Instituto Tecnológico de Costa Rica
Engineering School
Computer Engineering
CE-3104 - Lenguajes, Compiladores e Interpretes
II Semester 2017

Project

Knight Tour

Jóse Andres Rivera Tencio, 2015045476
Kenneth Jeanpol Alvarado Mendez, 2015095715
Group 01
Professor: Marco Rivera

October 27, 2017

Contents

1 Abstract

Abstract

The horse problem (Wikipedia, Horse problem, 2017) is an old problem mathematician in which he is asked to, having a grid of n x n squares and a chess horse placed in any position (x, y), the horse passes through all the boxes (Wikipedia, Horse (chess), 2017) and once. This projects implements the Knight Tour programmed in Racket.

Keywords: Knight Tour, Backtracking, Heuristic, Matrix

2 objectives

2.1 General Objective

Develop an application that allows to reaffirm the knowledge of the paradigm of functional programming.

2.2 Specific Objectives

- Create an application that solves the problem of using DrRacket.
- Apply functional programming concepts.
- Create and manipulate lists as data structures.

3 User Guide

- First, you must run the program DrRacket:

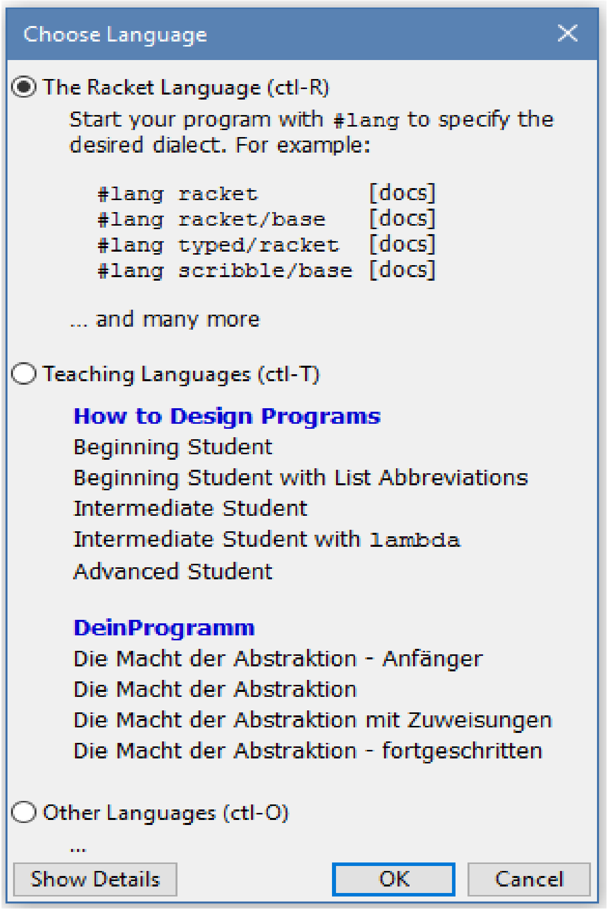
Figure 1

- Then open the file named "SchemeHorse"

SchemeHorse.rkt 25/10/2017 11:12 Racket Document 12 KB

Figure 2

- Make sure you select racket language in order to compile the program as follow in the next figure.



(●) The Racket Language (ctl-R)
Start your program with `#lang` to specify the desired dialect. For example:

```
#lang racket          [docs]
#lang racket/base     [docs]
#lang typed/racket    [docs]
#lang scribble/base   [docs]
```

... and many more

( ) Teaching Languages (ctl-T)

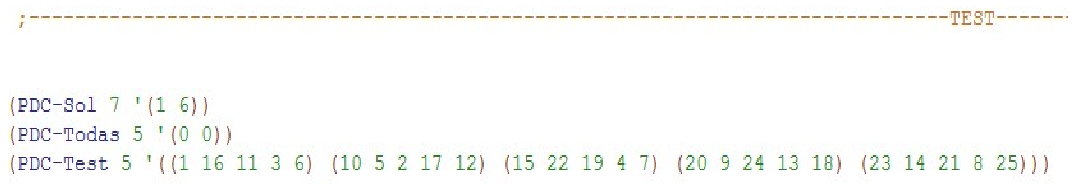
How to Design Programs
Beginning Student
Beginning Student with List Abbreviations
Intermediate Student
Intermediate Student with `lambda`
Advanced Student

DeinProgramm
Die Macht der Abstraktion - Anfänger
Die Macht der Abstraktion
Die Macht der Abstraktion mit Zuweisungen
Die Macht der Abstraktion - fortgeschritten

( ) Other Languages (ctl-O)
...

Figure 3

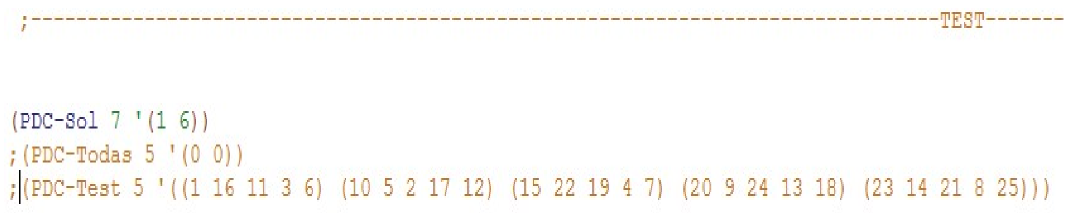
- Once the file is loaded, go to the section "TEST"



```
;-------------------------------------------------------------------------------TEST------


(PDC-Sol 7 '(1 6))
(PDC-Todas 5 '(0 0))
(PDC-Test 5 '((1 16 11 3 6) (10 5 2 17 12) (15 22 19 4 7) (20 9 24 13 18) (23 14 21 8 25)))
```

Figure 4

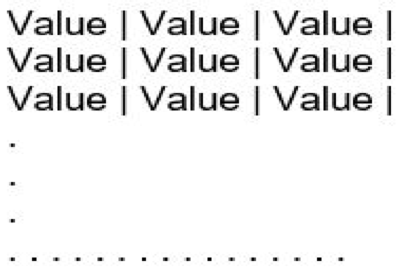
- In TEST, there are three functions to run, you must choose one and the remaining others should be commented as follows, placing a ("semicolon";) before making the call to the function.



```
;---------------------------------------------------------------------------------TEST-------


(PDC-Sol 7 '(1 6))
;(PDC-Todas 5 '(0 0))
;(PDC-Test 5 '((1 16 11 3 6) (10 5 2 17 12) (15 22 19 4 7) (20 9 24 13 18) (23 14 21 8 25)))
```

Figure 5

- To execute PDC-Sol you must pass certain parameters: first the value of the size of the matrix to be constructed being this nxn, so you must specify the value of n, and second you must pass a list containing the ordered pair (ij) where i is the row and j the column where the horse's path starts, the list in Scheme is specified as follows: '(), therefore the execution would look something thus: (PDC-Sol n '(ij)), and would return a matrix structure as follows



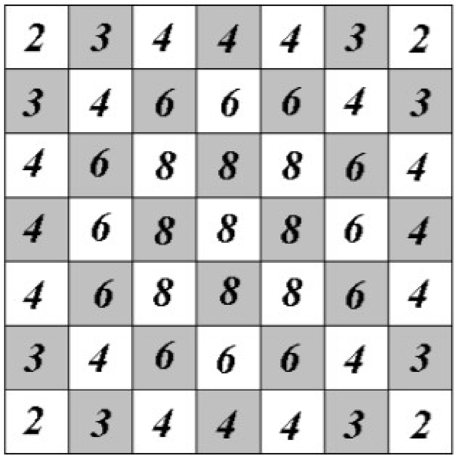
```
Value | Value | Value |
Value | Value | Value |
Value | Value | Value |
.
.
.
. . . . . . . . . . . . . . . . .
```

Figure 6

To solve this problem, we use the algorithm Backtracking, which has an efficiency

$$n^2$$

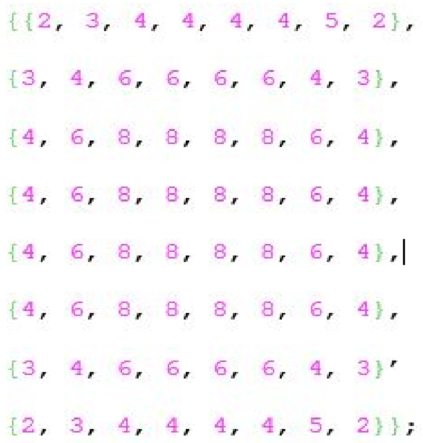
, so it is a slow method, the accessibility in a 7x7 matrix is as follows:



2	3	4	4	4	3	2
3	4	6	6	6	4	3
4	6	8	8	8	6	4
4	6	8	8	8	6	4
4	6	8	8	8	6	4
3	4	6	6	6	4	3
2	3	4	4	4	3	2

Figure 7

And the accessibility for an 8x8 is:

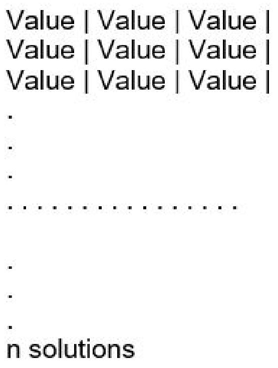


```
{{2, 3, 4, 4, 4, 4, 5, 2},
{3, 4, 6, 6, 6, 6, 4, 3},
{4, 6, 8, 8, 8, 8, 6, 4},
{4, 6, 8, 8, 8, 8, 6, 4},
{4, 6, 8, 8, 8, 8, 6, 4},
{4, 6, 8, 8, 8, 8, 6, 4},
{3, 4, 6, 6, 6, 6, 4, 3}'
{2, 3, 4, 4, 4, 4, 5, 2}};
```

Figure 8

Where 2 is the minimum value and 8 is the maximum, the greater the value of where to start the greater path is the duration and recursion that must be applied to reach the final solution, so if you want to apply PCD- Sol to an 8x8 matrix, we recommend one of the following ordered pairs '(0 0),' (0 7), '(7 0),' (7 7), which correspond to the corners of the board with a value of 2. These generates the solution in a time of: 5 minutes: 56 seconds, in a computer with the following characteristics: CPU: Intel Corei7, RAM: 8Gb. Any other ordered pair will take longer to run.

- To execute PDC-All you must pass certain parameters: first the value of the size of the matrix to be constructed being this nxn, so you must specify the value of n, and second you must pass a list containing the ordered pair (ij) where i is the row and j the column where the horse's path starts, the list in Scheme is specified as follows: '(), therefore the execution would look something as follows: (PDC-All n '(ij)) and would return several matrix structures as follows:



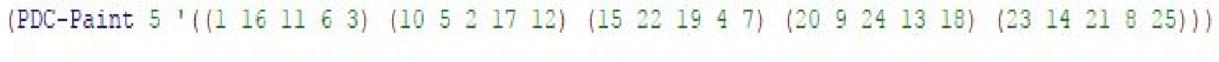
```
Value | Value | Value |
Value | Value | Value |
Value | Value | Value |
.
.
.
. . . . . . . . . . . . . . . .

.
.
.
n solutions
```

Figure 9

It is also recommended for this test a 5x5 matrix to be able to reach all the solutions in a humanly possible time.

- To execute PDC-Test you must pass certain parameters: first the value of the size of the matrix to be constructed being this nxn, so you must specify the value of n, and second you must pass a matrix containing the solution to the horse's path for later from this function to be able to validate if the solution is valid or not, the matrix in scheme is specified as follows: '(() () () ...) so the execution would look something like this: (PDC-Sol n '(() () () ....)).

- In order to execute PDC-Paint, that will show show the knight tour with an animation, is necessessary to call PDC-Paint with the first parameter the size of the square matrix, and the second parameter a valid solution expressed as a matrix. An example of the implementation of the function is the following one.



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

Figure 10

When it is exectued, a window will be shown with a grind on it, and red squares will be shown every second, in the position "i j" based on the matrix given a parameter as shown below

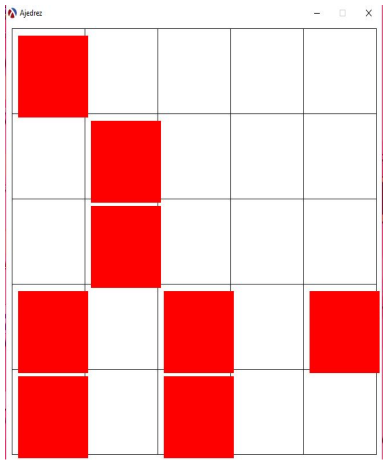


Figure 11

Finally, when all the tour is completed the grid will be fill with red squares as follows.

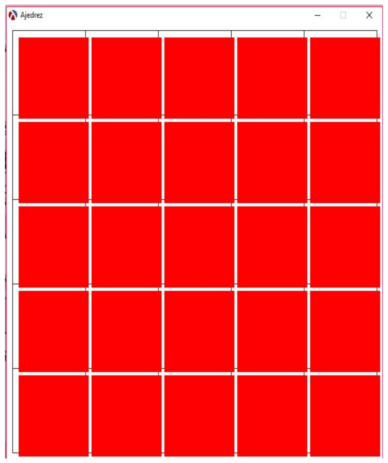


Figure 12

4 Description of the implemented functions.

4.1 (define (aux-Sol n ncuad matrix mov x y limit ejex ejey p sol))

This function is an auxiliary function used by PCD-Sol which receives (n, which is the number representing the size of the matrix), (ncuad, which represents the number of cells), (matrix, which refers to the (x, y), (y, position j), (axis x, motion matrix on axis x), (y axis, motion matrix on y axis), (p, counter), (sun, value 0 or 1), (limit, limit for movement).

4.2 (define (aux-Sol3 n ncuad matrix mov x y limit ejex ejey p sol)

This function is an auxiliary function used by PCD-All which receives (n, which is the number representing the size of the matrix), (ncuad, which represents the number of cells), (matrix, which refers to the (x, y), (y, position j), (axis x, motion matrix on axis x), (y axis, motion matrix on y axis), (p, counter), (sun, value 0 or 1), (limit, limit for movement).

4.3 (define (aux-Sol2 ncuad mov matrix matrixaux x y limit)

This function is an auxiliary function used by PCD-Test which receives (ncuad, which represents the number of squares), (matrix, which refers to the matrix generated at 0), (mov, motion), (x, position i), (y, position j), (axisx, motion matrix on axis x), (y axis, motion matrix on y axis), (limit, motion limit) .

4.4 (generateMatrixIJ n cont myList)

Generates a matrix fill wit 0s, where it size is the parameter nxn, cont is used to manage where will be the end of the matrix, and myList is a parameter where will be save the generation of the matrix, this parameter is initialize with an empty list.

4.5 (game n)

This function will initialize the GUI, which will show the knight tour, n is the size of the matrix.

4.6 (moveKnight i j color)

This function is used to paint red rectangles wich represent the path taken to complete the knight tour, where i is the position in the axis x, and j the position in the axis y, and color is the color selected to paint, in this case red.

4.7 (PDC-Paint n matrixSol)

This function calls PDC-PaintAux, which is in charge to show the GUI, with is knight tour, inside PDC-PaintAux is called the function game and moveKnight previously explained.

4.8 (printMatrix matrix N)

Is in charge to print the solution path given the matrix solution, and the size of the matrix, represented with the parameter N. It calls printList, for each element in the list matrix, and because is a matrix, will be able to print every row of the matrix.

5 Description of the data structures developed.

5.1 Lists

The linked list is an TDA that allows us to store data in an organized way, just like the vectors but, unlike these, this structure is dynamic, so we do not have to know "a priori" the elements that can contain. Lisp is based on List management

5.2 Matrixes

An array is a set of elements contiguous, all of the same type, sharing a name and which can be accessed through the (index) that occupies each of them in the matrix, as a vector or matrix in Algebra. Used basically to manage to grid of the GUI, and the knight tour.

6 Detailed description of the algorithms developed.

The algorithm used for PCD-Sol uses several cond, the first cond verifies if the movement is equal to 1, if it is then it updates the matrix in the initial position given with the value of the movement, makes the recursive call and verifies if the mov = 1 as already the mov has a value other than one then it passes to the next condition that verifies if there is solution, ie verify if sol has value of 1, as it is instantiated in 0 does not comply, otherwise it prints in console the matrix, then checks if the counter p is less than 8, this is because it is the subscript to use to go through the list of axis and axis and contains the movements which are only 8, if it is less then goes to verify if the x + axisx [p] is equal to 0 and less than the limit, verify the same with y + axisy [p], if that complies check if the box in that xyy updated with the movements equals 0, so that the horse has not passed out there, if that is true, verify that the movements are me n-squared, if it returns true then it makes a recursive call to the function with the updated array and the motion increased by 1, when p is no less than 8 does the backtracking returning to 8 the position it is evaluating at that time and Recalculates, when the motions are no longer smaller than n squared, then makes a recursive call to the function with sol = 1, thus checking if sol == 1 prints the final solution of the matrix.

The algorithm used for PCD-All uses several cond, the first cond verifies if the movement is equal to 1, if it is then it updates the matrix in the initial position given with the value of the movement, makes the recursive call and verifies if the mov = 1 as already the mov has a value other than one then it passes to the next condition that verifies if there is solution, ie verify if sol has value of 1, as it is instantiated in 0 does not comply, otherwise it prints in console the matrix, then checks if the counter p is less than 8, this is because it is the subscript to use to go through the list of axis and axis and contains the movements which are only 8, if it is less then goes to verify if the x + axisx [p] is equal to 0 and less than the limit, verify the same with y + axisy [p], if that complies check if the box in that xyy updated with the movements equals 0, so that the horse has not passed out there, if that is true, verify that the movements are smaller than n squared, if it returns true then it makes a recursive call to the function with the updated array and the movement increased by 1, when p is no less than 8 does the backtracking returning to 8 the position that it is evaluating at that instant and Recalculate, when the motions are no longer smaller than n squared, then makes a recursive call to the function with sol = 1, so when checking if sol == 1 prints the final solution of the array and re-runs.

The algorithm used in PCD-Test evaluates the first movement and updates the matrix in the xy position with a 0, this is to know that in positions with 0 there is no possibility of movement, then evaluates if all movements have already been reached, if not yet, then evaluates from the current position the next move to reach the box n + 1, if it meets all then returns the matrix, otherwise sends an error message.

7 Known Issues.

7.1 Optimize runtime of the algorithms.

Because of the use of bactracking algorithms, the runtime of some cases is not the best, so it takes a lot of time compiling the solution. We couldnt implement heuristic algoritms to solve this problem.

7.2 Matrix pixels out of bounds.

In the GUI, the route is painted with red rectangles, but they are painted out of the edges of the grid ("chess grid based o NXN given"). Because it is not a single case, we couldnt find a way to relate those pixels that are out of bounds based on the size of the grid

8 Problems Found

8.1 Matrix with zeroes

8.1.1 Problem

The path for the knight tour in the matrix were not completed, so many zeroes where in the "matrix solution"

8.1.2 Description

The matrix was printed with 0 in some indexes.

8.1.3 Solution Tries

- Debbuged the program with Dr Racket IDE to find where were the problem
- Reads ALOT of documentation from Racket to understand matrix management

8.1.4 Solution

Was solved applying the backtracking, since the 0 were due to incomplete solutions invalid.

8.1.5 Advice

Is recommended use Heuristic algorithms first, but if theres no solution to the problem then use bactracking algorithms.

8.1.6 Conclusion

The most effective way to find the horse's route is to use a heuristic algorithm, but it has its limitations, because when finding the best option, only one accepted route can be found.

8.1.7 Links

[Wikipedia]Wikipedia. (2017, 10 15). *https://es.wikipedia.org/wiki/Problema-del-caballoEl-problema-del-caballo-en-la-literatura* consultado el 22/10/2017

9 Work log

Table 1: Invested time on each task

Activity	**Performance**	**Date**	**Time**
José Rivera	Understand Knight Tour Algorithm	15/10/2017	6
José Rivera	Look for Knight Tour Examples	16/10/2017	6
José Rivera	Implementation of Knight Tour Examples	18/10/2017	12
José Rivera	Debugging Knight Tour algorithm in Racket	20/10/2017	8
José Rivera	Solved some bugs in the algorithm	23/10/2017	6
José Rivera	Documentation	25/10/2017	4
Kenneth Alvarado	Implementation of some matrix management algorithms	18/10/2017	6
Kenneth Alvarado	Look for information related to racket GUI	18/10/2017	8
Kenneth Alvarado	Implementation of the GUI for the Knight Tour animation	18/10/2017	6
Kenneth Alvarado	Comment code developed	18/10/2017	4
Kenneth Alvarado	Documentation	26/10/2017	3

10 Conclusions

- It can be concluded with the execution times obtained, that backtracking is extremely inefficient, however easier to program in the functional paradigm
- The most effective way to find the horse's route is to use a heuristic algorithm, but it has its limitations, because when finding the best option, only one accepted route can be found.
- If its needed to implement a GUI on Lisp language, is a good idea to include the library (lib "graphics.ss" "graphics")), in your project, and look information about it. It is a powerful tool to developed nice GUI in this programming language.
- The knight tour algorithm is only valid for grids nxn with n bigger than 4. As the n value increase, the run time will increase as well in order to get a solution.
- Not every i j position given will have a solution, nor the same number of solutions as in other position. It depends basically of the position given and the size of the matrix
- As the position given is close to the center of the matrix, will performed the solution faster. The reason is because it has more possibilities to start the tour, in the other hand, if the starting position is a corner will take more time, because it will only have 2 possible paths to star the tour.

References

[1] [Guzman]J. E. *Introducción a la programación con Scheme. Cartago: Editorial Tecnológica de Costa Rica.* consultado el 22/10/2017

[2] [Racket](2017, 08 15) *racket-lang.org. Retrieved from Racket: https://racket-lang.org/* consultado el 22/10/2017

[3] [Wikipedia]Wikipedia. (2017, 10 15). *https://es.wikipedia.org/wiki/Caballo-(ajedrez)* consultado el 22/10/2017

[4] [Wikipedia]Wikipedia. (2017, 10 15). *https://es.wikipedia.org/wiki/Problema-del-caballoEl-problema-del-caballo-en-la-literatura* consultado el 22/10/2017