Lecture 3

AI applications, Uninformed Search Strategies

(Ch 3.1-3.4)

Today's Lecture

- Discussion on AI applications
- Search
- Search Spaces
- Generic Search Algorithm
- Uninformed Search (time permitting)
- Depth first

AI agents in this course

Would like most general agents possible, but in this course, we need to restrict ourselves to:

- Flat representations (vs. hierarchical)
- Knowledge given (vs. knowledge learned)
- Goals and simple preferences (vs. complex preferences)
- Single-agent scenarios (vs. multi-agent scenarios)

We will look at

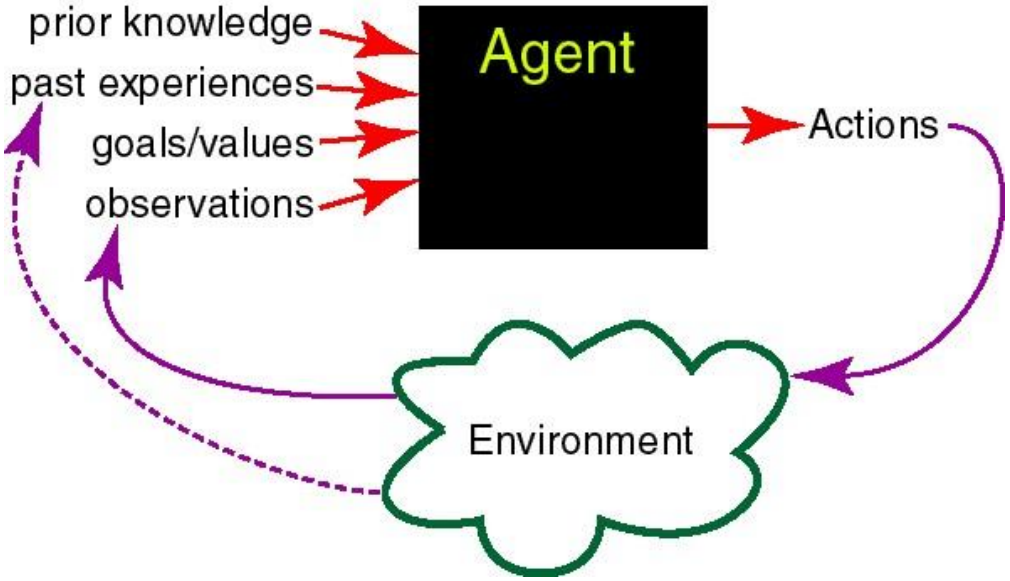
- Deterministic and stochastic domains
- Static and Sequential problems

And see examples of representations using

- Explicit state or features or relations

AI Application

- Today, we will look at some AI applications that you have found for your assignment 0



- You are asked to described them in terms of the elements above and some more

- What does it do

- Goals

- prior knowledge needed

- past experiences that it does (or could) learn from

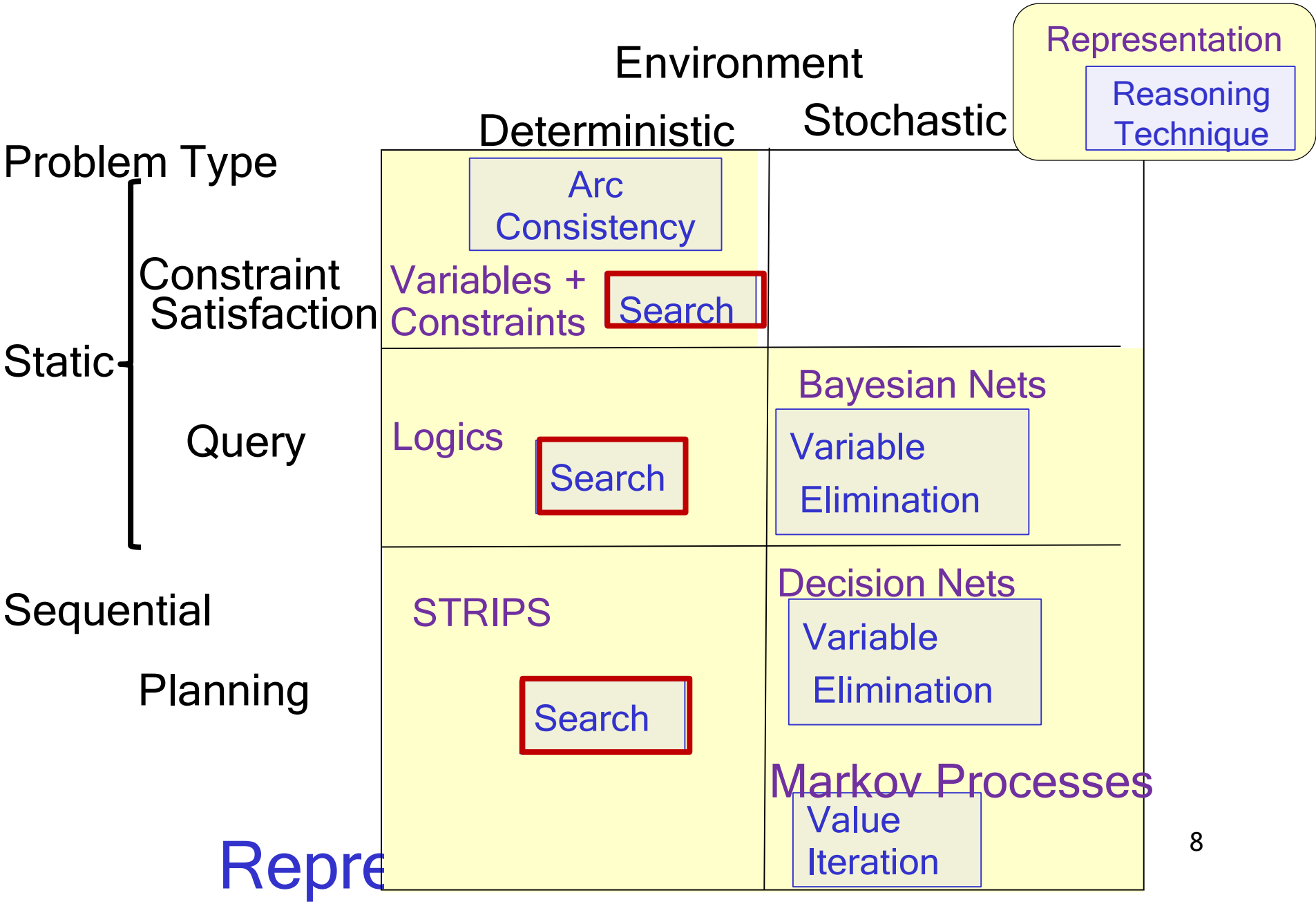
- Observations needed

- Actions performed
- AI technologies used
- Why is it intelligent?
- Evaluation?

Today's Lecture

- Discussion on AI applications
- Search

- Search Spaces
- Generic Search Algorithm
- Uninformed Search
- Depth first


Representation
Reasoning Technique
Environment
Deterministic
Stochastic
Problem Type
Static
Constraint Satisfaction
Query
Sequential
Planning
Arc Consistency
Variables + Constraints
Search
Logics
Search
Bayesian Nets
Variable Elimination
STRIPS
Search
Decision Nets
Variable Elimination
Markov Processes
Value Iteration

Repre

Recap

- Search is a key computational mechanism in many AI agents
- We study the basic principles of search via a simple deterministic search agent model
- Agent is in a start state
- Agent is given a goal (subset of possible states)
- Environment changes only when the agent acts
- Agent perfectly knows:
- actions that can be applied in any given state

- the state it is going to end up in when an action is applied in a given state
- The sequence of actions (and appropriate ordering) taking the agent from the start state to a goal state is the solution

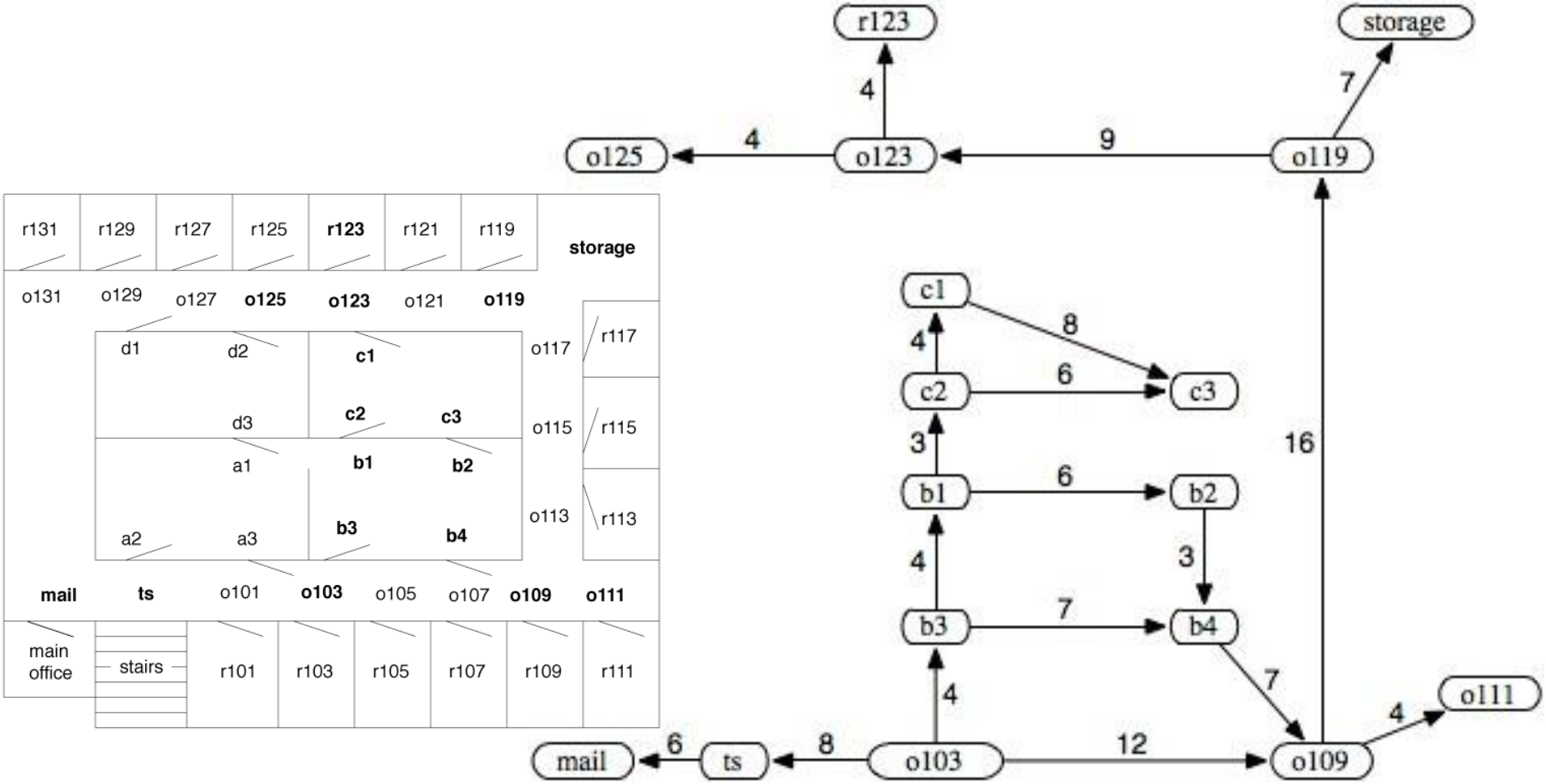
Slide

Definition of a search problem

- Initial state(s)
- Set of actions (operators) available to the agent
- An action function that, given a state and an action, returns a new state
- Goal state(s)

- Search space: set of states that will be searched for a path from initial state to goal, given the available actions
- states are nodes and actions are links between them.
- Not necessarily given explicitly (state space might be infinite)
- Path Cost (we ignore this for now)

Search Space for the Delivery Robot

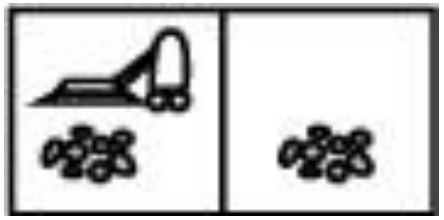


Today's Lecture

- Discussion on AI applications
- Search
- Search Spaces
- Generic Search Algorithm
- Uninformed Search
- Depth first

Example: vacuum world

- States
- Two rooms: r1, r2
- Each room can be either dirty or not
- Vacuuming agent can be in either in r1 or r2



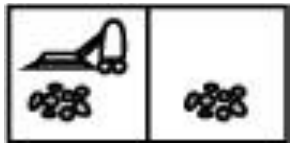
Example: vacuum world

Possible start state

Possible goal state

- States
- Two rooms: r1, r2
- Each room can be either dirty or not
- Vacuuming agent can be in either in r1 or r2

Feature-based representation:
- Features?
- how many states?



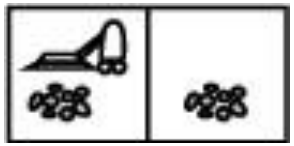
Possible start state

Possible goal state

Example: vacuum world

- States
- Two rooms: r1, r2
- Each room can be either dirty or not
- Vacuuming agent can be in either in r1 or r2

Feature-based representation:
- Features?
- how many states?



Possible start state

Possible goal state

Suppose we have the same problem with krooms.

The number of states is….

A. k^3

B. $k * 2k$

C. $k * 2^k$

D. $2 * k^k$

Suppose we have the same problem with krooms.

The number of states is….

A. k^3

B. $k * 2k$

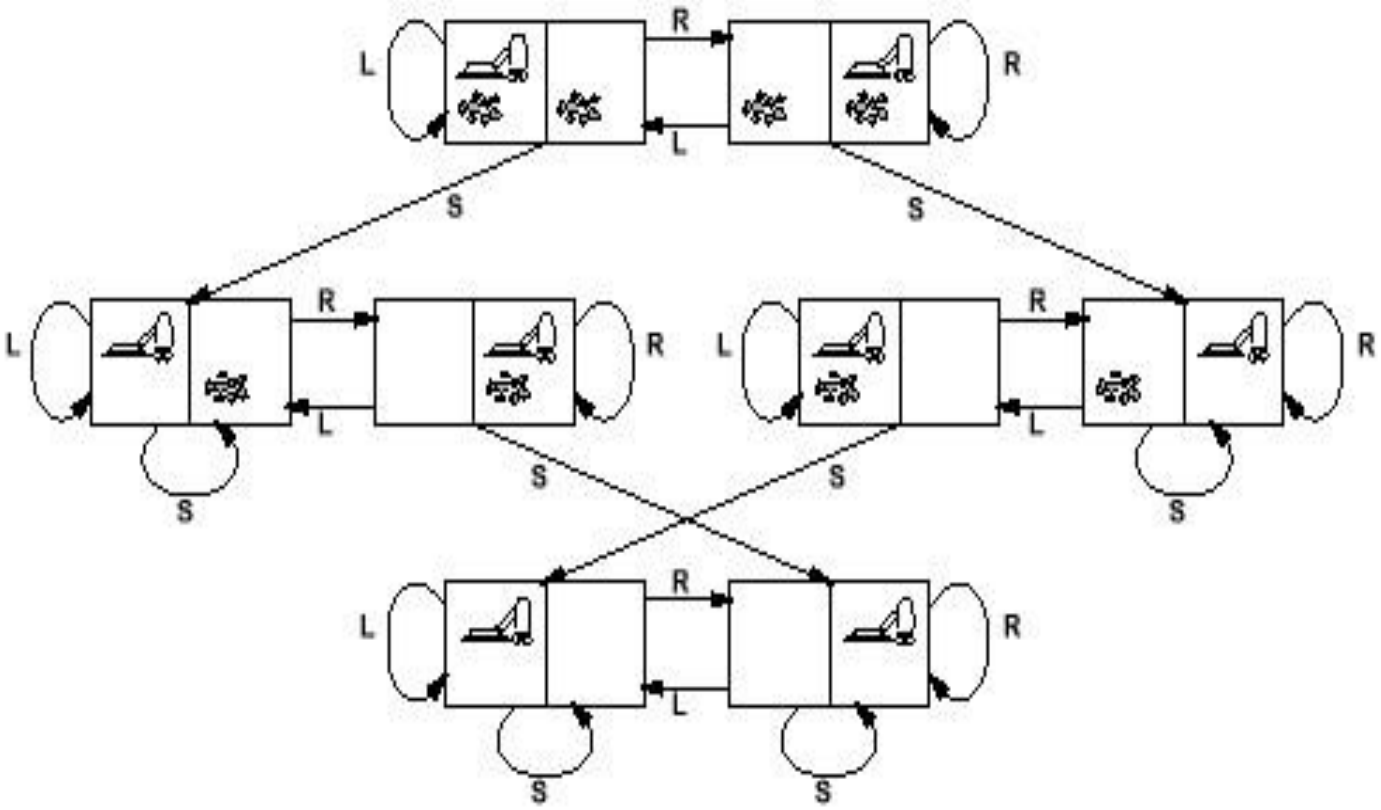
C. $k * 2^k$

D. $2 * k^k$

Loc() feature can take k possible values

For each room i, dirty_room_i can take 2 values, and there are k of these features

Search Space



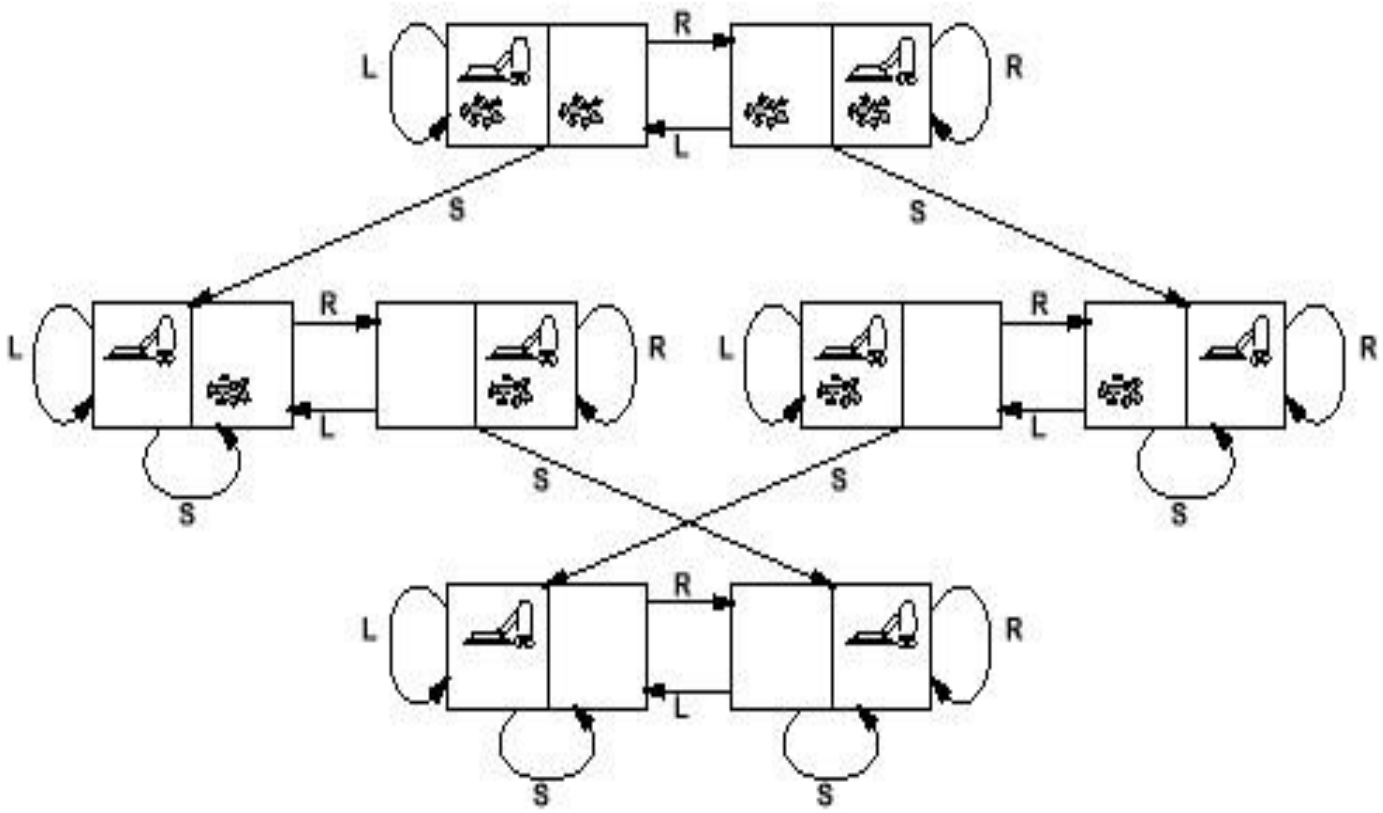
- Actions – left, right, suck

  action applied to a given state

- Possible Goal – no dirt

- Successor states in the graph describe the effect of each

Search Space



- Actions – left, right, suck

  action applied to a given state

- Possible Goal – no dirt

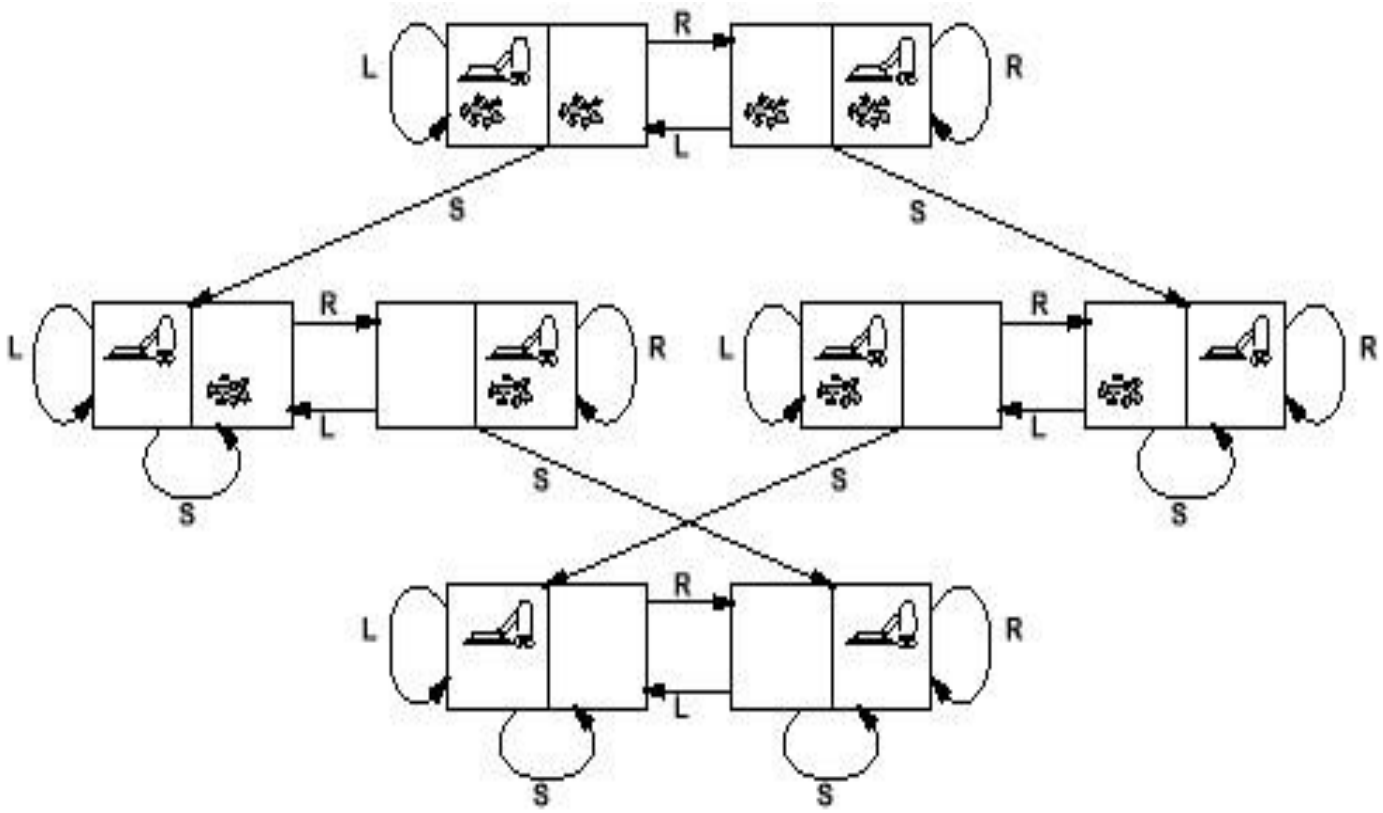
Search Space

- Successor states in the graph describe the effect of each

action applied to a given state

- Possible Goal – no dirt

Search Space



- Actions– left, right, suck

  action applied to a given state

- Possible Goal – no dirt

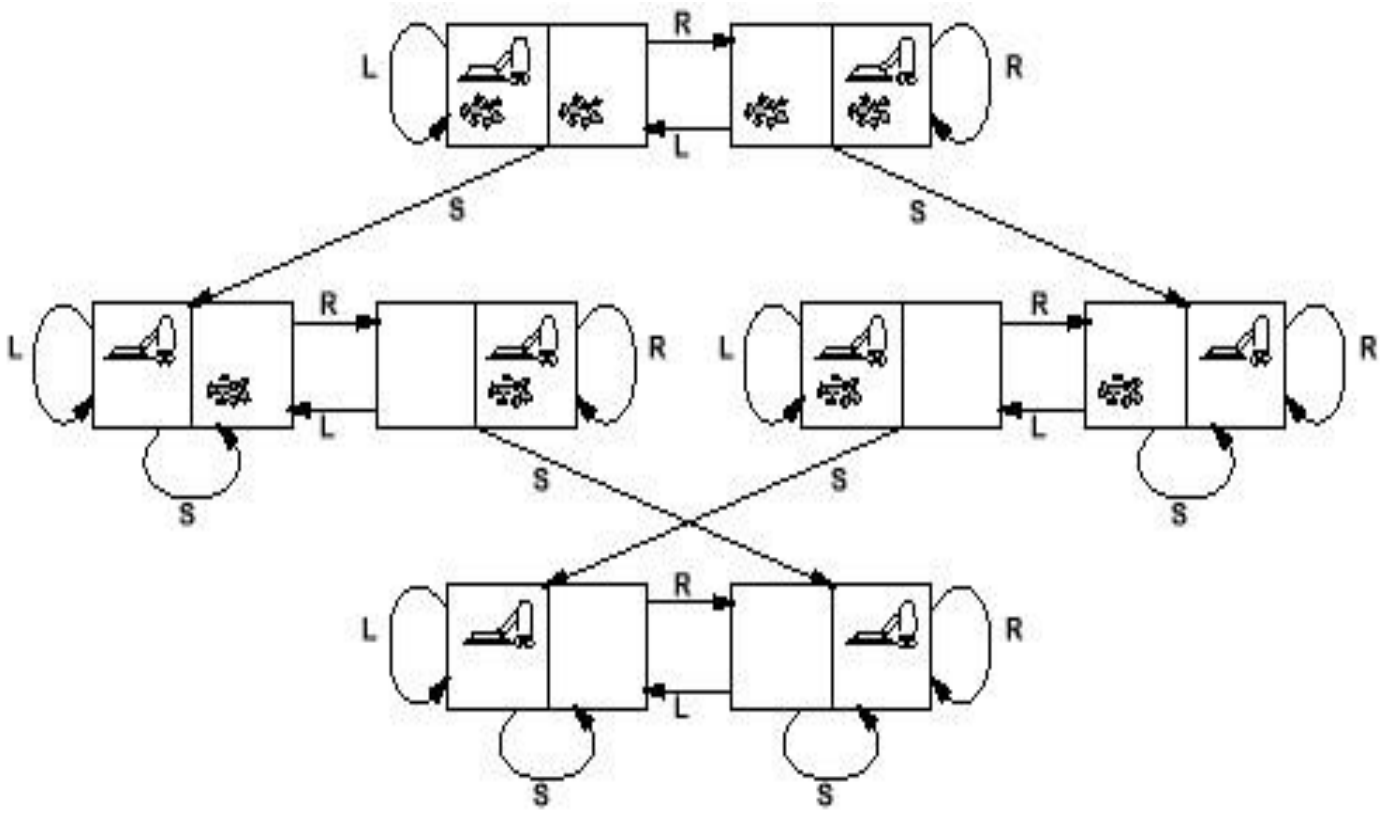
Search Space

- Successor states in the graph describe the effect of each

action applied to a given state

- Possible Goal – no dirt

Search Space



- Actions – left, right, suck

  action applied to a given state

- Possible Goal – no dirt

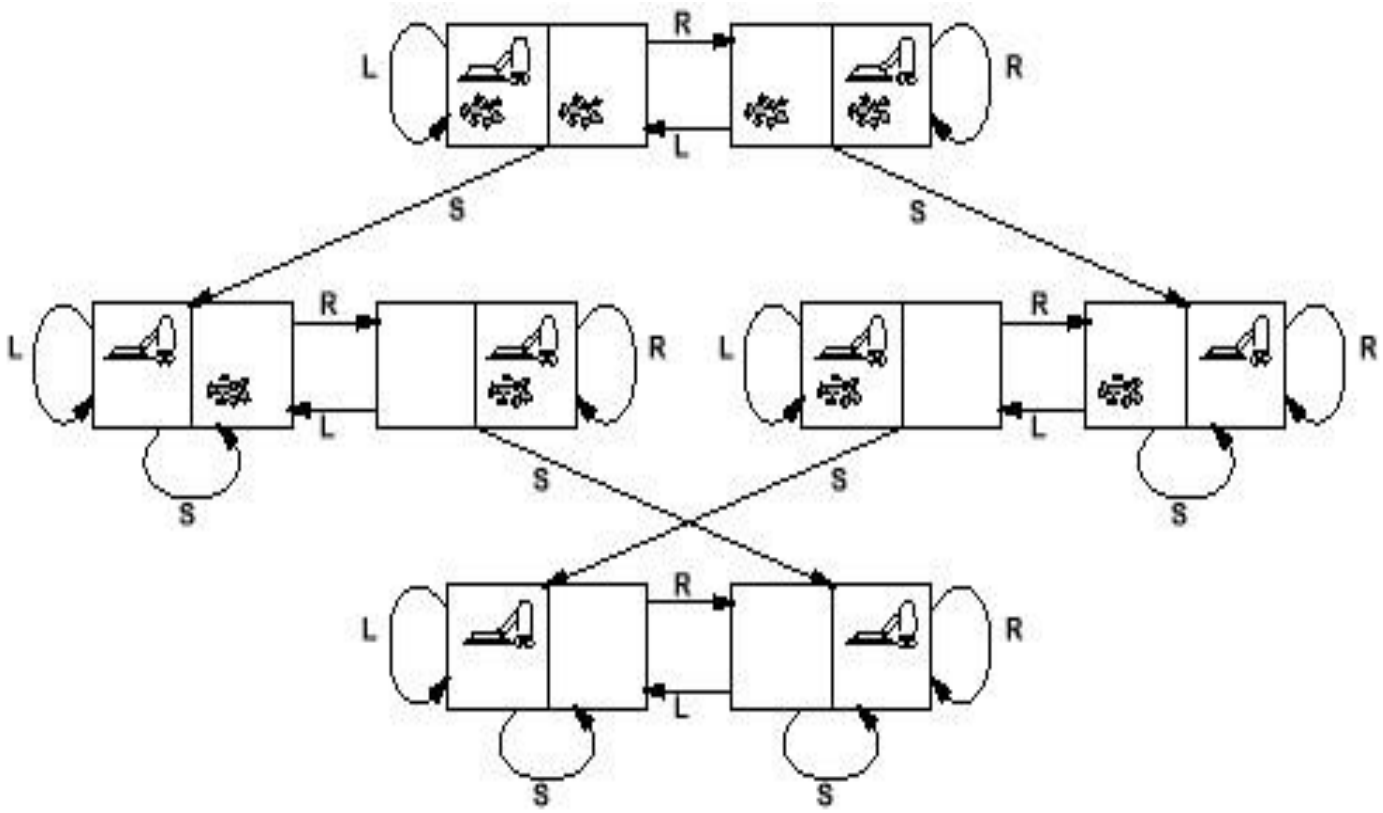
Search Space

- Successor states in the graph describe the effect of each

action applied to a given state

- Possible Goal – no dirt

Search Space



- Actions – left, right, suck

  action applied to a given state

- Possible Goal – no dirt

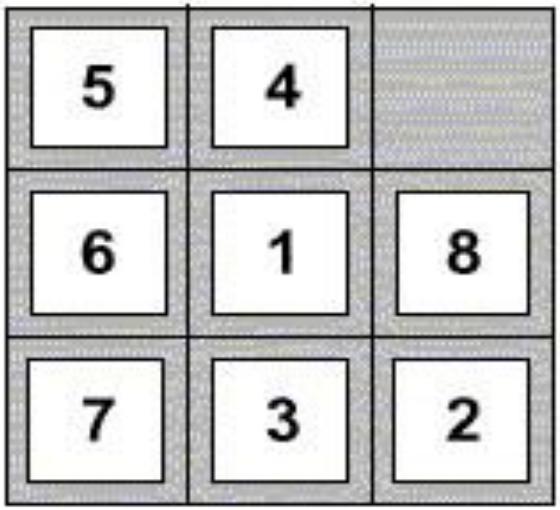
Search Space

- Successor states in the graph describe the effect of each

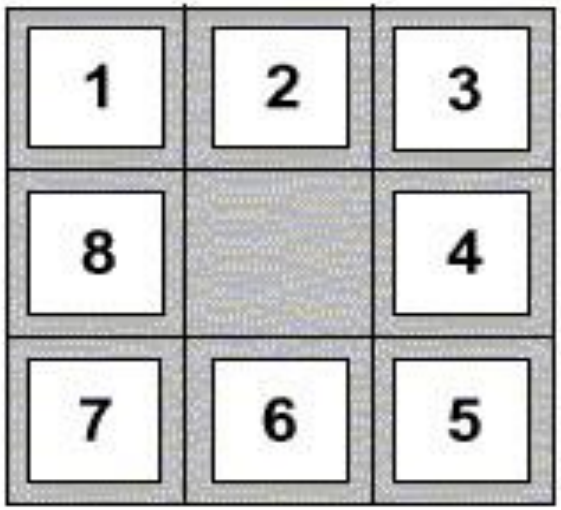
action applied to a given state

- Possible Goal – no dirt

Eight Puzzle

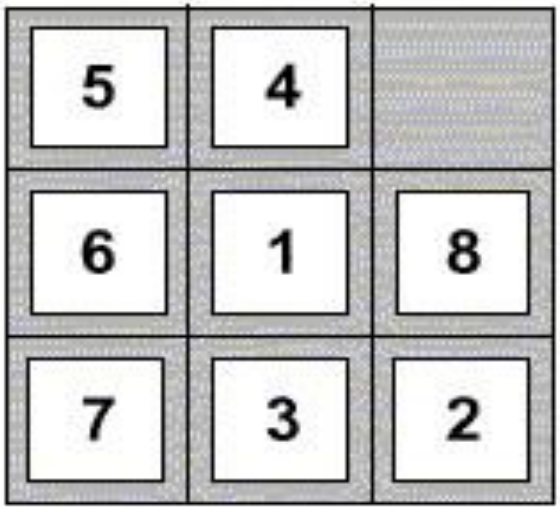


Start State

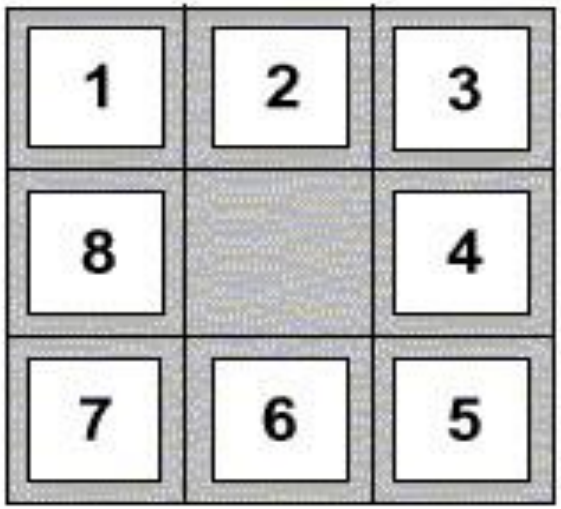


Goal State

Eight Puzzle



Start State



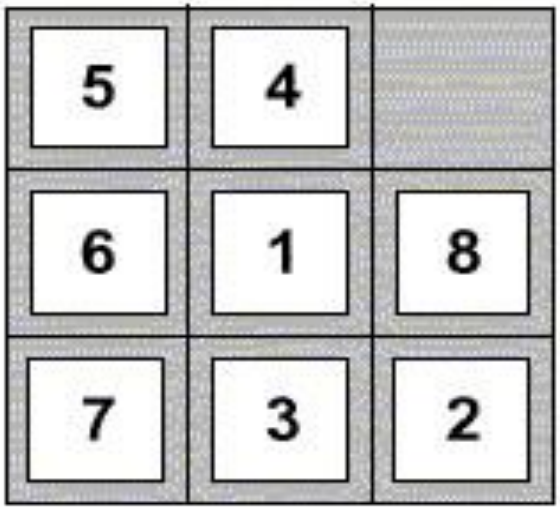
Goal State

States: each state specifies which number/blank occupies each of the 9 tiles

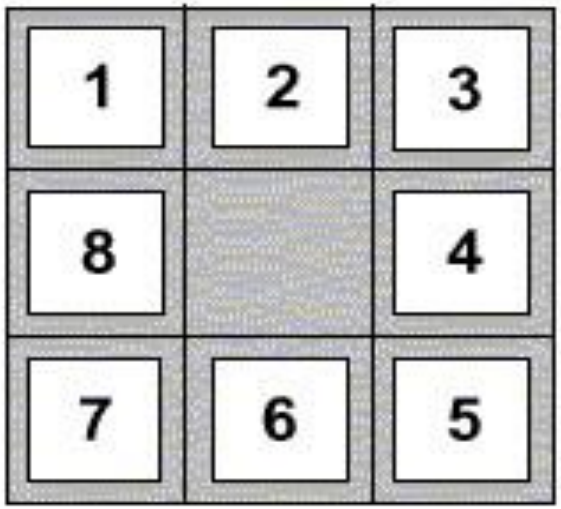
HOW MANY STATES ?

Actions:

Eight Puzzle



Start State



Goal State

Goal:

States: each state specifies which number/blank occupies each of the 9 tiles

HOW MANY STATES? *9!*

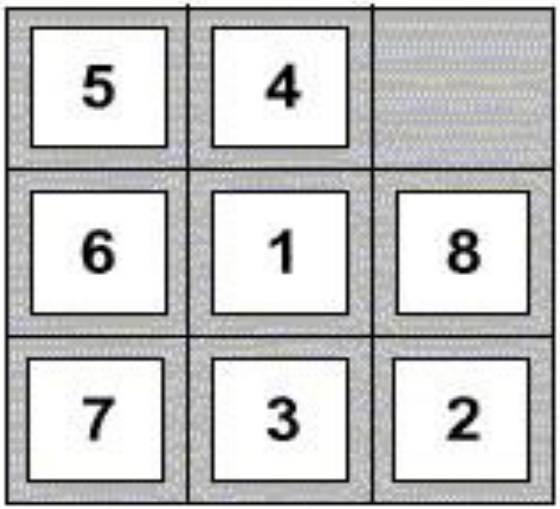
Eight Puzzle

Actions:

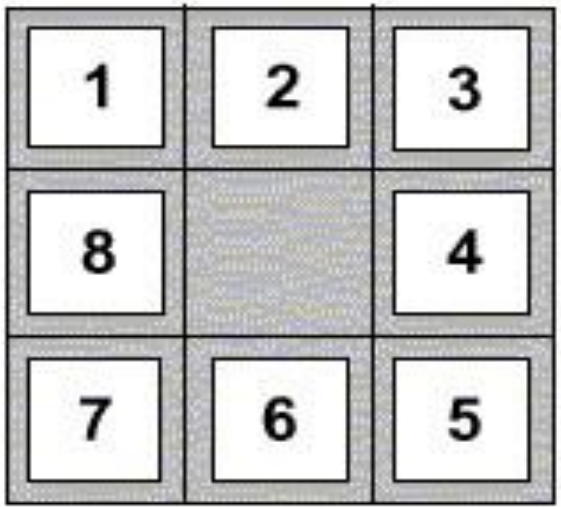
Goal:

States: each state specifies which number/blank occupies each of the 9 tiles

Eight Puzzle



Start State



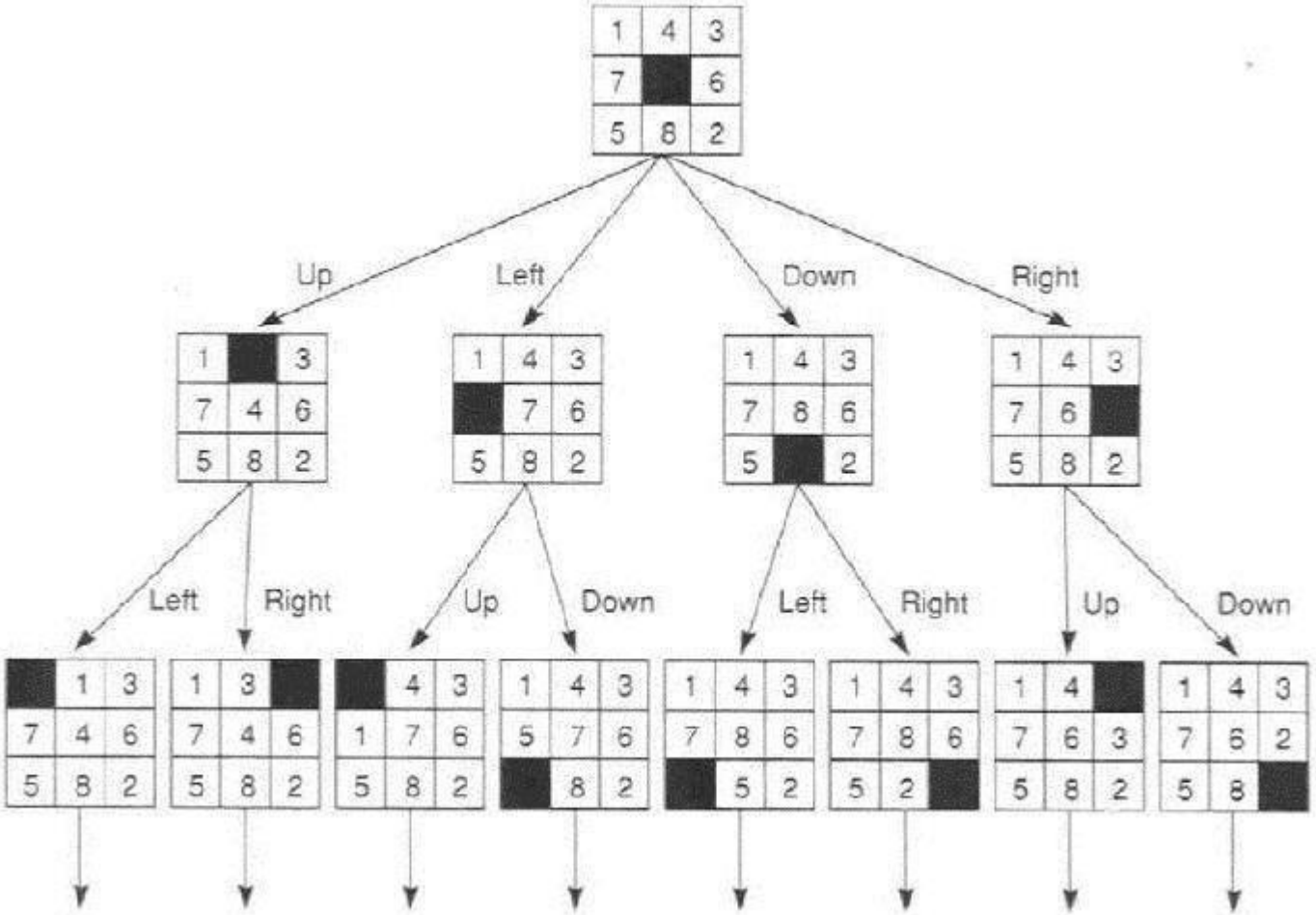
Goal State

HOW MANY STATES ? *9!*

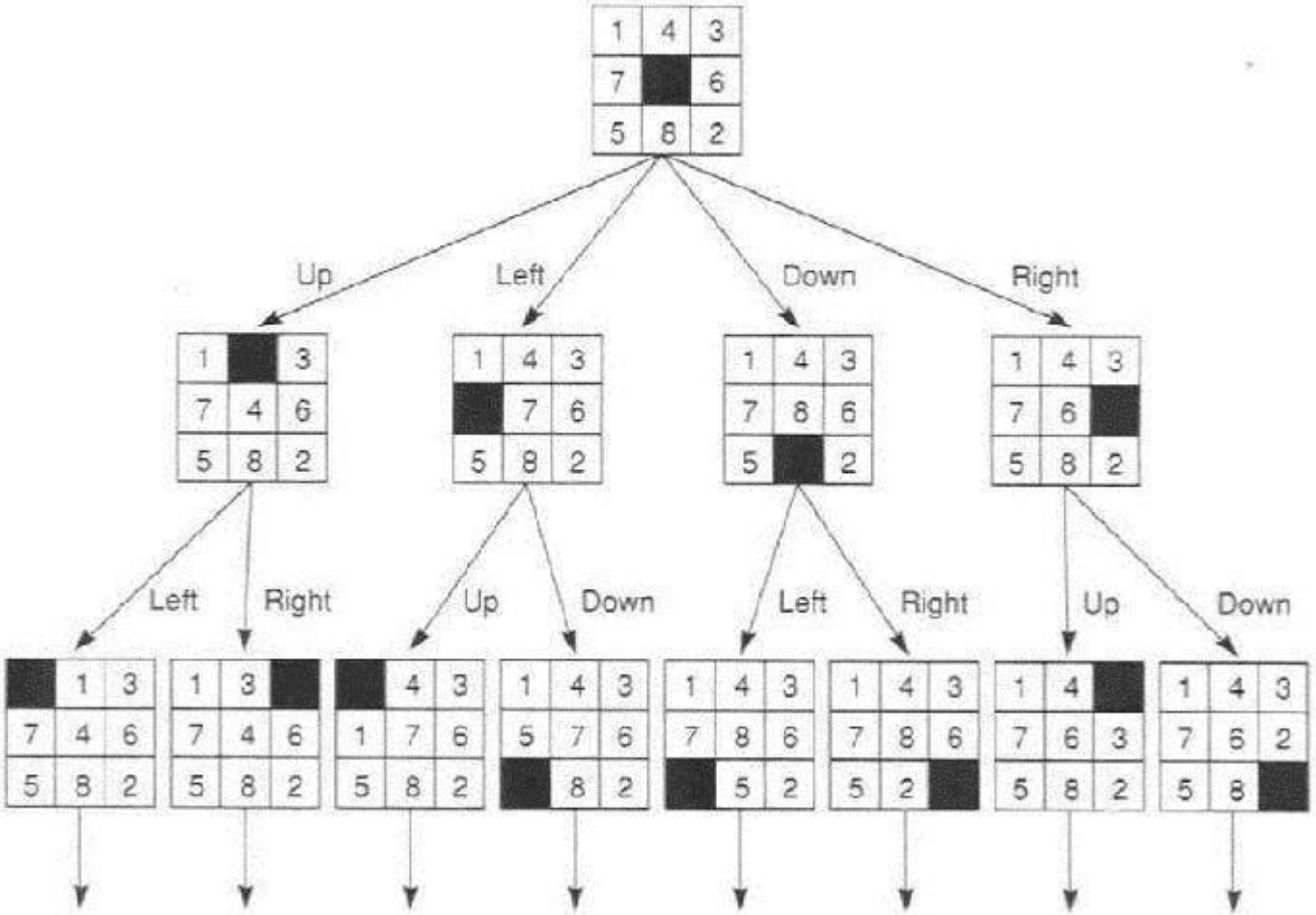
Actions: blank moves left, right, up down

Goal: configuration with numbers in right sequence

Search space for 8puzzle

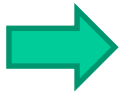

Up
Left
Down
Right
Left
Right
Up
Down
Left
Right
Up
Down

Search space for 8puzzle


Up
Left
Down
Right
Left
Right
Up
Down
Left
Right
Up
Down

How can we find a solution?

- How can we find a sequence of actions and their appropriate ordering that lead to the goal?
- Need smart ways to search the space graph



Today's Lecture

- Discussion on AI applications
- Search
- Search Spaces
- Generic Search Algorithm
- Uninformed Search (time permitting)
- Depth first

Search: Abstract Definition

How to search

- Start at the start state
- Evaluate the effect of taking different actions starting from states that have been encountered in the search so far
- Stop when a goal state is encountered

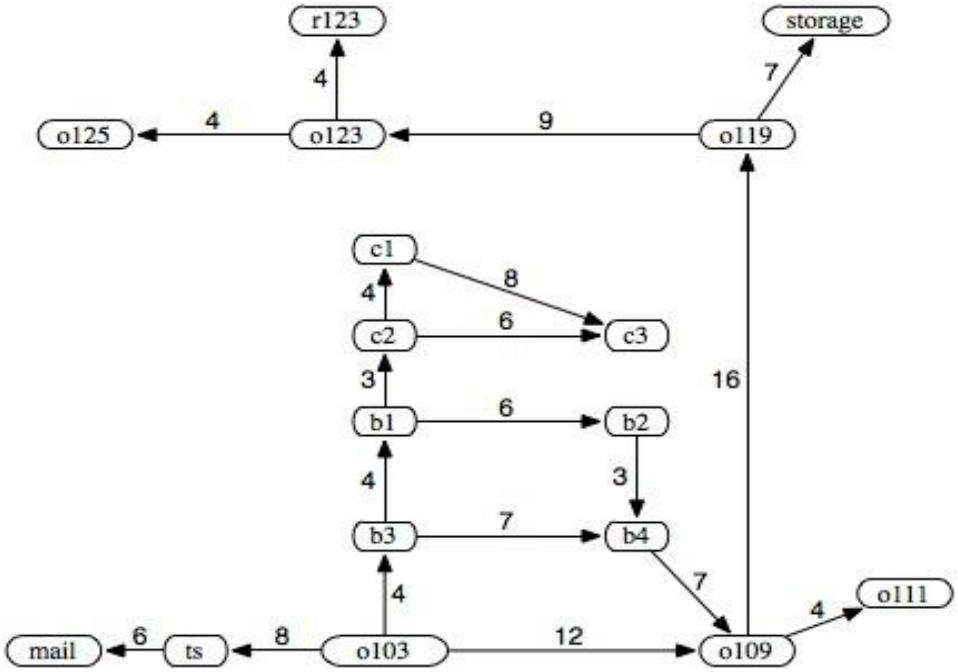
To make this more formal, we'll use the definition of a graph that you were asked to review for today

Graphs

- A directed graph consists of a set N of nodes (vertices) and a set A of ordered pairs of nodes, called edges (arcs).
- Node n_2 is a neighbor of n_1 if there is an arc from n_1 to n_2. That is, if □ n_1, n_2 □ □ A.
- A path is a sequence of nodes n_0, n_1,..,n_k such that □ n_{i-1}, n_i □ □ A.
- A cycle is a non-empty path such that the start node is the same as the end node.

- A directed acyclic graph (DAG) is a graph with no cycles
- Given a set of start nodes and goal nodes, a solution is a path from a start node to a goal node

Graph specification for the Delivery Robot



N={mail, ts, o103, b3, o109,...}

A={

⟨ts,mail⟩,

⟨o103,ts⟩, ⟨o103,b3⟩,

⟨o103,o109⟩,
...}

One of several solution paths:

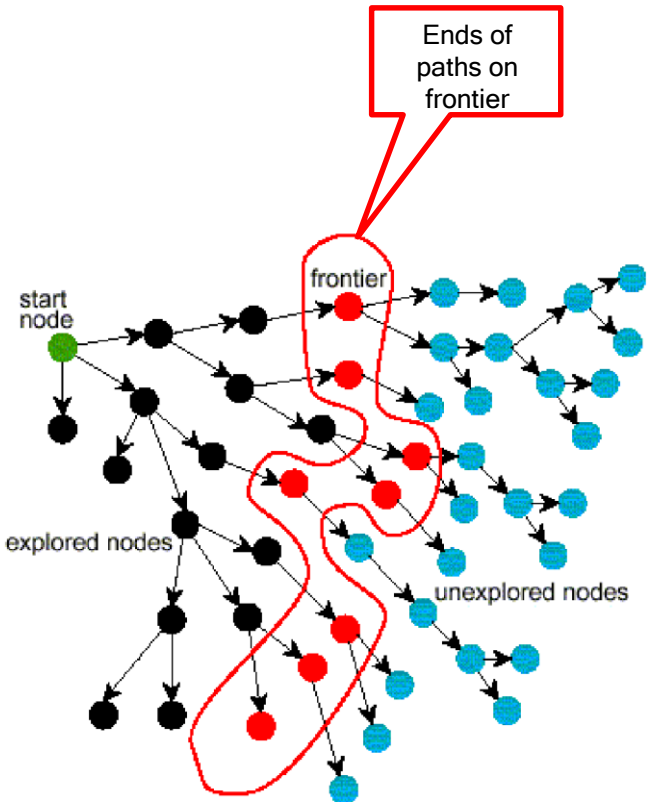
⟨o103, o109, o119, o123, r123⟩

Graph Searching

- Generic search algorithm:
  - given a graph, start nodes, and goal nodes, incrementally explore paths from the start nodes.

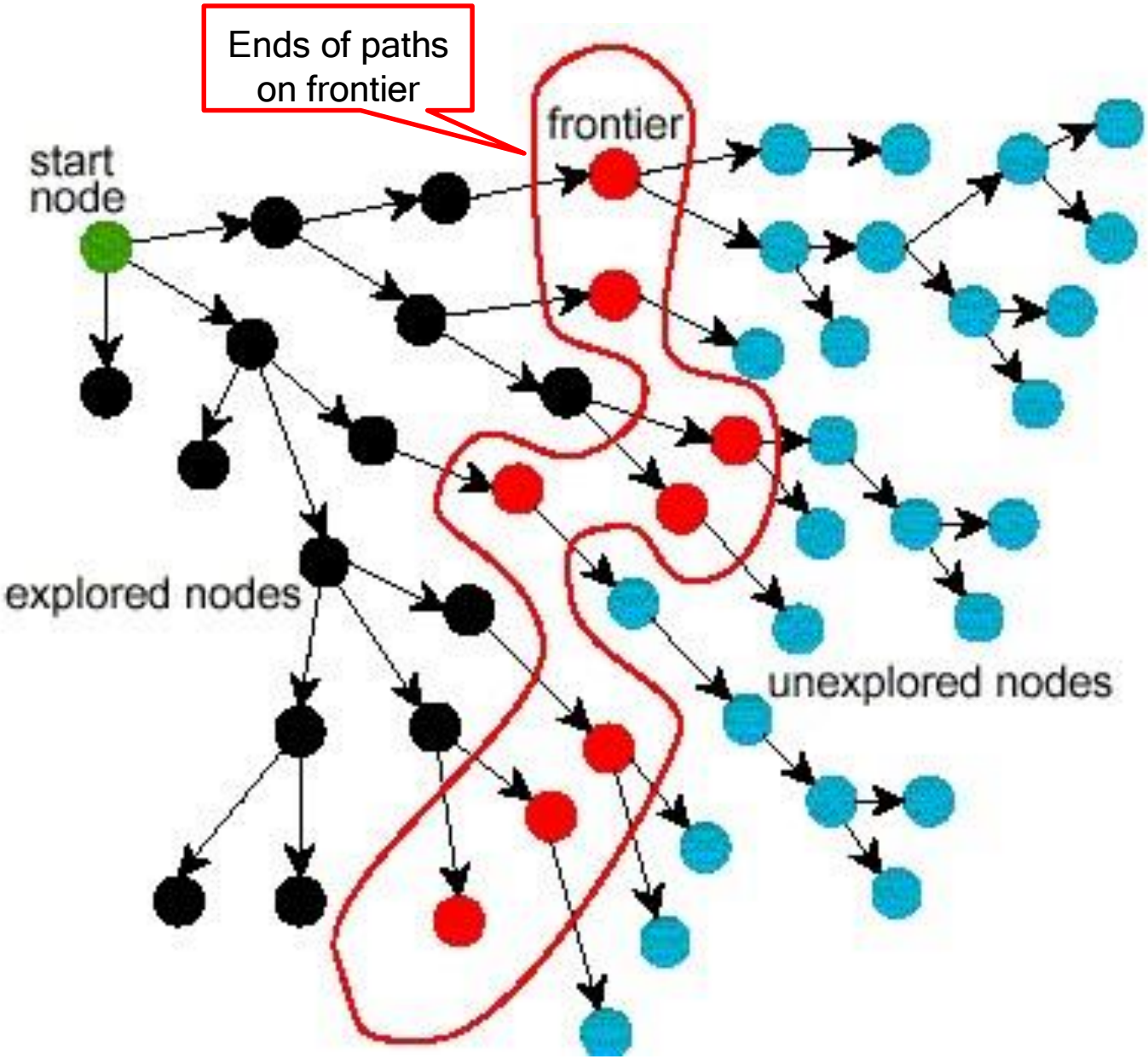
- Maintain a frontier of paths that have been explored from the start node
- As search proceeds, the frontier expands into the unexplored nodes until a goal node is encountered.

The way in which the frontier is expanded defines the search strategy.



If there is only one thing you want to remember about search, this is it.

Problem Solving by Graph Searching


Ends of paths
on frontier
frontier
start
node
explored nodes
unexplored nodes

Input:

- a graph
- a set of start nodes
- Boolean procedure goal(n) testing if n is a goal node

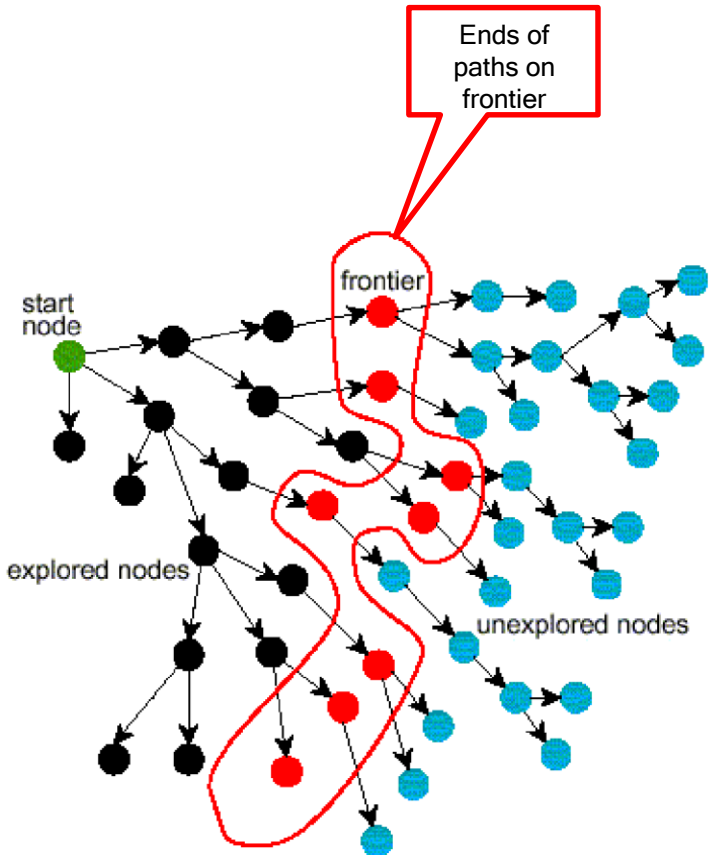
frontier:= [<s>: s is a start node]; While

frontier is not empty:

select and remove path $<n_o,\ldots.,n_k>$ from frontier;

If goal(n_k) return $<n_o,\ldots.,n_k>$;

For every neighbor n of n_k, add $<n_o,\ldots.,n_k, n>$ to frontier; end



Generic Search Algorithm

- The goal function defines what is a solution.

- Which path is selected from the frontier defines the search strategy.

Comparing Searching Algorithms: Will it find a solution? the best one?

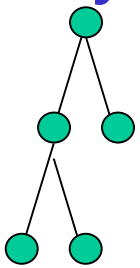
Def. : A search algorithm is complete if, whenever there is at least one solution, the algorithm is guaranteed to find it within a finite amount of time.

Def.: A search algorithm is optimal if, when it finds a solution, it is the best one

Slide

Comparing Searching Algorithms: Complexity

Branching factor b of a node in a graph is the number of arcs going out of the node



Def.: The time complexity of a search algorithm is the worst- case amount of time it will take to run, expressed in terms of

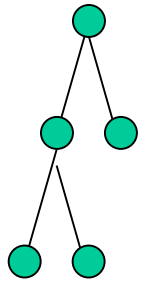
- maximum path length m
- maximum branching factor b.

Def.: The space complexity of a search algorithm is the worst-case amount of memory that the algorithm will use (i.e., the maximum number of nodes on the frontier),

- also expressed in terms of mand b.

Questions: Branching Factor

- The branching factor of a node is the number of arcs going out of the node



- If the forward branching factor of a node is b and the graph is a tree, how many nodes are nsteps away from that node?

A. nb

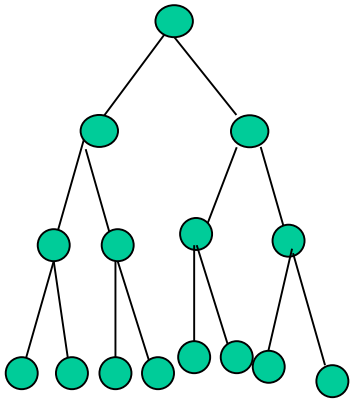
B. b^n

C. n^b D. n/b

Branching Factor

- The branching factor of a node is the number of arcs going out of the node

- If the forward branching factor of a node is band the graph is a tree, how many nodes are nsteps away from that node?



B. b^n

Learning Goals for today's class

- Identify real world examples that make use of deterministic, goal-driven search agents
- Formalize these examples in terms the components of a search problem
- Assess the size of the search space of a given search problem.
- Implement the generic solution to a search problem.

- Define completeness, optimality, time complexity and space complexity for search algorithms

TO DO

- Start Heuristic Search: Ch 3.6

- Work on Practice Exercise 3A and 3.B in AI space

(link also on class schedule)

- Not for marks, but useful to review material from today's lecture and get ready for the next class