resuelva el juego de `GSenku`.

Notas:

- Se puede hacer el uso que se desee del fichero `GSenku.hpp`, pero obligatoriamente deberán implementarse las funciones que se describen en el fichero suministrado.
- Si es necesario (seguramente lo será), el fichero `GSenku.cpp` puede contener más funciones que las que se dan descritas en `GSenku.hpp`.

Forma y fecha de entrega

- El trabajo se realizará por parejas.
- Se deberá entregar un fichero con nombre `AAAAAA_BBBBBB.zip` en la tarea de *Moodle* habilitada para dicha entrega, donde `AAAAAA` y `BBBBBB` son los NIAs de los miembros de la pareja y `AAAAAA` $<$ `BBBBBB`. El fichero contendrá exclusivamente lo siguiente:
  - Todos los ficheros de código fuente, `.hpp` y `.cpp`, que se hayan utilizado.
  - Un fichero `Makefile` que compile dichos ficheros y genere el ejecutable `mainGSenku` que resuelva el juego de `GSenku`.
- Las cabeceras de todos los ficheros de código fuente contendrán: los nombres, apellidos y NIAs de los dos miembros de la pareja.
- La entrega la realizará solamente el miembro de la pareja con NIA `AAAAAA`.
- El último día para entregar el trabajo será el 15 de mayo de 2025.