

# CPSC 322: Introduction to Artificial Intelligence

## Search: Introduction

Textbook reference: [3.1–3.4](#)

Instructor: Varada Kolhatkar  
University of British Columbia

Credit: These slides are adapted from the slides of previous offerings of the course. Thanks to all instructors for creating and improving the teaching material and making it available!

# Lecture outline

- **Recap from last lecture (~5 mins)** 
- Search motivation (~5 mins)
- Simple search agents and examples (~30 mins)
- Break (~5 mins)
- General search procedure (~25 mins)
- Summary and wrap-up (~5 mins)

# A rough CPSC 322 overview

Representation  
and reasoning

Static

Sequential

Problem

Constraint  
satisfaction

Query

Planning

Environment

Deterministic

Stochastic

Arc consistency

Variables +  
constraints

Search

Logics

Search

Belief networks

Variable elimination

STRIPS

Search

Decision networks

Variable elimination

Markov decision  
processes

Value iteration

# Recap: Choose the best answer

A deterministic agent

- A. knows what state it is in
- B. could predict the next state if it knew its current state and the action to be taken 
- C. is determined to get to its goal
- D. does not know exactly what state it is in

# Recap: How many states?

Mars explorer

Weather: {Sunny, Cloudy}

A possible state:  
{Sunny, -28, 320, 100}

Temperature: [-40, +40]

Longitude: [0, 359]

Latitude: [-1, 179]

How many total number  
of mutually exclusive states?



$$2 \times 81 \times 360 \times 179$$

# Questions?

# Today's class: Learning outcomes

By the end of the class you will be able to

- Define a directed graph.
- Represent a problem as a state-space graph.
- Assess the size of the search space of a given search problem.
- Trace through/implement a generic search algorithm.

# Lecture outline

- Recap from last lecture (~5 mins)
- **Search motivation** (~5 mins) 
- Simple search agents and examples (~30 mins)
- Break (~5 mins)
- General search procedure (~25 mins)
- Summary and wrap-up (~5 mins)

# Many problems can be solved by search

## Deepmind's AlphaGo wins 2nd game against Chinese Go champion

Ke Jie lost despite playing what Google's AlphaGo indicated was the best game any opponent has played against it.



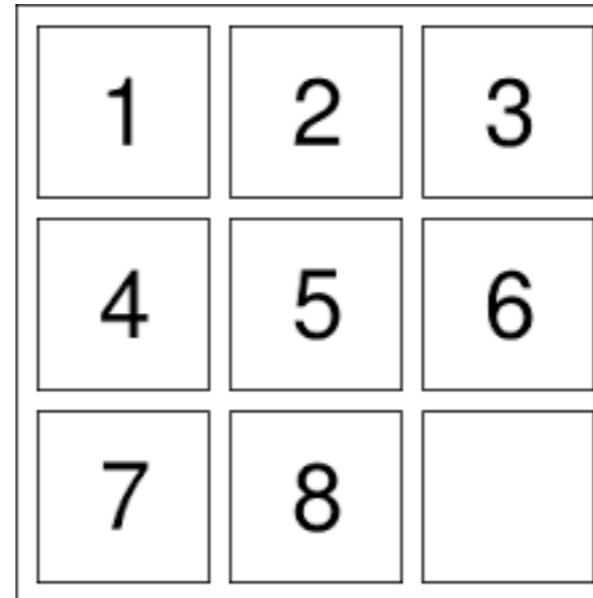
# More examples



Peg solitaire

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



8-puzzle



# Formulating a problem as search

- Often we are given a specification of what a solution is, but do not know how we would go about finding one.
- In that case we have to **search** for a solution.
  - Enumerate a set of potential partial solutions.
  - Check to see if they are solutions or could lead to one.
  - We can recognize a solution once we see one.

“Generate [potential [partial] solution] and test.”

# Why search?

- One of the most fundamental techniques in AI
  - Part of many AI systems
- Can solve problems that humans are not so good at solving
- Can achieve super-human performance on problems such as Chess and go
- A useful problem solving technique in AI as well in other areas
- Planning can be cast as a search problem

# Planning ahead as a search problem

- We want to build good agents that make good decisions
- In order to make good decisions we need to plan ahead
- What it means to have an agent that plans ahead?
  - Build an agent that thinks about the consequences of its actions so that we can use those consequences to make good decisions in the first place.
- Formalize this idea of planning ahead as a search problem

# Lecture outline

- Recap from last lecture (~5 mins)
- Search motivation (~5 mins)
- **Simple search agents and examples**  (~30 mins)
- Break (~5 mins)
- General search procedure (~25 mins)
- Summary and wrap-up (~5 mins)

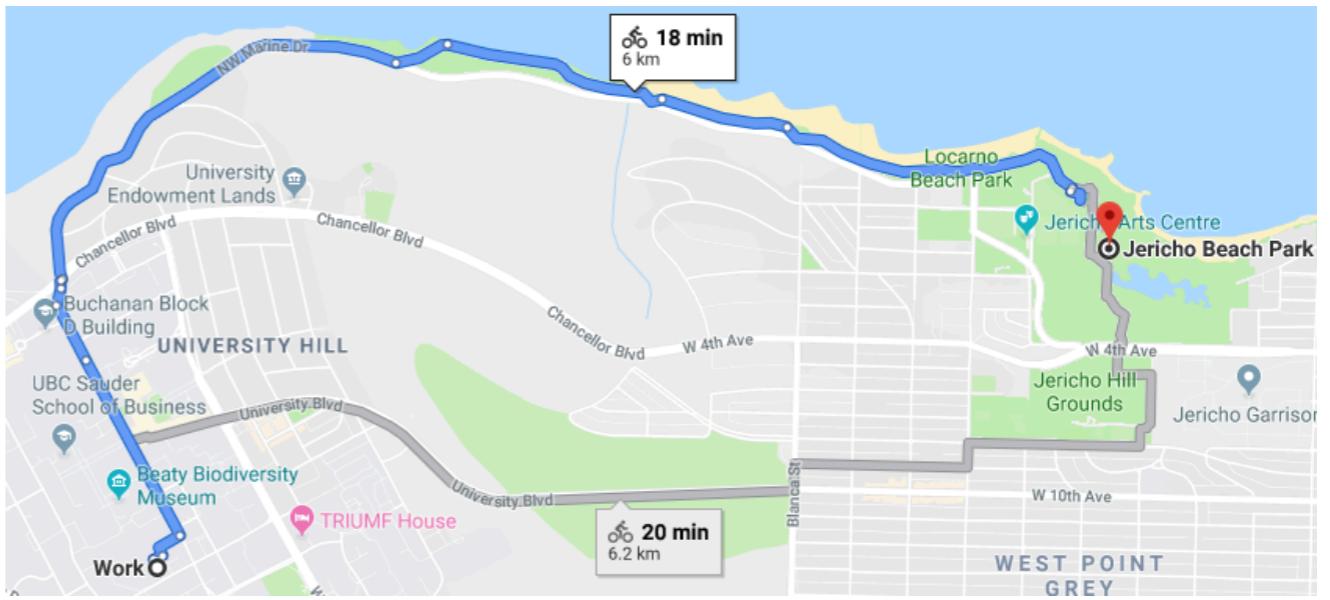
# Simple deterministic agent

- **Goal-driven deterministic** agent with
  - **Perfect knowledge** of the world
    - Using state-based world representation
    - Environment changes only when the agent acts, and in known ways
  - **Deterministic actions**
    - Agent perfectly knows actions that can be applied in any given state and the state it is going to end up in when an action is applied in a given state

# Simple deterministic agent

We assume that

- Agent is in a **start** state
- Agent is given a **goal** (subset of all states)
- A sequence of actions taking the agent from the start state to a goal state is a **solution** (a plan)



# Choose the correct answer

Our simple deterministic search agent has ...

- A. Perfect knowledge of its environment
- B. Perfect knowledge of the effect that its actions can have on the environment
- C. Both of the above 
- D. None of the above

# Definition of a search problem

- **State space:** set of all possible states
  - Not necessarily given explicitly (state space might be infinite)
- **Initial state(s)**
- **Set of actions (operators)** available to the agent: for each state and each operator, if operator applicable, defines the successor state: the state the agent would end up in
- **Goal state(s):** explicit set of states or predicate on states
- **Path cost** (we ignore this for now)

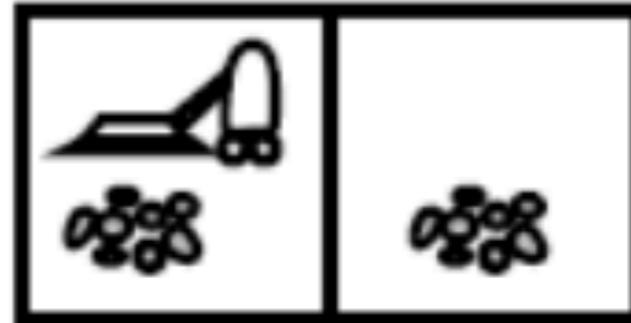
# Three examples

- Vacuum cleaner world
- Solving an 8-puzzle
- The delivery robot planning the route it will take in a bldg. to get from one room to another (Text: Section 1.6)

# Example 1: Vacuum cleaner world

- Two rooms:  $r_1, r_2$
- Operators: left, right, suck
- Each room can be either **dirty** or **clean**
- Vacuuming agent can be in either  $r_1$  or  $r_2$
- **How many total states?**

Possible start state

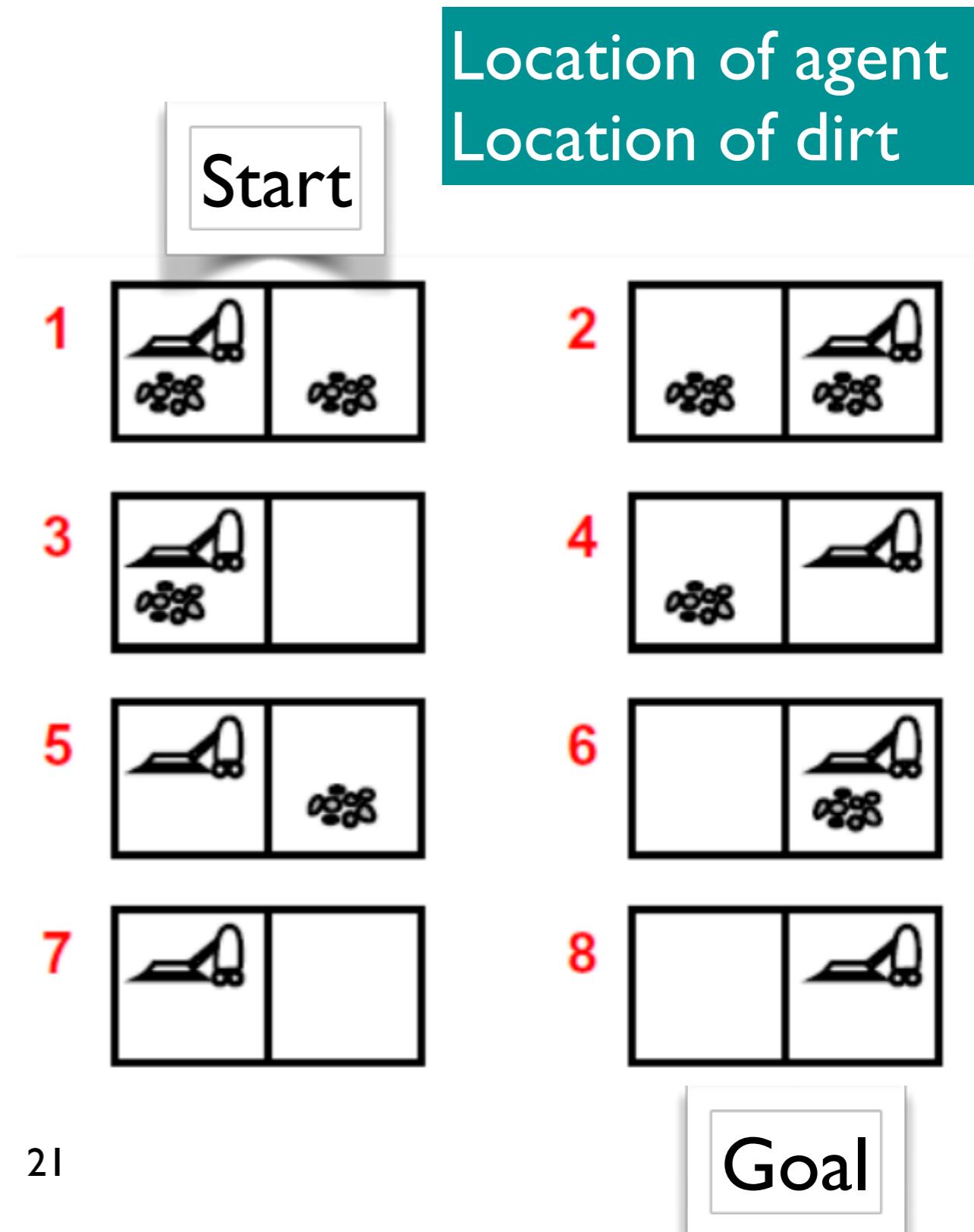


Possible goal state



# Example 1: Vacuum cleaner world

- Two rooms:  $r_1, r_2$
- Operators: left, right, suck
- Each room can be either **dirty** or **clean**
- Vacuuming agent can be in either  $r_1$  or  $r_2$
- **How many total states?**



# How many states?



Suppose we have the same problem with  $k$  rooms. The number of states is

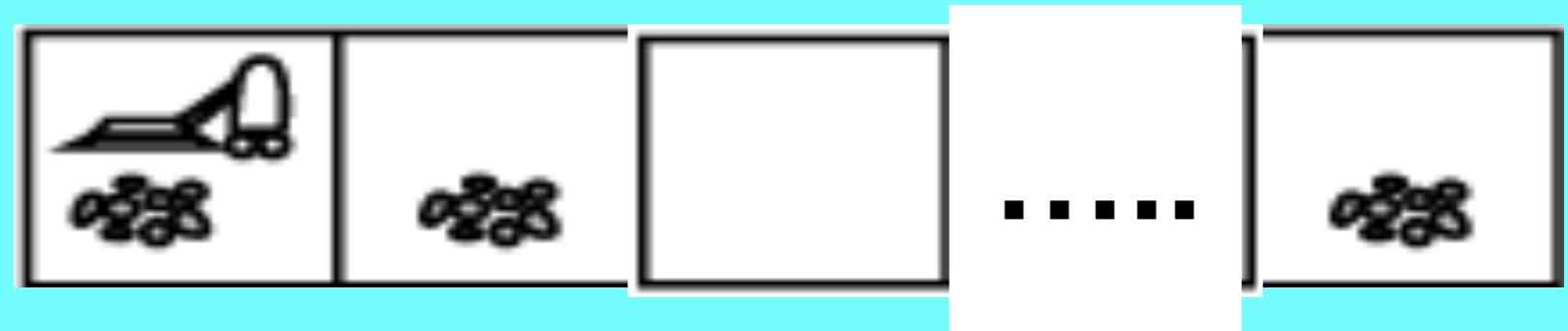
A.  $k^3$

B.  $k \times 2^k$

C.  $k \times 2k$

D.  $2 \times k^k$

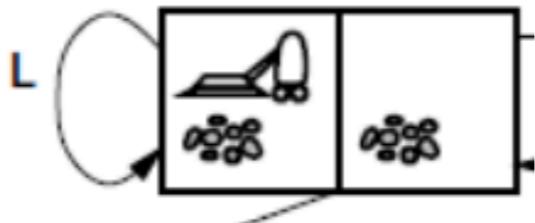
E. Never mind, I'll clean up the rooms myself.



# ASIDE: Think at leisure

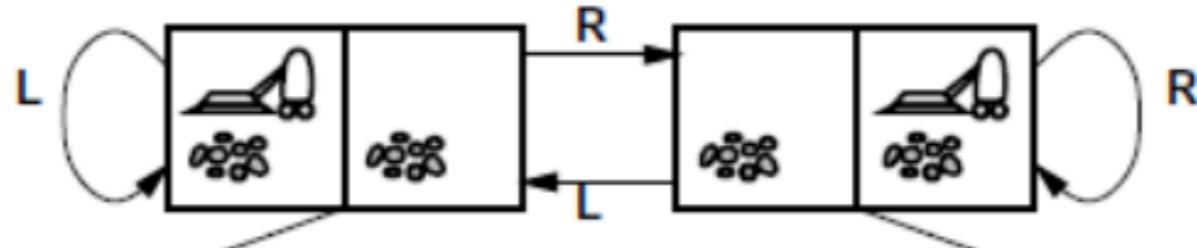
How many states if we had  $k$  rooms and a room can be either be dirty or swept or wiped (3 possible feature values instead of 2)?

# Search space



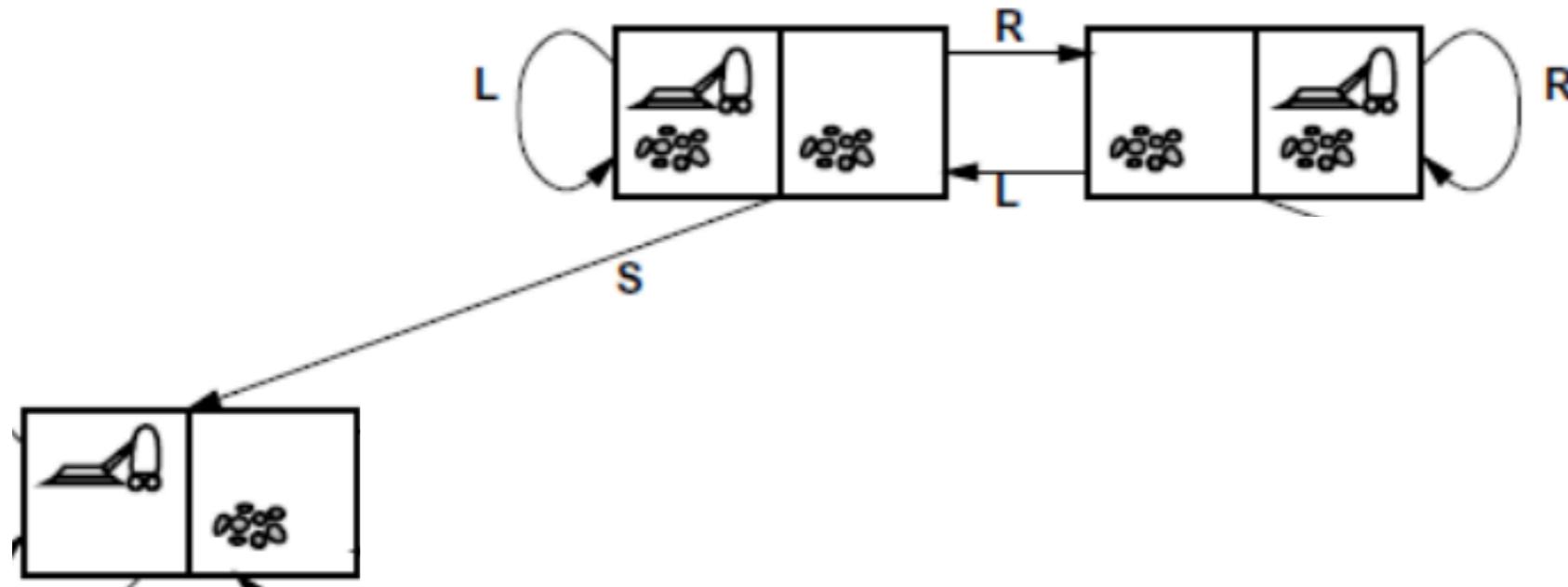
- Operators: left (L), right (R), suck (S)
- Successor states in the graph describe the effect of each action applied to a given state
- Possible Goal: no dirt

# Search space



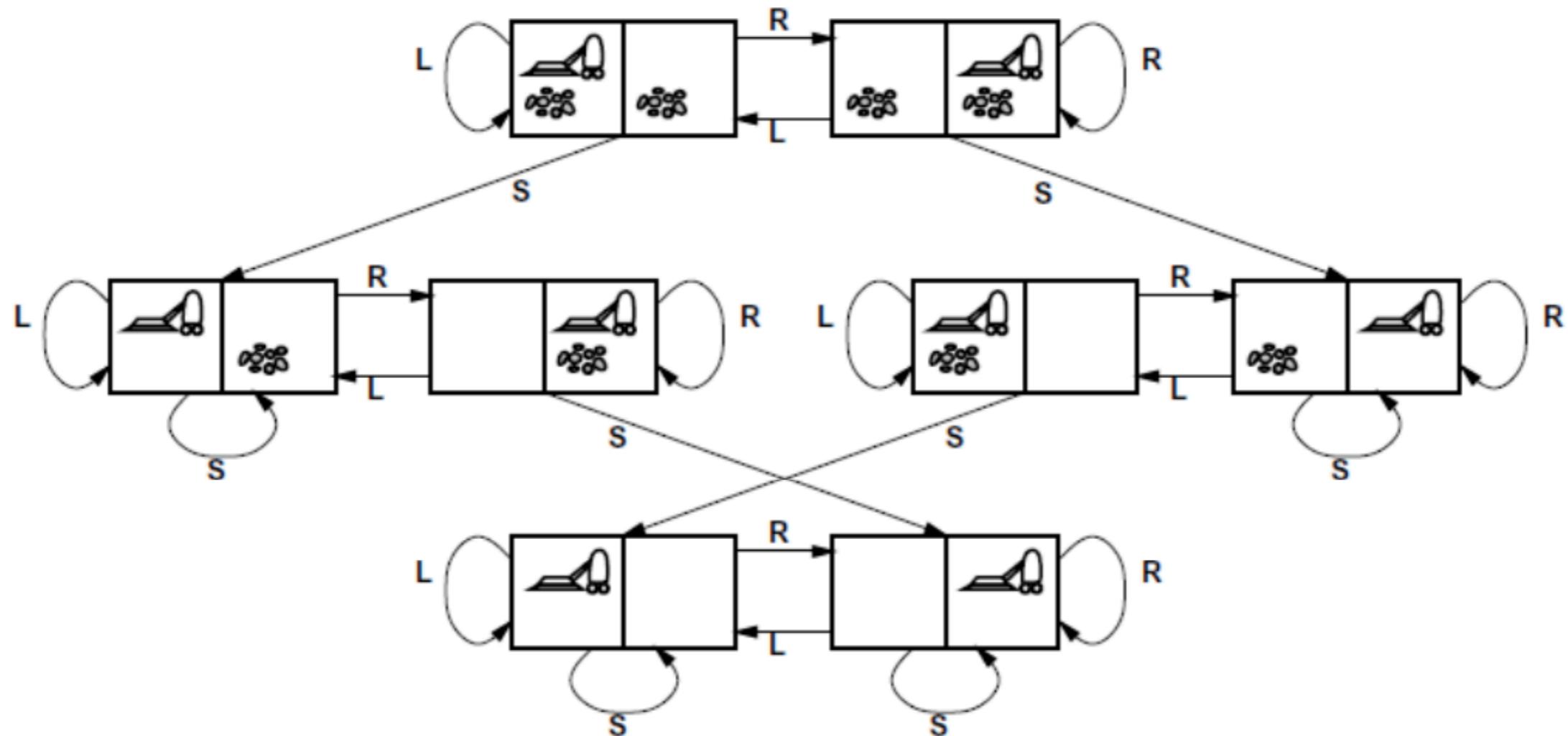
- Operators: left (L), right (R), suck (S)
- Successor states in the graph describe the effect of each action applied to a given state
- Possible Goal: no dirt

# Search space



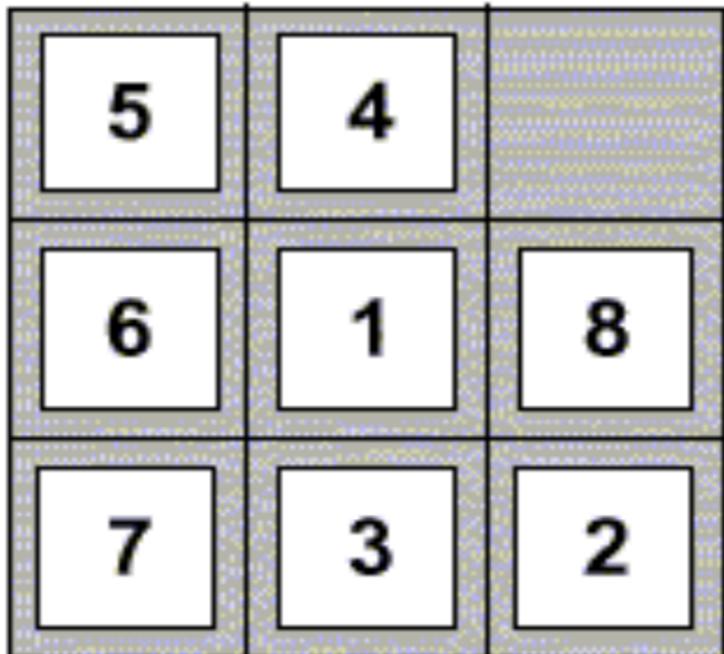
- Operators: left (L), right (R), suck (S)
- Successor states in the graph describe the effect of each action applied to a given state
- Possible Goal: no dirt

# Search space

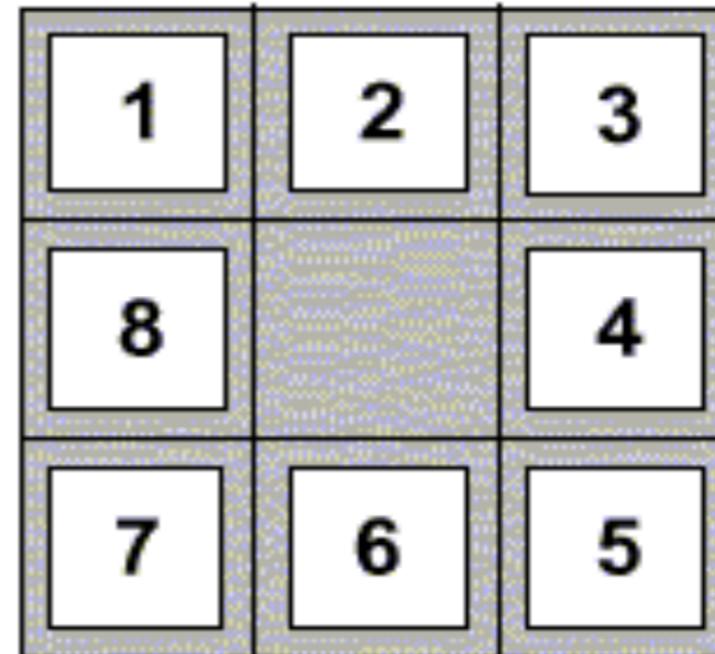


- Operators: left (L), right (R), suck (S)
- Successor states in the graph describe the effect of each action applied to a given state
- Possible Goal: no dirt

# Example 2: 8-puzzle



Start



Goal

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

**Actions:** blank moves left, right, up down

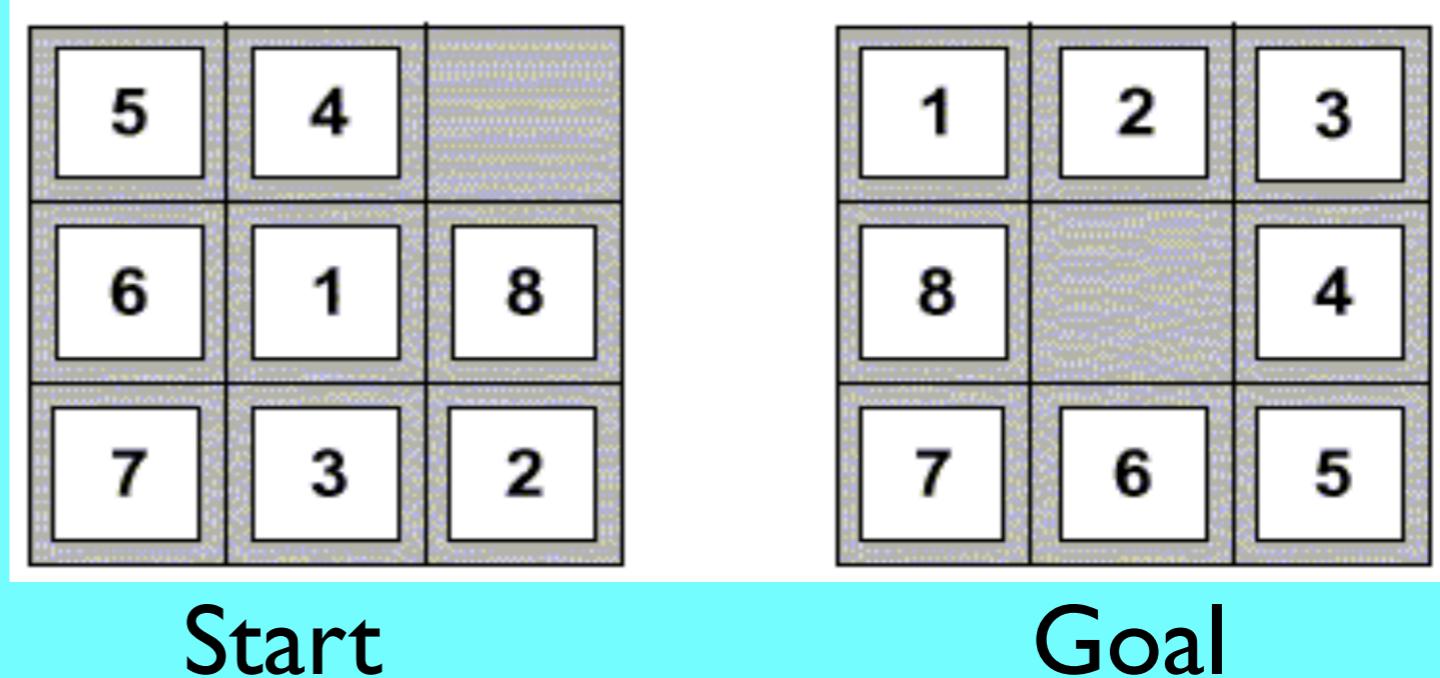
**Possible Goal:** configuration with numbers in right sequence

# Example 2: 8-puzzle



How many states?

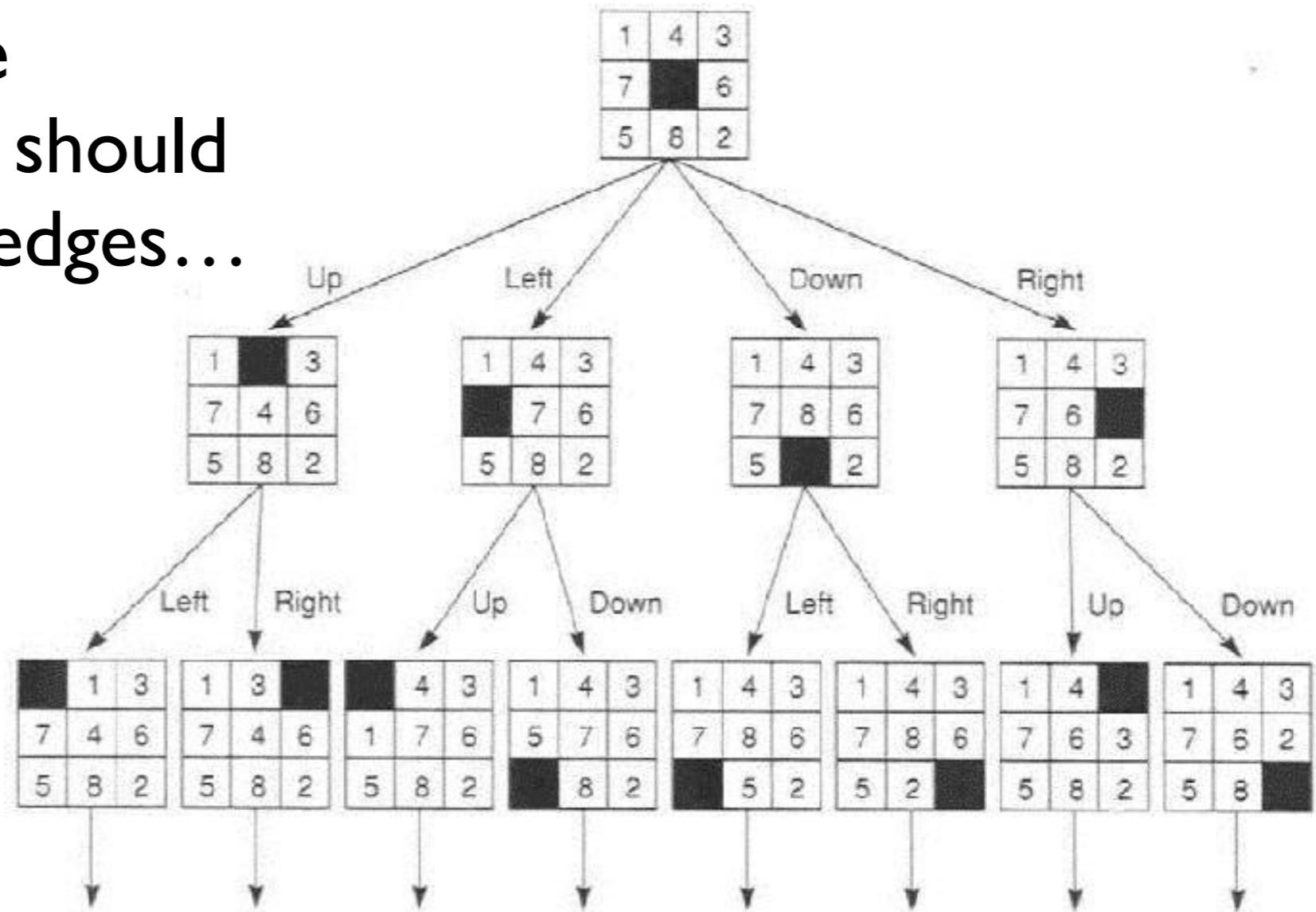
- A.  $8^9$
- B.  $2^9$
- C.  $9^9$
- D.  $9!$
- E. 42



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

# Search space for 8-puzzle

- Only a tiny fraction of the search space
- Each action can be reversed, so there should be twice as many edges...

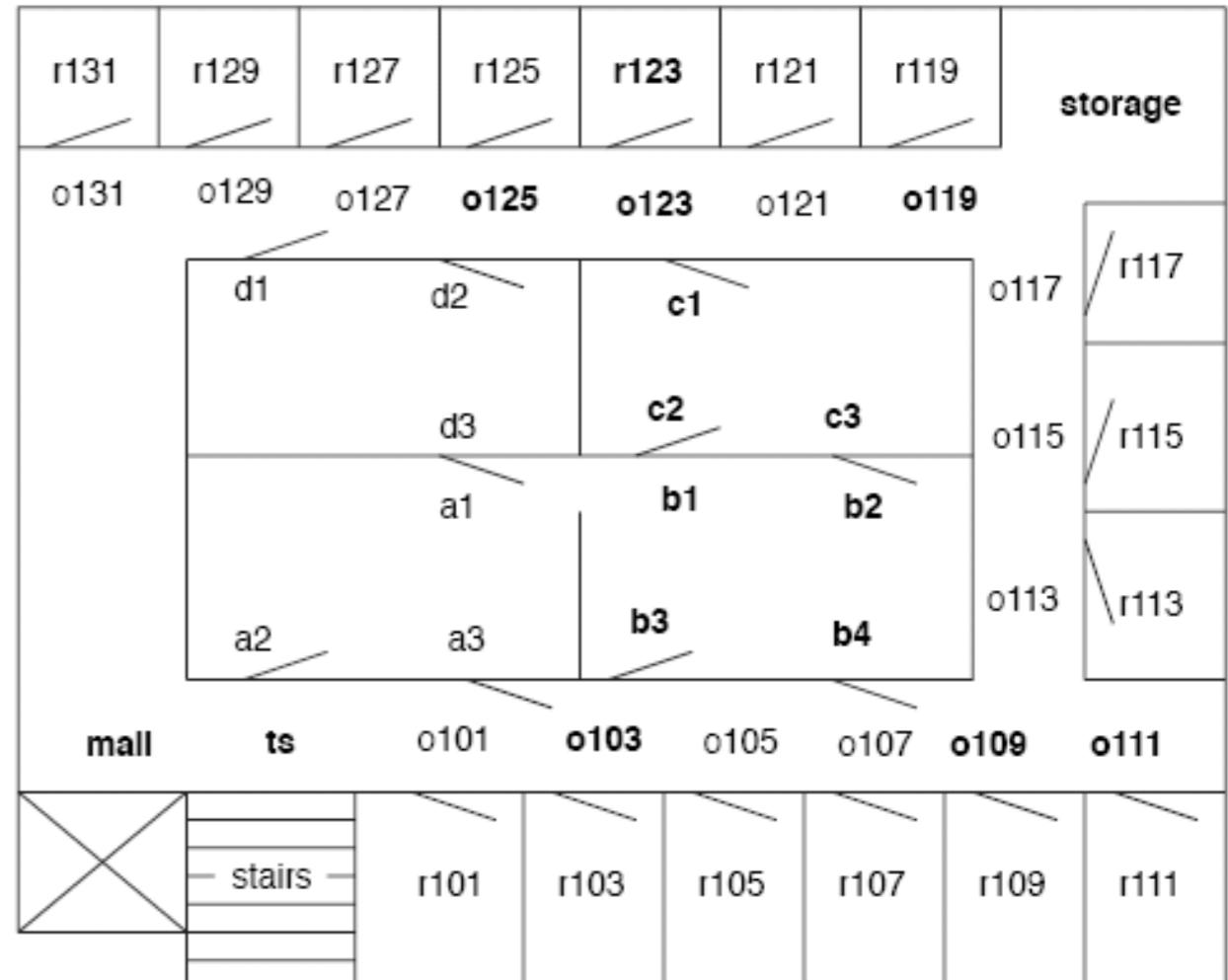


# Example 3: Delivery robot



Simplified version

- Consider only **bold** locations
- Limits in direction of movement (can move only in the direction doors open)

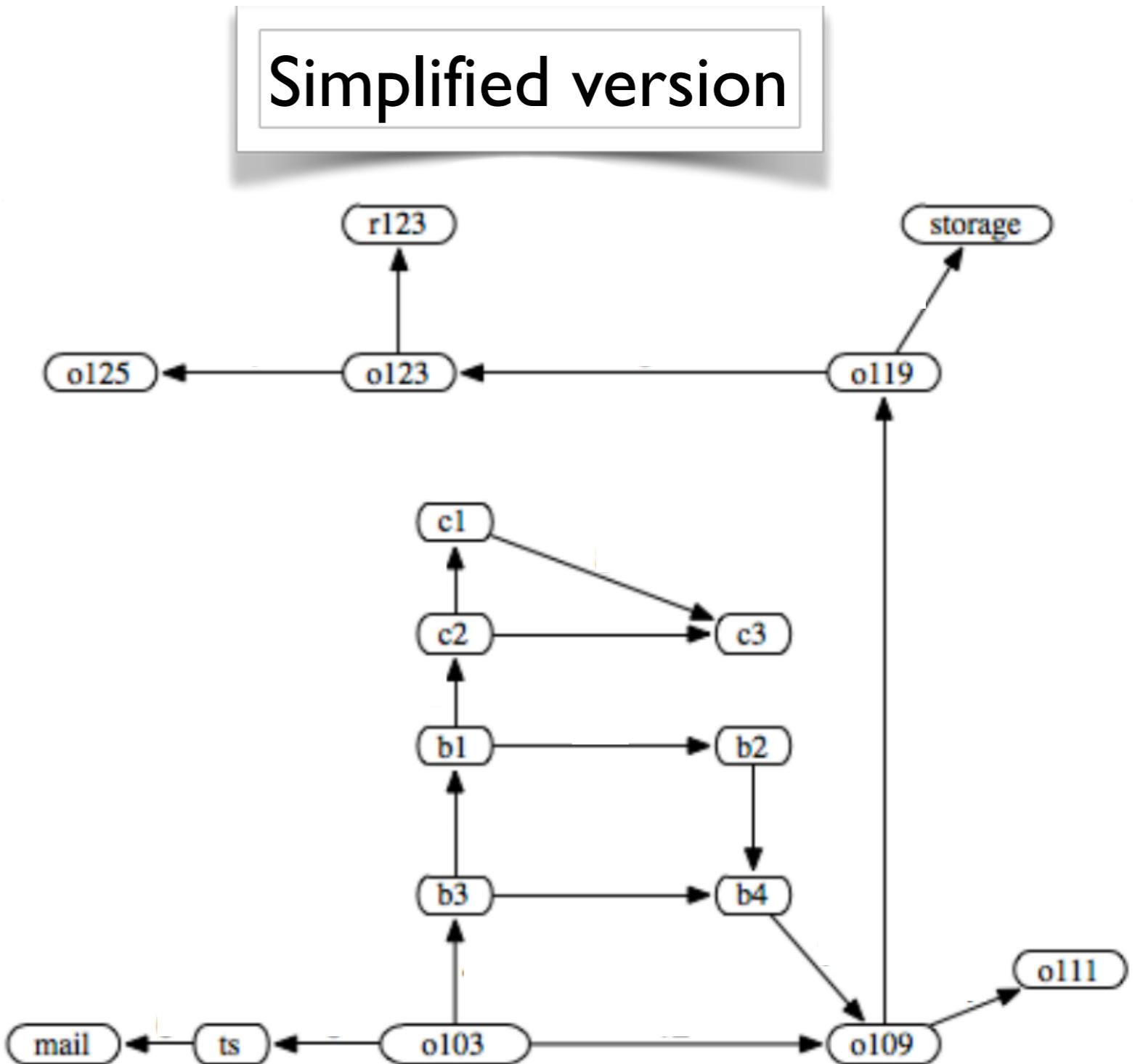


# Example 3: Delivery robot

- How can we find a sequence of actions and their appropriate ordering that lead to the goal?
- Define underlying search space graph where nodes are states and edges are actions.

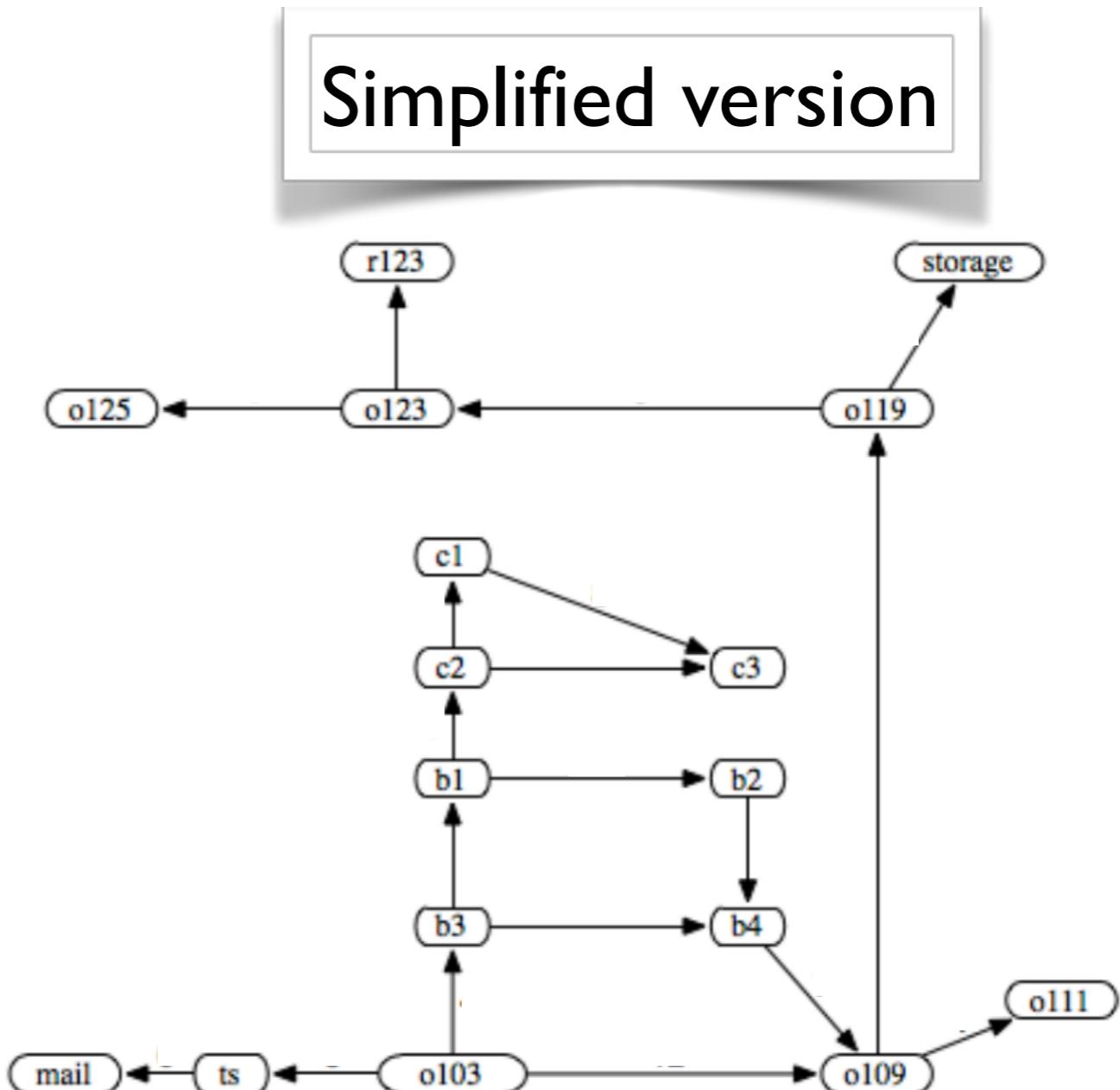
# Example 3: Delivery robot

- Start: o103
- Goal: r123
- Solution: ?



# Example 3: Delivery robot

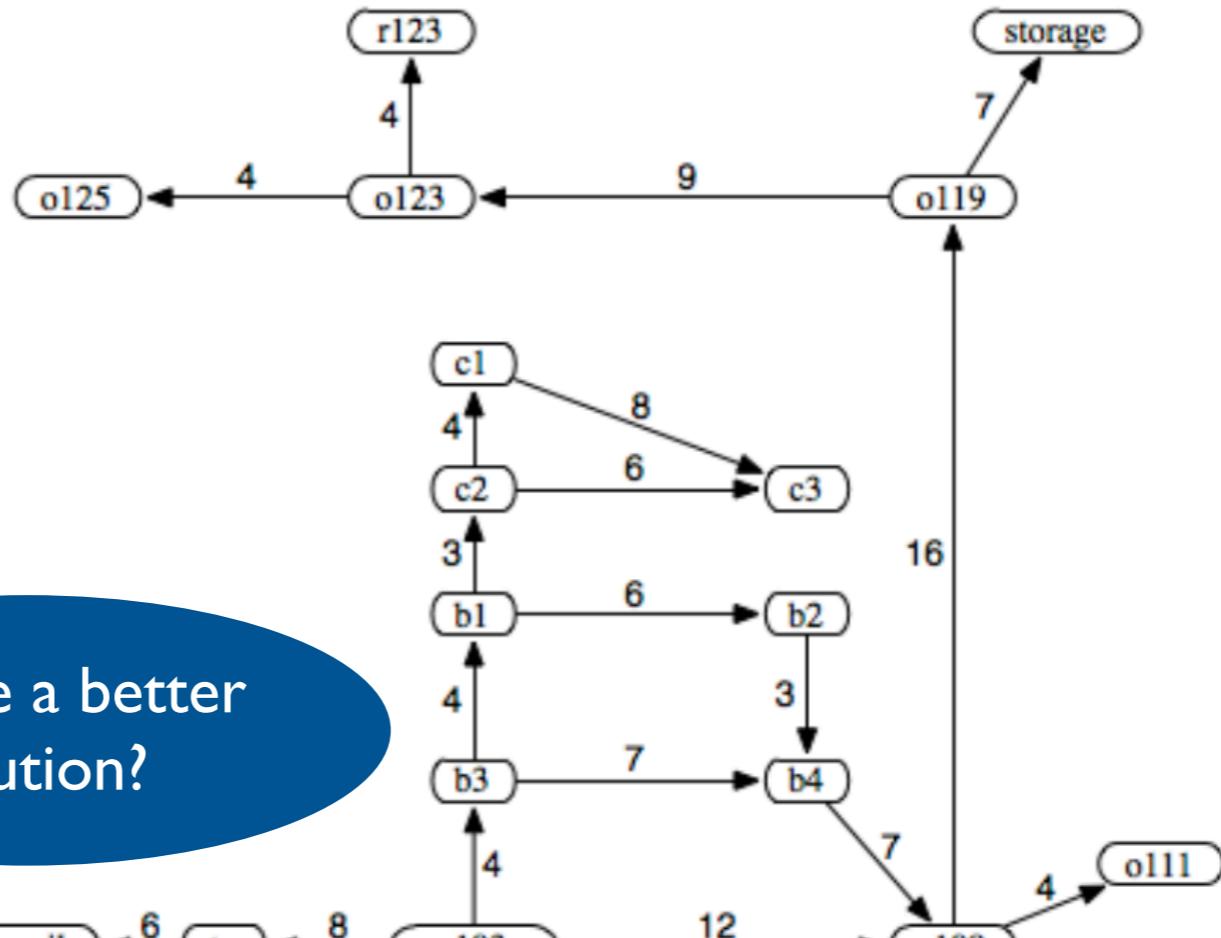
- Start: o103
- Goal: r123
- A solution:



$o103 \rightarrow b3 \rightarrow b4 \rightarrow o109 \rightarrow o119 \rightarrow o123 \rightarrow r123$

# Example 3: Delivery robot

Simplified version with costs



- Start: o103
- Goal: r123
- A solution:

Is there a better solution?

$\text{o103} \rightarrow \text{b3} \rightarrow \text{b4} \rightarrow \text{o109} \rightarrow \text{o119} \rightarrow \text{o123} \rightarrow \text{r123}$

# Lecture outline

- Recap from last lecture (~5 mins)
- Search motivation (~5 mins)
- Simple search agents and examples (~30 mins)
- Break (~5 mins)
- **General search procedure** (~25 mins) 
- Summary and wrap-up (~5 mins)

# Search abstract definition

## How to search

- Start at the start state
- Evaluate which actions can lead us from states that have been encountered in the search so far to new states
- Stop when a goal state is encountered

To make this more formal, we'll need to review the formal definition of a directed graph

# Directed graphs

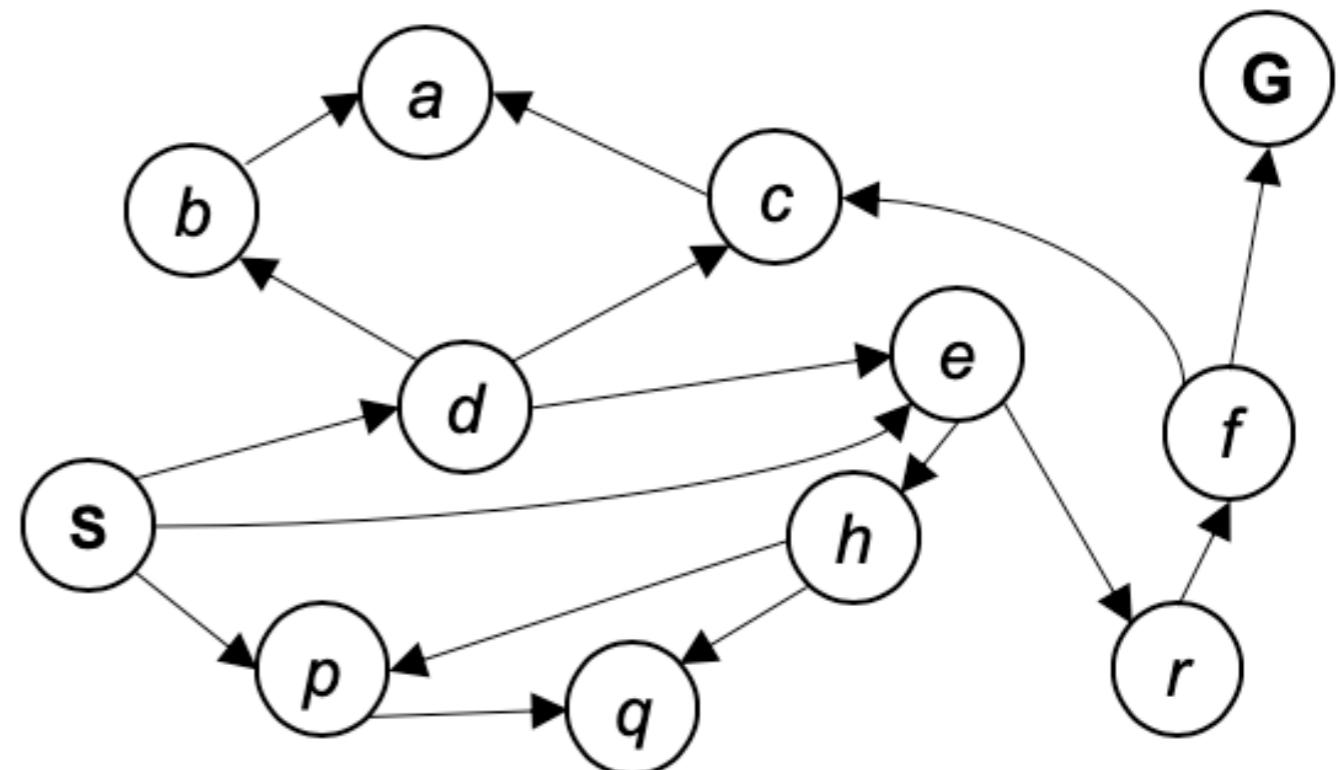
- A **directed graph** consists of
  - a set  $N$  of nodes (vertices)
  - a set  $A$  of ordered pairs of nodes, called edges (arcs).
- Node  $n_2$  is a **neighbour** of  $n_1$  if there is an arc from  $n_1$  to  $n_2$ . That is, if  $\langle n_1, n_2 \rangle \in A$
- A **path** is a sequence of nodes  $n_0, n_1, n_2, \dots, n_k$  such that  $\langle n_{i-1}, n_i \rangle \in A$ 
  - Start node =  $n_0$
  - End node =  $n_k$
- A **cycle** is a non-empty path such that the start node is the same as the end node.

# Directed graph

- A **path** is a sequence of nodes  $n_0, n_1, n_2, \dots, n_k$  such that  $\langle n_{i-1}, n_i \rangle \in A$ 
  - Start node =  $n_0$
  - End node =  $n_k$
- A **cycle** is a non-empty path such that the start node is the same as the end node.

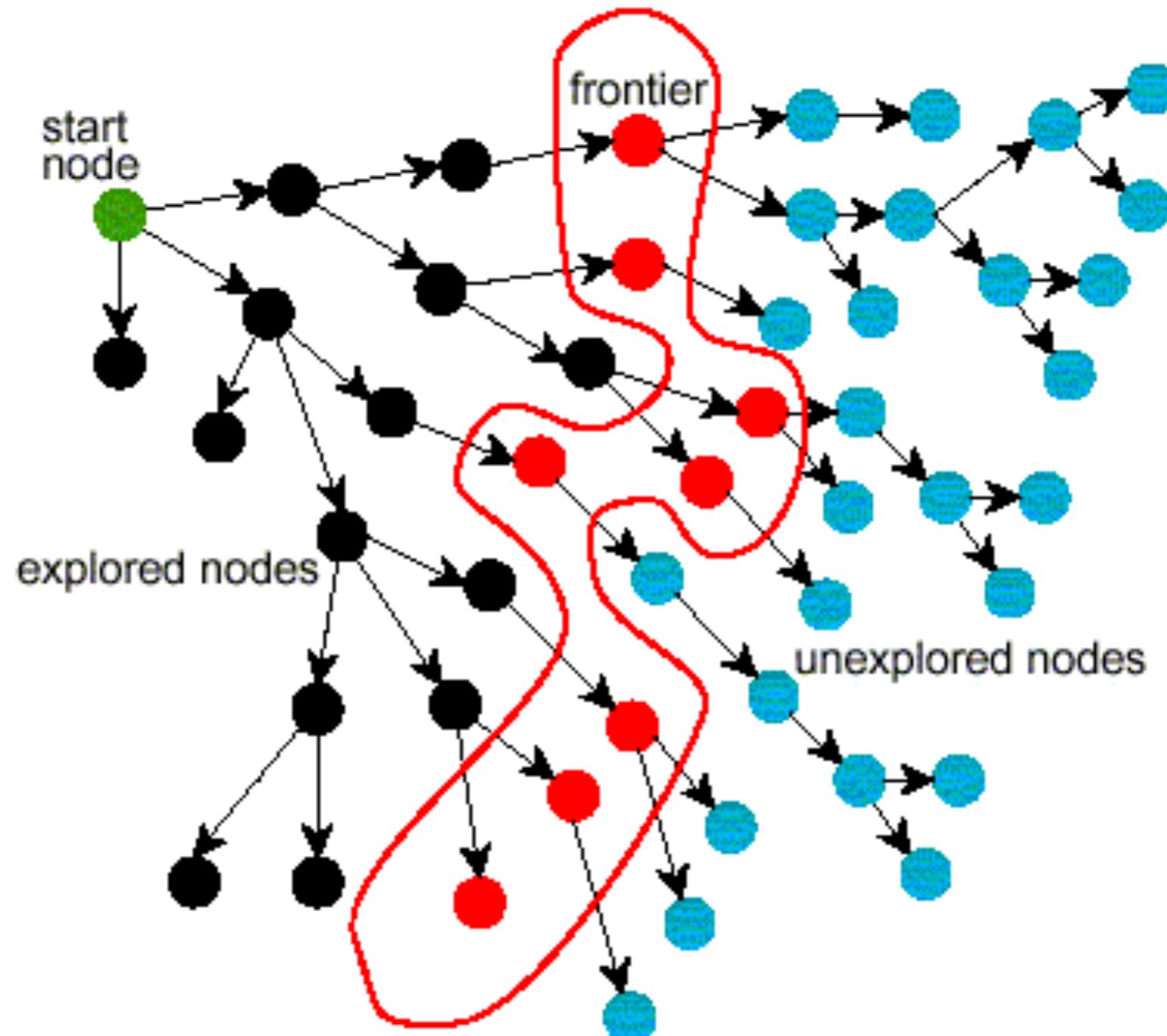
# Search graphs

- Nodes are search states
- Edges correspond to actions
- Given a set of start nodes and goal nodes, a **solution** is a path from a start node to a goal node: a **plan of actions**



Credit: Graph picture from [Berkeley AI course material](#)

# Problem solving by graph searching



# Branching factor



- **The forward branching factor** of a node is the number of arcs going out of the node.
- **The backward branching factor** of a node is the number of arcs going into the node.
- If the forward branching factor of any node is  $b$  and the graph is a tree, how many nodes are  $m$  steps away from a node?
  - A.  $mb$
  - B.  $m/b$
  - C.  $b^m$
  - D.  $m^b$

# 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 from the start node that have been explored.
- 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.**

# Generic search algorithm

**Inputs:** a graph,

a start node  $n_0$

a boolean procedure  $goal(n)$  that tests if  $n$  is a goal node

frontier:= [  $< n_0 >$ :  $n_0$  is a start node];

**While** frontier is not empty:

**select** and **remove** path  $< n_0, n_1, \dots, n_k >$  from frontier;

If  $goal(n_k)$

**return**  $< n_0, n_1, \dots, n_k >$  ;

**For every** neighbour  $n$  of  $n_k$

**add**  $< n_0, n_1, \dots, n_k, n >$  to frontier;

**return** NULL

# Activity: Generic search algorithm

Trace the algorithm on the graph below  
Start node: S  
Goal: X

**Inputs:** a graph,

a start node  $n_0$

a boolean procedure  $goal(n)$  that tests if  $n$  is a goal node

frontier:= [  $< n_0 >$ :  $n_0$  is a start node];

**While** frontier is not empty:

**select** and **remove** path  $< n_0, n_1, \dots, n_k >$  from frontier;

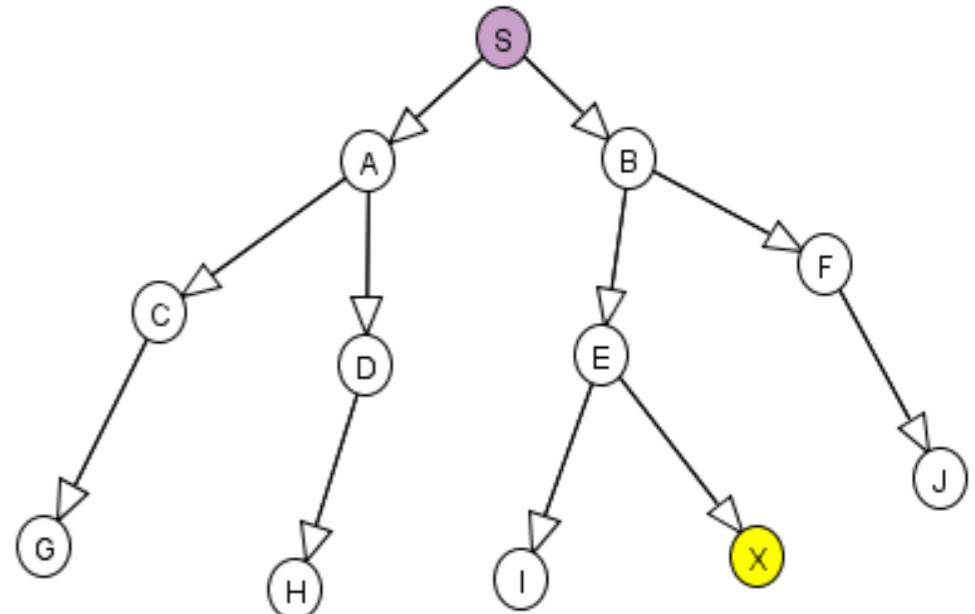
**If**  $goal(n_k)$

**return**  $< n_0, n_1, \dots, n_k >$  ;

**For every** neighbour  $n$  of  $n_k$

**add**  $< n_0, n_1, \dots, n_k, n >$  to frontier;

**return** NULL



# First few iterations of the algorithm

Assuming the frontier is a queue

iteration	frontier	selected & removed
1	$\langle s \rangle$	$\langle s \rangle$ goal(s)? False
2	$\langle s, A \rangle, \langle s, B \rangle$	$\langle s, A \rangle$ goal(A)? False
3	$\langle s, B \rangle, \langle s, A, C \rangle$ $\langle s, A, D \rangle$	$\langle s, B \rangle$ goal(B)? False
	$\vdots$	
	$\vdots$	
	$\langle s, B, E, X \rangle, \langle s, B, F, J \rangle$	$\langle s, B, E, X \rangle$ goal(X)? <u>True</u>

# Summary

- Search is a key computational mechanism in many AI agents
- We will study the basic principles of search on the simple deterministic goal-driven search agent model in state-based world representation
- Generic search approach:
  - Define a search space graph
  - Initialize the frontier with an empty path
  - Incrementally expand frontier until goal state is reached
- **The way in which the frontier is expanded defines the search strategy**

# Coming up

## Uninformed search

- Depth-first search
- Breadth-first search

Textbook reference: [3.5.1, 3.5.2]

