



# State Spaces

*Introduction to Artificial Intelligence*

Dr. Robin Burke



# Details

- Moodle

- still not set up

- TA

- Ola Ayoola

- [olapeju.ayoola@ucd.ie](mailto:olapeju.ayoola@ucd.ie)

- contact her for any questions about assignments, marking, practicals,

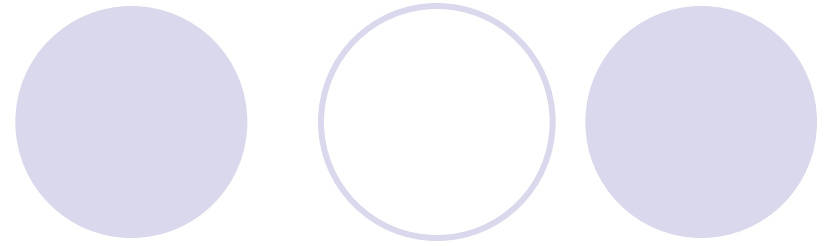
- contact me for content-related questions



# Problem Solving

- Specific subarea of AI
  - solving well-defined puzzle-like problems
- Classic example
  - Three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries.) The boat cannot cross the river by itself with no people on board.
  - (originally, the “jealous husbands problem” from medieval times)
- Solution = the sequence of boat trips that will get the groups across

# Problem Solving



- One of the first applications of AI
  - GPS (General Problem Solver)
  - Newell and Simon 1956
- Logical puzzles
  - “8 puzzle”

2	8	3
1	6	4
7		5

# Sudoku

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

# Sudoku solved

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



# An AI solution

- We need a representation
  - a formal description of the problem
  - the starting state of the problem
  - the termination condition (success)
  - the constraints
  - the possible moves
- We need an algorithm
  - how do we go about constructing a solution?



# Steps to solution

1. Define the problem precisely
2. Create a representation capable of explicitly encoding all important aspects of the problem
3. Encode the problem in the representation
4. Select and apply appropriate problem-solving method

# The Farmer Jones River Problem

Farmer Jones owns a small boat (Bounty), goat, a pet wolf (don't ask!), and a prize cabbage.

To get to the annual cabbage show he must cross from the south bank of the river to the north bank, but Bounty will only allow him to travel with one other object (it's a big cabbage!).

If left unattended, the wolf will eat the goat and the goat will eat the cabbage.

How can farmer Jones take his goat, wolf and cabbage to the show?



# Problem

- There is a lot of extraneous detail in the description
- What really matters in the problem?
  - objects
  - states of the world
  - constraints
- Simplifying assumptions
  - can ignore rowing of the boat
  - can ignore tying the boat to the dock, etc.

# States and Operators



- States

- conditions of the world relative to the problem

- Special States

- starting state (the problem)
- ending state (the solution)
- illegal states

- Operators

- actions that take the world from one state to another



Farmer

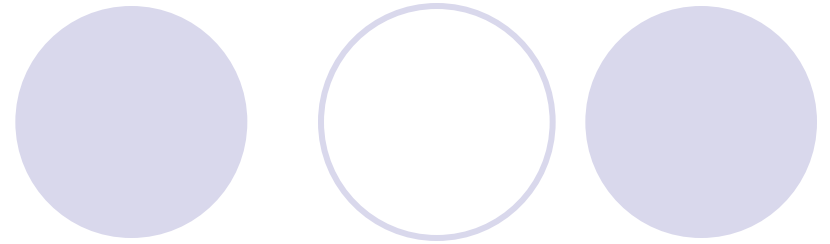
- States
- Operators
- Also, constraints
  - states that are illegal

# Representation Language

- How to encode the problem?
- We need to represent
  - the position of four objects
    - we ignore the boat because the farmer is always with the boat
  - either north or south bank
- A natural choice is a tuple

( farmer    wolf    goat    cabbage  
  A?    B?    C?    D? )

# Representation



- Initial State

- (S, S, S, S)

- Goal State

- what constitutes a solution

- (N, N, N, N)

# Actions



- An action takes one state of the world and turns it into another
- Carry Goat North
  - start with a world in which the Goat is on the South bank
  - end up with a world in which the Goat is on the North bank
  - nothing else changes
- Representation
  - Move(Goat, North)
    - (S, ?wolf, S, ?cabbage) => (N, ?wolf, N, ?cabbage)
  - ?*wolf* is a variable representing the position of the wolf
  - using variables saves me from having to write four rules

# Constraints



- Certain states of the world are not permitted
  - wolf eats goat (only if farmer not present)
    - (N, S, S, ?cabbage)
    - (S, N, N, ?cabbage)
  - goat eats cabbage (only if farmer not present)
    - (N, ?wolf, S, S)
    - (S, ?wolf, N, N)

# 8-puzzle

- States?
- Operators?

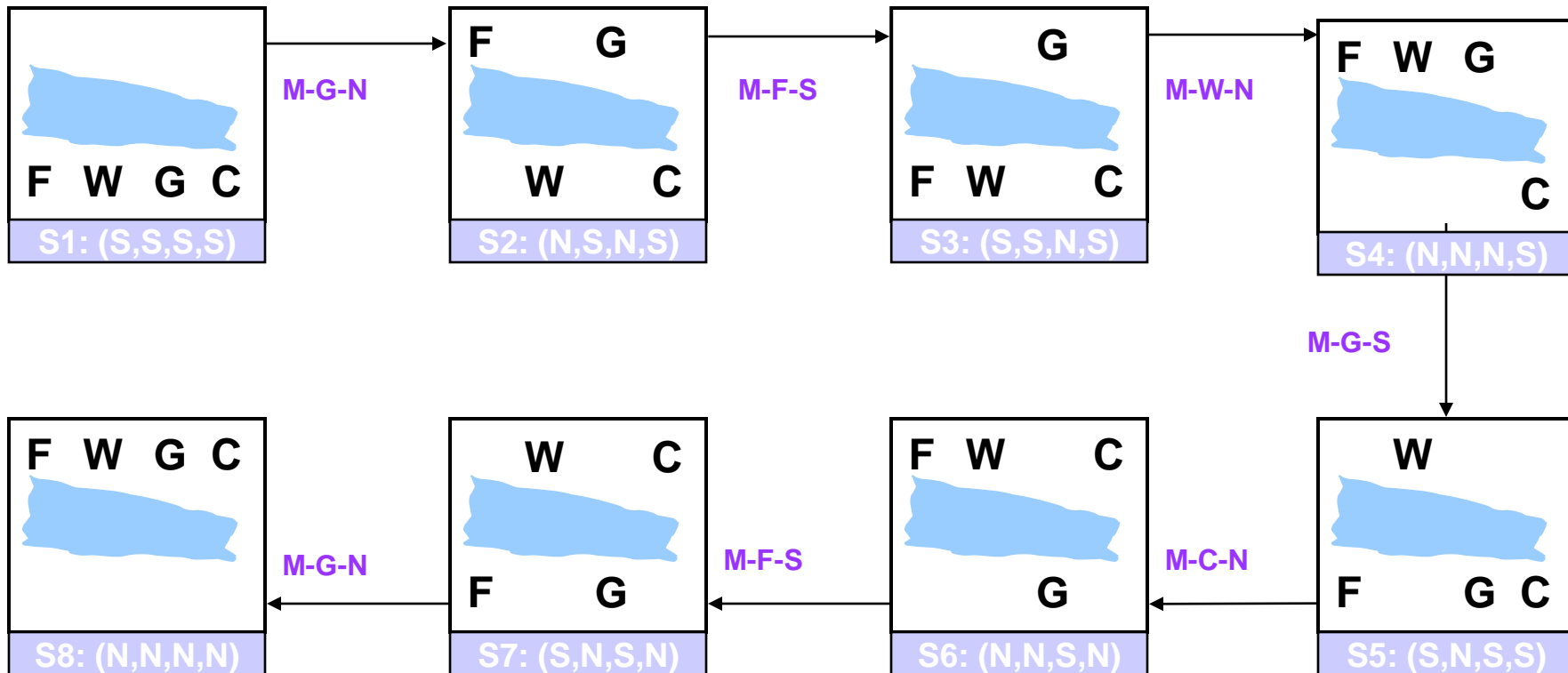
2	8	3
1	6	4
7		5

# Solving the Farmer Problem

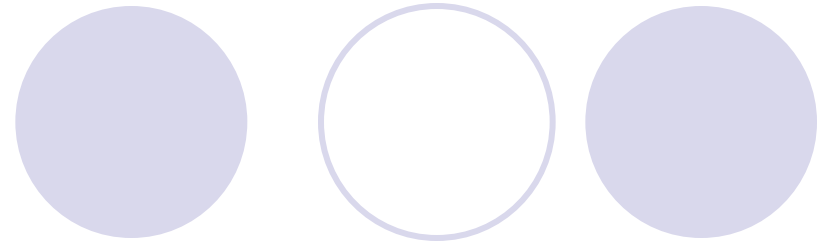
- Representation is complete
  - there are eight actions
  - there are  $2^4 = 16$  states
- How to solve?
- Simplest approach
  - try all possible combinations of actions
    - the shortest solution is seven steps
    - there are  $8^7 =$  about 2 million 7 step sequences
  - many combinations are impossible
    - $\langle \text{Move}(\text{goat}, \text{north}), \text{Move}(\text{goat}, \text{north}), \dots \rangle$
  - some combinations lead to illegal situations
    - $\text{Move}(\text{cabbage}, \text{north})$  can't be the first move

# Example – One Solution...

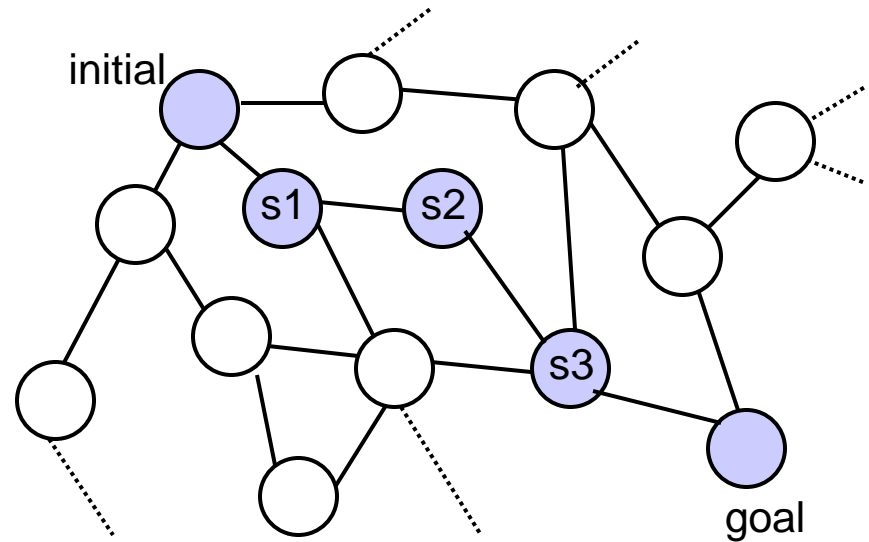
One 7 step solution to the river problem.



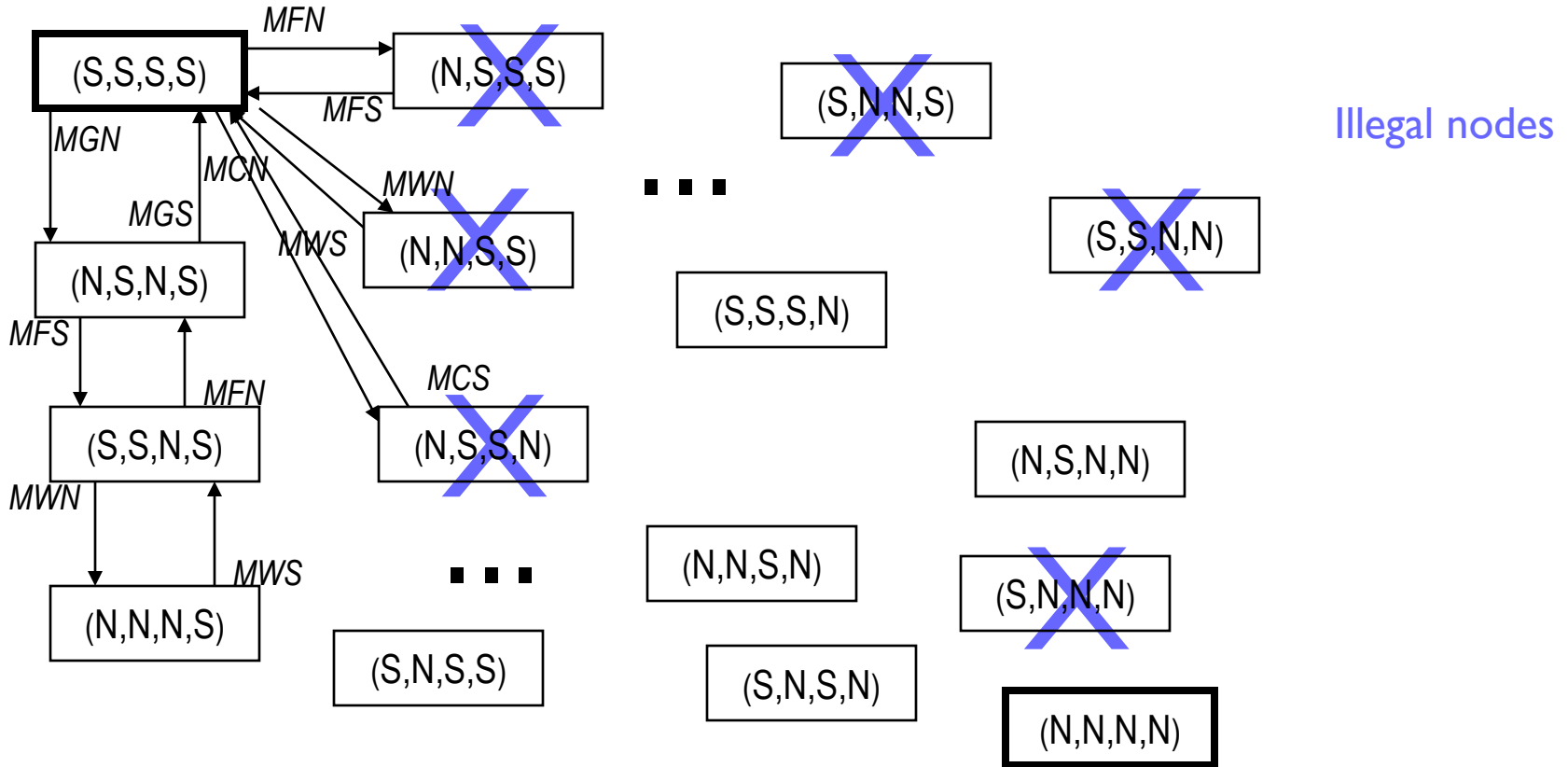
# State Space



- Conceptualize the problem as a graph
- Each state is a node
- Each action is an edge
- We start at some initial node
  - trying to set to some destination
  - the edges we cross constitute the list of actions to take



# Our State Space

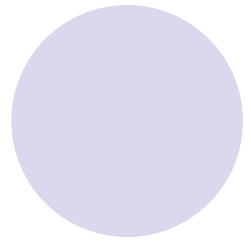
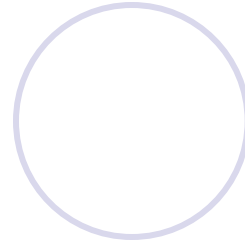
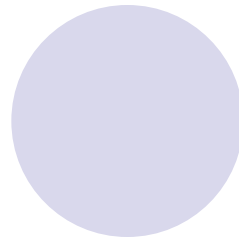
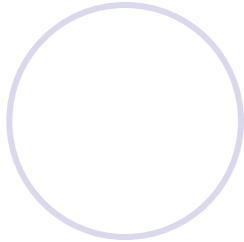
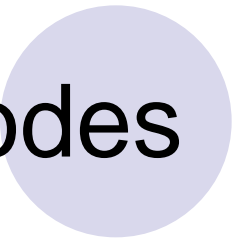


# Search



- Formulate the problem as one of search
- We start with the initial state
  - we define “successor states” as ones that we can reach legally from that state
  - at each successor state
    - we check to see if we are in the final state, if so, we are done
    - we check to see if the state is illegal, if so, we ignore this state
    - otherwise, we keep searching

# Nodes



- Repeated states

- we never want to go back to the same state

- Move (Goat, North), Move (Goat, South), Move(Goat, North) etc.

- wasted actions

- but there may be multiple ways to reach the same state

- need to distinguish

- some paths might be better than others

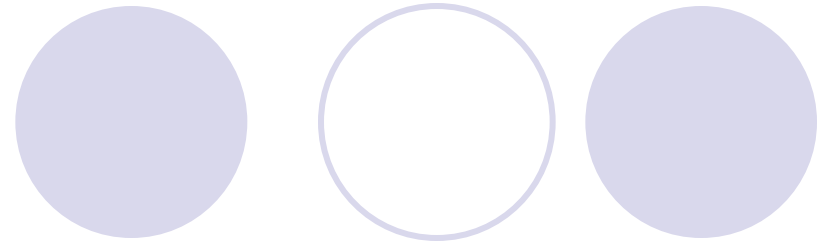
- Search Nodes

- a search node is

- state

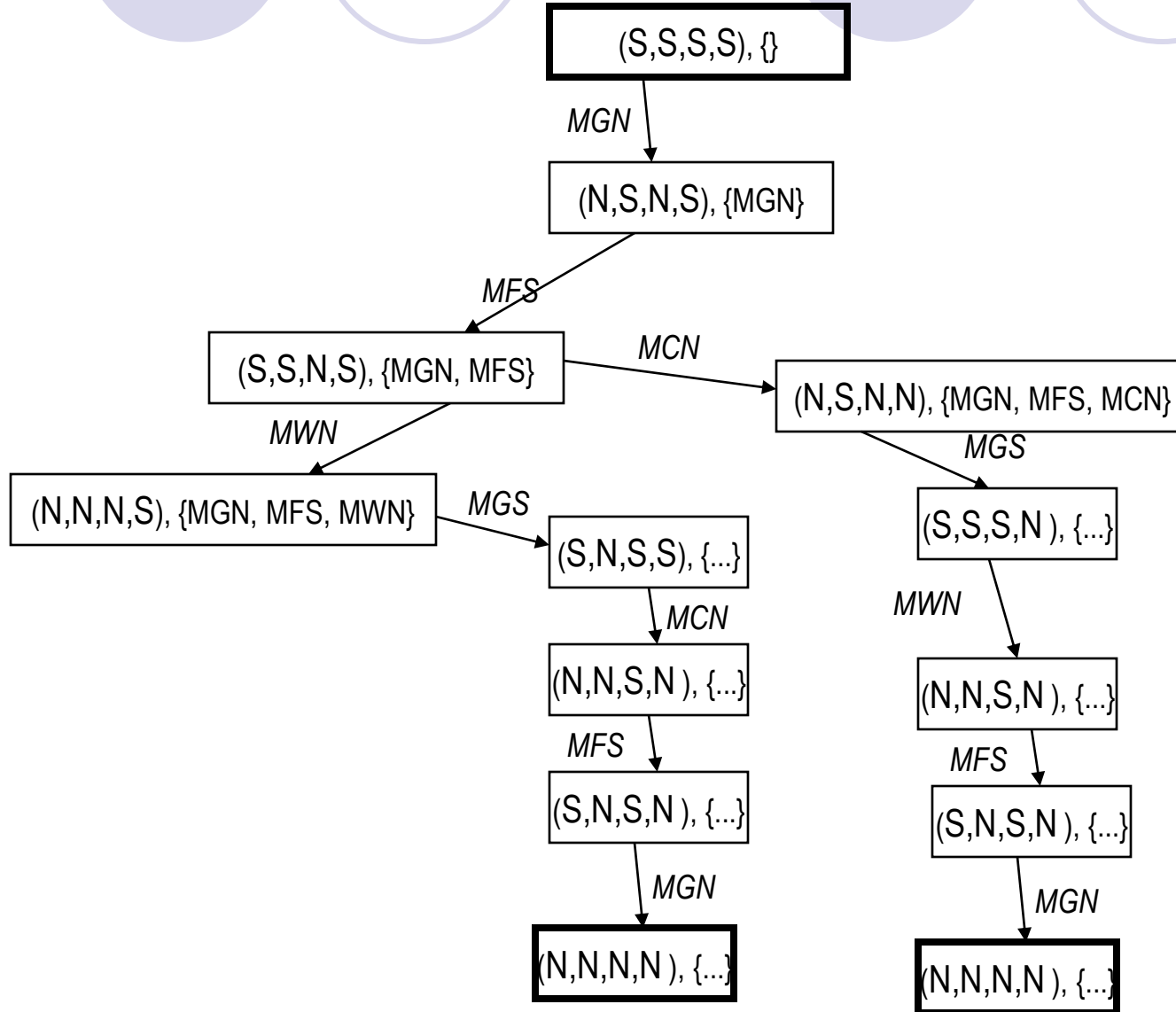
- path to the state

# Search Tree



- Use search nodes
  - (instead of states)
- Useful properties
  - the graph is acyclic
  - only one way to reach a given node
    - therefore, the state space becomes a search tree
  - the solution / path is part of the representation

# Search Tree (ignore illegal states)



# Toy example



- Search space quite small
  - 10 legal states of 16 possible
- Every path leads to success
  - or quick failure
- Only one legal first move
- People sometimes have trouble, though
  - the idea of “progress” is defeated

# Search algorithms



- For a real problem
  - impossible to examine entire search space
- We need a search algorithm
  - determines how to move through the search tree
  - which nodes to examine and in what order

# Basic idea



- Given
  - a set of nodes  $N$
  - a successor function  $f$  that
    - takes a node  $n$
    - returns all of the nodes  $S$  reachable from  $n$  in a single action
- Algorithm
  - pick  $n$  from  $N$  (somehow)
  - $s = \text{state}(n)$
  - $p = \text{path}(n)$
  - $S = f(s)$
  - for all  $s', a \in S$ 
    - if  $s$  is solution, done
    - if  $s$  is illegal, discard
    - else
      - create new node  $n_a = \langle s', p + a \rangle$
    - $N = N - n + n_a$
  - repeat

# Search strategies



- All of the interesting action is here
  - “Pick n from N”
  - Selecting the node to “expand”
- Classes of strategies
  - uninformed search
    - no knowledge of the search space is assumed
    - do not prefer one state over another
  - informed search
    - use knowledge of the search space
    - heuristic search

Tuesday

- Read Chapter 3.4-3.7