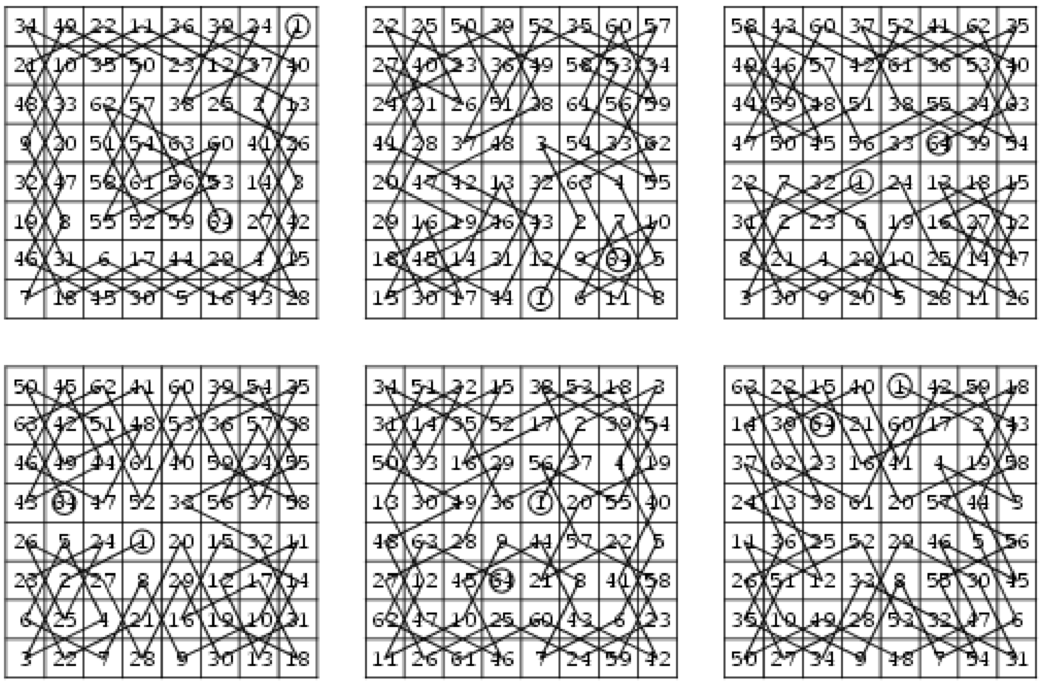
PHONG HUYNH
Final Project
Math 107

# KNIGHT'S TOUR

The purpose of this project is to explore all the possible moves that the Knight (chess piece) can move in an nxn board. This is also known as the Knight's Tour. A Knight's tour is a sequence of moves by a knight chess piece, which may only make moves that simultaneously shift one square along one axis and two along the other forming an L shape, such that each square of the board is visited exactly once. This is also a way in which we explore the Hamiltonian path.
The following is an illustration of how the graphs should look.



```
> restart;
> with(networks): with(linalg):
Warning, the names charpoly and rank have been redefined

Warning, the protected names norm and trace have been redefined and
unprotected
```

The Initial Purpose of the Knight's Tour

```
In this section, I am hoping to accomplish the initial
stage of developing a chessboard and hopefuly get some of
the knight moving procedures to work.  Most importantly get
Maple to produce all the possible moves that a knight can
move.
```

```
> new(G):
```

```
With the new graph G.  I want to create a board with an nxn
vertices using the following procedure called "board".  The
purpose of this procedure is to give the general nxn board
of vertices.

By assigning an initial size value I don't have to keep on
changing the values within my procedures later in the
program.  I can just change the Size value here and then
reproducing a new set of board and possible moves.
```

**WARNING:** DO NOT assign a big number for "Size > 5", unless you have the time to sit and wait..........for a really realllllllllly long time.

```
> Size:=5;
                                                Size := 5
> board := proc(n): for i from 1 to n
                    do for j from 1 to n
                      do addvertex([[i,j]],G):
                      od;
                    od;
         end proc:
Warning, `i` is implicitly declared local to procedure `board`

Warning, `j` is implicitly declared local to procedure `board`

> board(Size);
```

By assigning a Size to this procedure we can have all the vertices stored in the graph G with a Size-by-Size table.

```
                                                 [5, 5]
> vertices(G);
  {[1, 2], [2, 1], [4, 5], [5, 1], [4, 3], [4, 4], [3, 5], [4, 1], [4,
2], [3, 3], [3, 4],

        [3, 1], [3, 2], [2, 3], [2, 4], [2, 5], [1, 4], [1, 5], [2, 2],
[5, 5], [5, 4], [5, 2],

        [5, 3], [1, 3], [1, 1]}
```

The following procedure is to figure out the possible moves that a knight can move within an nxn board by assigning a vertex [i,j] to the procedure.  Since the knight is moving in an [i+1,j+2], [i+1,j-2], [i-1,j+2], [i-1,j-2], [i+2,j+1], [i+2,j-1], [i-2,j+1], or [i-2,j-1] position, the procedure can be written as follow.

```
> possiblemoves := proc(v,n):
>   w:=[seq(0,i=1..8)];
  w[1] := v + [2,1];
>   w[2] := v + [1,2];
>   w[3] := v + [-2,1];
>   w[4] := v + [1,-2];
>   w[5] := v + [-1,-2];
>   w[6] := v + [-2,-1];
>   w[7] := v + [-1,2];
>   w[8] := v + [2,-1];
>   for i from 1 to 8 do
    if ((0 < w[i][1]) and (w[i][1] < n+1) and (0 < w[i][2])
and (w[i][2] < n+1))
    then connect({v},{w[i]}, G)
    end if;
  end do;
end proc:
Warning, `w` is implicitly declared local to procedure `possiblemoves`

Warning, `i` is implicitly declared local to procedure `possiblemoves`
```

Notice that the result will not display all the vertices that connected to the initial vertex. Instead it displays the edge that is being connected to the initial vertex.

```
> possiblemoves([2,3],4);
                                            e4
```

The resulting vertices of the initial vertex can be obtain by using the neighbors method. This method will display all the vertices that are connecting to the initial vertex. Unfortunately, this does not tell me anything.  I don't want Maple to give me just a few vertices that connected to the initial vertex.

```
neighbors([2,3],G);
                              {[4, 4], [4, 2], [3, 1], [1, 1]}
```

I'm hoping that edges(G) would tell me something about all the edges that are being connected.

```
> edges(G);
                                     {e3, e4, e2, e1}
>
```

Finally, the following procedure chessTable will generate a chess table for the possiblemoves procedure that contains all the possible moves (in vertices) with the given Size value.

```
> chessTable:= proc(n): for i from 1 to n
                        do for j from 1 to n
                          do possiblemoves([i,j],n)
                          od;
                        od;
             end proc:
```

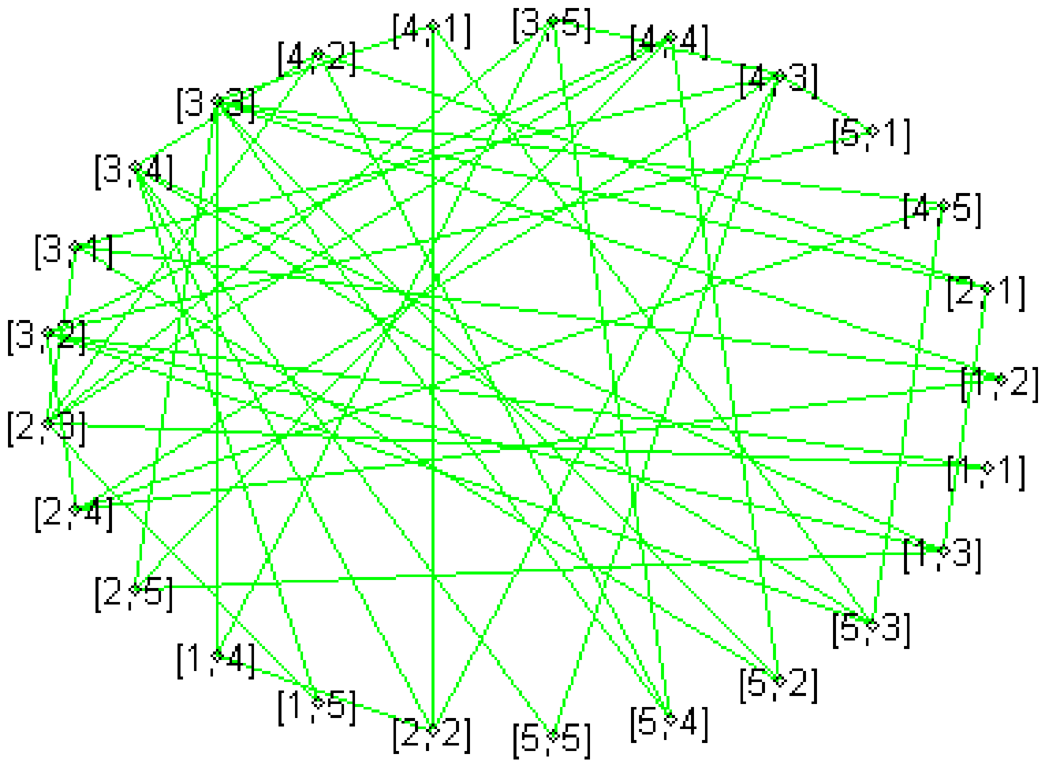
```
Warning, `i` is implicitly declared local to procedure `chessTable`

Warning, `j` is implicitly declared local to procedure `chessTable`
```

By calling Size in chessTable, I am able to generate all the vertices on the chess board. YES!!!!! This is what I want Maple to display. Hopefuly this will give me all the vertices that I needed to work with.

```
> chessTable(Size);
ends(G);
  {{[3, 3], [5, 4]}, {[5, 1], [3, 2]}, {[4, 5], [3, 3]}, {[3, 2], [1, 
3]}, {[4, 1], [3, 3]},

         {[4, 3], [2, 2]}, {[3, 3], [5, 2]}, {[3, 5], [1, 4]}, {[1, 4], 
[2, 2]}, {[4, 2], [3, 4]},

         {[1, 2], [3, 3]}, {[3, 4], [5, 5]}, {[3, 2], [2, 4]}, {[2, 3], 
[1, 5]}, {[4, 3], [3, 5]},

         {[3, 4], [5, 3]}, {[4, 5], [2, 4]}, {[3, 3], [1, 4]}, {[3, 4], 
[1, 5]}, {[1, 2], [2, 4]},

         {[3, 5], [5, 4]}, {[4, 3], [2, 4]}, {[2, 1], [4, 2]}, {[1, 2], 
[3, 1]}, {[4, 1], [5, 3]},

         {[2, 1], [3, 3]}, {[3, 3], [2, 5]}, {[4, 2], [5, 4]}, {[2, 3], 
[1, 1]}, {[3, 1], [2, 3]},

         {[5, 1], [4, 3]}, {[4, 4], [2, 3]}, {[4, 4], [2, 5]}, {[4, 3], 
[5, 5]}, {[3, 4], [1, 3]},

         {[3, 2], [1, 1]}, {[4, 2], [2, 3]}, {[3, 4], [2, 2]}, {[3, 1], 
[5, 2]}, {[2, 5], [1, 3]},

         {[4, 1], [2, 2]}, {[4, 3], [3, 1]}, {[4, 4], [5, 2]}, {[3, 2], 
[5, 3]}, {[4, 5], [5, 3]},

         {[4, 4], [3, 2]}, {[3, 5], [2, 3]}, {[2, 1], [1, 3]}}
```

Here is the result when use the draw(G) method. Notice this is not the result that I want Maple to display. Unfortunately, Maple is unable to produce a grid table that I want it to.

```
> draw(G);
```



AHH! You would think that I would be giving up by now. Wait until you see the next section.
>

## Knight's Tour

With the help of  Sonja Willis's project on Hamiltonian Paths and Cayley Digraphs of Algebraic Group.  I'm hoping that it would help me in finding the tour for Maple to graph.  The following codes are taken from Sonja Willis's project.

```
> continuepath := proc(Graph,path)
  local daugh;
  daugh:=sort([op(departures(path[1],Graph) minus 
{op(path)})]);

    if daugh = []
```

```
      then RETURN ([path,false]);
      else RETURN ([[daugh[1],op(path)],true]);
    fi;
end:
> backtrack := proc( Graph, origpath)
  local bad, daugh, position, path ;
  path := origpath;
  do
    bad:=path [1];
    path:=path [2..nops(path)];
    if path = []
      then RETURN ([path, false]);
    fi;
     #print (path);
    daugh:=sort([op(departures(path[1],Graph) minus
{op(path)})]);
    member (bad,daugh,'position');
     #print(`da-po-ba`,daugh,position,bad);
    if position < nops(daugh)
      then RETURN ([[daugh[position + 1],op(path)],true]);
    fi;
     #print(`bac-path`,path);
  od;
end:
> HamiltonianPath := proc(Graph,start)
  local path, result;
  path := [start];
  do
    do
      result := continuepath (Graph,path);
      path := result[1];
      # print (path);
      if result[2] = false
        then break;
      fi;
    od ;

    if nops(result[1]) = nops(vertices(Graph))
      then RETURN (path);
    fi ;

    result:=backtrack (Graph,path);
    if result[2]=false
      then print (`NO PATH TRY ANOTHER VERTEX`); break ;
      else path :=result[1];
    fi;
  od;
```

```
end:
```

It is time for me to check out what this HamiltonianPath procedures would give me.  By calling my original graph G and assigning an initial vertex, this is what HamiltonianPath gives me.

```
> tour := HamiltonianPath(G,[1,1]);
>
  tour := [[5, 5], [3, 4], [1, 5], [2, 3], [3, 1], [5, 2], [4, 4], [2,
5], [1, 3], [2, 1],

        [4, 2], [5, 4], [3, 3], [1, 2], [2, 4], [4, 5], [5, 3], [4, 1],
[2, 2], [1, 4], [3, 5],

        [4, 3], [5, 1], [3, 2], [1, 1]]
```

It turns out that HamiltonianPath procedure does give me a path that connects all the vertices on my chess board.  It works, but now, how am I going to transfer all these points (vertices) into a grid table that I have always wanted?

Since Maple cannot produce a grid table, I'm hoping that matrix will give me some kind of output that would be similar to the grid table.  Then again, it seems as though matrix is the only thing that is close to a grid table. Worth a try!

```
> m := matrix(Size,Size,0);
```

$$m := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

I am crossing my finger and hoping that the following codes will produce the right matrix that display the path that the Knight's tour gives.

```
> n := nops(tour):
                for i from 1 to n
                   do m[tour[n-i+1][1],tour[n-i+1][2]] :=
i;
                   od:
> matrix(m);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

YEA HEW!!!!!!!!!!!!!!  This is exactly what I wanted since the beginning of the project.  This is really exciting!

The following results are displayed by generating a 5 by 5 table.  Notice that every time Maple generates, the results are different.  The reason for this is that the HamiltonianPath is generating in a random form and it is depending on the points that Maple pick to generate the graph. The chronological number from 1 to Size-by-Size represent the order in which the knight is moving.

$$\begin{bmatrix} 1 & 10 & 5 & 18 & 3 \\ 14 & 19 & 2 & 11 & 6 \\ 9 & 22 & 13 & 4 & 17 \\ 20 & 15 & 24 & 7 & 12 \\ 23 & 8 & 21 & 16 & 25 \end{bmatrix} \quad \begin{bmatrix} 1 & 18 & 5 & 12 & 3 \\ 8 & 13 & 2 & 17 & 6 \\ 19 & 22 & 7 & 4 & 11 \\ 14 & 9 & 24 & 21 & 16 \\ 23 & 20 & 15 & 10 & 25 \end{bmatrix} \quad \begin{bmatrix} 1 & 4 & 11 & 16 & 25 \\ 12 & 17 & 2 & 5 & 10 \\ 3 & 20 & 7 & 24 & 15 \\ 18 & 13 & 22 & 9 & 6 \\ 21 & 8 & 19 & 14 & 23 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 14 & 19 & 8 & 25 \\ 6 & 9 & 2 & 13 & 18 \\ 15 & 20 & 7 & 24 & 3 \\ 10 & 5 & 22 & 17 & 12 \\ 21 & 16 & 11 & 4 & 23 \end{bmatrix}$$

```
>
```

**Final Thoughts**

WELL, The project turns out to be as what I wanted it to be but the task did not reach its conclusion as easy as  I thought.  When I first started the project, I did not know what to do and how to begin the task.  Knight's tour seems to have a simple task, as I begin to explore the tour and with the helps of Stefan Erickson, I find out that the simple task become a really big task.  The biggest challenge for me is to write a procedure that teaches Maple to move the initial vertex to its final destination and then print out the results in a gridded board.  As you have read the step by step that I have written during the mission in accomplishing the Knight's tour,  it have taken me this long to have mission accomplished!  I am a happy man!!  Thank you for following my task.

Special thanks to Professor J. Wavrik and Stefan Erickson for helping me with this mission. And thanks Sonja Willis for her HamiltonianPath codes.