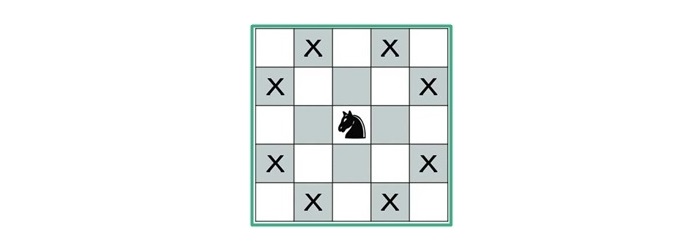
## What is Knight's tour problem?

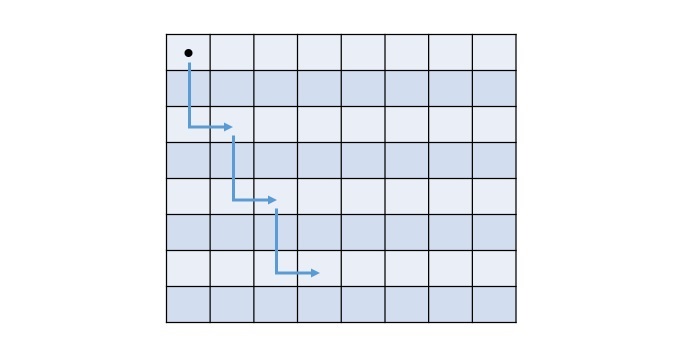
In the **knight's tour** problem, we are given with an empty chess board of size NxN, and a knight. In chess, the knight is a piece that looks exactly like a horse. Assume, it can start from any location on the board. Now, our task is to check whether the knight can visit all of the squares on the board or not. When it can visit all of the squares, then print the number of jumps needed to reach that location from the starting point.

There can be two ways in which a knight can finish its tour. In the first way, the knight moves one step and returns back to the starting position forming a loop which is called a **closed tour**. In the second way i.e. the **open tour**, it finishes anywhere in the board.

For a person who is not familiar with chess, note that the knight moves in a special manner. It can move either two squares horizontally and one square vertically or two squares vertically and one square horizontally in each direction. So, the complete movement looks like English letter **'L'**.



Suppose the size of given chess board is 8 and the knight is at the top-left position on the board. The next possible moves are shown below −



Each cell in the above chess board holds a number, that indicates where to start and in how many moves the knight will reach a cell. The final values of the cell will be represented by the below matrix −

0 59 38 33 30 17 8 63

37 34 31 60 9 62 29 16

58 1 36 39 32 27 18 7

35 48 41 26 61 10 15 28

42 57 2 49 40 23 6 19

47 50 45 54 25 20 11 14

56 43 52 3 22 13 24 5

51 46 55 44 53 4 21 12

Remember, this problem can have multiple solutions, the above matrix is one possible solution.

One way to solve the knight's tour problem is by generating all the tours one by one and then checking if they satisfy the specified constraint or not. However, it is time consuming and not an efficient way.

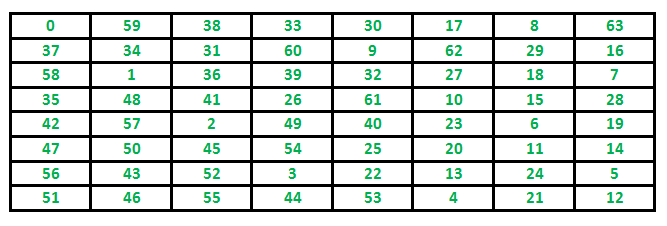
## Backtracking Approach to Solve Knight's tour problem

The other way to solve this problem is to use backtracking. It is a technique that tries different possibilities until a solution is found or all options are tried. It involves choosing a move, making it, and then recursively trying to solve the rest of the problem. If the current move leads to a dead end, we backtrack and undo the move, then try another one.

To solve the knight's tour problem using the backtracking approach, follow the steps given below −

* Start from any cell on the board and mark it as visited by the knight.
* Move the knight to a valid unvisited cell and mark it visited. From any cell, a knight can take a maximum of 8 moves.
* If the current cell is not valid or not taking to the solution, then backtrack and try other possible moves that may lead to a solution.
* Repeat this process until the moves of knight are equal to 8 x 8 = 64.

**The path followed by Knight to cover all the cells**  
Following is a chessboard with 8 x 8 cells. Numbers in cells indicate the move number of Knight.



**Time Complexity :**  
There are N2 Cells and for each, we have a maximum of 8 possible moves to choose from, so the worst running time is O(8N^2).

**Auxiliary Space:**O(N2)

**Important Note:**  
No order of the xMove, yMove is wrong, but they will affect the running time of the algorithm drastically. For example, think of the case where the 8th choice of the move is the correct one, and before that our code ran 7 different wrong paths. It’s always a good idea a have a heuristic than to try backtracking randomly. Like, in this case, we know the next step would probably be in the south or east direction, then checking the paths which lead their first is a better strategy.

Note that Backtracking is not the best solution for the Knight’s tour problem. See the below article for other better solutions. The purpose of this post is to explain Backtracking with an example.