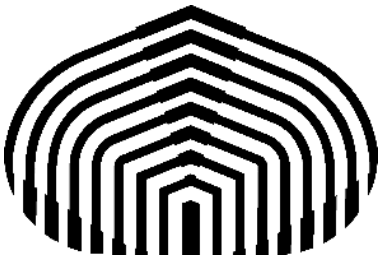


Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

Laboratorio de Algoritmos y Estructuras I CI-2691

Prof. Carmen Rosseline Rodríguez

Especificaciones del Proyecto

4 En Línea

Integrantes:

José Barrera. Carnet: 15 - 10123

Alfredo Cruz. Carnet: 14 – 10261

Sartenejas, abril de 2018

Introducción

El juego “4 En Línea” tiene como finalidad colocar cuatro fichas en línea (horizontal, vertical o diagonal) en un tablero vertical de 6x7. En cada turno el jugador, deja caer una ficha en alguna de las 7 columnas. Cuando se encuentran alineadas 4 fichas del mismo jugador este queda ganador del juego y se concluye la partida actual. Sin embargo, puede llenarse el tablero y que no se logre ninguna conexión en cuyo caso se declara un empate. O puede que el jugador decida retirarse y automáticamente conceder la victoria al rival.

El problema a resolver consiste en lograr desarrollar en el lenguaje de programación Python y en el pseudo-lenguaje GCL, una adaptación del juego 4 En Línea. Para hallar la solución al problema implementaremos el código utilizando la estructura de análisis descendente, partiendo de una carta estructurada (que se anexará más adelante). Ésta permite tener una perspectiva más clara del código, pues divide y jerarquiza las tareas a realizar en pequeños procedimientos y funciones más fáciles de entender. Las cuales nos permitirán vislumbrar la solución del problema de forma más clara y organizada.

Descripción de la finalidad de las definiciones de tipos de datos:

Para hacer más legible el código y facilitar su lectura, decidimos establecer los siguientes tipos de datos al inicio del juego:

Nombre	Tipo	Descripción
nombreusuario	String/Entrada	Nombre del Usuario
nivel	Integer/Entrada	Dificultad de la Máquina
partida	Integer/Entrada	Nueva Partida(0), Cargar Partida(1), Salir del Juego(2)
seguir	Bool/Entrada	Si se desea rendirse
guardar	Bool/Entrada	Si se desea guardar la partida actual
columna	Integer/Entrada	Columna donde el usuario desea jugar
tabl	List	Matriz Tablero de Juego
tabv	List	Tablero de Victorias
jugando	Bool	Controla si está en partida o en el menú
turno	Integer	Contador de los turnos de la partida
juegauser	Bool	Si le toca jugar al usuario
ganador	Integer	Ganador de la partida finalizada: Empate(0), Usuario(1) o Máquina (2)
dentro	Bool	Controla si está dentro del juego
movida	Bool	Controla el ciclo de intentar una juagada para el Nivel1
i	Integer	Fila donde la Máquina juega
j	Integer	Columna donde la Máquina juega
Ruser	List	Almacena los resultados de jugadaUser()
Rvictoria	List	Almacena los resultados de Rvictoria()
RIA	List	Almacena los resultados de IA()
datosdejuego	Class	Atributos: nombreusuario, turno, nivel, tabl, i, j, tabv
anterior	datosdejuego	Almacena los datos de juego de una partida anterior.
actual	datosdejuego	Almacena los datos de juego la partida actual.

Objetivos de los subprogramas de la carta estructurada:

Juego/Obtener Jugada/jugadaUser: La función **jugadaUser**(), recibe como argumento el tablero de juego **tabl**, solicita al usuario la fila y la columna que desea jugar, si son válidas modifica el tablero de juego lógico y gráfico.

Juego/Obtener Jugada/GuardarJuego: La función **GuardarJuego**(), recibe como argumento un archivo de texto (siendo **guardado.txt** el predeterminado) donde escribe los datos de juego actuales, contenidos en el segundo argumento (siendo **actual** el predeterminado, una variable miembro de la clase datosdejuego). No tiene datos de salida.

1.1.1 Inicializar Partida/Cargar Partida/CargarJuego: La función CargarJuego, recibe como argumento recibe un archivo de texto (siendo **guardado.txt** el predeterminado) desde donde lee la información de una partida anterior y la escribe en una lista, donde cada elemento es un dato de juego. No tiene datos de salida.

2.1.1.1 Juego/Obtener Jugada/Jugador/GuardarJuego: La función GuardarJuego, recibe como argumento recibe un archivo de texto (siendo **guardado.txt** el predeterminado) y un miembro de la clase valoresdejuego (siendo **anterior** el predeterminado) y escribe en el archivo, una línea para cada atributo de **anterior**. No tiene datos de salida.

2.1.1.3 Juego/Obtener Jugada/Jugador/jugadaUser: La función jugadaUser, recibe como argumento recibe el tablero de juego (tabl) y retorna el tablero con la jugada aplicada (también en la interfaz gráfica), la jugada y las coordenas de dicha movida.

2.1.2.2.2.1 Juego/Obtener Jugada/Máquina/Dificultad Media/ IA: La función **IA**(), recibe como argumento el tablero de juego **tabl**, la fila de la jugada anterior de la máquina **i**, y la columna de la jugada anterior de la máquina **j**. Retorna el tablero de juego **tabl** con la jugada realizada, la fila y columna de la jugada que acaba de realizar **i**, **j**.

2.2 Juego/Validar Jugada: Cuando se quiere saber si una posible jugada es válida, para luego ejecutarla, se deben evaluar sus coordenadas en la función **valida**(), la cual toma como argumento: la matriz del tablero de juego **tabl**, la fila y columna donde se desea jugar. Retorna verdadero o falso dependiendo de si el movimiento cumple o no las reglas del juego. Estas reglas se pueden ver en [1].

2.3 Juego/Verificar 4 en Línea: Cuando se quiere saber si se ha formado una línea ganadora en el tablero, se pasan como argumento a la función **victoria**(): la matriz del tablero de juego **tabl**, el bool **jugando**, el entero **ganador** y las coordenadas de la última jugada **i**, **j**. Se retorna la matriz del tablero **tabl**, el bool **jugando** y el entero **ganador** modificados si había efectivamente una línea ganadora además del respectivo cambio en la interfaz gráfica. En caso contrario, no se modifican.

<u>Implementación del programa:</u>

- Decidimos no seguir la sugerencia planteada en el enunciado del proyecto referente al nivel 2. En lugar de que la maquina intente completar un tipo de línea que se define

de manera random, hasta que vea que no puede hacerlo. Se optó por una estrategia del "mejor siguiente", que es ligeramente más inteligente. Esto se explicará detalladamente más adelante.

- El tablero posee "números guía", para que sea más sencillo visualizar en la interfaz gráfica cual es el número de la columna donde el usuario desea jugar.
- En pro de que la entrada de datos por parte del usuario sea la mínima posible. Solo se pide la columna (y el programa calcula la fila), y para decidir si guardar partida o seguir jugando (ambas se piden cada turno) se puede simplemente pulsar Enter, cualquier otra entrada se tomará como un no.
- Para mostrar más sencillamente las victorias acumuladas por el usuario, la máquina y los empates se creó una lista llamada tabv donde el primer elemento son los empates, el segundo las victorias del usuario y el ultimo las de la máquina. Para hacerlo más ilustrativo esta información se da en la propia consola, ya que el tablero se muestra siempre que el usuario no esté en una partida.

Manera en que la máquina hace la toma de decisiones:

Para el turno de la máquina, las jugadas siguen 2 estrategias bien diferenciadas de acuerdo al nivel de dificultad (1 o 2) que selecciona el usuario:

Nivel 1: Se intenta un par de coordenadas random hasta que estas se ubiquen en una casilla válida para jugar. Entonces se ejecuta la jugada.

Nivel 2: La máquina en su primer turno intenta hacer la jugada fila 5, columna 3. Esto porque como se explica en [2], es la mejor jugada para empezar una partida. Si el jugador la realiza primero entonces hace la jugada fila 5, columna 2, pudo haberse escogido la 5,4 también. A partir de la primera jugada, construye su estrategia en base a la jugada anterior. Busca las jugadas validas en las 7 direcciones posibles y ejecuta aquella dirección y sentido con más jugadas validas en un alcance de 3 espacios contiguos. Si la jugada anterior se encuentra rodeada, es decir, todas las 7 direcciones y sentidos posibles se evalúan 0, se busca y ejecuta la primera jugada valida obtenida recorriendo la matriz de abajo hacia arriba y de derecha a izquierda.

Conclusiones

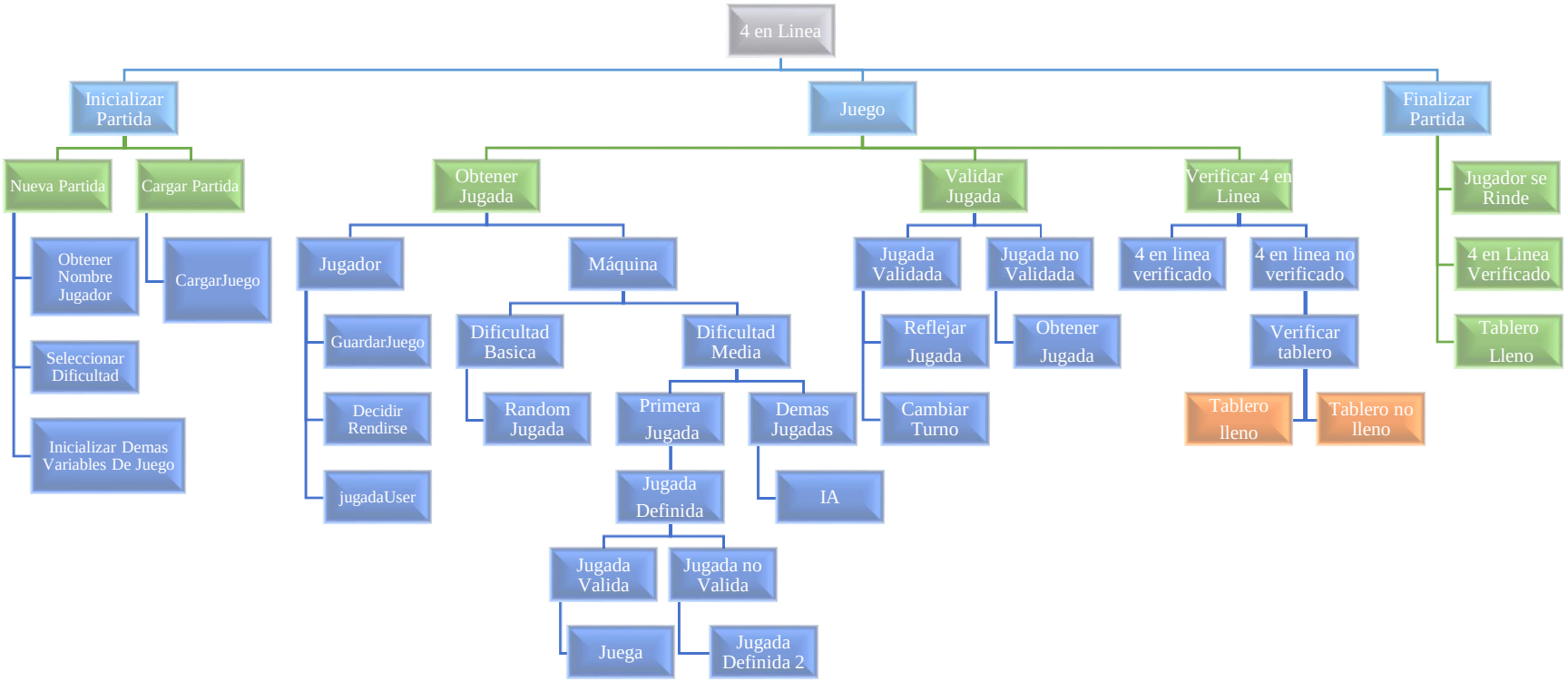
La realización del presente proyecto de laboratorio, nos permitió implementar la estructura de análisis descendente para resolver un problema de complejidad sin precedentes dentro del curso. Durante su desarrollo, se hizo evidente que el dividir la tarea en pequeños problemas que trabajan de forma interconectada, realmente permite atacar los problemas de forma más eficaz. Esto debido a que se puede analizar cada componente como una nueva mini-tarea, enfocando la capacidad de resolución de problemas en algo más simple, que incluso pudiese aplicarse en futuras macro-tareas.

Nos dimos cuenta, además, de que el análisis descendente es un proceso que no es ajeno a los cambios, pues las primeras soluciones a cada mini-tarea, por lo general no escapaban de mejoras en pro de simplificar el código, hacerlo más legible e inclusive agregarle nuevas funcionalidades al programa.

En cuanto a los algoritmos y las estructuras de datos, podemos concluir que fijar las herramientas adecuadas es determinante para desarrollar soluciones elegantes a problemas concretos. Un aspecto básico que, de descuidarse, deviene en una pérdida de tiempo para comprender el código tanto por los lectores como por el propio desarrollador.

Referencias

- [1] Instrucciones Oficiales Conecta 4 de Hasbro. Disponible en la web: https://www.hasbro.com/common/documents/dad2614d1c4311ddbd0b0800200c9a66/808133DE50569047F5FF971AE13D0BA4.pdf
- [2] Victor Allis. “A Knowledge-based Approach of Connect-Four The Game is Solved: White Wins”. Vrije Universiteit. 1988. Disponible en la web: http://www.informatik.uni-trier.de/~fernau/DSL0607/Masterthesis-Viergewinnt.pdf
- [3] The Python Standard Library, Document version 3.6.5, capítulo 2. https://docs.python.org/3/library/functions.html


4 en Linea
Inicializar Partida
Juego
Finalizar Partida
Nueva Partida
Cargar Partida
Obtener Jugada
Validar Jugada
Verificar 4 en Linea
Jugador se Rinde
Obtener Nombre Jugador
CargarJuego
Jugador
Máquina
Jugada Validada
Jugada no Validada
4 en linea verificado
4 en linea no verificado
4 en Linea Verificado
Seleccionar Dificultad
GuardarJuego
Dificultad Basica
Dificultad Media
Reflejar Jugada
Obtener Jugada
Verificar tablero
Tablero Lleno
Inicializar Demas Variables De Juego
Decidir Rendirse
Random Jugada
Primera Jugada
Demas Jugadas
Cambiar Turno
Tablero lleno
Tablero no lleno
jugadaUser
Jugada Definida
IA
Jugada Valida
Jugada no Valida
Juega
Jugada Definida 2