TEC | Tecnológico de Costa Rica

Área de Ingeniería en Computadores

Lenguajes, Compiladores e Intérpretes

Grupo 01

Tarea #1: Problema del Caballo

Elaborado por:

Juan Pablo Martínez - 2016140404

Bradly

Esteban

Profesor:

Marco Rivera M.

14 de Mayo, 2019

1. Manual de usuario

El usuario debe abrir el archivo llamado PDC.rkt el cual contiene el código fuente del programa que está compuesto de las funciones madre y funciones auxiliares que hacen posible la solución, las funciones madre son las que el usuario puede ejecutar para usar el programa. Las funciones madre se deberán ejecutar en la consola del editor de DrRacket junto con las entradas que se especificarán en la sección de descripción de las funciones implementadas, la solución de las tres primeras funciones serán mostradas en la misma consola mientras que para la última función, la solución será mostrada en una interfaz animada. A continuación se mostrarán ejemplos de cómo se deben hacer las llamadas a las funciones en la consola:

> (PCD-Sol 5 '(1 1))

> (PCD-Todas 5 '(1 1))

> (PCD-Test 5 '(7 4 15 24 17 6 3 10 19 22 11 2 20 9 13 16 23 12 21 18 25 14 5 8 1))

> (PCD-Paint 5 '(7 4 15 24 17 6 3 10 19 22 11 2 20 9 13 16 23 12 21 18 25 14 5 8 1))

2. Descripción de las funciones implementadas

PDC-Sol: esta función retorna una solución del algoritmo del caballo, se debe ingresar el tamaño del tablero y un par ordenado que indica donde se encuentra inicialmente el caballo.

PDC-Todas: esta función retorna todas las soluciones posibles del algoritmo del caballo, se debe ingresar el tamaño del tablero y un par ordenado que indica donde se encuentra inicialmente el caballo.

PCD-Test: esta función recibe una solución del problema del caballo junto con el tamaño del tablero y verifica que la solución sea correcta y retorna una matriz correspondiente a los movimientos de la solución dada.

PDC-paint: esta función recibe un tamaño de matriz y una "Solución" del problema del caballo que se verificará como correcta antes de pasar a la animación la cual hará una matriz de acuerdo al tamaño acordado y pintará cada casilla en orden de la solución antes ingresada.

3. Descripción de las estructuras de datos desarrolladas

Grafo: Esta estructura es la utilizada para el manejo de las casillas vecinas en un tablero y para la búsqueda de soluciones al problema del caballo. Este grafo está representado por una lista de sublistas, donde cada sublista posee dos elementos: la casilla y sus vecinos válidos (que pertenezcan a la matriz). A continuación se muestra un ejemplo de un grafo usado para una matriz 4x4:

```
'(
  ((0 0) ((1 2) (2 1)))
  ((0 1) ((1 3) (2 2) (2 0)))
  ((0 2) ((1 0) (2 3) (2 1)))
  ((0 3) ((1 1) (2 2)))
  ((1 0) ((2 2) (0 2) (3 1)))
  ((1 1) ((2 3) (0 3) (3 2) (3 0)))
  ((1 2) ((2 0) (0 0) (3 3) (3 1)))
  ((1 3) ((2 1) (0 1) (3 2)))
  ((2 0) ((3 2) (1 2) (0 1)))
  ((2 1) ((3 3) (1 3) (0 2) (0 0)))
  ((2 2) ((3 0) (1 0) (0 3) (0 1)))
  ((2 3) ((3 1) (1 1) (0 2)))
  ((3 0) ((2 2) (1 1)))
  ((3 1) ((2 3) (1 2) (1 0)))
  ((3 2) ((2 0) (1 3) (1 1)))
  ((3 3) ((2 1) (1 2)))
)
```

4. Descripción detallada de los algoritmos desarrollados

Quicksort: Es un algoritmo de ordenamiento, principalmente para conjuntos de elementos que pueden ser representados por listas.

Primero, este algoritmo escoge un elemento de la lista como pivote. Este puede ser cualquier elemento; el algoritmo implementado escoge el primer elemento de la lista.

Después, el algoritmo crea dos lista: una para elementos mayores al pivote, y otra para aquellos menores al pivote. Cuando ya no haya más elementos, se aplica recursivamente quicksort a las dos listas creadas, y así hasta que los elementos estén ordenados.

El quicksort implementado usa como valor para comparar los elementos la cantidad de vecinos que poseen estos, obtenida del grafo; y ordena los elementos de forma ascendente con respecto a este valor. Este ordenamiento optimiza el algoritmo 'PDC-Sol', ya que al escoger como nuevo elemento de la ruta a el vecino con menos vecinos, aumenta la probabilidad de encontrar una solución y reduce la cantidad de backtracking necesaria.

5. **Problemas conocidos:**

Debido a la cantidad de información que debe manejar la computadora y la cantidad de recursiones que esta debe hacer, el algoritmo de PDC-Todas no funciona para matrices de tamaño 6 o más. Se investigó y se encontró que las personas que han podido realizar tal análisis ocuparon varias computadoras funcionando en conjunto y varias semanas de ejecución, por lo que el código presentado no es capaz de ser ejecutado por una computadora para estas matrices grandes.

Por el hecho que el programa debe expandirse en forma de grafo para bucar todas las soluciones posibles de este algoritmo, se debe aplicar backtracking en multiples veces hasta encontrar las soluciones deseadas, esto in

6. **Actividades realizadas por estudiante:**

Actividad	Integrante	Horas	Fecha
Elaboración de los algoritmos 'PDC-Sol' y 'PDC-Todas'	Esteban	7	05/05/19 - 06/05/19
Implementación del quicksort para 'PDC-Sol'	Esteban	0.5	09/05/19
Reducción de código y optimización de 'PDC-Todas'	Esteban	1.5	12/05/19
Validar que una solución es correcta	Juan Pablo	3.5	08/05/19 - 09/05/19
Traducción de solución a matriz	Juan Pablo	2.5	13/05/19
Aprender a utilizar Graphics: Legacy Library e implementar un tablero	Bradly	3	10/5/19
Implementar la animación del PDC sin	Bradly	2	11/5/19

imagen del caballo			
Implementar toda la animación con imagen del caballo y agregar funciones mejoradas	Bradly	1.5	13/5/19

Estudiante	**Total de horas invertidas**
Esteban	9 horas
Juan Pablo	6 horas
Bradly	6 horas 30 minutos

7. **Problemas encontrados:**

Para la función PDC-Paint hubo un problema de poder eliminar la imagen del caballo de su posición actual hacia la casilla que se iba a mover, la solucion para esto fue hacer 2 funciones aparte de la de pintar el caballo que pintara los lugares por donde pasó y su rastro encima de la posición anterior del caballo.

Para las funciones PDC-Sol y PDC-Todas habían muchos problemas con el tiempo de ejecución, donde tardaba mucho más de lo estimado (5 veces más aproximadamente). Para optimizar PDC-Sol se implementó un quicksort dirigido para ordenar los vecinos de una casilla según la cantidad de vecinos que estos mismos poseían, así se reducía la cantidad de backtracking necesaria para encontrar una solución.

Para la función PDC-Todas se tuvo que corregir un problema lógico que tenía el algoritmo, ya que al usar las mismas funciones auxiliares que PDC-Sol, sucedía que se realizaban más recursiones de las debidas, llegando a recorrer un mismo camino varias veces cuando no es necesario. Se corrigió este problema realizando una función auxiliar única que solo usa PDC-Sol para su backtracking, para que PDC-Todas pudiese ejecutarse eficientemente.

8. **Conclusiones y Recomendaciones del proyecto**

Conclusiones

- Racket es un IDE que está basado en Scheme y utiliza funciones primitivas internas para trabajar operaciones simples, además, posee librerías para realizar diferentes proyectos como este.

- El paradigma funcional se refiere a resolver problemas utilizando solamente funciones, estas funciones vienen dentro del lenguaje y se conocen como funciones primitivas, a partir de ellas se hacen funciones más complejas que resuelven los problemas de la misma complejidad.
- Las listas utilizadas en este proyecto se manejaron como grafos para dar una serie de soluciones que varían dependiendo de la ruta que se encuentre.
- El problema del caballo se basa en buscar caminos en una matriz determinada, por lo cual utilizar grafos para realizar búsquedas en profundidad para solucionar este problema es una opción muy eficiente, considerando la facilidad que brinda DrRacket para definir grafos con listas.

Recomendaciones

- Buscar la definición sobre el problema del caballo.
- Realizar funciones auxiliares en el paradigma funcional para realizar pequeñas tareas sencillas, las cuales al trabajar en conjunto, pueden resolver problemas complejos. Además, facilita la lectura del código.
- Dar nombres representativos a los parámetros y a las funciones para dar un mayor entendimiento del programa.

9. Bibliografía consultada en todo el proyecto

http://interactivepython.org/runestone/static/pythoned/Graphs/ElProblemaDeLaGiraDelCaballo.html

https://docs.racket-lang.org/graphics/index.html

https://docs.racket-lang.org/guide/concurrency.html

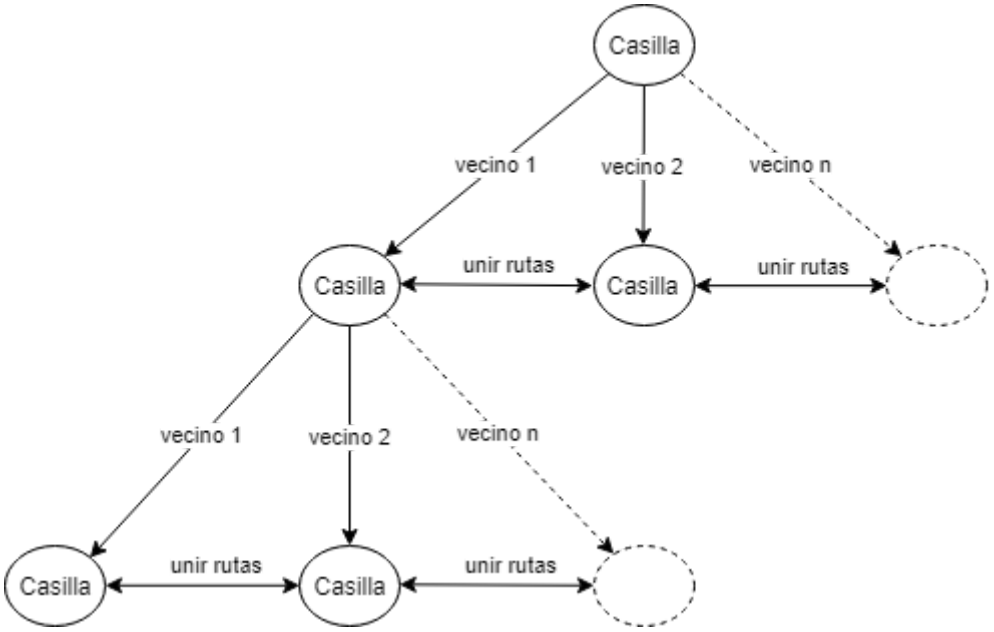
https://docs.racket-lang.org/guide/for.html

10. Bitácora

Actividades	Integrantes	fecha
Reunión para leer y entender el enunciado del proyecto y dar tareas a cada integrante, además se propusieron posibles soluciones para cada una de las tareas entre todos los integrantes	Juan Pablo Bradly Esteban	03/05/19
Implementación de los algoritmos ‘PDC-Sol’ y	Esteban	05/05/19 -

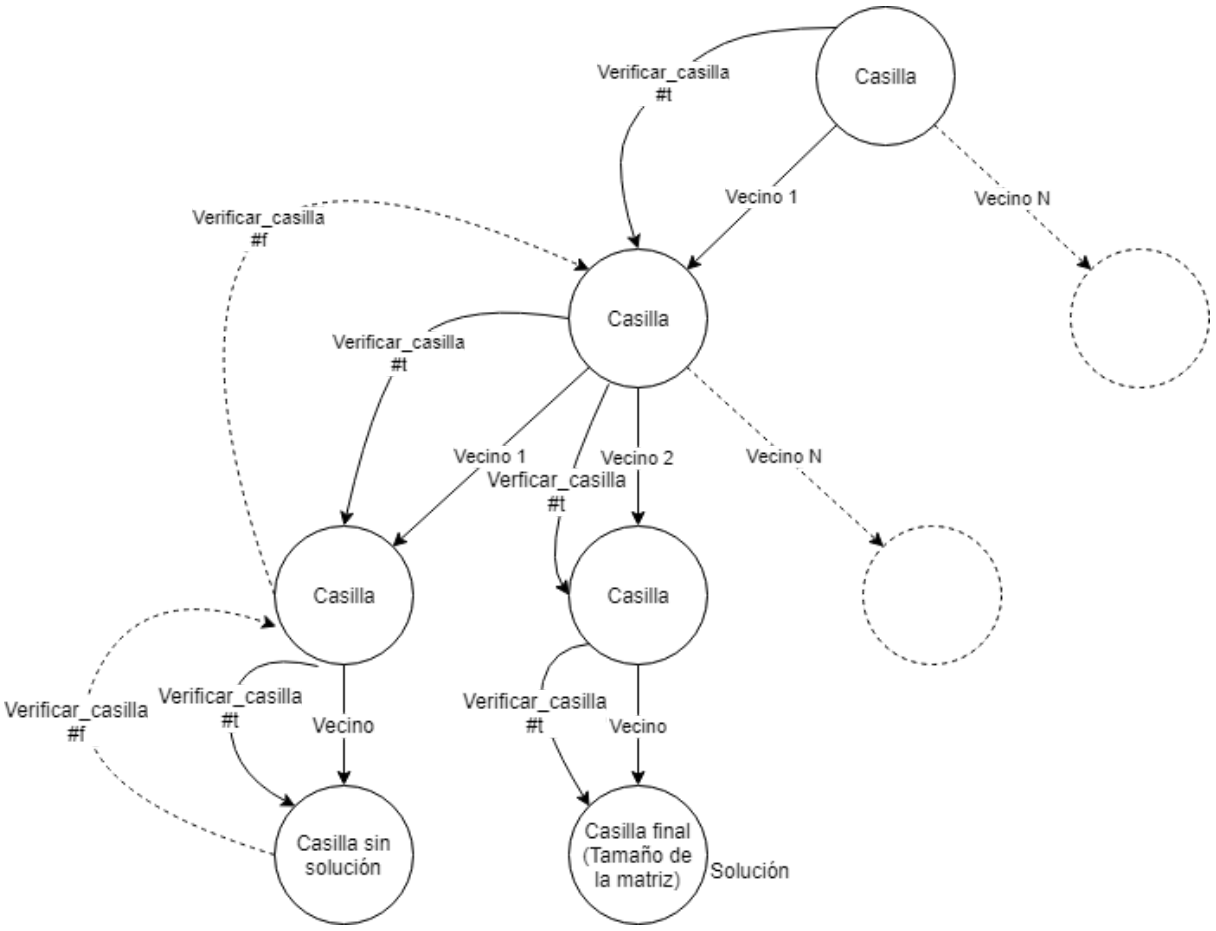
‘PDC-Todas’ en su totalidad; además de funciones básicas necesitadas para el avance de los 4 algoritmos principales del proyecto, como crear grafos o funciones de análisis de listas		06/05/19
Consulta a conocidos para decidir qué librería se utilizará para la interfaz gráfica	Bradly	08/05/19
Implementación del quicksort enfocado a ordenar los vecinos de forma ascendente, con respecto a la cantidad de vecinos que posean	Esteban	09/05/19
Planteamiento de la solución de PDC-Paint y que funciones llevaría	Bradly	09/05/19
Implementación de la función PDC-Test y de función que valida si solución es correcta que además se usará para verificar una solución antes de animarla en PDC-pain	Juan Pablo	08/05/19 - 09/05/19
Reunión para ver el avance de cada integrante, explicar lo que cada uno había hecho y plantear posibles soluciones para lo que faltaba y empezar con parte de la documentación externa del proyecto	Juan Pablo Bradly Esteban	11/05/19
Optimización de la función PDC-Test	Juan Pablo	13/05/19
Creacion e implementacion para la animación de PDC-Paint	Bradly	10/05/19- 11/05/19- 13/05/19
Eliminación de variables no usadas, recursiones innecesarias y cambios en la lógica del algoritmo ‘PDC-Todas’ para su optimización. También se realizó la documentación interna para un mayor entendimiento del programa	Esteban	12/05/19
Reunión para verificar que todo el proyecto estuviera funcionando a la perfección y terminar la documentación externa	Juan Pablo Bradly Esteban	14/05/19

11. Diagrama de clases y un documento que explique el porqué del diseño



PDC-Todas

La figura anterior es una representación del proceso que realiza la función PDC-Todas. Esta, buscará todos los vecinos que posea la casilla donde está situado el caballo, y unirá las rutas que se puedan encontrar yendo por dichos vecinos. Situando el caballo en uno de los vecinos, se vuelve a realizar este proceso para los nuevos vecinos, y así hasta que no encuentre vecinos que no estén en la ruta. Llegado a este punto, se verifica si el camino recorrido posee la cantidad de casillas del tablero, y si es así, retorna el camino en cuestión; si no, retorna una lista vacía. El algoritmo une todas las listas vacías y rutas que encontró al irse por todos los vecinos posibles y pasando por estos una única vez, y al final retorna una sola lista con todos los caminos posibles que representen una solución al problema del caballo.



PDC-Sol

Esta imagen representa el funcionamiento de PDC-Sol, para su función está verificará que cada casilla del grafo sea un vecino solución de la misma, pero de no ser así hará Backtracking y buscará otra ruta con los vecinos hasta llegar a una casilla donde se iguale el tamaño de la matriz y así dará esa solución.

12. Repositorio en GitHub

https://github.com/tsg3/PDC