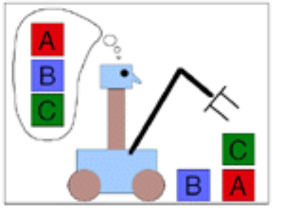
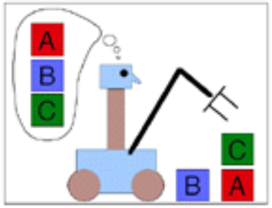
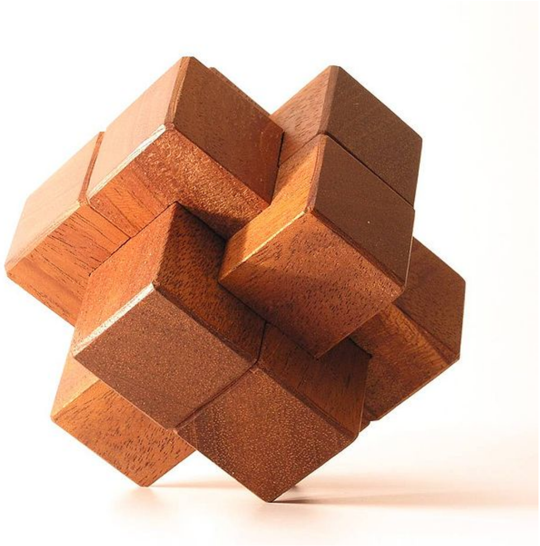


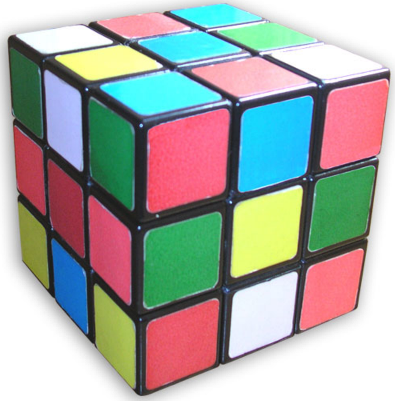
Problem Solving by Searching

Russell and Norvig, chapter 3.1-3.3

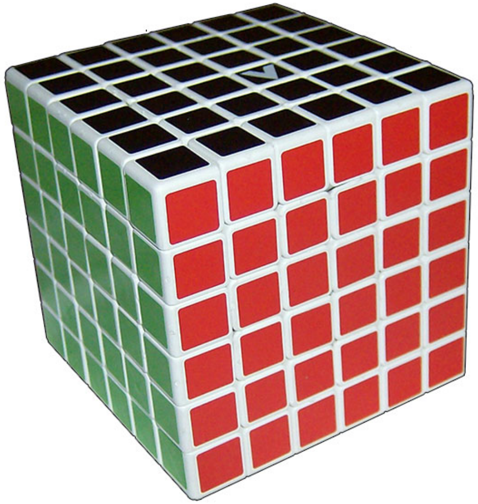
Puzzles!

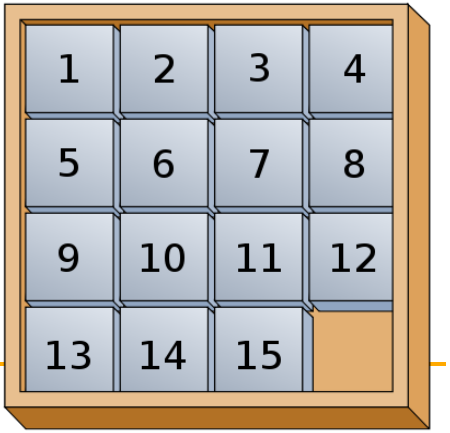






5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

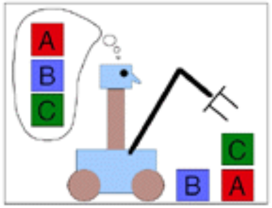




The missionaries and cannibals problem

- Goal: transport the missionaries and cannibals to the right bank of the river.
- Constraints:
  - Whenever cannibals outnumber missionaries, the missionaries get eaten
  - Boat can hold two people and can't travel empty

Formulating the problem



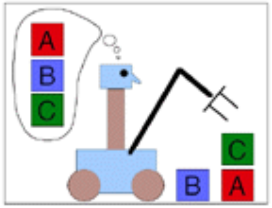
- A state description that allows us to describe our state and goal:

$$(M_L, C_L, B)$$

M_L: number of missionaries on left bank
C_L: number of cannibals on left bank
B: location of boat (L,R)

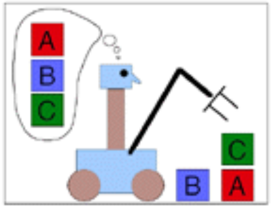
- Initial state: (3,3,L) Goal: (0,0,R)

Problem solving agents



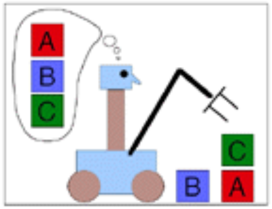
- Problem Formulation
  - States and actions (successor function).
- Goal Formulation
  - Desired state of the world.
- Search
  - Determine a sequence of actions that lead to a goal state.
- Execute
  - Perform the actions.
- Assumptions:
  - Environment is fully observable and deterministic
  - Agent knows the effects of its actions

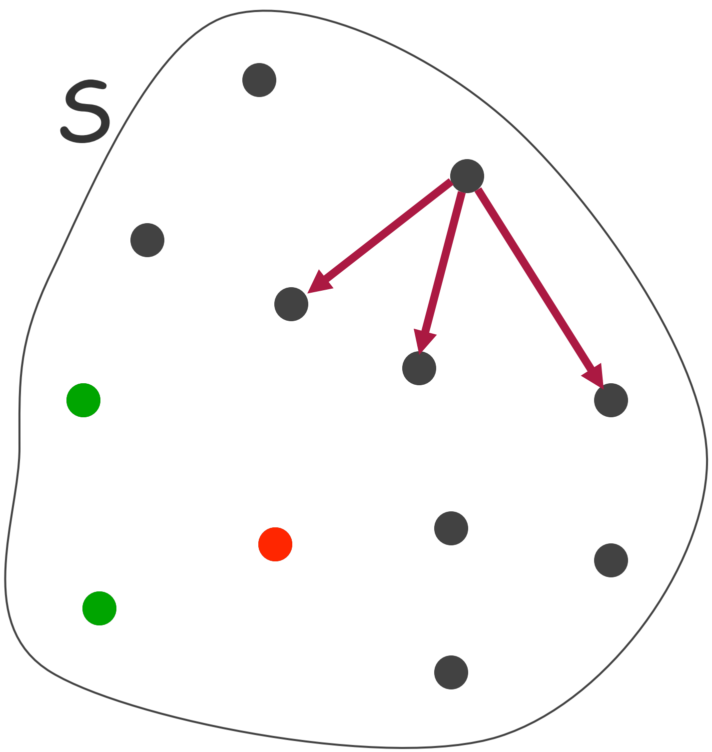
Graph formulation of the problem



- Nodes: all possible states.
- Edges: edge **from** state *u* to state *v* if *v* is reachable from *u* (by an action of the agent)
- Edges for missionaries and cannibals problem?
- Problem is now to find a path from (3,3,L) to (0,0,R).
- In general, paths will have costs associated with them, so the problem will be to find the lowest cost path from initial state to the goal.

Formulating a search problem

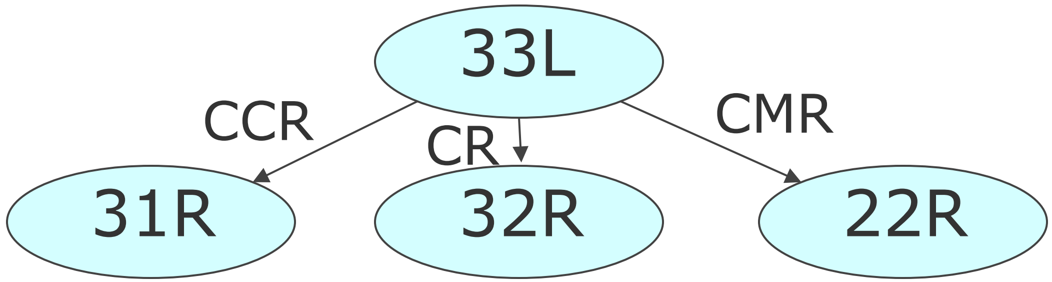




- State space S (nodes)
- Successor function: the states you can move to by an action (edge) from the current state
- Initial state
- Goal test

  is a state x a goal?
- Cost

Back to our problem



State: 33L three missionaries and three cannibals on left bank, boat is on left bank

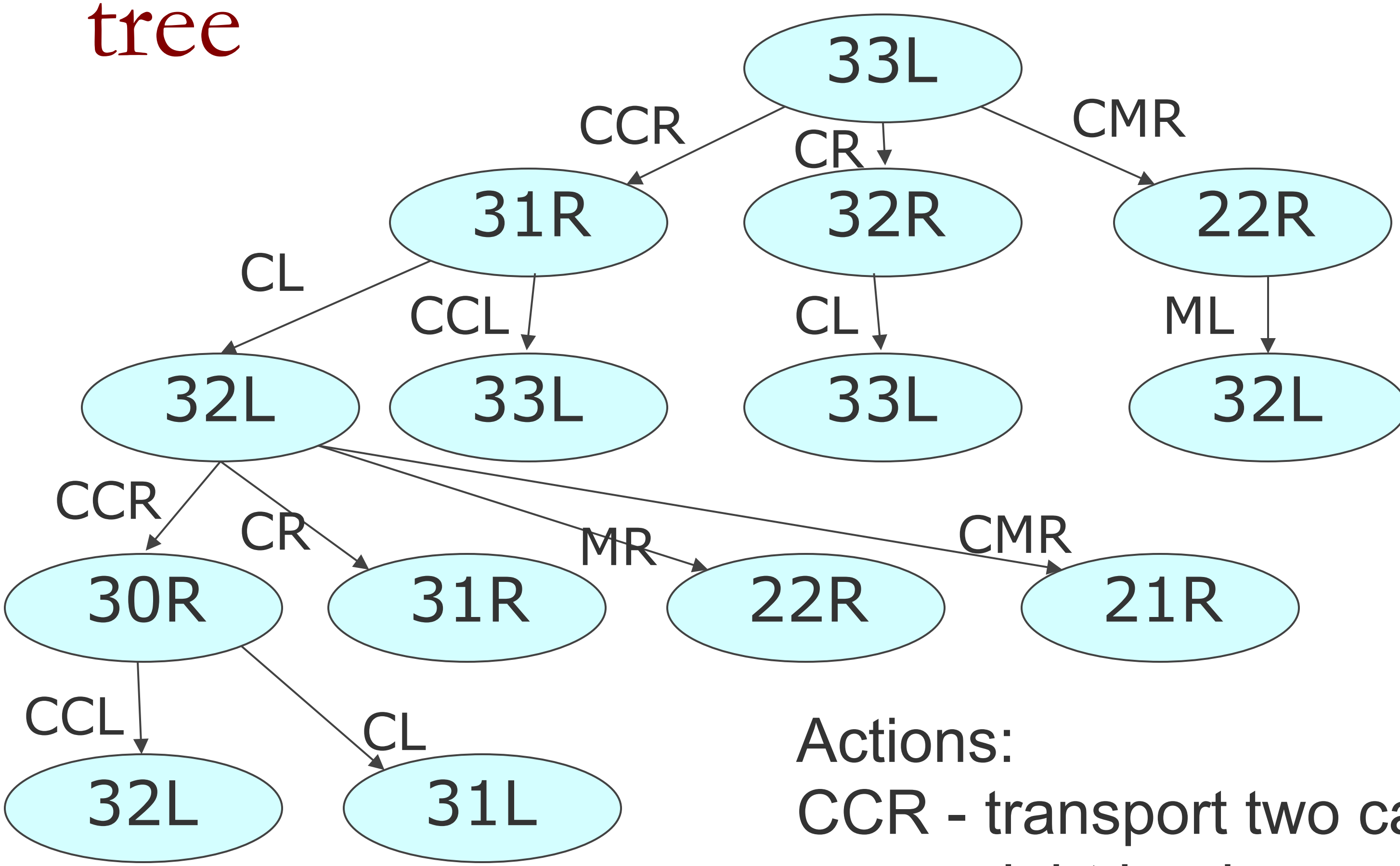
Actions (operators):

CCR - transport two cannibals to the right bank

CMR - transport a missionary and a cannibal to the right bank

Why no MR or MMR transition from this state?

The (partially expanded) search tree

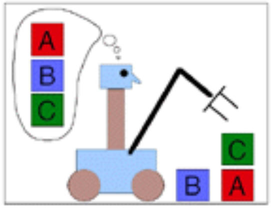


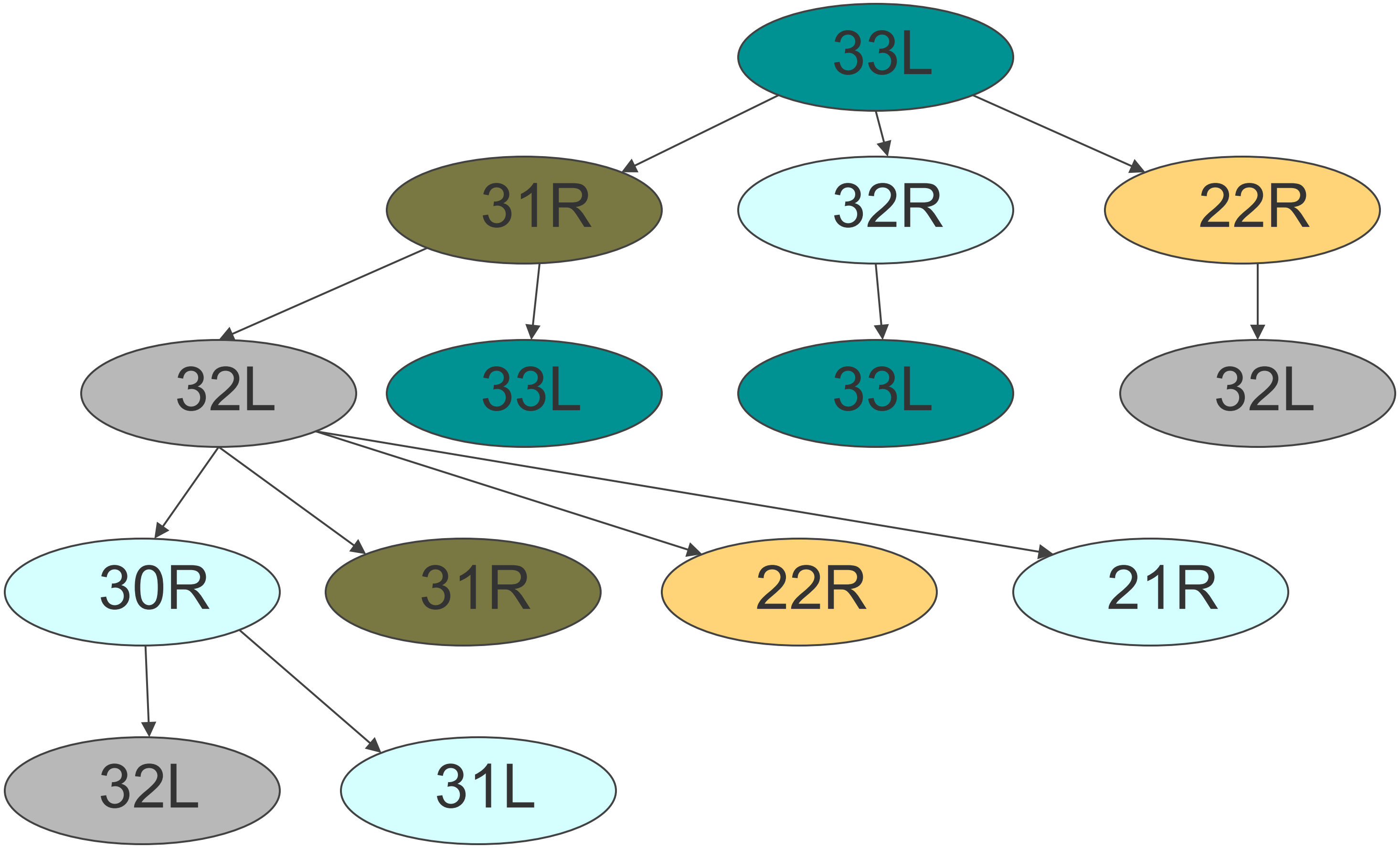
Actions:

CCR - transport two cannibals to the right bank

CML - transport a missionary and a cannibal to the left bank

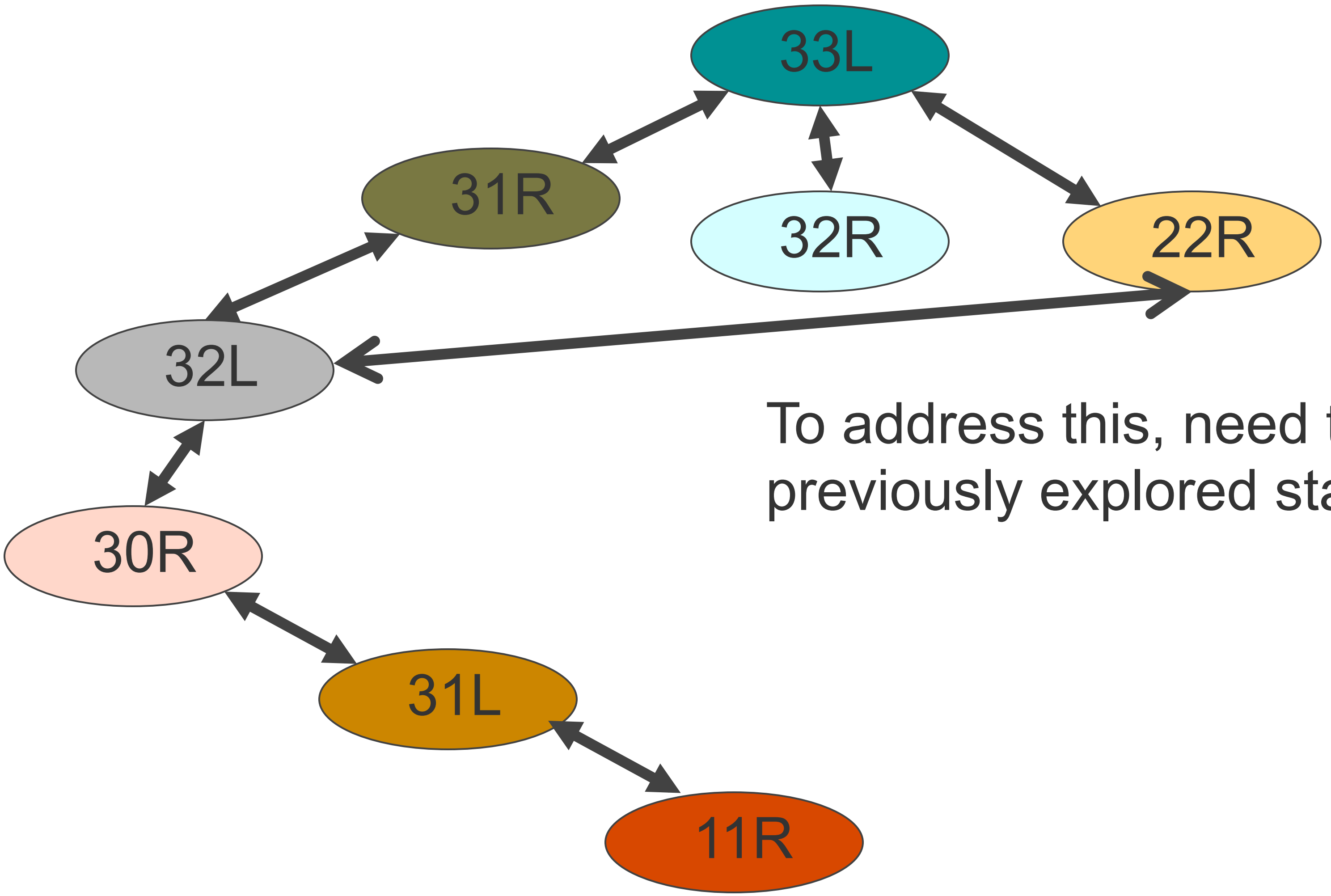
Repeated states





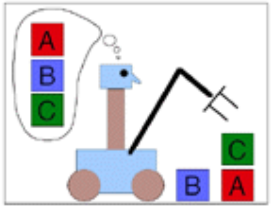
The search tree contains duplicate nodes, since the underlying graph in which we are searching is not a tree!

The state space graph



To address this, need to track previously explored states

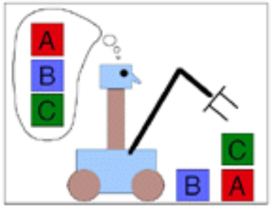
Aside: Searching the graph

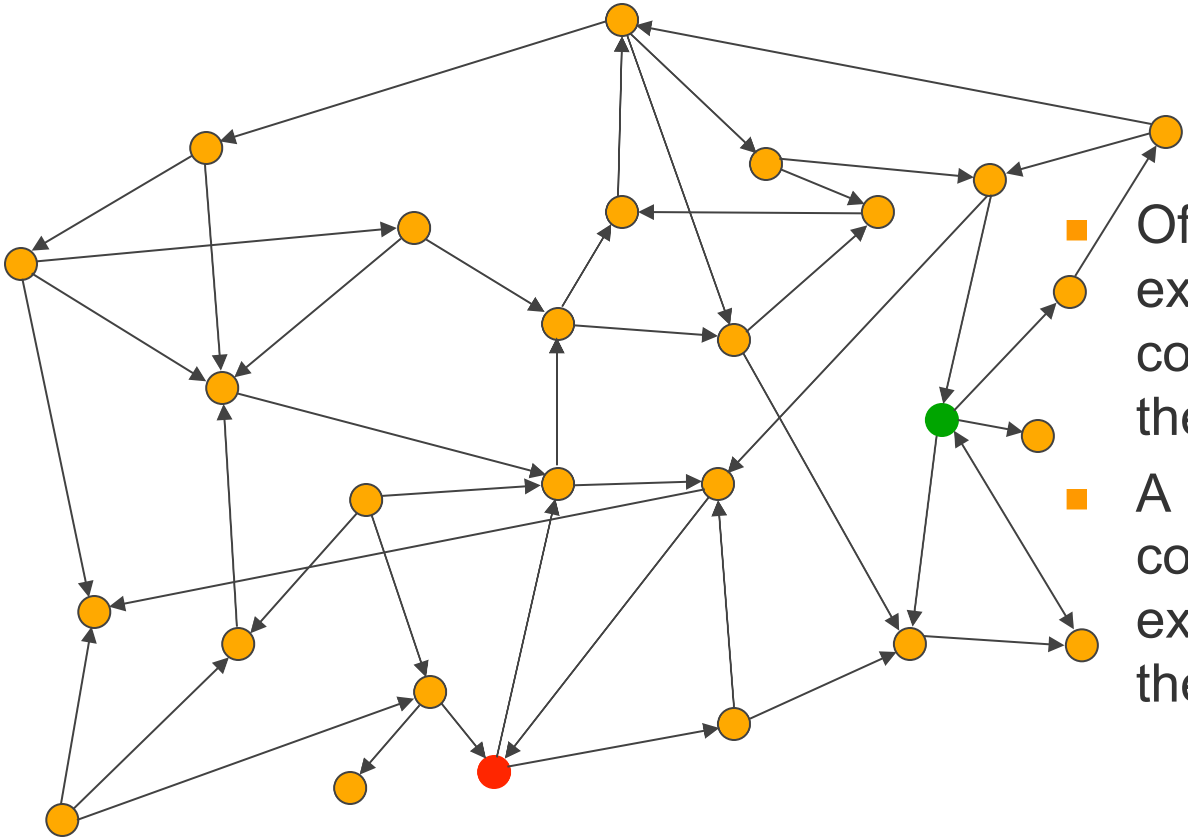


function GRAPH-SEARCH(*problem*) returns a solution, or failure
initialize the *frontier* using the initial state of problem
initialize the *explored* set to be empty
loop do
if the *frontier* is empty **then return** failure
choose a leaf *node* and remove it from the frontier
if the *node* contains a goal state **then return** the corresponding solution
add the *node* to the *explored* set
expand the chosen *node*
add the resulting *nodes* to the *frontier* only if not in the *frontier* or *explored*

The difference between BFS and DFS is in how the node is chosen and/or how it is added to the frontier: queue versus stack.

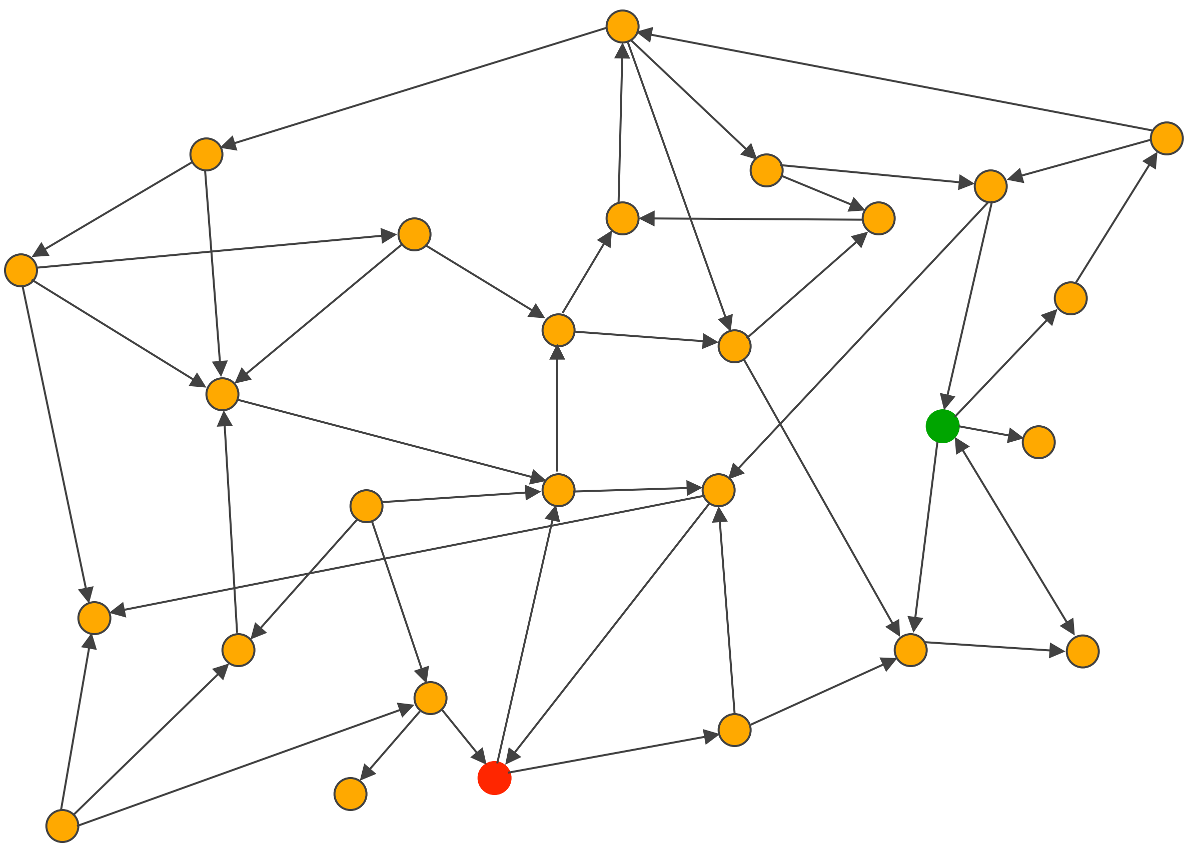
Searching the state space

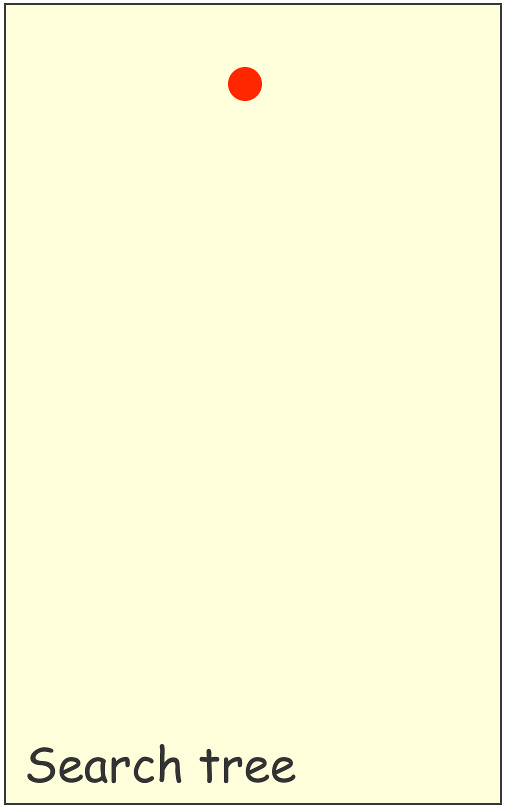




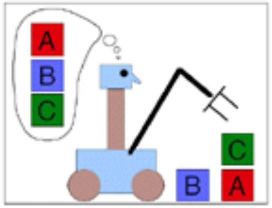
- Often it is not feasible (or too expensive) to build a complete representation of the state graph
- A problem solver must construct a solution by exploring a small portion of the graph

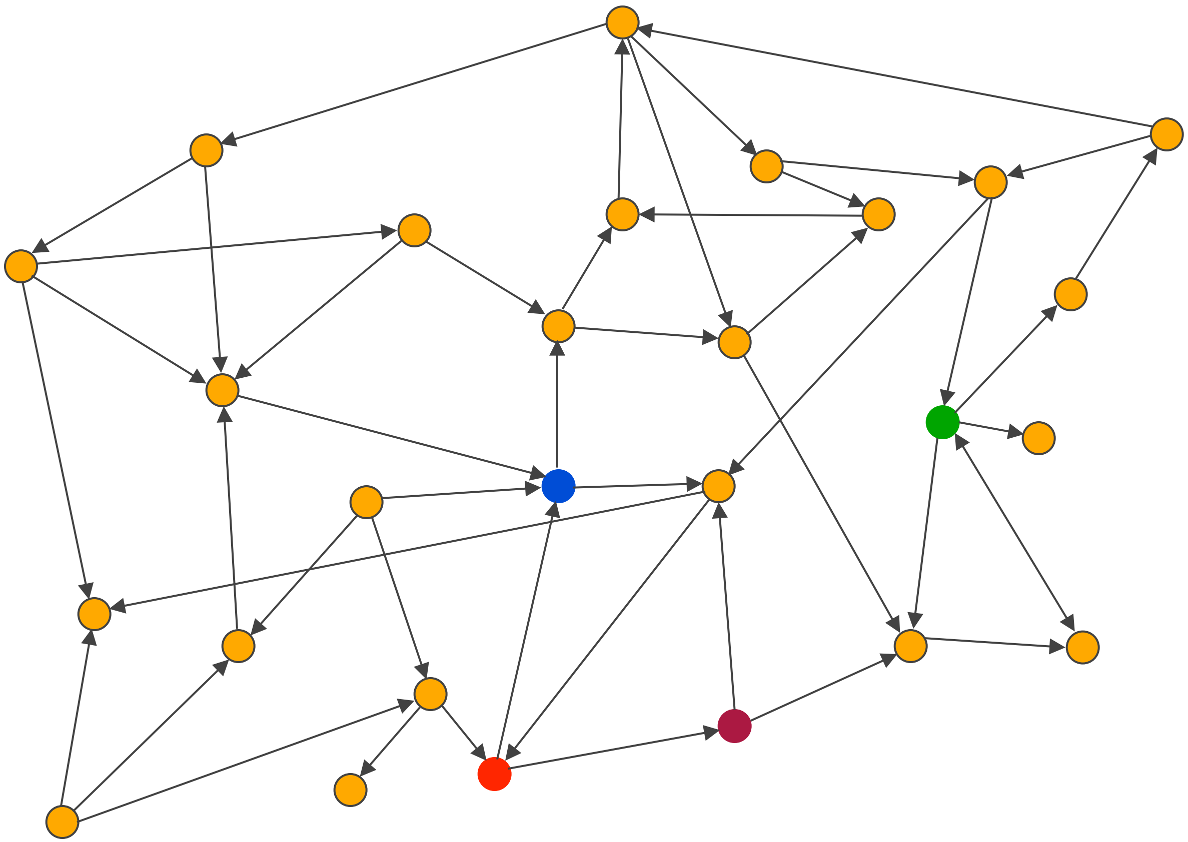
Searching the state space

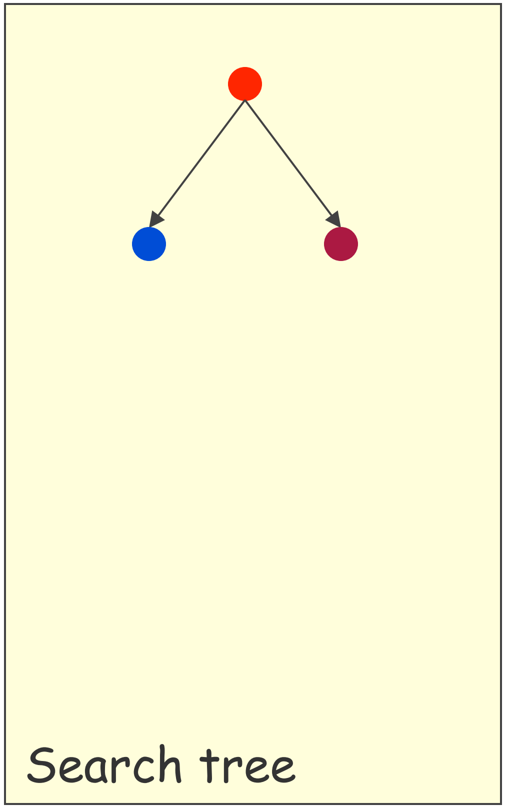




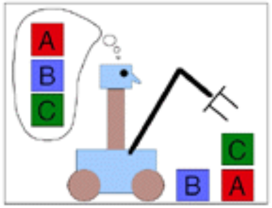
Searching the state space

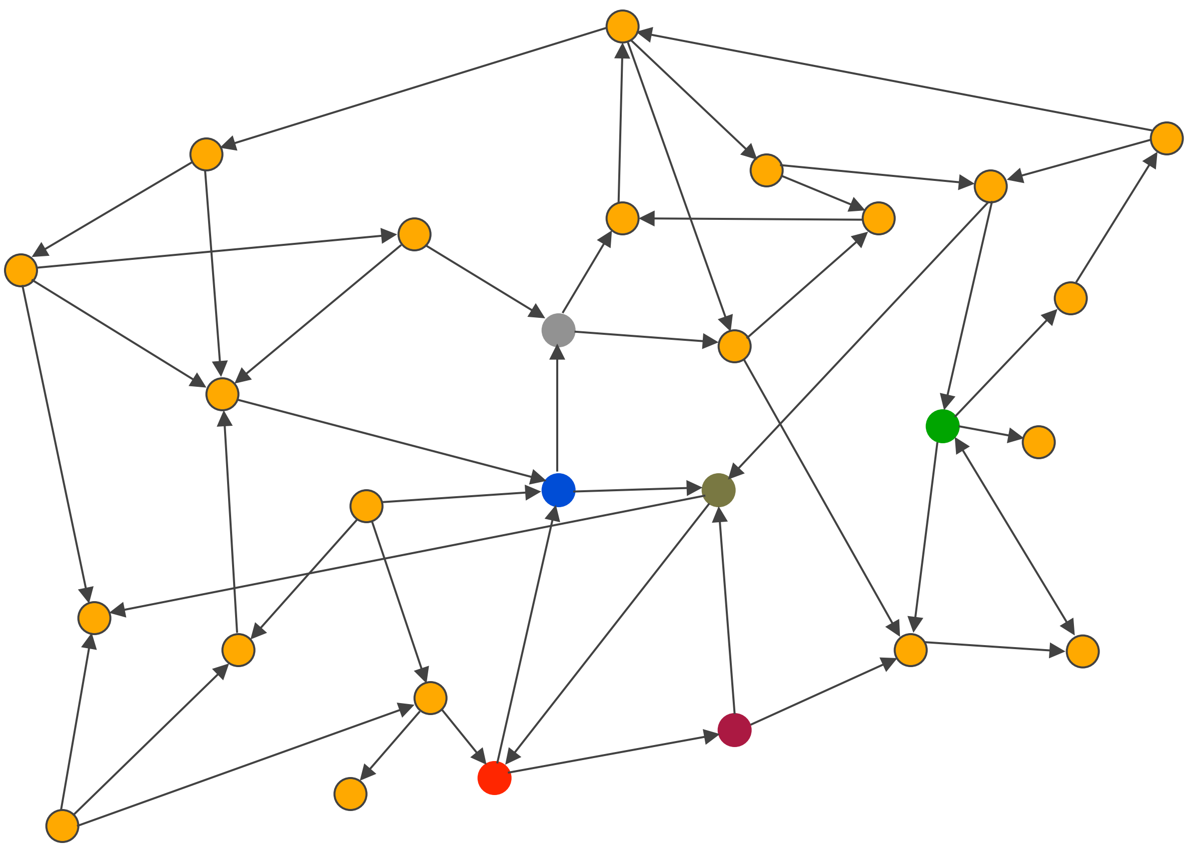


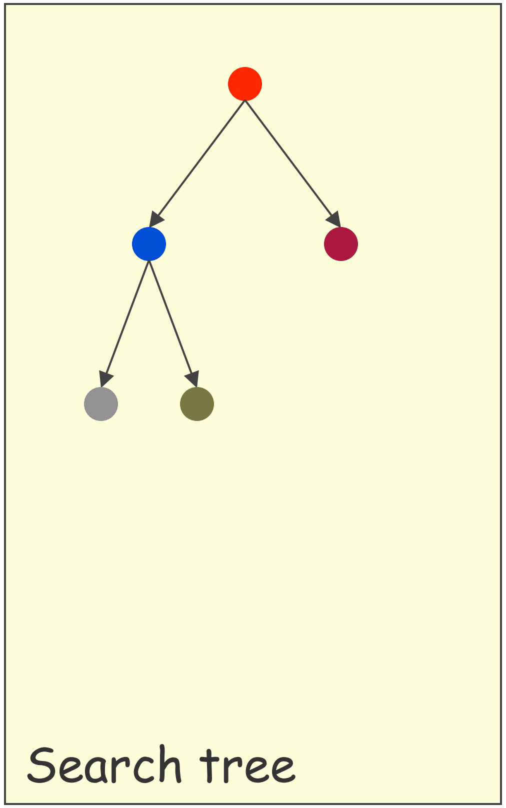




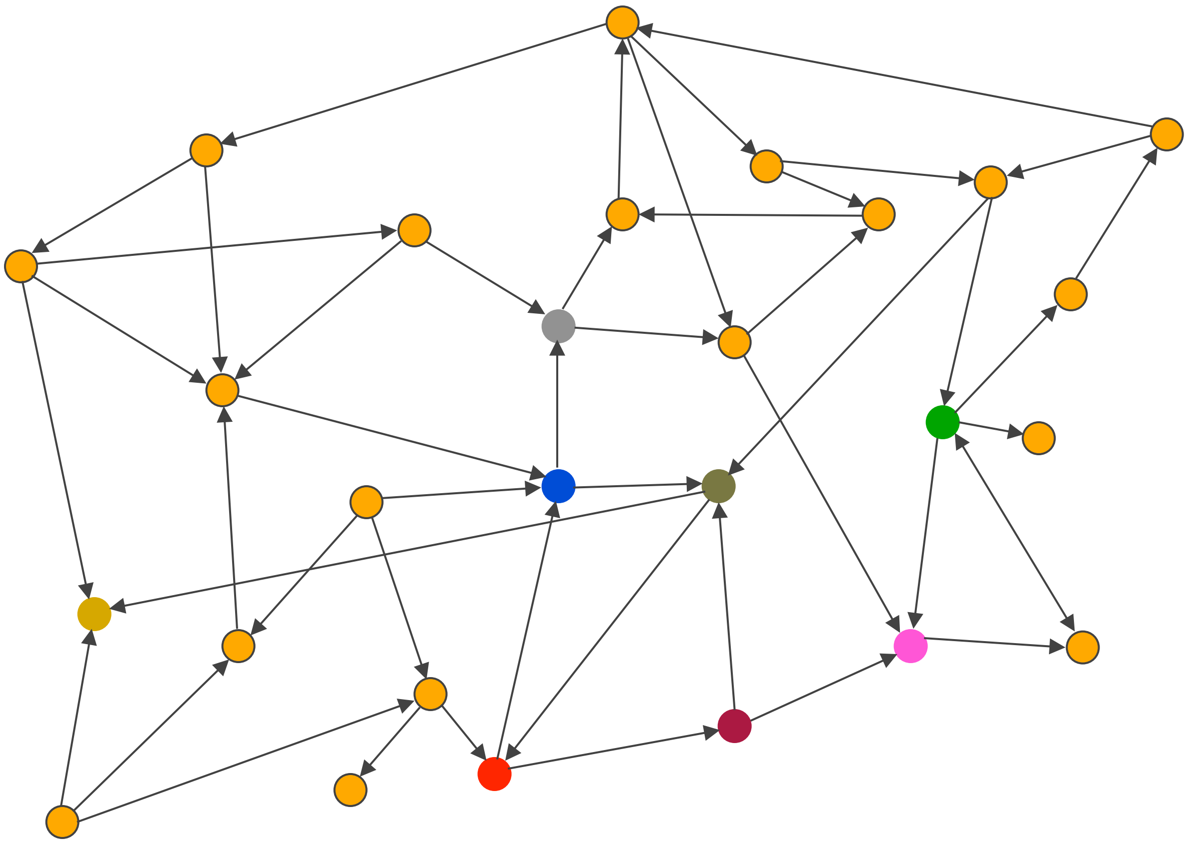
Searching the state space

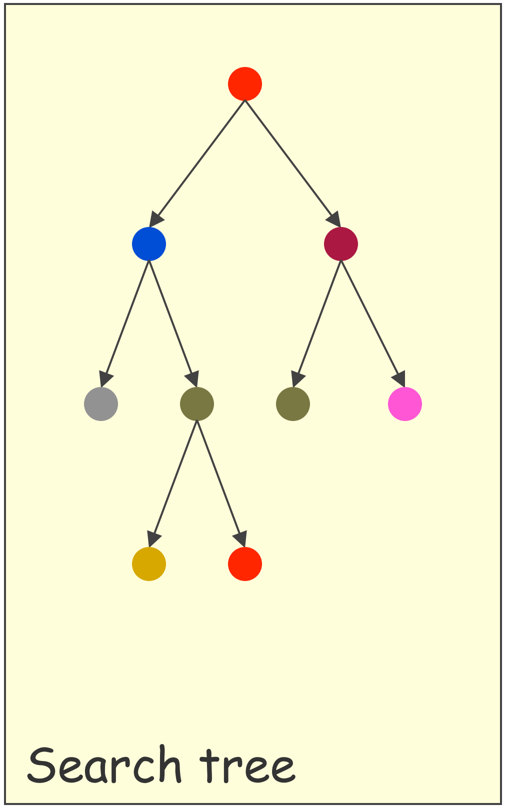




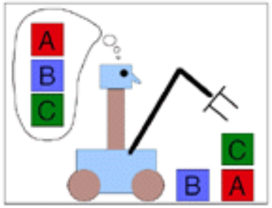


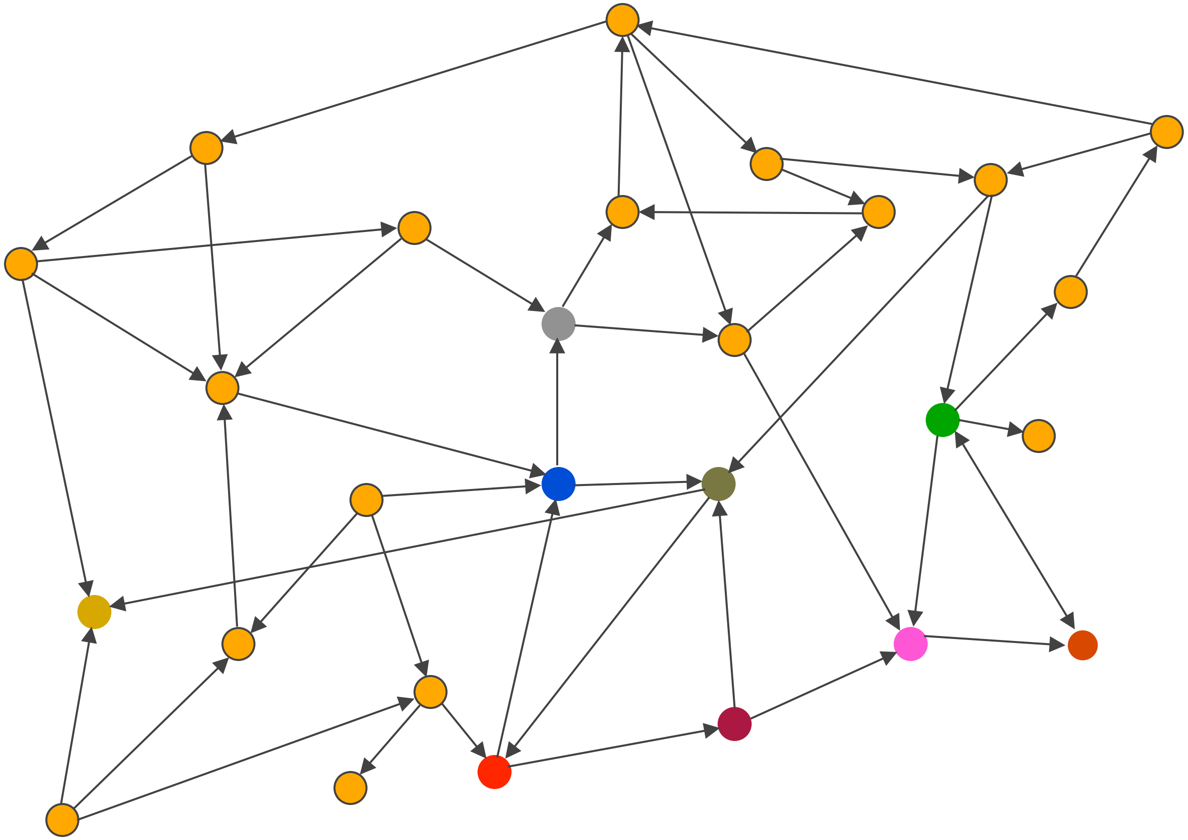
Searching the state space

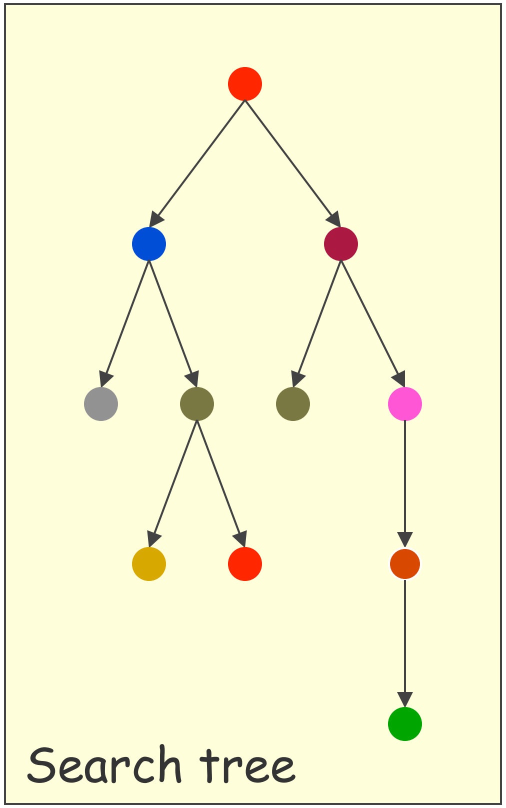




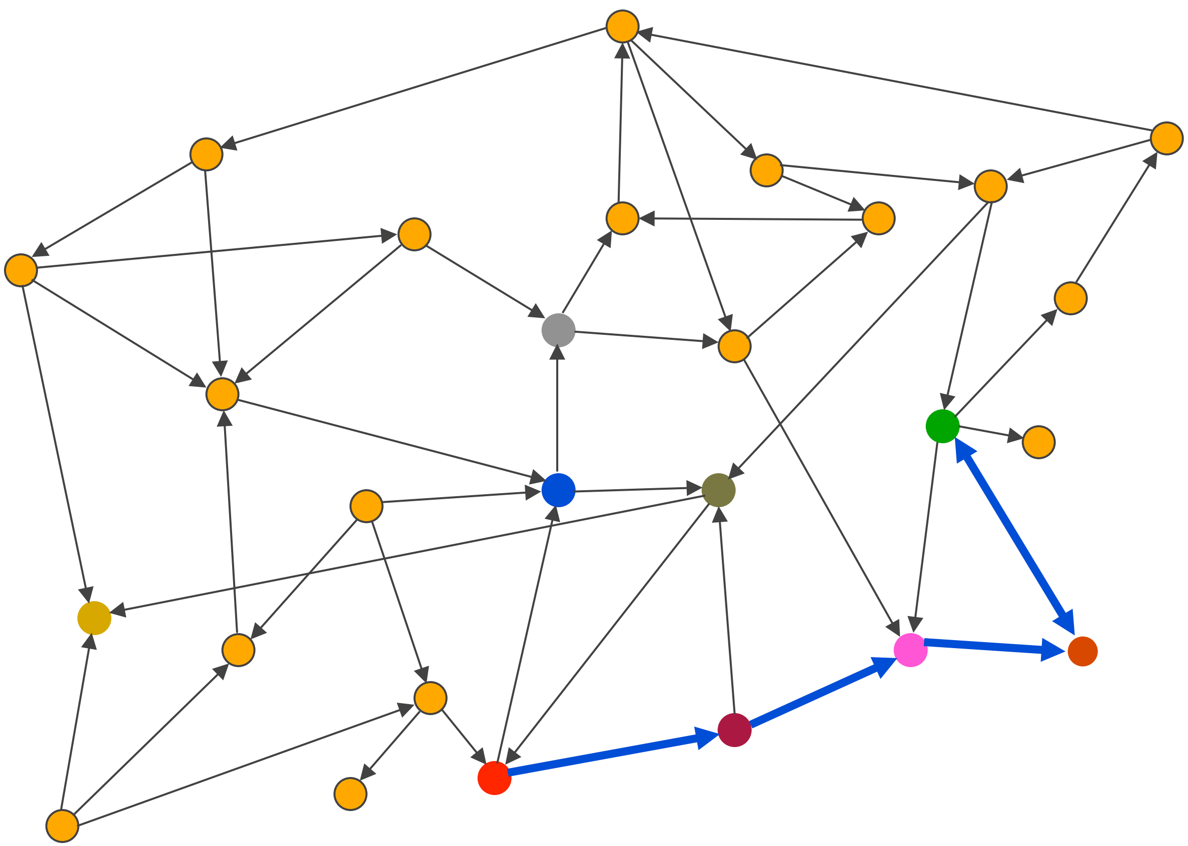
Searching the state space

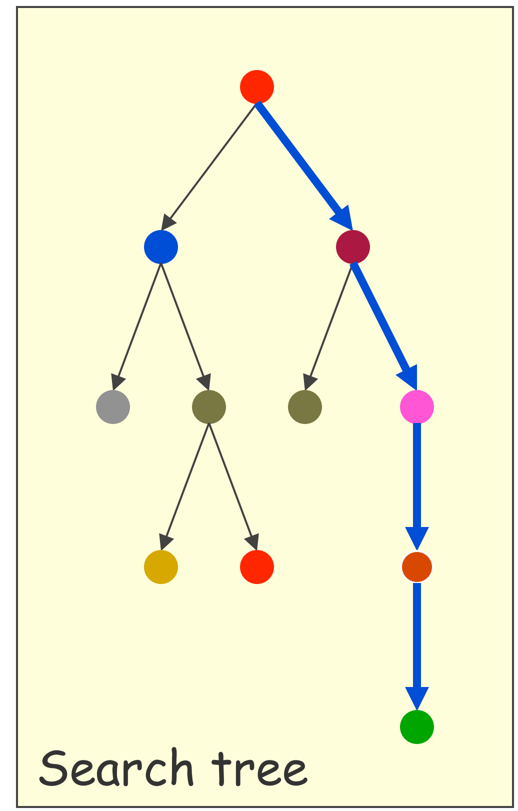




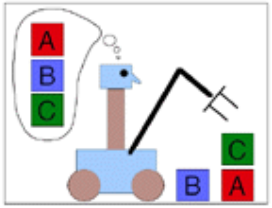


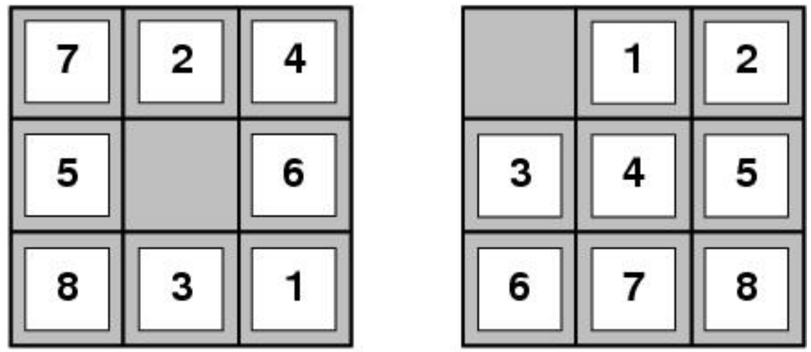
Searching the state space





The 8 puzzle

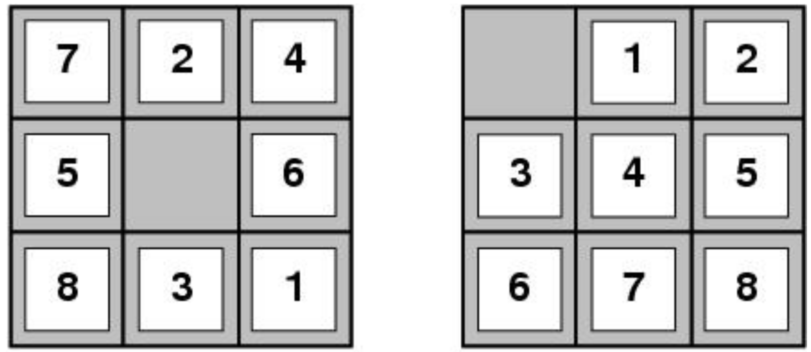




Start State Goal State

- States?
- Initial state?
- Actions?
- Goal test?
- Path cost?

The 8 puzzle



Start State Goal State

- States? Integer location of each tile. How many of them are there?
- Initial state? Any state
- Actions? (tile, direction)

  where direction is one of {*Left*, *Right*, *Up*, *Down*}
- Goal test? Check whether goal configuration is reached
- Path cost? Number of actions to reach goal
- Is the search graph a tree?

(n^2-1)-puzzle

8	2	
3	4	7
5	1	6

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

....

The 15-puzzle

Sam Loyd offered $1,000 of his own money to the first person who would solve the following problem:

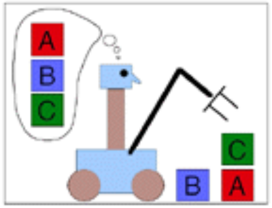
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

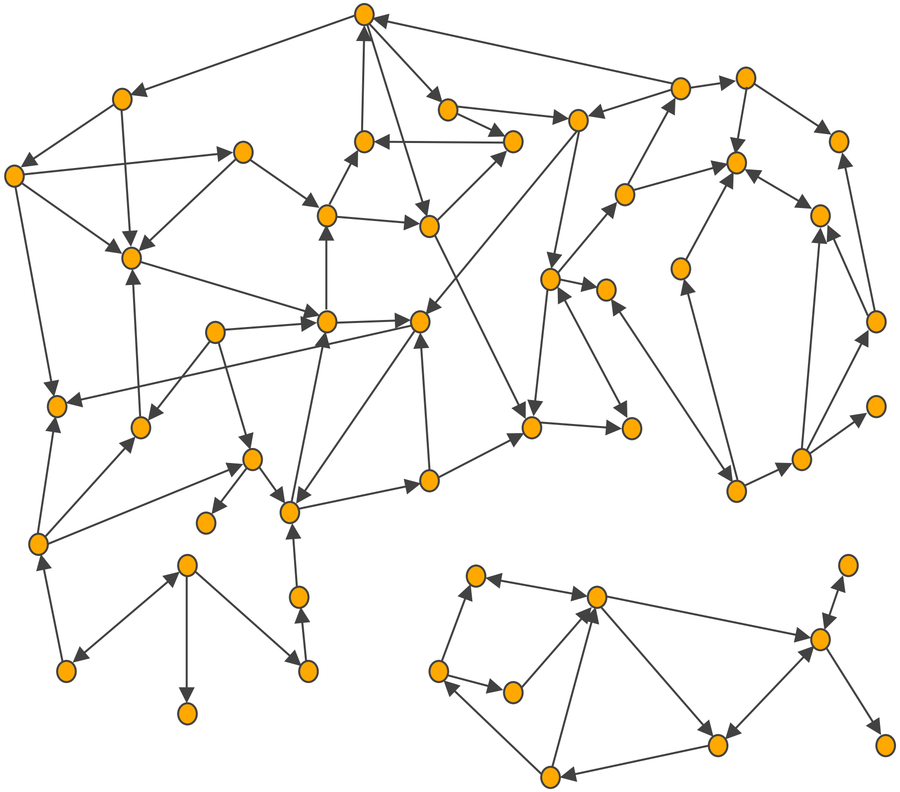
? →

1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	

But no one ever won the prize !!

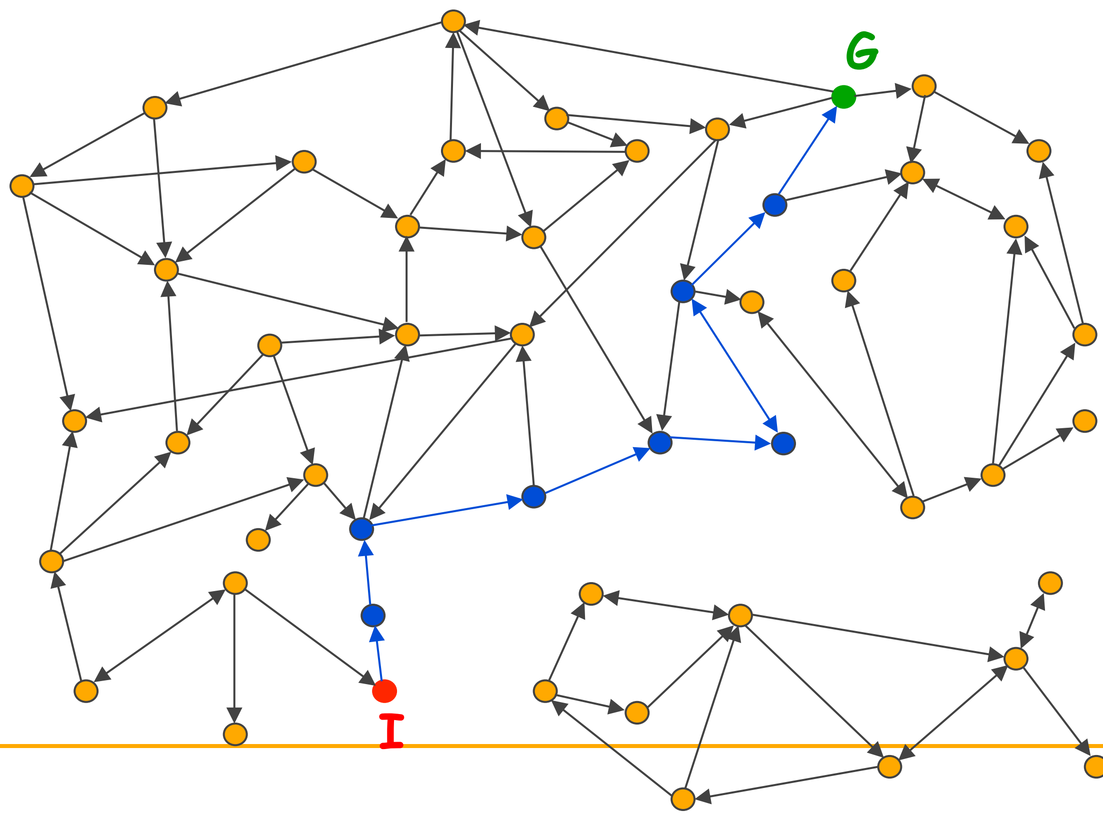
Why?



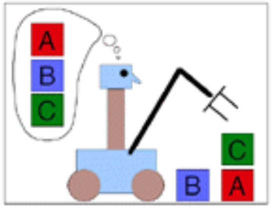


Solution to the Search Problem

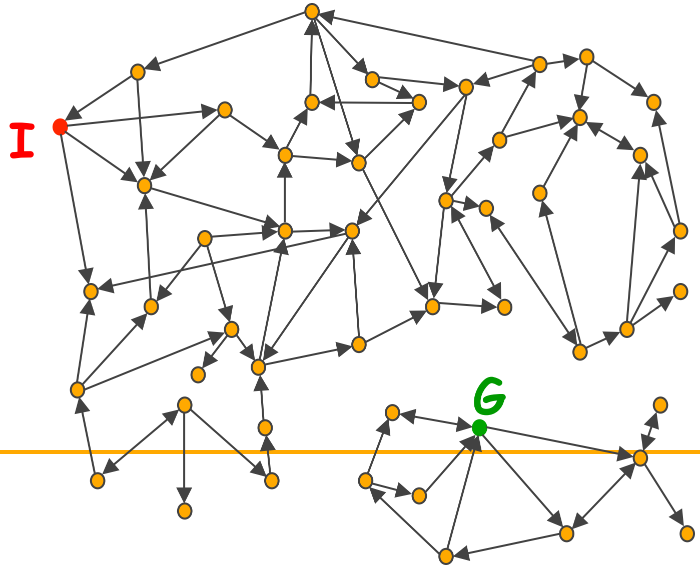
- A solution is a path connecting the initial node to a goal node (any one)



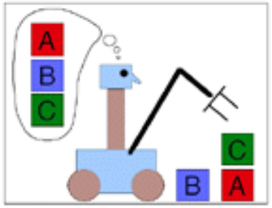
Solution to the Search Problem



- A solution is a path connecting the initial node to a goal node (any one)
- The cost of a path is the sum of the edge costs along this path
- An optimal solution is a solution path of minimum cost
- There might be no solution !



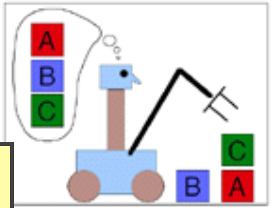
Path Cost



- An edge cost is a positive number measuring the “cost” of performing the action corresponding to the edge, e.g.:
  - 1 in the 8-puzzle example
- We will assume that for any given problem the cost c of an edge always satisfies:

  $c \geq \varepsilon > 0$, where ε is a constant
  - Why? Has to do with the cost of arbitrarily long paths

Goal State



- It may be explicitly described:

1	2	3
4	5	6
7	8	

- or partially described:

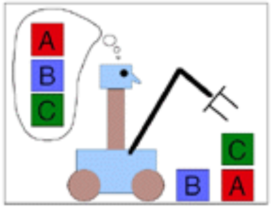
1	a	a
a	5	a
a	8	a

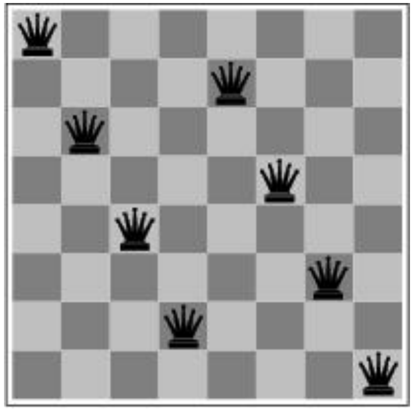
("a" stands for "any" other than 1, 5, and 8)

- or defined by a condition, e.g., the sum of every row, of every column, and of every diagonal equals 30

15	1	2	12
4	10	9	7
8	6	5	11
3	13	14	

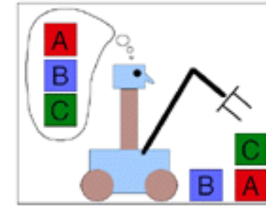
Another example: the 8 queens problem

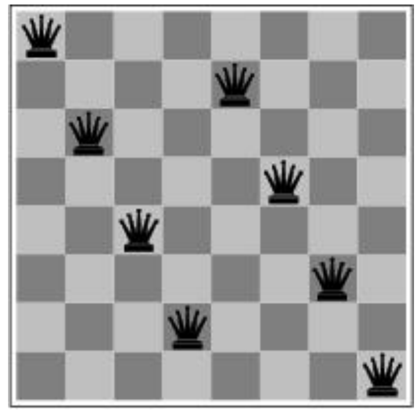




- Incremental vs. complete state formulation:
  - Incremental formulation starts with an empty state and involves operators that augment the state description
  - A complete state formulation starts with all 8 queens on the board and moves them around

8 queens problem: representation is key





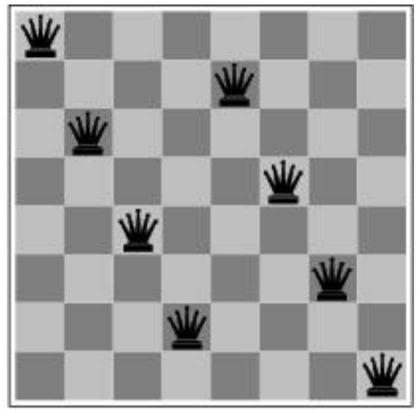
Incremental formulation

- States? Any arrangement of 0 to 8 queens on the board
- Initial state? No queens
- Actions? Add queen in empty square
- Goal test? 8 queens on board and none attacked
- Path cost? None

$64 \times 63 \times \ldots 57 \sim 3 \times 10^{14}$ states to investigate

Is the search graph a tree?

A better representation

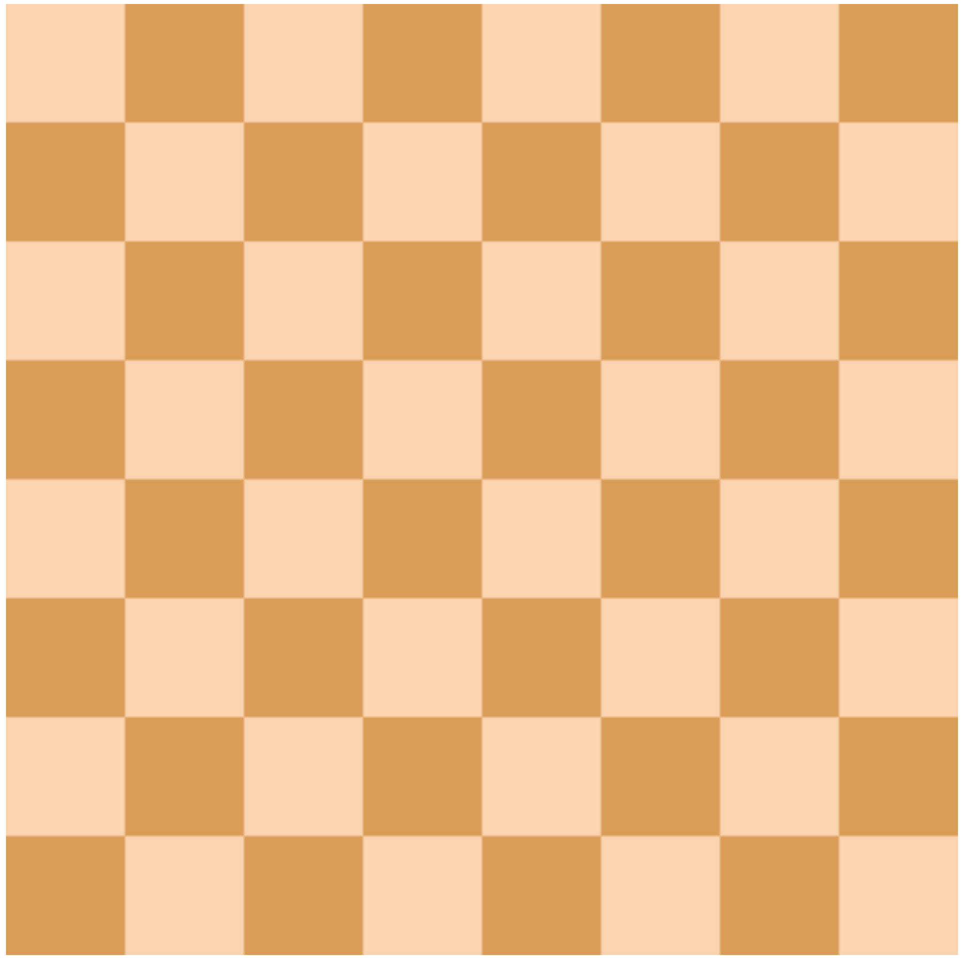


Another incremental formulation:

- States? n (between 0 and 8) queens on the board, one in each of the n left-most columns; no queens attacking each other.
- Initial state? No queens
- Actions? Add queen in leftmost empty column such that it does not attack any of the queens already on the board.
- Goal test? 8 queens on board

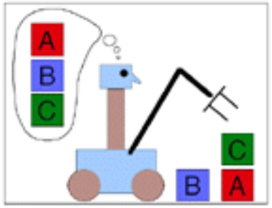
2057 possible sequences to investigate

8 queens solved



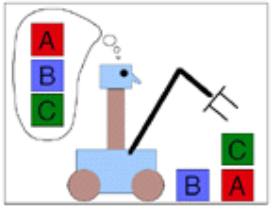
Source:
https://sites.google.com/a/lclark.edu/drake/courses/ai/search-and-n-queens

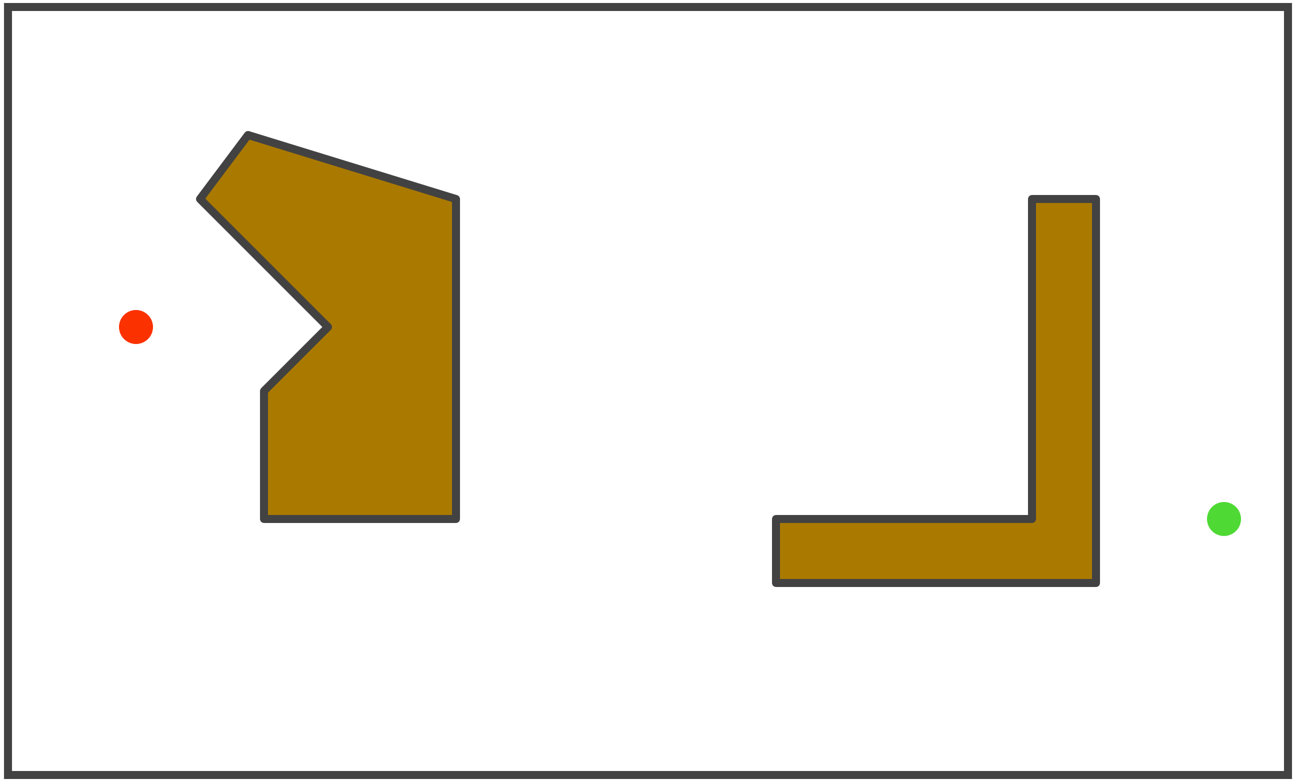
n-queens problem



- A solution is a goal node, not a path to this node (typical of design problem)
- Number of states in state space:
  - 8-queens → 2,057
  - 100-queens → 10^{52}
- But techniques exist to solve n-queens problems efficiently for large values of n

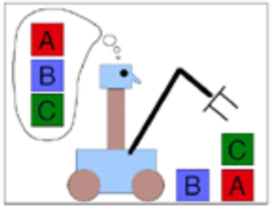
Path planning

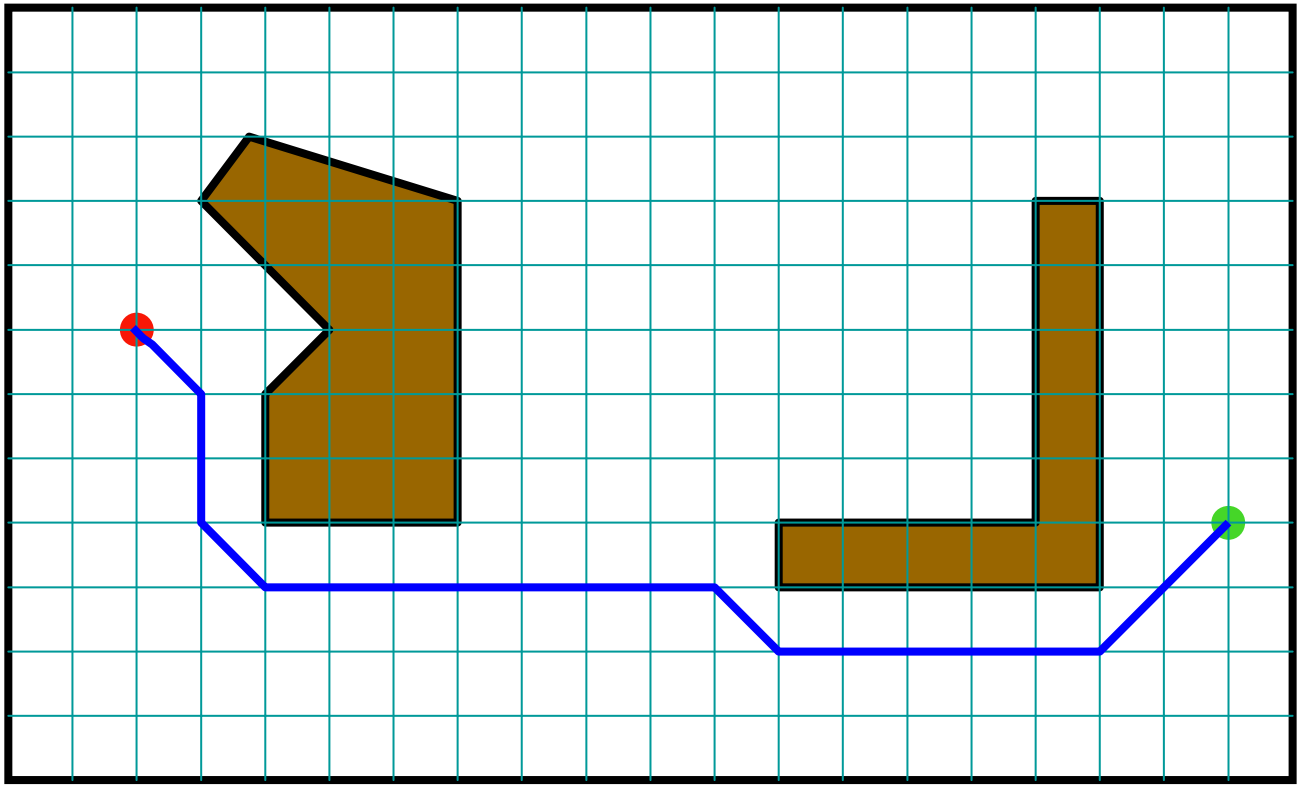




What is the state space?

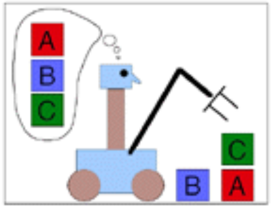
Path planning

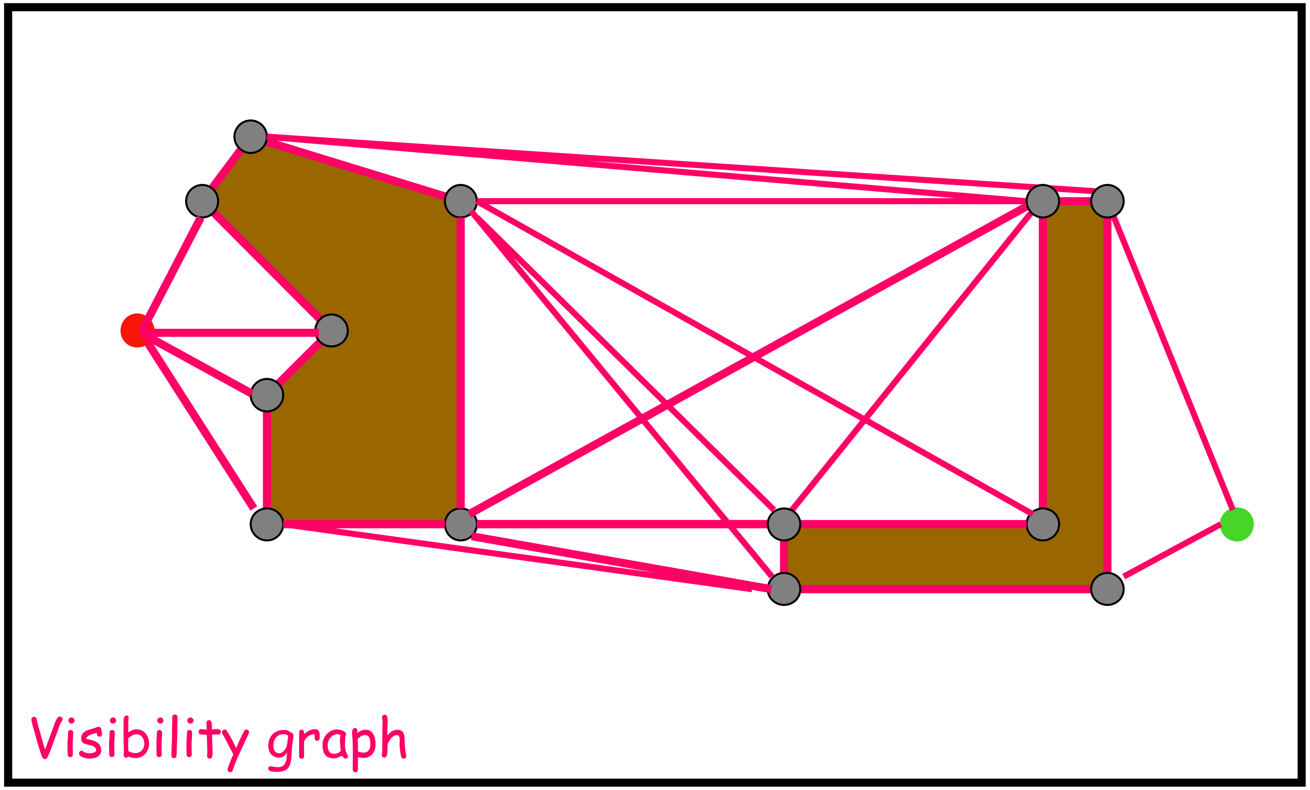




This path is the shortest one in the discretized space, but not in the original space.

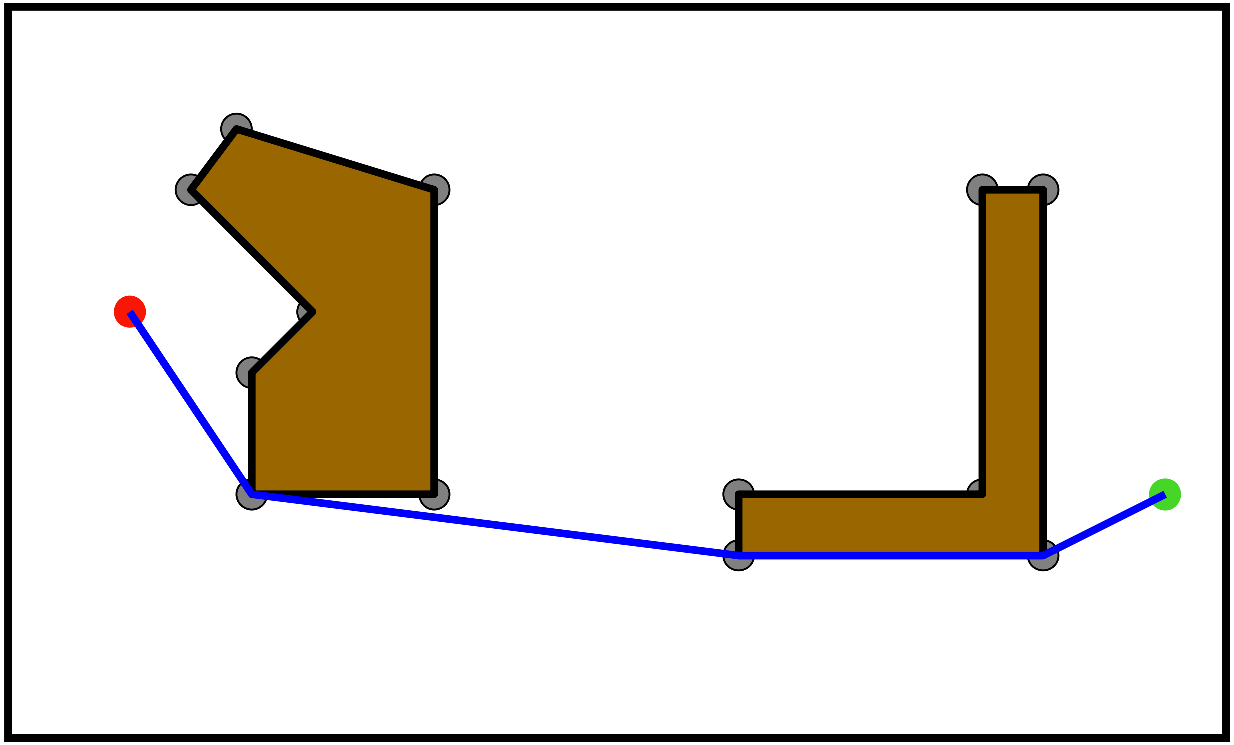
An alternative formulation





Cost of a step is the length of the step

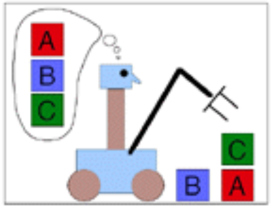
An alternative formulation



The solution in this space is the same as in the continuous space!

Lozano-Pérez, Tomás; Wesley, Michael A. (1979), "An algorithm for planning collision-free paths among polyhedral obstacles", *Communications of the ACM* **22** (10): 560–570,

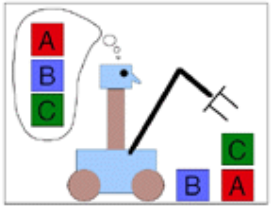
Assumptions in Basic Search



- The world is static
- The world is discretizable
- The world is fully observable
- The actions are deterministic

Search remains an important tool even if not all these assumptions are satisfied

Applications



- Search plays a key role in many AI systems, e.g.:
  - Route finding (google/mapquest, internet, airline)
  - VLSI Layout
  - Robot navigation.
  - Pharmaceutical drug design, protein design
  - Video games