INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA EN COMPUTACIÓN
ÁREA ACADÉMICA DE INGENIERÍA EN COMPUTADORES
LENGUAJES, COMPILADORES E INTÉRPRETES (CE-3104)

Tarea 1
PDC

Realizado por:
OLMAN CASTRO 2016003915
PABLO MORA
EDUARDO MOYA BELLO - 2015096664

Profesor:
Ing. Marco Hernández Vásquez

Cartago, 14 de mayo de 2019

Índice

Manual de usuario

Introducción

El programa PDC se encarga de resolver el problema del caballo, el cual es un antiguo problema matemático. Teniendo una cuadrícula de n x n casillas y un caballo de ajedrez colocado en una posición cualquiera (x, y) el caballo debe pasar por todas las casillas y una sola vez (Hernández, 2019).

Cómo ejecutar el programa

El programa está realizado en el lenguaje de programación Scheme y cuenta con cuatro funciones principales:

```
(PDC-Sol n pos)
(PDC-Todas n pos)
(PDC-Test n sol)
(PDC-Paint n sol)
```

`PDC-Sol` es una función que recibe un entero que dice el tamaño de la matriz (n) y una posición y un par ordenado (pos) que indica dónde se encuentra inicialmente el caballo. Por ejemplo:

```
> (PDC-Sol 5 '(4 4))
(list (list 4 21 16 11 6) (list 15 10 5 22 17) (list 20 3 24 7 12)
(list 9 14 1 18 23) (list 2 19 8 13 0))
```

`PDC-Todas` es igual que `PDC-Sol`, pero devuelve todas las soluciones posibles para el n definido.

`PDC-Test` es una función que recibe dos argumentos, un entero que indica el tamaño de la matriz y una estructura solución. Indica si es correcta o no mediante un booleano. Por ejemplo:

```
> (PDC-Test 5 '((4 21 16 11 6) (15 10 5 22 17) (20 3 24 7 12) (9
14 1 18 23) (2 19 8 13 0)))
#true
> (PDC-Test 5 '((0 21 16 11 6) (15 10 5 22 17) (20 3 24 7 12) (9
```

```
14 1 18 23) (2 19 8 13 4)))
#false
```

PDC-Paint recibe un entero que indica el tamaño de la matriz y una estructura solución, la idea es probar la solución encontrada pero utilizando la librería Racket Graphical Interface Toolkit y representarla de una forma gráfica/animada.

Funciones utilizadas

PDC-Sol

esta función recibe dos argumentos un entero que indica el tamaño de la matriz y un par ordenado que indica donde se encuentra inicialmente el caballo.
Como resultado debe retornar una estructura (a definir por el estudiante) con una solución de la secuencia de casillas que el caballo va visitando.

>(PDC-Sol 8 '(1 1))

(solución)

Para realizar esta función, se ocupan de varias funciones auxiliares, principalmente de (PDC-Sol-aux), pero también ocupa varias para el manejo de matrices, entre ellas podemos mencionar:

- (imprimir): Función para imprimir matriz, esta función básicamente recibe la matriz que se desea imprimir en consola y mientras no sea nula la devuelve.
- (crear-matriz): Función para crear una matriz con todos los valores que se encuentran en su interior iguales a 0. Esta función recibe una matriz y el tamaño (n*n) que desea el usuario que tenga la matriz y mientras n sea diferente de 1, se llama a (crear-matriz-aux)
- (crear-matriz-aux): Esta función básicamente llena la matriz de 0's recorriendo toda la matriz y usando la función (lista-ceros), hasta que el contador que es igual a 'n' sea 0.
- (lista-ceros): Al igual que la función anterior con ayuda de un contador llena una lista con 0's.
- (obt-ele-matriz): Esta función es de vital importancia, pues nos permite obtener un elemento específico de la matriz a partir de la fila y la columna asignada. Esta función recorre las listas y con ayuda de las funcione (obt-ele-lista) obtiene el elemento en la matriz.

- (obt-ele-lista): Esta función nos permite obtener un elemento específico de una lista, recorriendola con un contador hasta que el contador sea 0.
- (cambiar-ele-lista): Esta función recorre la lista hasta encontrar el elemento solicitado y cambia el valor sin modificar los otros elementos dentro de la lista.
- (cambiar-ele-matriz): Esta función recorre la matriz por listas y con ayuda de la función (cambiar-ele-lista) encuentra el elemento solicitado y cambia el valor sin modificar los otros elementos dentro de la matriz.

Todas estas funciones en conjunto me permite utilizar correctamente la función (PDC-Sol-aux), que verifica si es el primer valor para colocarlo, si el flag es 1 para imprimir la matriz, si los movimientos son válidos y en caso que se cumlpla todo llama recursivamente a la funcion sobreescribiendo los valores de la matriz para dar la resultante, sino cumple el movimiento válido este aumenta un contador para verificar si otro movimiento le permite lograrlo.

PDC-Todas

Al igual que la función anterior recibe dos parámetros, un entero que indica el tamaño de la matriz y un par ordenado que indica donde se encuentra inicialmente el caballo y retorna todas las soluciones posibles.

>(PDC-Todas 8 '(1 1))
((solución 1)
(solución 2)
…….
……..
(solución N))

Esta función realiza lo mismo que la anterior, solo que utiliza un backtracking para encontrar mas de una solución y por lo tanto encontrar todas las posibles.

PDC-Test

Validaciones

Se deben realizar varias validaciones para tener certeza de que la solución es correcta. La primera es que sea una matriz cuadrada NxN, por lo que la función `is-nxn` se encarga de verificar que el tamaño de la matriz (número de filas) sea igual al número de columnas. La función recibe una matriz y un n a verificar, y retorna un booleano. Para esto se requieren dos funciones auxiliares:

- `len`: recibe una lista y retorna la cantidad de elementos en ella (sumando 1 cada vez que se realiza un `cdr` mientras la lista no esté vacía).
- `rows-n`: es una función que recibe una matriz y verifica que todas sus filas sean del mismo tamaño (n).

Además, se debe validar que los elementos de la matriz no se repitan, por lo que se creó la función `not-repeated`, que llama a su función auxiliar `not-repeated-aux` la cual busca que haya un único número x en un intervalo de 0 a $n^2 - 1$.

Otra validación esencial es que el movimiento realizado sea aceptado como movimiento de caballo, es decir, que forme una letra L en la cuadrícula. De esto se encarga la función `l-move` la cual recibe dos pares ordenados y verifica que tengan una forma de L.

Funciones adicionales

Una función muy importante en `PDC-Test` es `pos` la cual indica la posición de un elemento de una matriz. La función recibe una matriz y un elemento y retorna el par ordenado de donde se encuentra dicho valor. Se utilizan tres funciones auxiliares:

- `pos-in-list`: indica la posición (índice) de un elemento en una lista.
- `pos-in-list-aux`: función auxiliar que permite la recursividad recibiendo la posición inicial de la lista (0 por defecto). Se va recorriendo la lista mediante el uso del `cdr` a medida que se aumenta la posición. Cuando ambos elementos son iguales, se retorna la posición. Si la lista se vacía, significa que no está el elemento por lo que se decidió retornar un -1.
- `pos-aux`: función auxiliar de `pos`. Verifica que el elemento se encuentre en las filas de la matriz mediante la función integrada `member`, y obtiene su

posición mediante `pos-in-list`. Si la matriz está vacía o no se encuentra el elemento, retorna (-1, -1).

Función final

La función final de `PDC-Test` cuenta con la función base y una auxiliar. La función base se encarga de validar que sea una matriz NxN (`is-nxn`) y que no se repitan elementos (`not-repeated`: cabe reiterar que también valida que haya un único elemento para cada posición de la matriz). La función auxiliar (`pdc-test-aux`) verifica que x y $x+1$ formen una L (o un movimiento de caballo válido.

PDC-Paint

Lorem ipsum…

Estructuras de datos utilizadas

Listas

Las listas son un conjunto de elementos ordenados. En Scheme, se escriben de la siguiente manera:

```
'(1 2 3 4 5)
```

Matrices

Una matriz no es más que una lista de listas. En el caso particular del problema del caballo, las matrices son de tamaño NxN (es decir, son cuadradas) por lo que la cantidad de elementos de la lista debe ser igual que la cantidad de elementos de las sublistas. Por ejemplo:

```
'((1 2) (3 4)) ;Matriz 2x2
'((1 2 3) (4 5 6) (7 8 9)) ;Matriz 3x3
'((1 2 3 4) (5 6 7 8) (9 10 11 12) (13 14 15 16)) ;Matriz 4x4
```

Algoritmos detallados

Como el paradigma que se trabaja en este caso es el funcional, los algoritmos detallados se encuentran explicados anteriormente en la sección de funciones utilizadas, ya que todos los algoritmos son funciones.

Problemas conocidos

Función index-of

Se investigó la existencia de una función integrada de Scheme llamada *index-of* para encontrar el índice de un elemento en una lista, pero en DrRacket se encontró el error de que la función no estaba definida.

Actividades realizadas por estudiante

Lorem ipsum…

Problemas encontrados

Lorem ipsum…

Conclusiones y Recomendaciones

Conclusiones

Lorem ipsum…

Recomendaciones

Lorem ipsum…

Referencias bibliográficas

- Enunciado
- https://docs.racket-lang.org/reference/pairs.html

Anexos

Anexo 1: Bitácora digital

Bitácora 1

Fecha: sábado 11 de mayo de 2019

Participante(s): Eduardo Moya

Asunto: Elaboración de PDC-Test

Descripción detallada: Se comenzó a la 1:08 p.m. con el desarrollo de la función PDC-Test. Debido a conversaciones informales que se tuvieron con los compañeros del equipo después del horario de clases, se tiene una idea de cómo proceder a encontrar una solución. Se empieza a crear una función, pero después de varios minutos se llega a la conclusión de que es más efectivo dividir el problema en varias funciones. Se inició con las validaciones.

Se procede a realizar una función que verifica que una matriz sea NxN (is-nxn). Se encuentra el problema de necesitar el tamaño de las filas y columnas por lo que se crea una función que lo soluciona. Se observa que la función is-nxn tiene una parte recursiva (ir verificando todas las filas de la matriz), por lo que se crea una función auxiliar. Asimismo, se crea una función que indica la posición de un elemento en la matriz. Se investigó la existencia de una función integrada de Scheme llamada *index-of* pero no funcionó en DrRacket. Para saber la posición de un elemento en una matriz también se necesita saber la posición de un elemento en una lista por lo que se crea una función para tal fin.

Continuando con las validaciones, se analiza alguna manera para validar que la solución propuesta sea la correcta. Además de que sea una matriz sea NxN, los elementos no se pueden repetir y tienen que ser todos diferentes, por lo que se crea una función que solucione este problema (not-repeated). También, todos los movimientos tienen que ser válidos, por lo que deben formar una letra L, entonces se crea una función que lo valide. Para realizar esta función, fue necesario tener una ayuda visual, ya que se torna muy complejo mediante el uso de *car* y *cdr*:

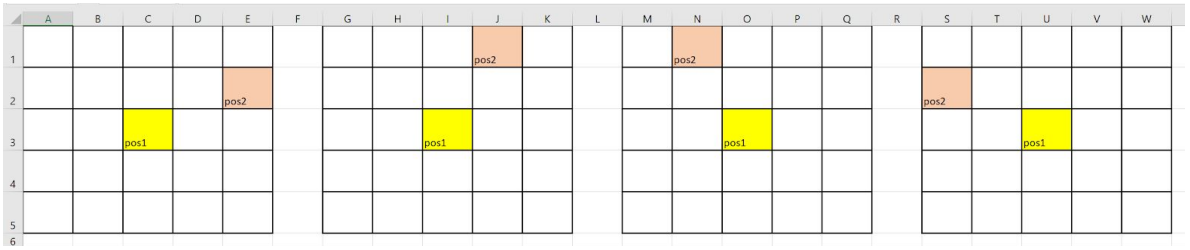

Figura 1: Ayuda visual de los posibles movimientos correctos. Realizado en Excel. Se muestran 4 de 8 soluciones

Posteriormente se unen todas las validaciones y se realiza la función final. De esta sesión, la mayoría de problemas fueron sintácticos en el manejo de paréntesis. Fue de gran ayuda el uso de paréntesis cuadrados. Se finalizó la sesión a 9:48 p.m. No hubo pausas durante el tiempo de trabajo.

Anexo 2: Diagrama de clases

Lorem ipsum...

Links que se usaron como base:

- https://www.geeksforgeeks.org/the-knights-tour-problem-backtracking-1/
- https://www.geeksforgeeks.org/warnsdorffs-algorithm-knights-tour-problem/