Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

CE-3104 Lenguajes, Compiladores e Intérpretes

Tarea 1

Prof. Marco Rivera Meneses

Estudiantes:

Fernanda Alvarez Martínez

Daniel Cob Beirute

Eder Vega Suazo

I Semestre 2023

Descripción del algoritmo goloso desarrollado

Para describir el algoritmo desarrollado se utilizarán los cinco componentes de un algoritmo goloso y cómo cada parte fue desarrollada en algoritmo presentado en la solución.

Conjunto de candidatos:

Al analizar la lógica del juego 4 en línea se concluyó que el conjunto de candidatos siempre serán las columnas del tablero. Es decir, si un tablero tiene 8 columnas el conjunto de candidatos serán las columnas (1, 2, 3, 4, 5, 6, 7, 8). No importa si las columnas están llenas o vacías serán parte del conjunto de candidatos. Por tanto, en el algoritmo no se programó una función que entregue el conjunto de candidatos ya que vienen implícitos en la matriz.

Función de viabilidad:

La función de viabilidad se encarga de eliminar candidatos que no sirven para contribuir a una solución. De modo que, se programó una función que obtuviese las columnas que no estuviesen llenas y puedan servir para insertar fichas. Estas columnas viables se entregan como una lista donde cada elemento es el número correspondiente a una columna viable. Por ejemplo: (1 3 4 5 7 8) donde las columnas viables serían la 1, 3, 4, 5, 7 y 8 de una matriz con 8 columnas.

Función objetivo:

La función objetivo se encarga de asignar un valor a cada solución. Para esta implementación se aplicó un cálculo de heurística a cada columna viable. Este cálculo se realiza en base a las fichas adyacentes, del contrincante o propias, y su posibilidad de completar una solución de 4 en línea. El cálculo se describe de la siguiente manera como una suma de las fichas horizontales, verticales y diagonales cercanas a la casilla en una columna. Por tanto, una ficha individual suma 1, 2 o más fichas suman esa cantidad de fichas más uno. Cabe mencionar que estas fichas se suman únicamente si son iguales, es decir, si son varias del contrincante se suman, si son varias propias se suman, pero nunca se suman las fichas del contrincante con las propias. Adicionalmente a las 2 o más fichas del contrincante en posición vertical suman esa cantidad más 2. Se programó una función que calcule esta heurística por cada columna y retorne una lista con sublistas por cada columna que contengan el número de columna y su valor de heurística. Por ejemplo, se retorna la siguiente lista: ((1 0) (2 1) (3 1) (4 1) (5 3) (6 4) (7 3) (8 2)), este resultado nos indicaría que en la columna 6 el valor de heurística es 4, y en la columna 7 el valor es 3.

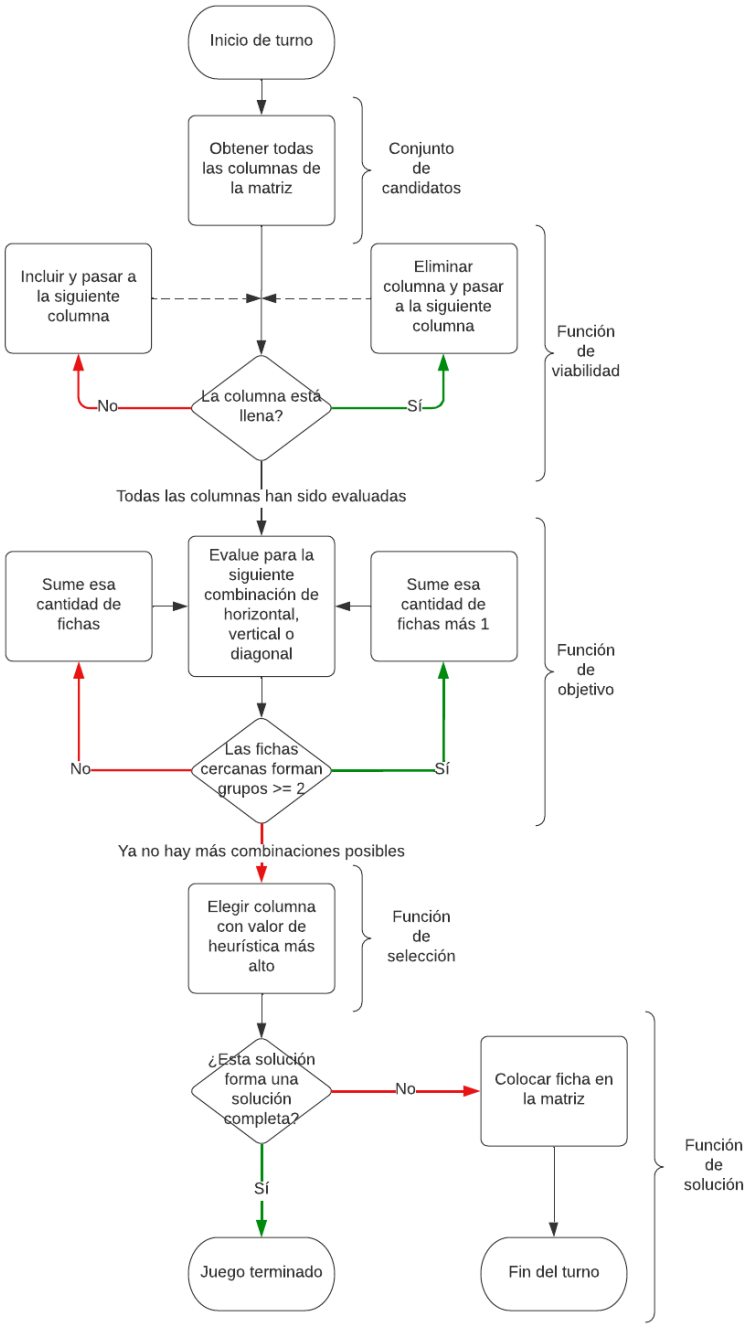
Función de selección:

La función selección elige el mejor candidato para agregar a la solución. Por tanto, se programó una función que tomara la lista que retorna la función objetivo y devolviera la columna con el valor de heurística más alto. En caso de haber dos o más mayores iguales se elige el primero en la lista, es decir el que esté más a la izquierda.

Función de solución:

La función de solución indica cuando se ha descubierto una solución completa. En este caso la solución completa corresponde a 4 o más fichas que estén en serie de manera horizontal, vertical o diagonal. Se programó una función que verificara si la columna elegida corresponde a una solución completa, si lo es retorna #t, sino retorna la matriz con la nueva ficha insertada.

Diagrama del algoritmo:



Descripción de las funciones implementadas

Elemento matriz: Esta función es usada para indexar en la matriz para obtener el elemento en la fila y columna solicitada, hace uso de la función elemento, la cual recibe una lista y un índice, y devuelve el elemento de la lista que indique el índice. Hace una llamada a la función elemento a la cual le pasa el número de fila, y de lista la columna, la cual se obtiene al llamar elemento y pasarle el número de columna y la matriz.

Agregar: La función agregar recibe el número de jugador, un numero de columna y la matriz, y se encarga de agregar el número de jugador en el primer espacio disponible en la columna indicada por el numero columna. Para ello hace uso de dos funciones, agregar-aux que se encarga de encontrar la columna correcta a la cual agregar el número de jugador; la función usa recurrencia de pila, en el caso de que la matriz recibida ya no tenga columnas entonces devuelve la matriz actual (una matriz vacía), en caso de que el número de columna sea 1, entonces llama la función agregarFicha, a la cual le pasa el número de jugador y la columna (car de la matriz), y la concatena con lo que resta de la matriz. Si la matriz no está vacía, y el contador columna no es uno, entonces concatena con un append la columna actual, con la matriz que devuelva la función agregar-aux a la cual se le pasará el número de jugador la columna – 1, y lo que queda de resta de la matriz (cdr matriz).

Agregar-ficha: Esta función se encarga de agregar una ficha de jugador (número de jugador) a la primera fila disponible, como recibe la columna (una lista) y el número de jugador, crea por recursividad de pila una lista, extrayendo los elementos de la lista (filas) y agregándolos a la lista creada, hasta encontrar un cero, en ese caso agrega a la lista el número de jugador y lo que queda de lista.

Verificar: La función verificar encapsula a las funciones verifHorizontral, verifVertical y verifDiagonal, recibe el número de fila y columna donde se ubica la casilla a verificar, el combo que es el objetivo de fichas en línea, se agregó este parámetro para flexibilizar la aplicación al algoritmo codicioso, y la matriz asociada al tablero; cada una de las funciones encapsuladas reciben la fila, columna y matriz. Si alguna de los resultados de la función verificar mas uno (debido a la ficha agregada en la casilla) es igual o mayor al combo, entonces devuelve #t, indicando que al colocar una ficha ahí uno de los jugadores ganó, en otro caso retorna #f. Las funciones verif-x comparan si el numero de la siguiente fiche de la dirección de la función es igual al número de la ficha que se encuentra en la casilla central.

Descripción de la ejemplificación de las estructuras de datos desarrolladas

Para el manejo del tablero del juego se hizo uso de un arreglo de listas, formando con ellas una matriz, cada lista dentro de la matriz corresponde a una columna del tablero, y cada elemento de esa lista corresponde a una fila. Al iniciar el juego se empieza con el tablero vacío y por lo tanto con una matriz nula (conformada por ceros), al seleccionar una columna se colocará la ficha correspondiente en la primera fila disponible de la columna. Las fichas del jugador son representadas con el número 1 y se representan en el tablero con fichas de color rojo; las fichas colocadas por el algoritmo goloso las representamos con el número 2 y se visualizan con fichas de color azul. En un tablero 8x10, la matriz generada es 10x8, al agregar una ficha a la columna 5 y después de que el algoritmo goloso agregue su ficha tendremos la siguiente matriz:
((0 0 0 0 0 0 0 0) (0 0 0 0 0 0 0 0) (0 0 0 0 0 0 0 0) (2 0 0 0 0 0 0 0) (1 0 0 0 0 0 0 0) (0 0 0 0 0 0 0 0) (0 0 0 0 0 0 0 0) (0 0 0 0 0 0 0 0))
Donde: (2 0 0 0 0 0 0 0) se refiere a la columna 4 del tablero, donde 2 es el valor de la ficha colocada por el algoritmo goloso; y (1 0 0 0 0 0 0 0) es la columna 5 y el 1 corresponde a la ficha colocada por el jugador.

Problemas sin solución

La manera en que se calcula la heurística hace que ciertos movimientos o conjunto de movimientos por parte del jugador no sean bloqueados por el algoritmo. De modo que, puede ocurrir fácilmente que se gane acomodando 4 fichas seguidas horizontalmente. Esto se puede evitar cambiando

ligeramente el cálculo de la heurística, por ejemplo, sumar un número más alto cuando hay un conjunto de fichas iguales a 3. Aunque también esta modificación podría dar surgimiento a otro tipo de acciones no óptimas.

Problemas encontrados

Error al colocar la última ficha: Este error se presenta a la hora de agregar una ficha a la última fila, en el error sucedía debido a que si bien se agregaba en la ficha del jugador en lo alto de la columna, luego en la llamada a la función verificar, había una función que dada una columna, "escalaba" aumentando un índice e indicaba cual era la última fila, esto lo hacía comparando si el siguiente era cero, pero en uno de los cond teníamos que si la lista (columna) estaba vacía, devolviese #f, y como ocupábamos un entero para indexar en la matriz y pasamos un booleano entonces se nos caía el programa, esto lo solucionamos al cambiar el #f por índice, donde devolvería el índice ya sea si la lista estaba vacía o si el siguiente número fuese cero.

Plan de Actividades

Tareas	Descripción	Persona asignada	Fecha límite
Algoritmo goloso	Programar un algoritmo con las partes descritas en las instrucciones de la tarea, que maneje el juego de la computadora	Daniel Cob Beirute Eder Vega Suazo	6/3/2023
Interfaz gráfica	Programar una interfaz gráfica con todas las partes descritas en las instrucciones de la tarea.	María Fernanda Álvarez Martínez	6/3/2023
Integración	Integrar el algoritmo goloso a la interfaz gráfica y asegurarse de que funcionen bien en conjunto.	María Fernanda Álvarez Martínez Eder Vega Suazo Daniel Cob Beirute	9/3/2023
Documentación	Documentar el código y el proceso de desarrollo.	María Fernanda Álvarez Martínez Eder Vega Suazo Daniel Cob Beirute	10/3/2023
Manual de Usuario	Crear un manual de usuario para el programa.	María Fernanda Álvarez Martínez	6/3/2023
Bitácora	Documentar el progreso del proyecto y cualquier problema o solución que surja durante el	María Fernanda Álvarez Martínez Eder Vega Suazo Daniel Cob Beirute	10/3/2023

	proceso de desarrollo.		

Conclusiones

Racket usa un paradigma simple y fácil de entender, a su vez al implementar el paradigma funcional brindó ventajas de eficiencia.

Al implementar el tablero como un tipo de matriz transpuesta facilitó la interpretación de las filas y columnas del tablero en la matriz, gracias a eso el colocar las fichas de fila menor a mayor fue una tarea simple.

Lograr acceder a las casillas de manera intuitiva fue un factor importante para simplificar el manejo de la matriz de Racket, debido a que en Racket no hay algo tal como indexar en una lista

Recomendaciones

Se recomienda utilizar otro tipo de algoritmo que cambie de comportamiento cuando reciba las mismas fichas para dejar de ser tan predecible y variar las partidas y la dificultad de estas. Por ejemplo, podría servir un algoritmo que aprenda del contrincante y tome mejores decisiones conforme avanza el juego. O bien, se puede reutilizar este mismo algoritmo pero que elija una columna al azar si hay varias opciones con el mismo valor de heurística.

Bibliografía consultada en todo el proyecto

Guzman, J. E. (2006). Introducción a la programación con Scheme. Cartago: Editorial Tecnológica de Costa Rica.

Flatt, M., Findler, R. B., & Clements, J. (n.d.). The Racket Graphical Interface Toolkit. Retrieved from https://docs.racket-lang.org/gui/

Bitácora

Fecha	Tareas realizadas	Integrantes
01/03	Reunión de equipo para planificar el proyecto y asignar tareas. Investigación sobre el juego de 4 en línea y cómo implementarlo en un programa.	María Fernanda Álvarez Martínez Eder Vega Suazo Daniel Cob Beirute
02/03	Programación de la lógica del juego de 4 en línea, incluyendo la creación del tablero y la detección de victorias. Pruebas preliminares para asegurarnos de que funcione correctamente.	Eder Vega Suazo Daniel Cob Beirute
03/03	Programación de la interfaz gráfica del juego de 4 en línea, según las instrucciones de la tarea. Revisión de diseño y funcionalidades necesarias.	María Fernanda Álvarez Martínez
04/03	Integración de la lógica del juego de 4 en línea a la interfaz gráfica. Corrección de errores y pruebas para verificar el correcto funcionamiento.	María Fernanda Álvarez Martínez Eder Vega Suazo Daniel Cob Beirute
05/03	Mejora de la interfaz gráfica del juego de 4 en línea. Agregado de elementos visuales y detalles para una mejor experiencia de usuario.	María Fernanda Álvarez Martínez
06/03	Programación del algoritmo goloso para el juego de 4 en línea, que permita a la computadora tomar decisiones sobre qué movimiento realizar. Pruebas preliminares para asegurarnos de que funcione correctamente.	Eder Vega Suazo Daniel Cob Beirute
07/03	Integración del algoritmo goloso a la lógica del juego de 4 en línea. Corrección de errores y pruebas para verificar el correcto funcionamiento.	Eder Vega Suazo Daniel Cob Beirute
08/03	Documentación del código del algoritmo, la lógica del juego y la interfaz gráfica. Inclusión de comentarios y explicaciones para facilitar la comprensión y futuras modificaciones.	María Fernanda Álvarez Martínez Eder Vega Suazo Daniel Cob Beirute
09/03	Creación del manual de usuario para el programa del juego de 4 en línea. Inclusión de instrucciones detalladas para jugar y utilizar las funcionalidades de la interfaz gráfica.	María Fernanda Álvarez Martínez
10/03	Revisión general del proyecto. Pruebas finales para asegurarnos de que todo funciona correctamente y sin errores. Entrega del proyecto completo, incluyendo el código, la documentación y el manual de usuario.	María Fernanda Álvarez Martínez Eder Vega Suazo Daniel Cob Beirute

Minutas

Fecha y hora	Participantes	Decisiones tomadas
1/03	María Fernanda Álvarez Martínez Eder Vega Suazo Daniel Cob Beirute	- Creación del plan de actividades. - Definición de roles.
10/03	María Fernanda Álvarez Martínez Eder Vega Suazo Daniel Cob Beirute	- Revisión del código