Backtracking Problems

We will consider some problems that can be solved by Backtracking, i.e. in trying to find a solution, the program tries potential solutions. If it fails at a particular point it backtracks, i.e. it undoes the current attempt and tries another. It contnues until success or it fails having tried all possiblities.

We will consider (maybe with variations) the following problems.

Knights Journey / Tour

Generating Permutations	Generating Combinations
The Eight Queens Problem	Generating subsets

The Knight's Journey

Problem: (See Wirth "Algorithms & Data Structures" Ch 3.4)

Given an NxN "chessboard" (N^2 squares), determine, if possible, all the moves starting from (x_0,y_0) such that the knight visits all squares on the board.

Can a knight starting from (x_0, y_0) visit all the other (N^2-1) squares on board exactly once, i.e. the knight visits all the squares and visits each square just once. This may not be possible, and the program should indicate so.

```
Try(k, x, y)  -- attempt move k from (x,y)
        ...
do
        if "Board Full" then
                "Finished"
        else
                from
                        "start move"
                until
                        "No more Moves" or "success"
                loop
                        "Select next Move"
                        If "acceptable" then
                                "record move"
                                Try(k+1, "New (x,y)")
                                If not "success' then
                                        "reset  New(x,y)"
                                end
                        end
                end -- loop
        end
end -- Try
```

Determining a Knights Move

Given a position (x,y) on the board we want to find all possible locations for the next move by a knight. A knight can move one square straight in any direction and then one move diagonal away from start. That is, a knight can move one square N, S, E or W and then if it moved N it can continue one square in either NW or NE.

A single move of a Knight can be implemented using arrays.

Let row, col : ARRAY[INTEGER] then (indexing from 1 to 8)

```
row := << -2, -1, 1, 2, 2, 1, -1, -2>>
col := <<1, 2, 2, 1, -1, -2, -2, -1>>
```

To find a next position for the Knight from (x,y),

```
New_x := x + row(k)
New_y := y + col(k)
```

Algebraic Formula for a single Knight's Move

We note that the "distance" from (x,y) to (New_x, New_y) is

$\sqrt{\text{row}(k)^2 + \text{col}(k)^2} = \sqrt{5}$ i.e. $\text{row}(k)^2 + \text{col}(k)^2 = 5$

Rather than have to work out the array initialisations, we can write a procedure to calculate the arrays row and col.

```
Init_Moves is
        local
                i,j,k : INTEGER
        do
                k := 1
                from
                        i := -2
                until
                        i > 2
                loop
                        from
                                j := -2
                        until
                                j > 2
                        loop
                                if i*i + j*j = 5 then
                                        row.put(i,k)
                                        col.put(j,k)
                                        k:=k+1
                                end
                                j := j+1
                        end
                        i := i+1
                end
        end --Init_Moves
```

Representation of Chessboard

Let the attribute

board : ARRAY2[INTEGER] -- 2-dim array / matrix

be used to represent the chessboard.

board.item(x,y) = 0 indicates square (x,y) is free,

board.item(x,y) = k, square (x,y) visited on the kth move.

Initially we will set square (x_0,y_0) to 1.

The attribute

success : BOOLEAN

indicates whether success was achieved.

We can expand the "pseudo-code" in the Try procedure

is replaced by "i > N*N"

<u>"Acceptable"</u>

The next move for the knight is acceptable if it is on the board and the square has not already been visited.

i.e. $1 \le new\_x \le N$

and $1 \le new\_y \le N$

and board(new_x, new_y) = 0.

<u>"record / reset"</u>

board.put(k,x,y) records the kth move on square (x,y) and board.put(0,x,y) resets the square (x,y) so that it can be visited later.

The procedure Try

Our problem is to find a Knights Journey around the board. There may be none or they may be many. We are interested in finding the 'first' solution. The critical routine Try is given as:

```
Try_Next_Move(k, x,y :INTEGER) is
    local
        size, d, new_x, new_y : INTEGER
    do
        size := board.size1*board.size2
                                        -- board may be rectangular
        if k > size then
            success := True
        else
            from
                d := 0
            until
                d = 8 or success
```

```
                        loop
                                d := d+1
                                new_x := x + row.item(d)
                                new_y := y + co.item(d)
                                if Acceptable(new_x, new_y) then
                                         board.put(k, new_x,new_y)
                                        Try_Next_Move(k+1,new_x,new_y)
                                        if not success then
                                                        board.put(0,new_x,new_y)
                                        end
                                end
                        end --loop
                end
        end -- Try_Next_Move
```

```
Acceptable(s, t : INTEGER) : BOOLEAN is
        do
                result   :=      s >= 1 and s <= board.size1
                                        and t >= 1 and t <= board.size2
                                        and then board.item(s,t) = 0
        end -- Acceptable
```

The main routine, Knights

The routine Knights initialises the board and calls the procedure, Try_Next_Move. It then checks for a solution.

```
        Knights is
        local
                ...
        do
                ...

                Init_Moves -- set row & col for a knight move
                board.put(1,x0,y0) -- start at (x0,y0)
                Try_Next_Move(2,x0,y0)
                -- Try next move from the current square
                if not success then
                        io.put_string("No Solution")
                else
                        "Display board / solution"
                end
        end -- Knights
```

Find All solutions to the Knights Journey

The above procedure finds the ‘first’ solution if any. In finding all solutions we do not have to exit the loop using ‘success’.

```
Try_All(k, x,y : INTEGER) is
    local
        d, New_x, New_y : INTEGER
    do
        if k > board.size1*board.size2 then
            “Display Board”
        else
            from
                d := 1
            until
                d > 8
            loop
                New_x := x + row.item(d)
                New_y := y + col.item(d)
                if Acceptable(New_x, New_y) then
                    board.put(k, New_x,New_y)
                    Try_All(k+1, New_x,New_y)
                    board.put(0,New_x,New_y)
                end
                d := d+1
            end
        end
    end -- Try_All
```

Related Knight Journey Problems

The Circular Journey or the Knights Tour. (tour = travel round; journey = go to a place)
In Wirth, the tour may not come back to the start.
Problem:
Is there a tour starting from, say (1,1) that ends within a knights move of (1,1). On a 8x8 board the 65th move could lead back to the start.
? Why is there no (circular) tour for a 5x5 board.?

Delete Opposite corners of the board.
Is there a knights journey when opposite corner squares are removed.
If n is even then a knights journey is impossible.
Reason:
Consider the board as a chessboard. A knights moves alters colours. So starting with black the moves alternate with **black - white - black- .....-black**
Since opposite corners are the same colour, say white, then the knight has to cover 32 black and 30 white squares. But the best that can be done is 31 black and 30 white i.e. start on a black and alternate. After the 61st move there will be no white square to move to.

Solution: Knights Tour (closed journey) for an 8x8 board

	1	2	3	4	5	6	7	8
1	1	8	11	18	35	6	33	20
2	10	17	2	7	12	19	36	5
3	15	64	9	52	3	34	21	32
4	50	25	16	13	22	57	4	37
5	63	14	51	24	53	38	31	58
6	26	49	42	45	60	23	56	39
7	43	62	47	28	41	54	59	30
8	48	27	44	61	46	29	40	55

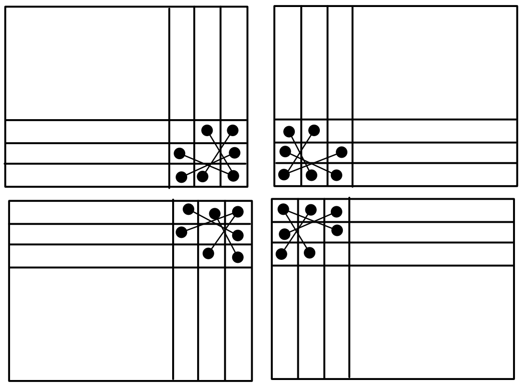
Solution: Knights Tour (closed journey) for an 8x8 board that satisfies the 'join property' -- from Parberry

Parberry's article can be downloaded (.ps format) at

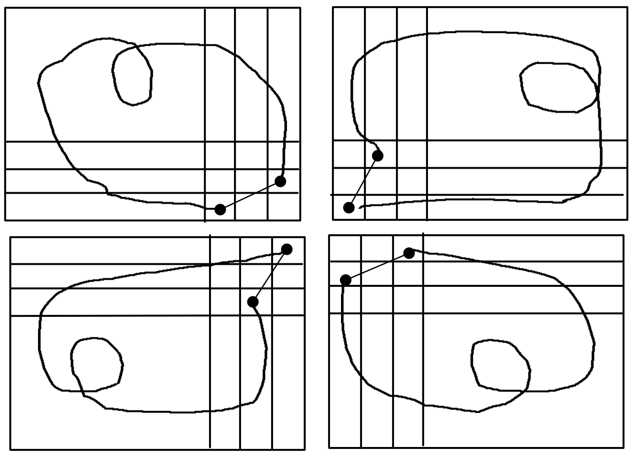
http://hercule.csci.unt.edu/~ian/papers/knight2.html

	1	2	3	4	5	6	7	8
1	1	46	17	50	3	6	31	52
2	18	49	2	7	30	51	56	5
3	45	64	47	16	27	4	53	32
4	48	19	8	29	10	55	26	57
5	63	44	11	22	15	28	33	54
6	12	41	20	9	36	23	58	25
7	43	62	39	14	21	60	37	34
8	40	13	42	61	38	35	24	59

Join Property



Initial 4 tours



Combine them to get