Search: Intro

CPSC 322 Lecture 4

R&R systems we'll cover in this course

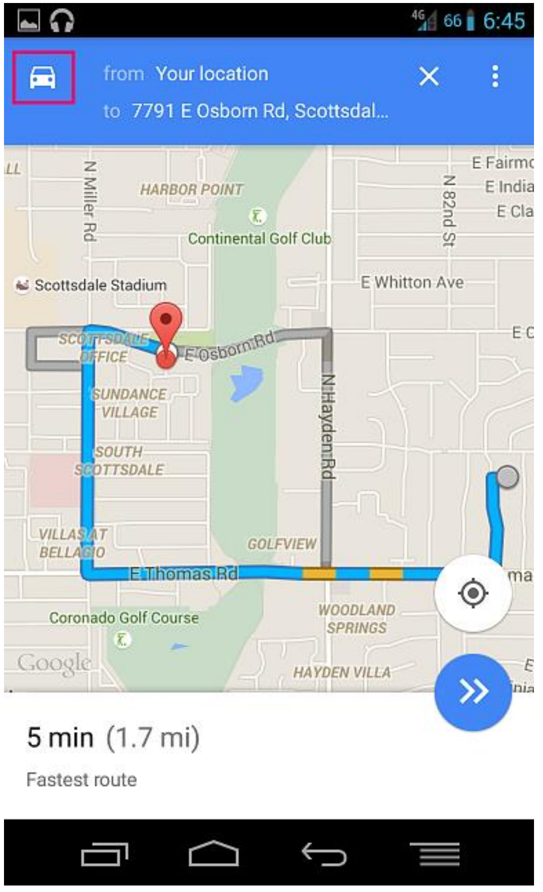
		Environment	
Problem		Deterministic	Stochastic
Static	Constraint Satisfaction	*Variables + Constraints* Search Arc Consistency Local Search	
	Query	*Logics* Search	*Bayesian (Belief) Networks* Variable Elimination
Sequential	Planning	*STRIPS* Search	*Decision Networks* Variable Elimination

Representation
Reasoning Technique

First part of course

What is search?

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



Intelligence is search

Search is to be avoided

Learning Goals for this class

- **Identify** real world examples that make use of deterministic, goal-driven planning agents
- **Assess** the size of the search space of a given search problem.
- **Trace through/implement** a **generic search algorithm**.

Lecture Overview

- **Simple Agent and Examples**
- Search Space Graph
- Search Procedure

Simple Planning Agent

Deterministic, goal-driven agent

- 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:
  - **what actions 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 (in the proper order) is the **solution**

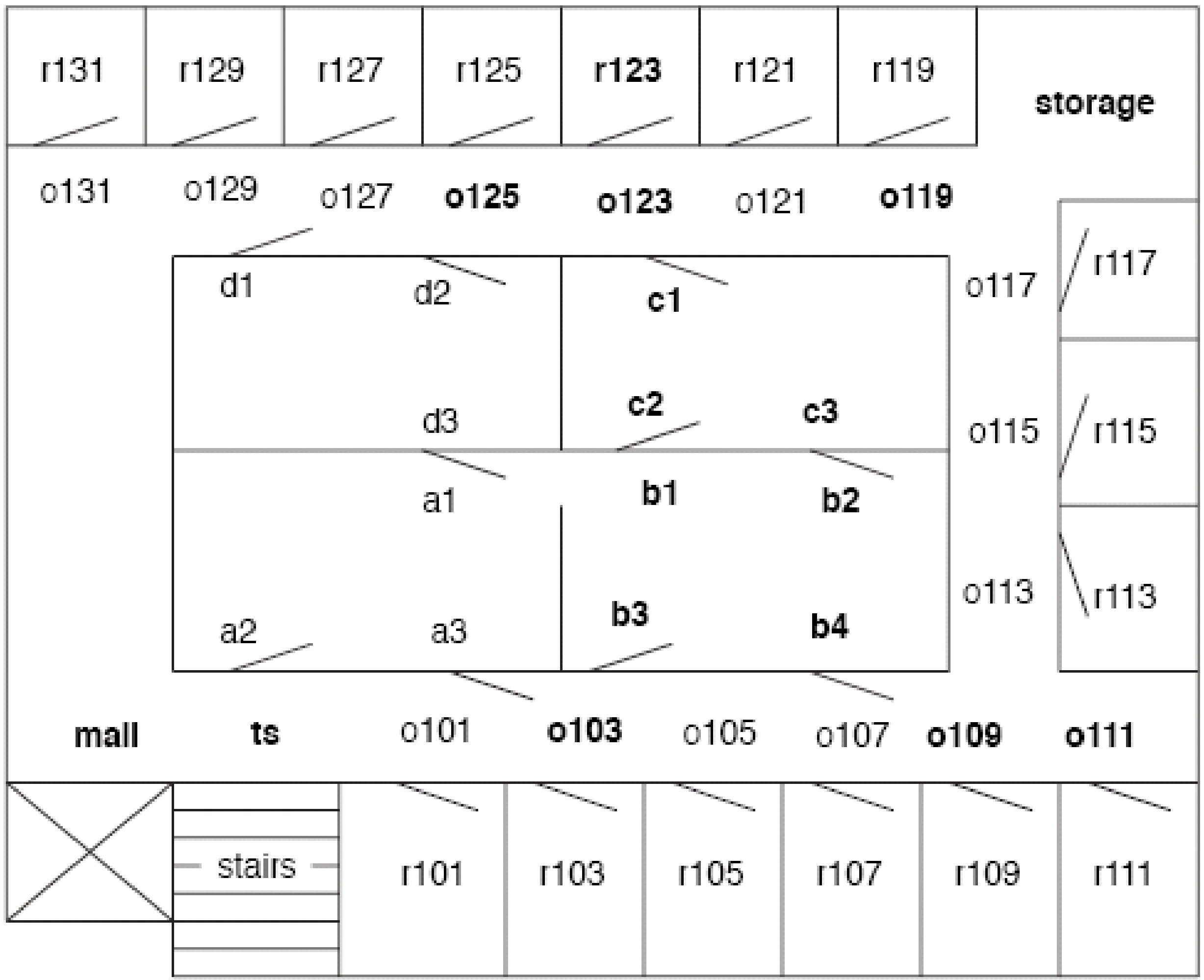
Three examples

1. A **delivery robot** planning the route it will take to get from one room to another (same building)

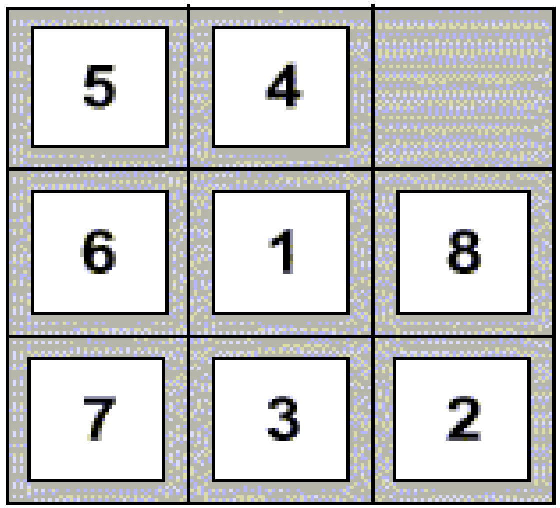
2. Solving an **8-puzzle**

3. Automated **vacuum cleaner**

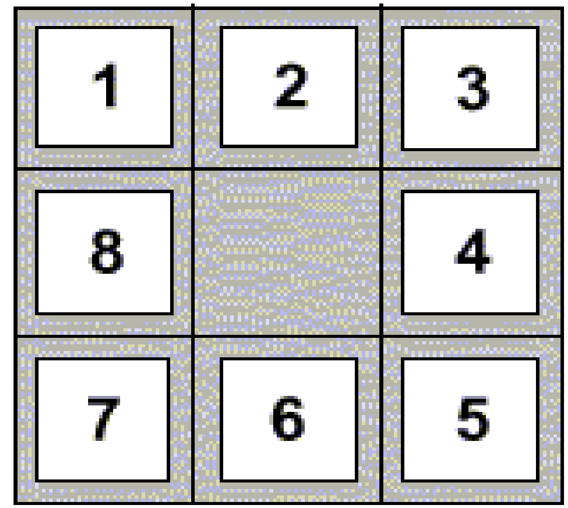
Example1: Delivery Robot



Eight Puzzle



Start State



Goal State

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

HOW MANY STATES ?

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

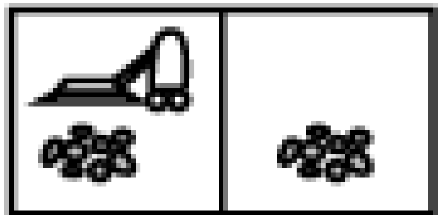
Actions: blank moves left, right, up down

Possible Goal: configuration with numbers in right sequence

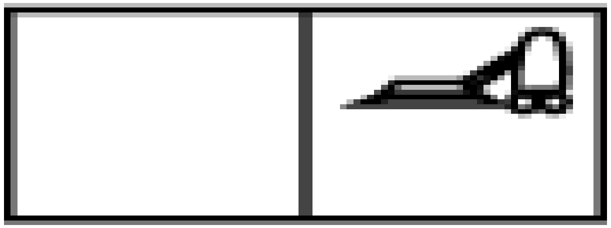
Example: vacuum world

States

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



Possible start state

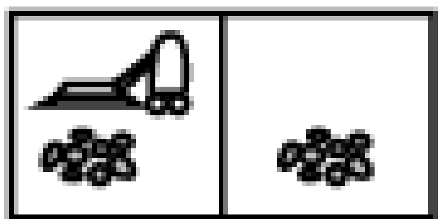


Possible goal state

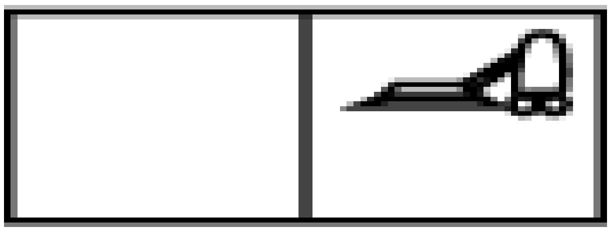
Example: vacuum world

States

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



Possible start state



Possible goal state

How many possible states?

Example: vacuum world

Suppose we have the same problem with ***k*** rooms.

The number of states is….

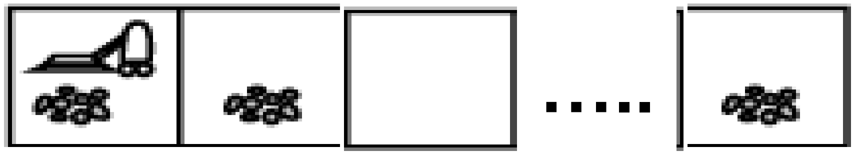
A. k^3

B. $k * 2k$

C. $k * 2^k$

D. $2 * k^k$

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

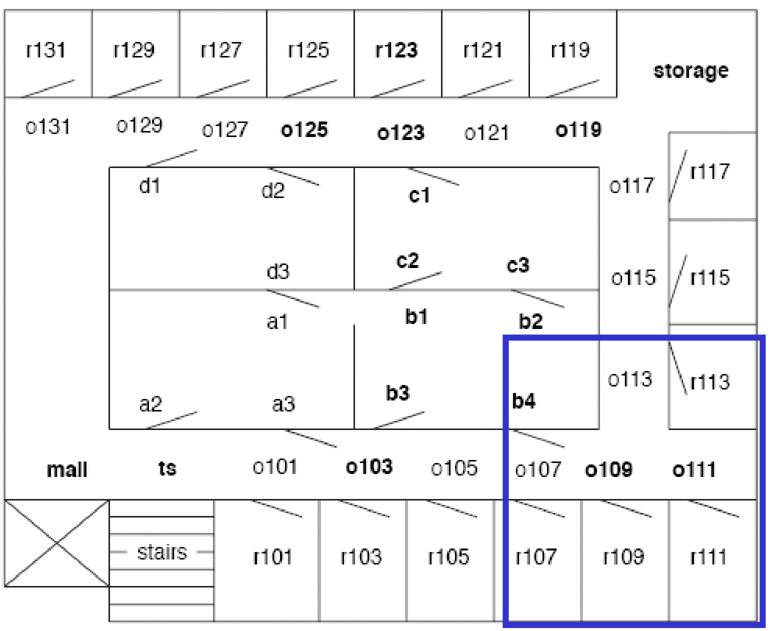


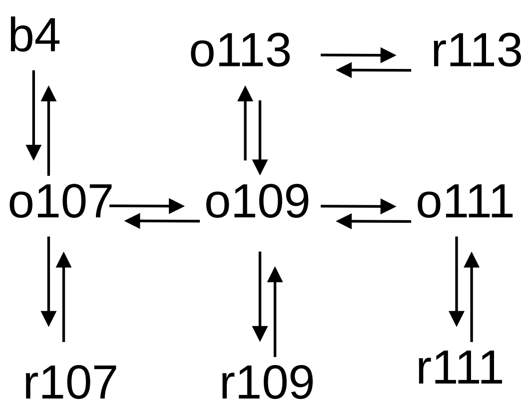
Lecture Overview

- **Simple Agent and** Examples
- Search Space Graph
- Search

How can we find a solution?

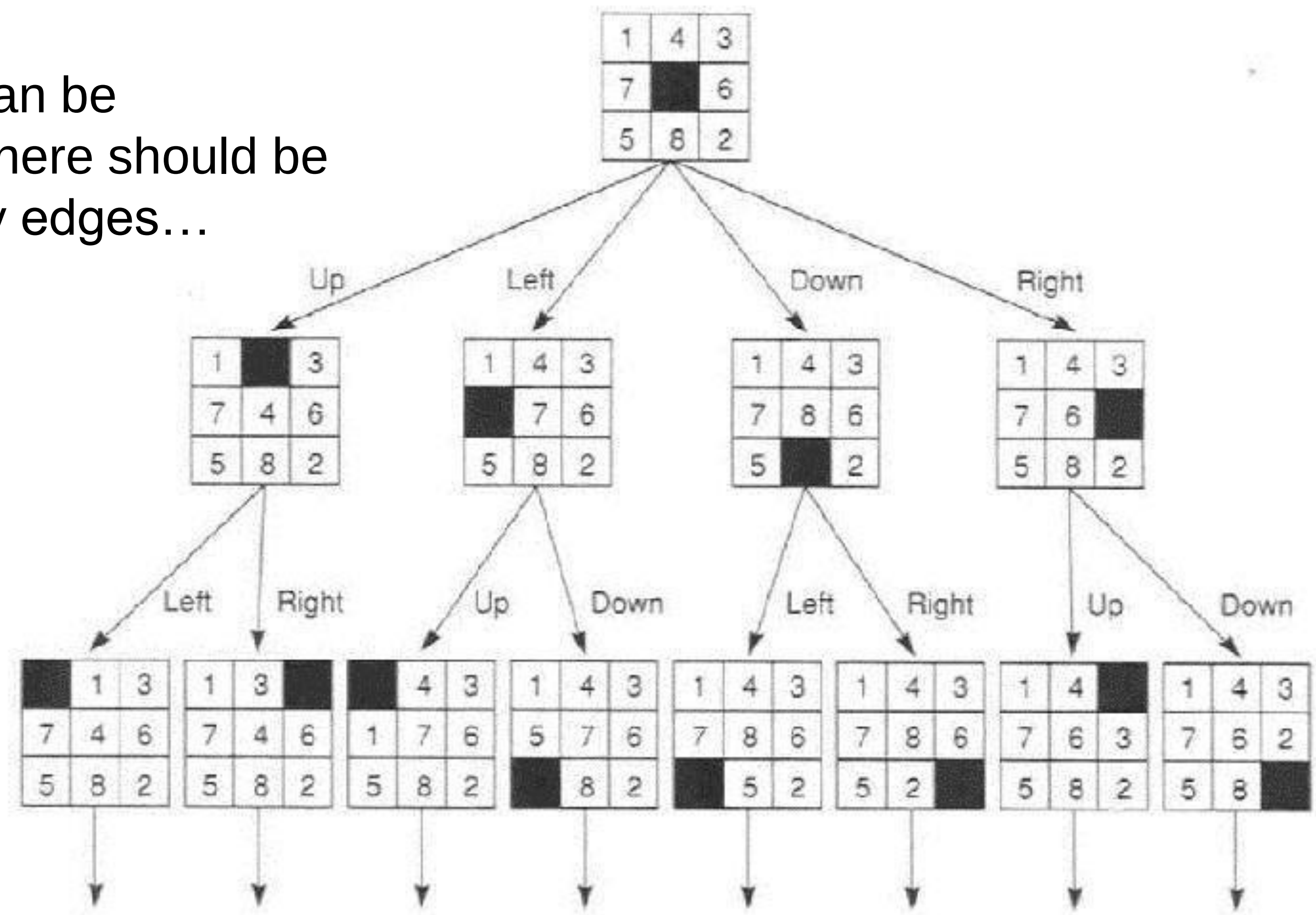
- How can we find a sequence of actions and their appropriate ordering that lead to the goal?
- Define underlying **search space graph** where **nodes are states** and **edges are actions**.



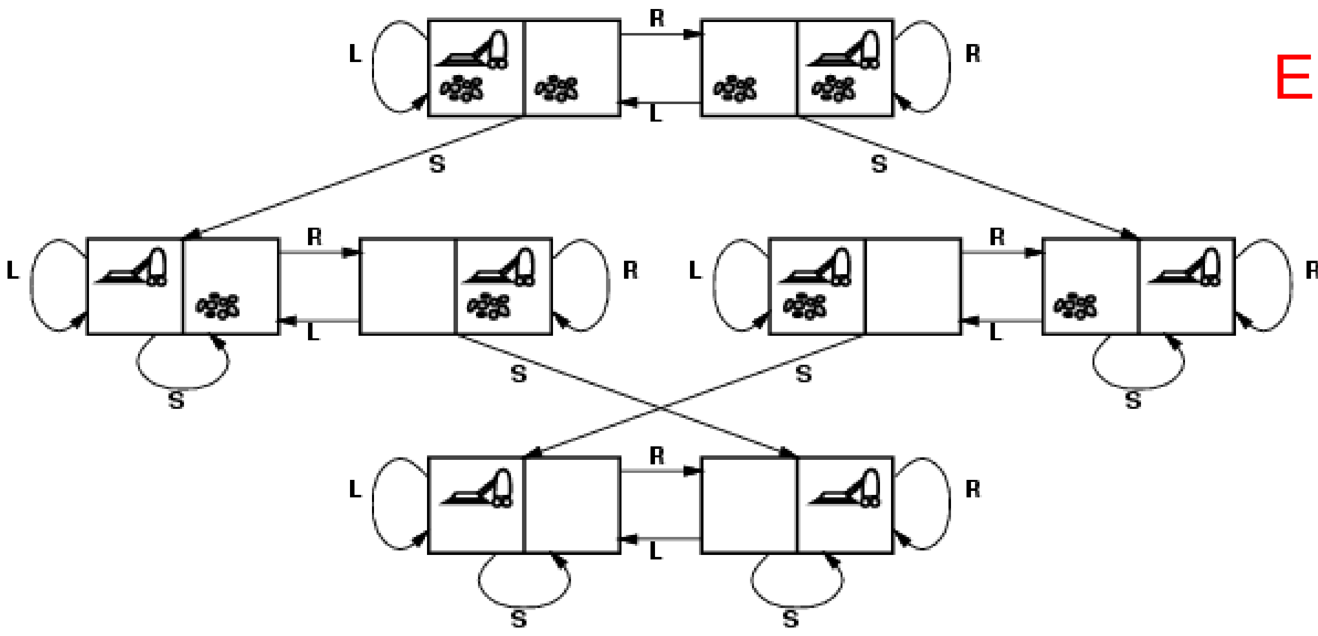


Search space for 8puzzle

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



Vacuum world: Search space graph



Entire search space

states? Where it is dirty and robot location

actions? *Left*, *Right*, *Suck*

Possible goal test? no dirt at all locations

Lecture Overview

- **Simple Agent and** Examples
- State Space Graph
- **Search Procedure**

Search: Abstract Definition

How to search

- Start at the start state
- Consider 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 need to review the **formal definition of a graph**...

Search Graph

A ***graph*** consists of a set *N* of ***nodes*** and a set *A* of ordered pairs of nodes, called ***arcs***.

Node n_2 is a ***neighbor*** 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 $n_0, n_1, n_2, \ldots, n_k$ such that $\langle n_{i-1}, n_i \rangle \in$ A.

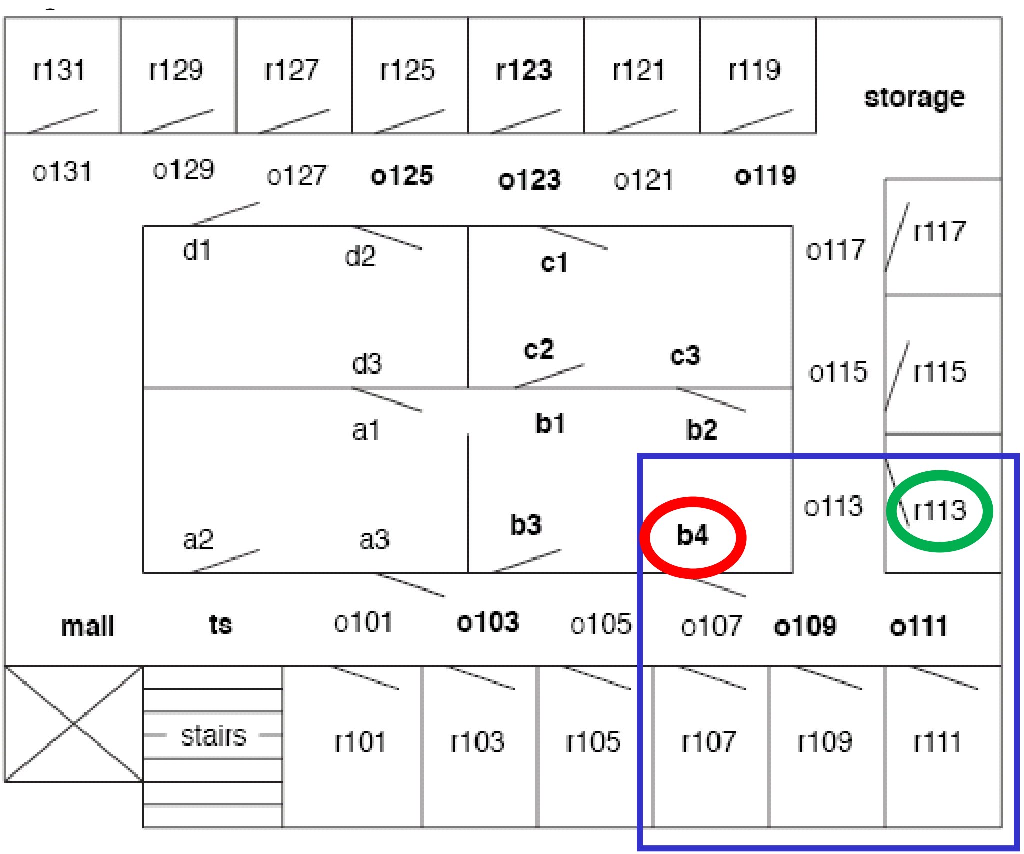
A ***cycle*** is a path with two or more nodes such that the start node is the same as the end node.

A ***directed acyclic graph*** (DAG) is a graph with no cycles and where the arcs have associated directions.

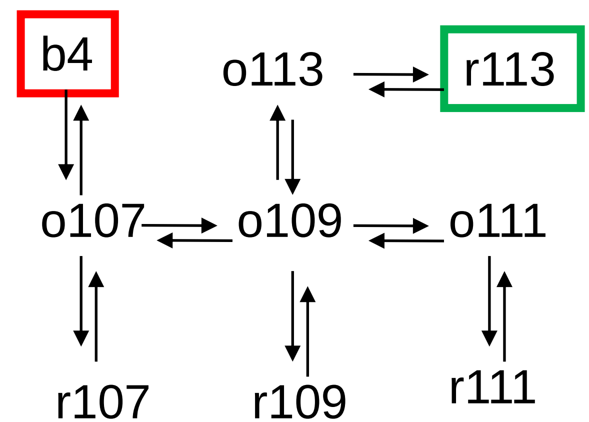
Given a start node and goal nodes, a ***solution*** is a path from a start node to a goal node.

Examples of solutions

- Start state **b4**, goal **r113**
- Solution <b4, o107, o109, o113, r113>



But there are many others!



Generic Search Algorithm

Inputs: a graph, a start node n_o, Boolean procedure *goal(n)* that tests if *n* is a goal node
frontier:= [<n_o>: n_o 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;*
return NULL

In-class activity: figure out the algorithm

- What is ***frontier***?
- How can you get **different kinds** of search?

Graph Searching

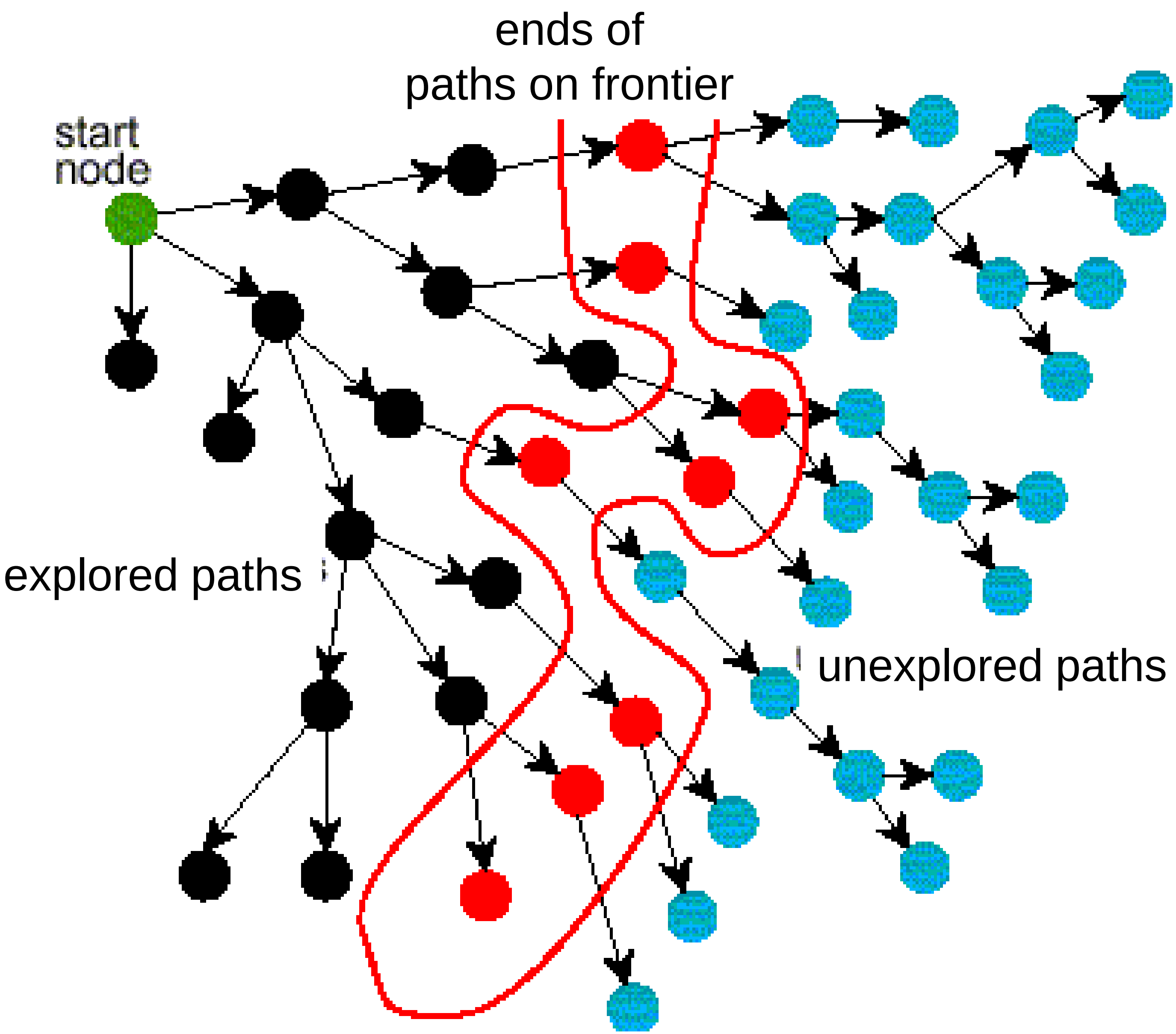
Generic search algorithm: given a graph, start node, and goal node(s), incrementally explore **paths** from the start node(s).

Maintain a **frontier of paths** from the start node that have been explored.

As search proceeds, the frontier expands into the unexplored nodes until (hopefully!) a goal node is encountered.

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

Problem Solving by Graph Searching



Generic Search Algorithm

Inputs: a graph, a start node n_o, Boolean procedure *goal(n)* that tests if *n* is a goal node

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

While *frontier* is not empty:

select and **remove** path $\langle n_o, \ldots, n_k \rangle$ from *frontier;*

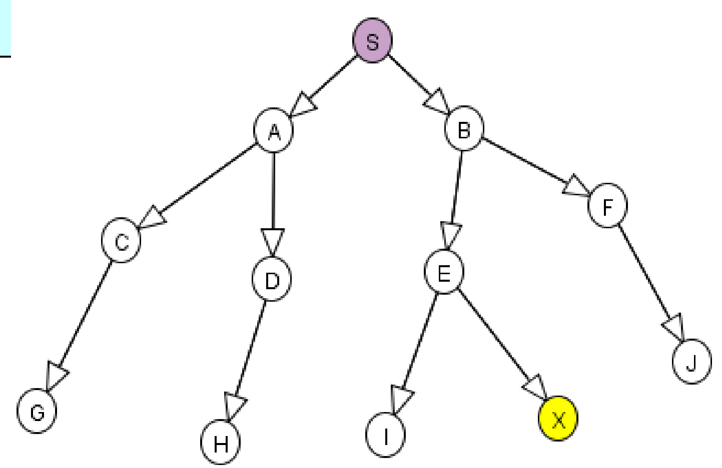
If $goal(n_k)$

return $\langle n_o, \ldots, n_k \rangle$;

For every neighbor *n* of n_k

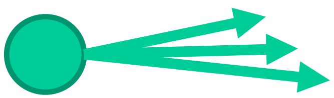
add $\langle n_o, \ldots, n_k, n \rangle$ to *frontier;*

return NULL

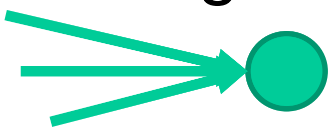


Branching Factor

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.** b^m **C.** m^b **D.** m/b

Lecture Summary

- Search is a key computational mechanism in many AI agents
- We will study the basic principles of search on the simple **deterministic planning agent model**

Generic search approach:

- define a search space graph,
- start from current state,
- incrementally explore paths from current state until goal state is reached.

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

Next “class”

- **Uninformed search** strategies

(read textbook Sec. 3.5)

- First **Practice Exercise** 3.A
- http://www.aispace.org/exercises.shtml