Backtracking Algorithms

Dr. Navjot Singh

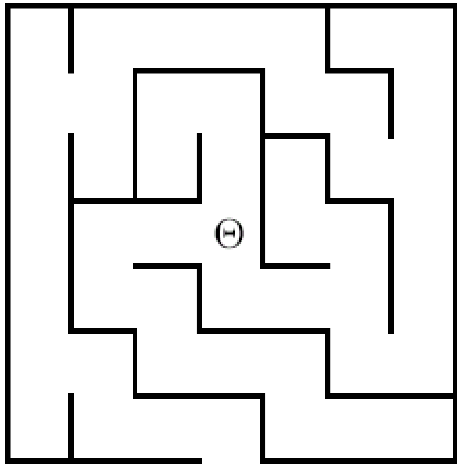
Design and Analysis of Algorithms

Backtracking

- Suppose you have to make a series of *decisions*, among various *choices*, where
  - You don't have enough information to know what to choose
  - Each decision leads to a new set of choices
  - Some sequence of choices (possibly more than one) may be a solution to your problem
- Backtracking is a methodical way of trying out various sequences of decisions, until you find one that "works"

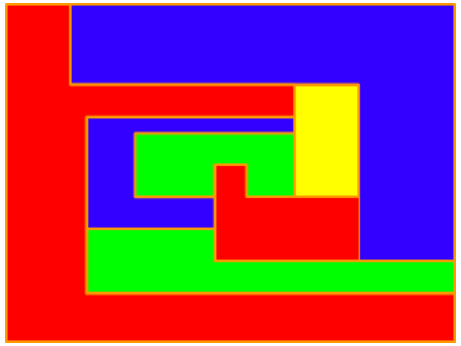
Solving a maze

- Given a maze, find a path from start to finish
- At each intersection, you have to decide between four or fewer choices:
  - Go left
  - Go right
  - Go up
  - Go down



- You don’t have enough information to choose correctly
- Each choice leads to another set of choices
- One or more sequences of choices may (or may not) lead to a solution

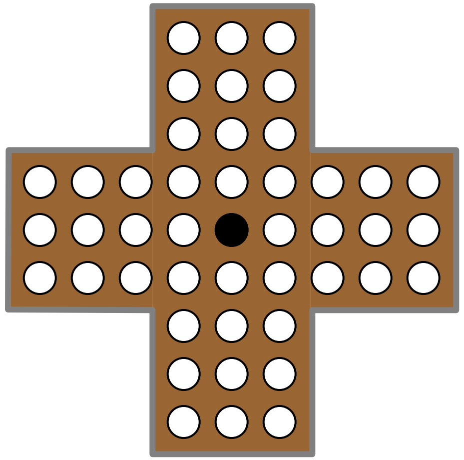
Coloring a map



- You wish to color a map with not more than four colors
  - red, yellow, green, blue
- Adjacent countries must be in different colors
- You don’t have enough information to choose colors
- Each choice leads to another set of choices
- One or more sequences of choices may (or may not) lead to a solution
- Many coloring problems can be solved with backtracking

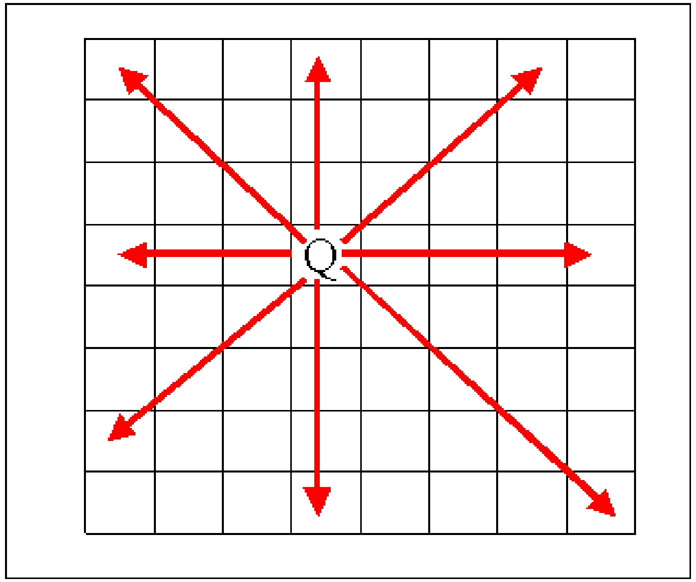
Solving a puzzle

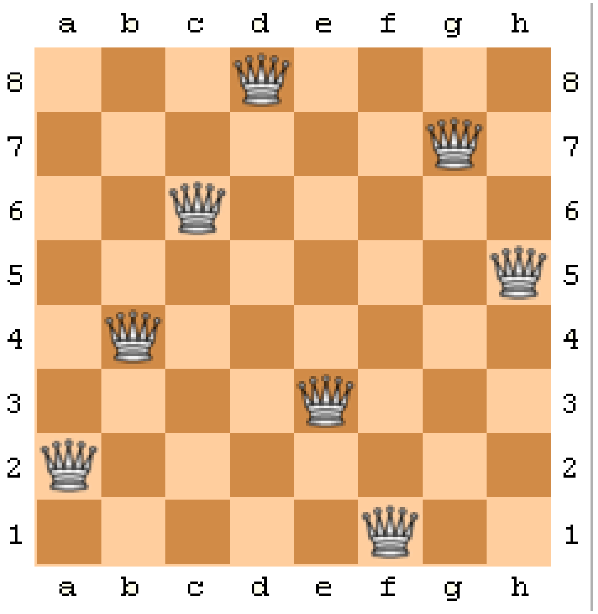
- In this puzzle, all holes but one are filled with white pegs
- You can jump over one peg with another
- Jumped pegs are removed
- The object is to remove all but the last peg
- You don't have enough information to jump correctly
- Each choice leads to another set of choices
- One or more sequences of choices may (or may not) lead to a solution
- Many kinds of puzzle can be solved with backtracking



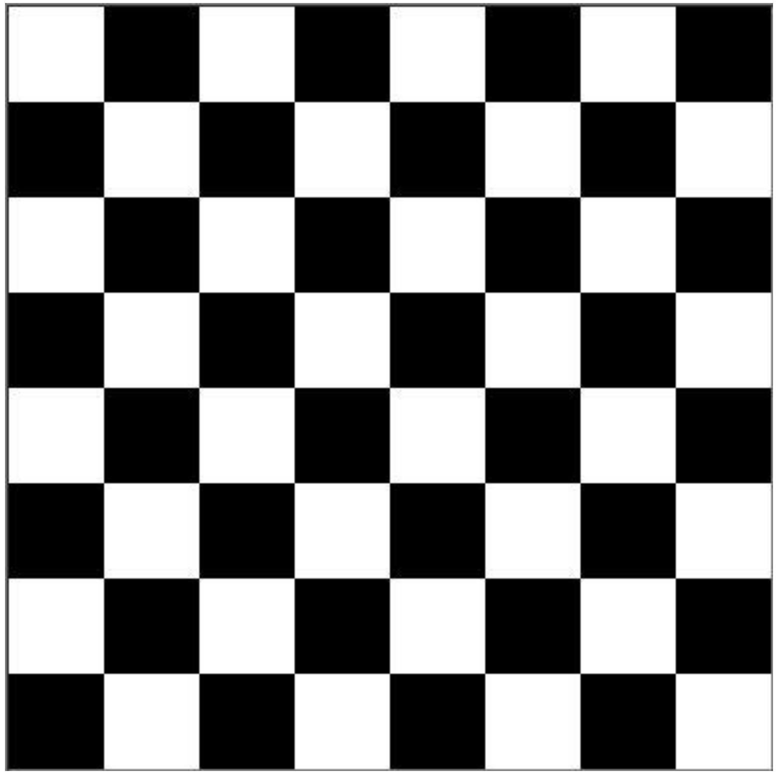
The 8 Queens Problem

- A classic chess puzzle
  - Place 8 queen pieces on a chess board so that none of them can attack one another



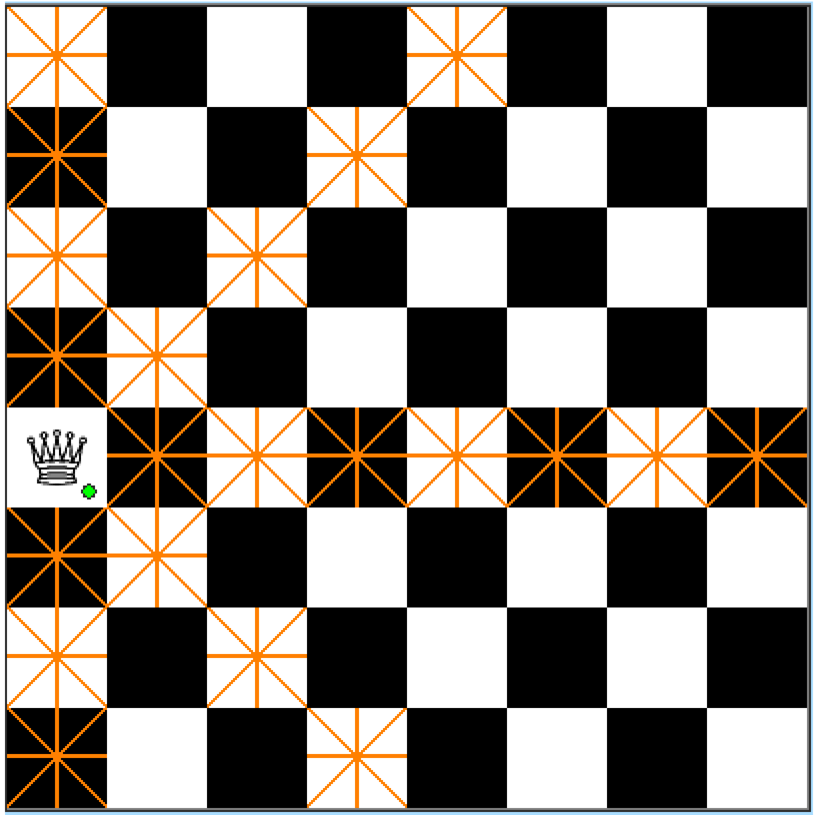


The 8 Queens Problem



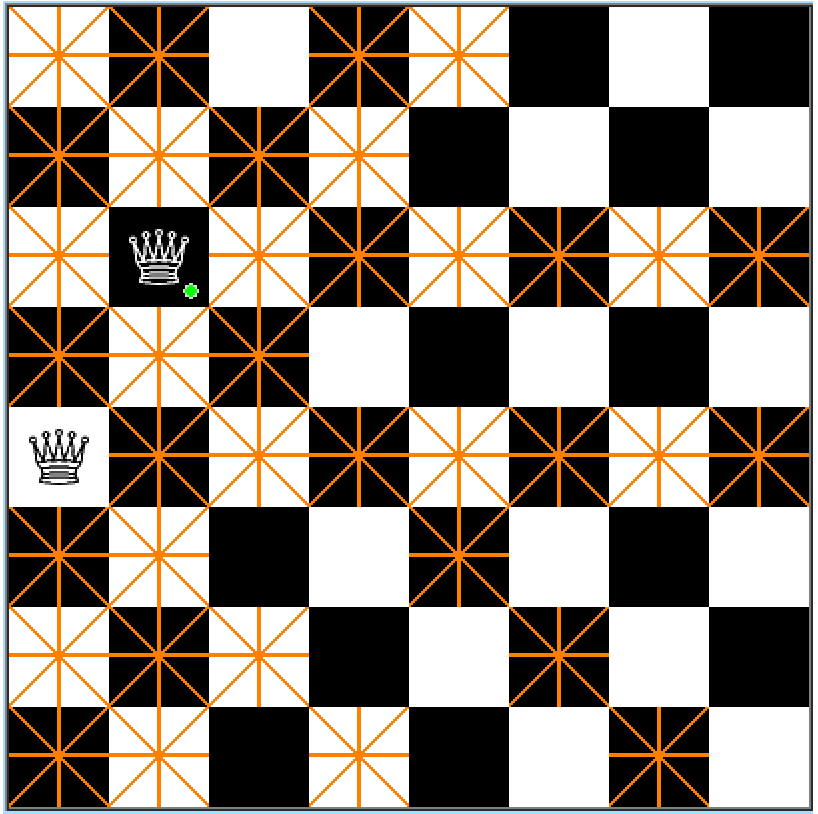
- It is an empty 8 x 8 chess board. We have to place the queens in this board.

The 8 Queens Problem



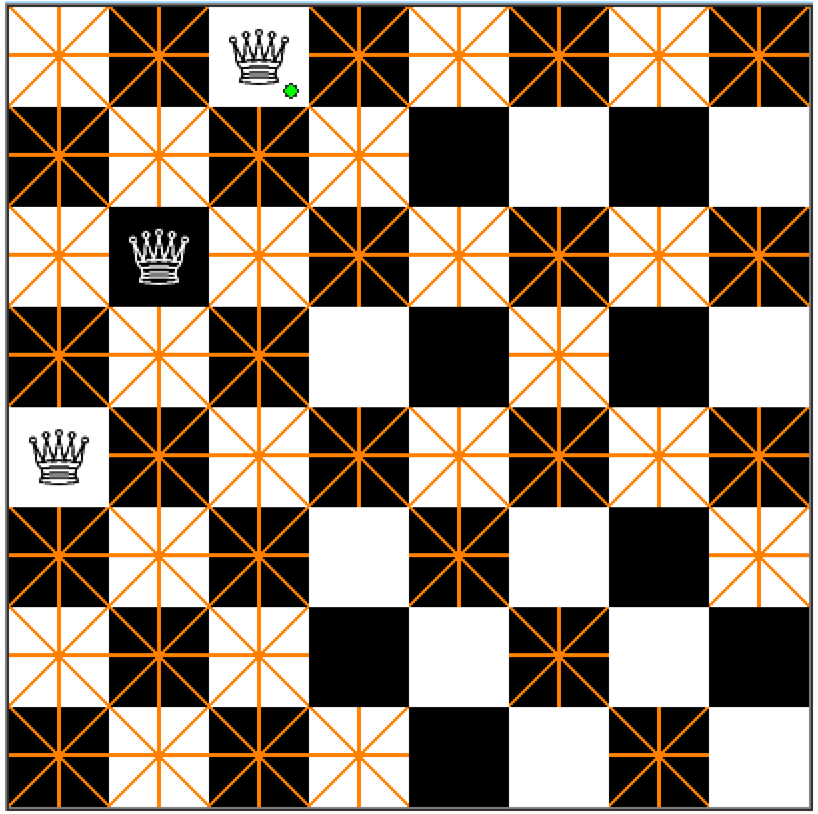
- We have placed the first queen on the chess board

The 8 Queens Problem



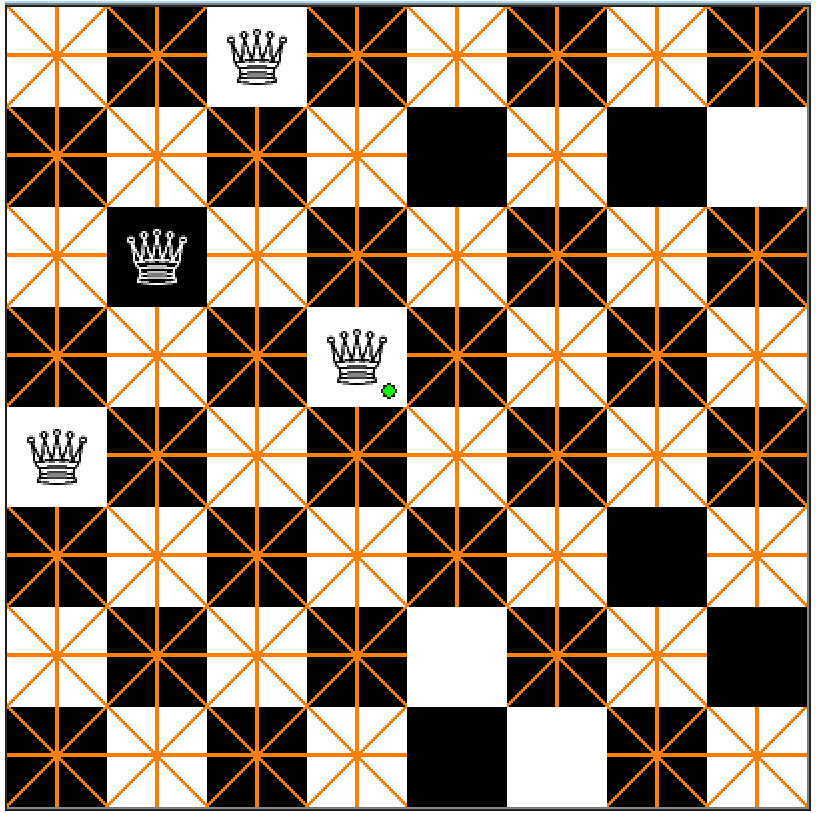
- Then we have placed the second queen on the board.
- The darken place should not have the queens because they are horizontal, vertical, diagonal to the placed queens.

The 8 Queens Problem



- We have placed the third queen on board.

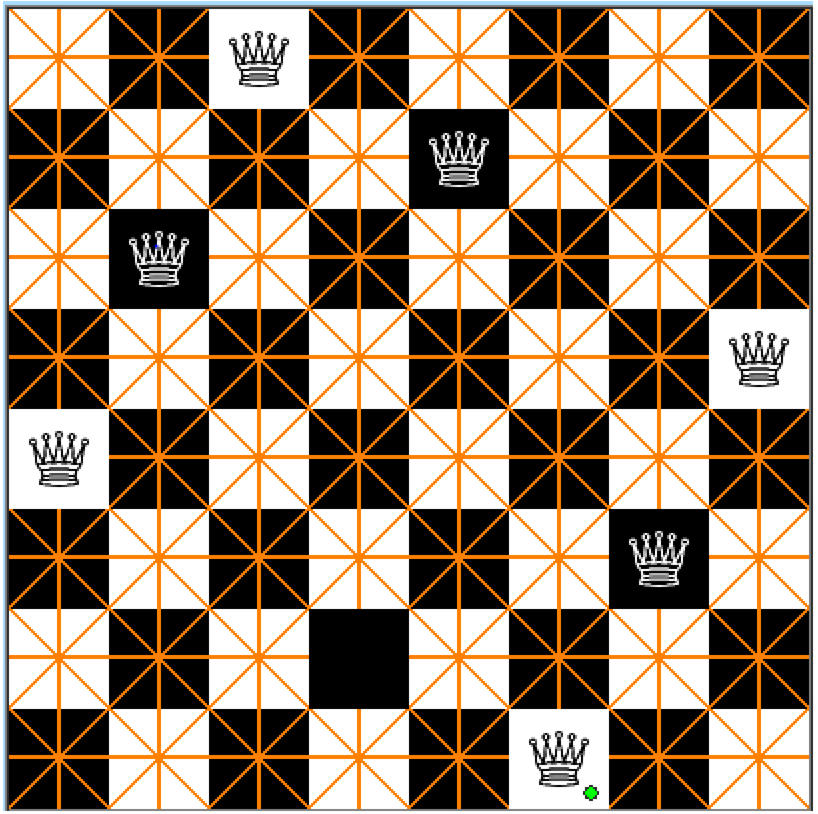
The 8 Queens Problem



- We have placed the 4th queen on the board.
- We have placed that in the wrong spot, so we backtrack and change the place of that one.

The 8 Queens Problem

- In this way, we have to continue the process untill our is reached ie., we must place 8 queens on the board.



The N Queens Problem

- Place N Queens on an N by N chessboard so that none of them can attack each other
- Number of possible placements?
- In 8 x 8

64 * 63 * 62 * 61 * 60 * 59 * 58 * 57 / 8!
 = 178,462, 987, 637, 760 / 8!
 = 4,426,165,368

$$\binom{n}{k} = \frac{n \cdot (n-1) \cdots (n-k+1)}{k \cdot (k-1) \cdots 1} = \frac{n!}{k!(n-k)!} \quad \text{if } 0 \leq k \leq n$$

n choose k

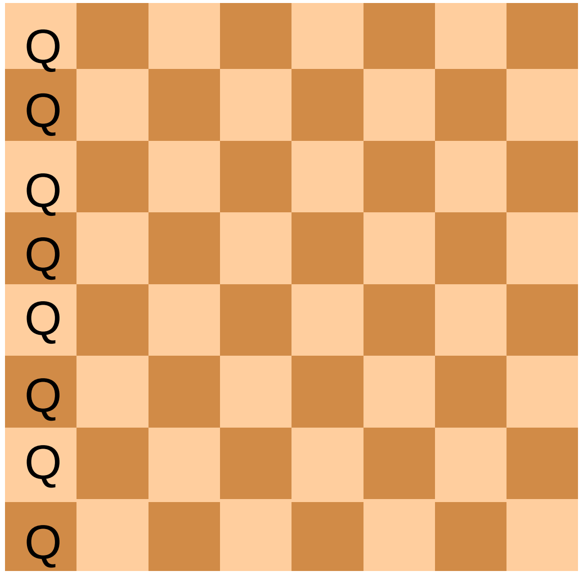
- How many ways can you choose k things from a

set of n items?

- In this case there are 64 squares and we want to choose 8 of them to put queens on

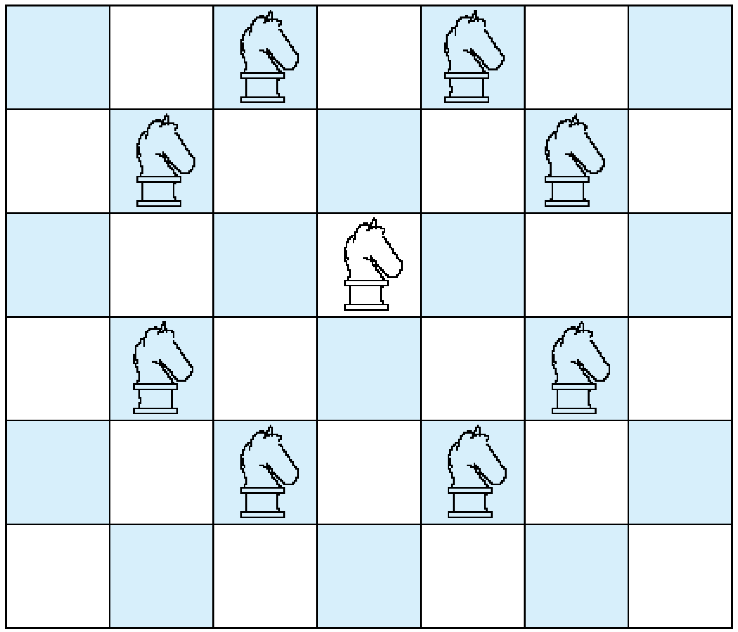
Reducing the Search Space

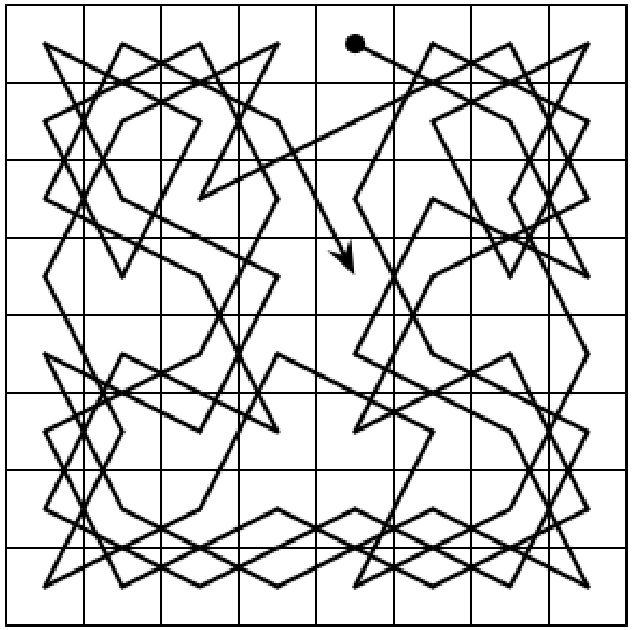
- The previous calculation includes set ups like this one
- Includes lots of set ups with multiple queens in the same column
- How many queens can there be in one column?
- Number of set ups
  8 * 8 * 8 * 8 * 8 * 8 * 8 * 8 = 16,777,216
- We have reduced search space by two orders of magnitude by applying some logic



Knight’s Tour Problem

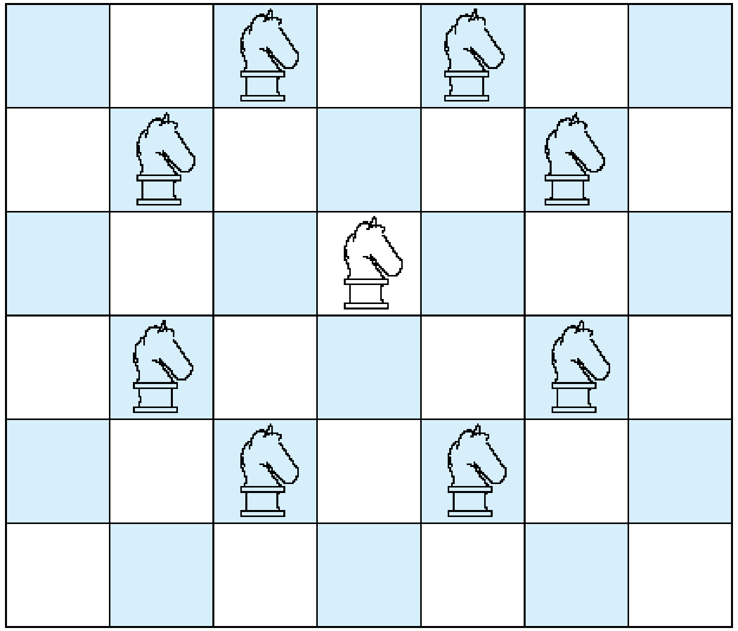
- Problem: find a series of legal moves in which the knight lands on each square of the chessboard exactly once
- Legal moves of a chess knight.

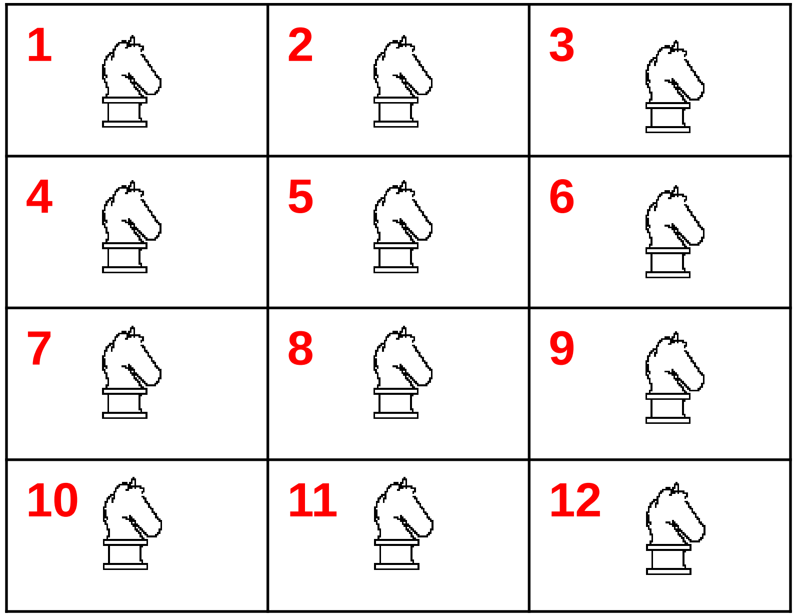




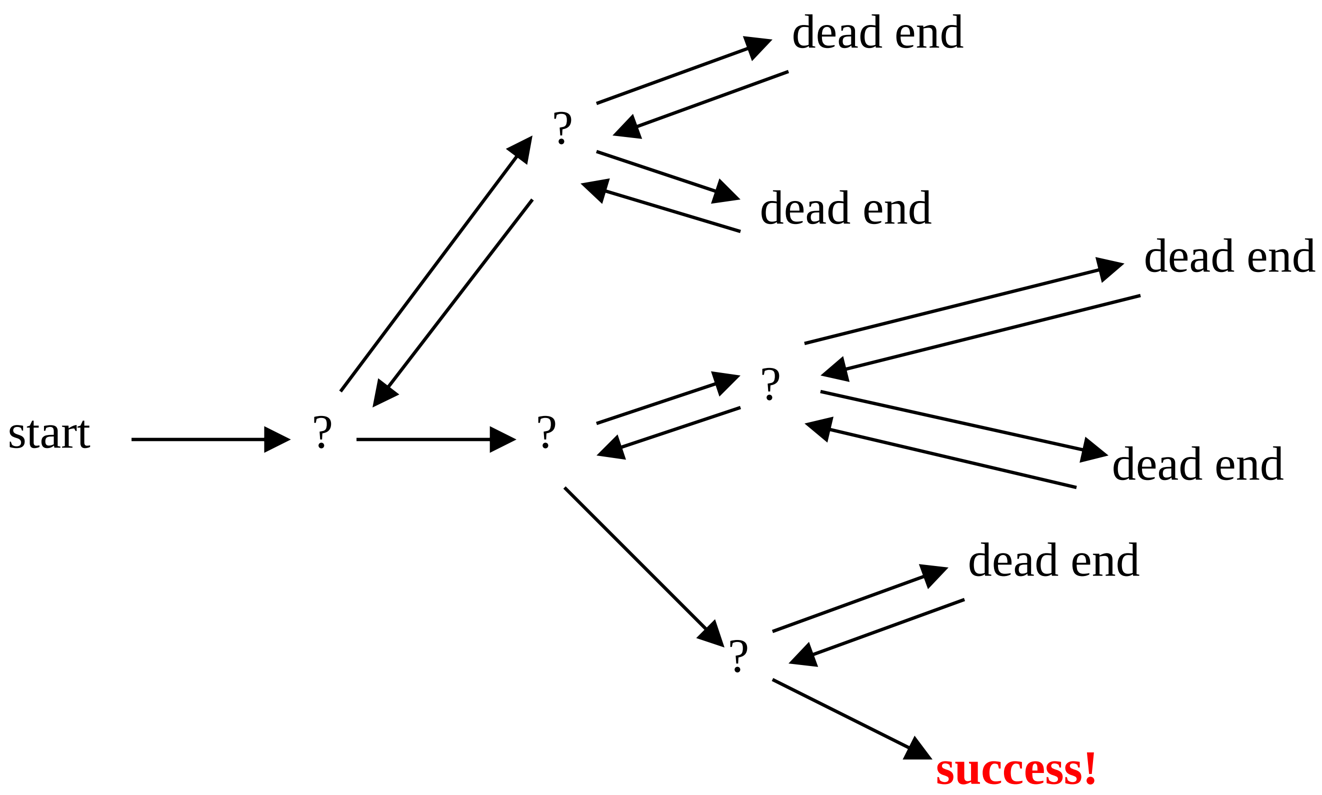
Knight’s Tour Problem

- Problem: find a series of legal moves in which the knight lands on each square of the chessboard exactly once
- Legal moves of a chess knight.

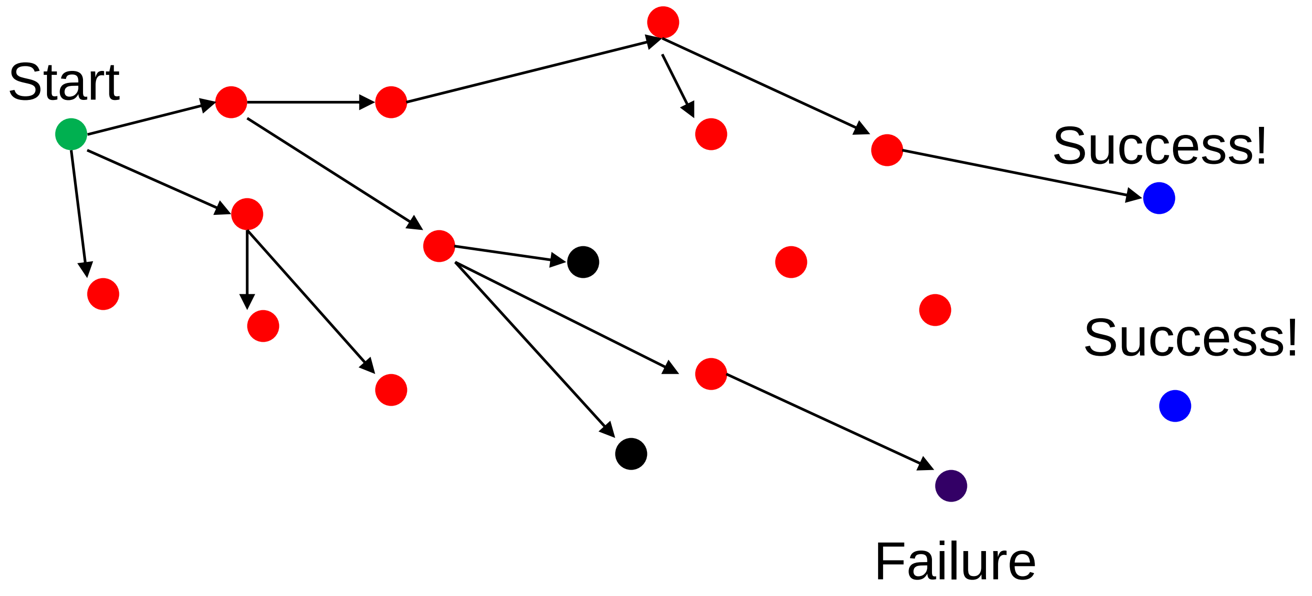




Backtracking



Backtracking

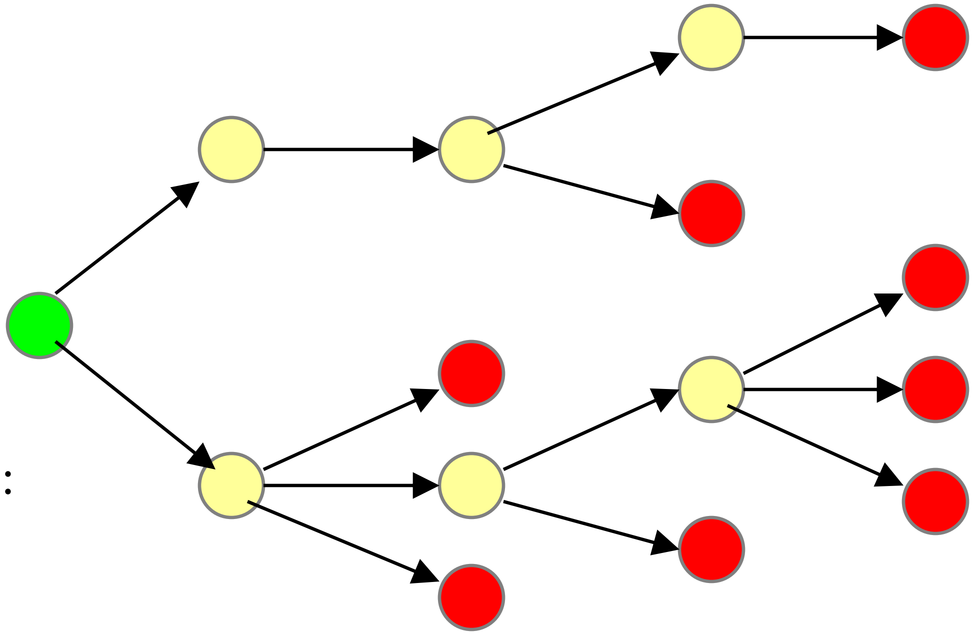


Problem space consists of states (nodes) and actions (paths that lead to new states). When in a node can can only see paths to connected nodes.

If a node only leads to failure go back to its "parent“ node. Try other alternatives. If these all lead to failure then more backtracking may be necessary.

Terminology I

A tree is composed of nodes



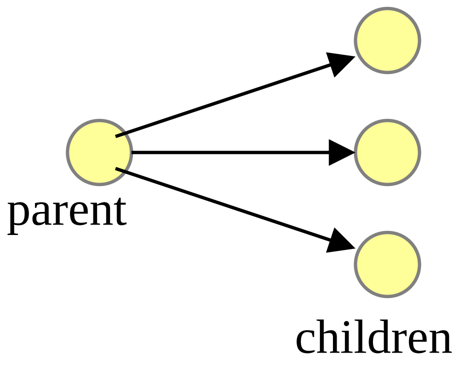
There are three kinds of nodes:

- The (one) root node
- Internal nodes
- Leaf nodes

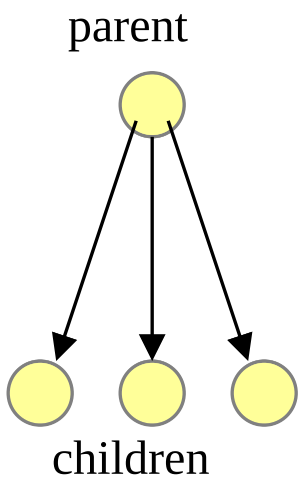
Backtracking can be thought of as searching a tree for a particular "goal" leaf node

Terminology II

- Each non-leaf node in a tree is a parent of one or more other nodes (its children)
- Each node in the tree, other than the root, has exactly one parent



Usually, however, we draw our trees *downward*, with the root at the top



The backtracking algorithm

- Backtracking is really quite simple--we “explore” each node, as follows:
- To “explore” node N:

1. If N is a goal node, return “success”
2. If N is a leaf node, return “failure”
3. For each child C of N,
    3.1. Explore C
        3.1.1. If C was successful, return “success”
4. Return “failure”

A More Concrete Example

- Sudoku
- 9 by 9 matrix with some numbers filled in
- all numbers must be between 1 and 9
- Goal: Each row, each column, and each mini matrix must contain the numbers between 1 and 9 once each
  - no duplicates in rows, columns, or mini matrices

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Solving Sudoku – Brute Force

- A *brute force* algorithm is a simple but general approach
- Try all combinations until you find one that works
- This approach isn't clever, but computers are fast
- Then try and improve on the brute force results

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Solving Sudoku

- Brute force Sudoku Soluton
  - if not open cells, solved
  - scan cells from left to right, top to bottom for first open cell
  - When an open cell is found start cycling through digits 1 to 9.
  - When a digit is placed check that the set up is legal
  - now solve the board

5	3	1		7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Solving Sudoku – Later Steps

5	3	1		7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

→

5	3	1	2	7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

→

5	3	1	2	7	4			
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	1	2	7	4	8		
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	1	2	7	4	8	9	
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

uh oh!

Sudoku – A Dead End

- We have reached a dead end in our search

5	3	1	2	7	4	8	9	
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

- With the current set up none of the nine digits work in the top right corner

Backing Up

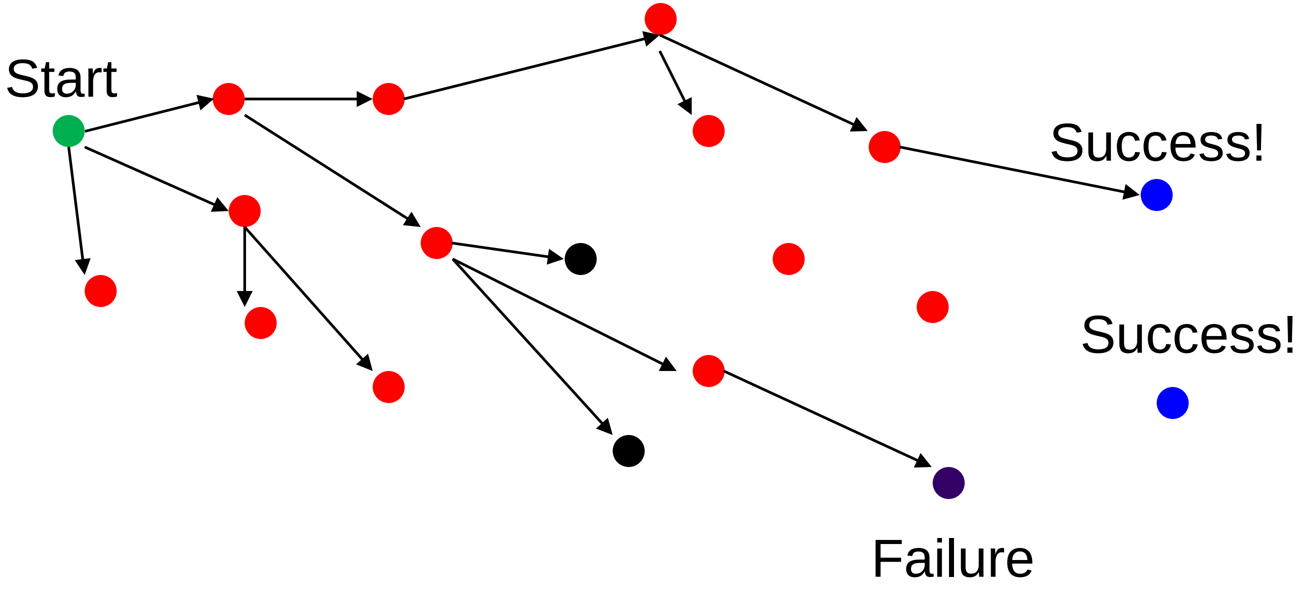
- When the search reaches a dead end in ***backs up*** to the previous cell it was trying to fill and goes onto to the next digit
- We would back up to the cell with a 9 and that turns out to be a dead end as well so we back up again
  - so the algorithm needs to remember what digit to try next
- Now in the cell with the 8. We try and 9 and move forward again.

5	3	1	2	7	4	8	9	
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	1	2	7	4	9		
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Goals of Backtracking

- Possible goals
  - Find a path to success
  - Find all paths to success
  - Find the best path to success
- Not all problems are exactly alike, and finding one success node may not be the end of the search



Acknowledgements

- Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., Introduction to algorithms. MIT press, 2009
- Dr. David Kauchak, Pomona College
- Prof. David Plaisted, The University of North Carolina at Chapel Hill