Learning Objectives

At the end of the class you should be able to:

- define a directed graph
- represent a problem as a state-space graph
- explain how a generic searching algorithm works
Often we are not given an algorithm to solve a problem, but only a specification of what is a solution — we have to search for a solution.

A typical problem is when the agent is in one state, it has a set of deterministic actions it can carry out, and wants to get to a goal state.

Many AI problems can be abstracted into the problem of finding a path in a directed graph.

Often there is more than one way to represent a problem as a graph.
State-space Search

- flat or modular or hierarchical
- explicit states or features or individuals and relations
- static or finite stage or indefinite stage or infinite stage
- fully observable or partially observable
- deterministic or stochastic dynamics
- goals or complex preferences
- single agent or multiple agents
- knowledge is given or knowledge is learned
- perfect rationality or bounded rationality
A state-space problem consists of

- a set of states
- a subset of states called the start states
- a set of actions
- an action function: given a state and an action, returns a new state
- a set of goal states, specified as function, goal(s)
- a criterion that specifies the quality of an acceptable solution.
Example Problem for Delivery Robot

The robot wants to get from outside room 103 to the inside of room 123.
Directed Graphs

A (directed) graph consists of a set $N$ of nodes and a set $A$ of ordered pairs of nodes, called arcs.
A (directed) **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 (directed) 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 $\langle n_0, n_1, \ldots, n_k \rangle$ such that $\langle n_{i-1}, n_i \rangle \in A$. 
A (directed) 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 $\langle n_0, n_1, \ldots, n_k \rangle$ such that $\langle n_{i-1}, n_i \rangle \in A$.

The length of path $\langle n_0, n_1, \ldots, n_k \rangle$ is $k$. 

Given a set of start nodes and goal nodes, a solution is a path from a start node to a goal node.
A (directed) 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 \( \langle n_0, n_1, \ldots, n_k \rangle \) such that \( \langle n_{i-1}, n_i \rangle \in A \).

The length of path \( \langle n_0, n_1, \ldots, n_k \rangle \) is \( k \).

Given a set of start nodes and goal nodes, a solution is a path from a start node to a goal node.
Example Problem for Delivery Robot

The robot wants to get from outside room 103 to the inside of room 123.
State-Space Graph for the Delivery Robot
Example: Google Maps
Robot Cleaner

- 2 rooms, one cleaning robot
- rooms can be clean or dirty
- robot actions:
  - suck: makes the room that the robot is in clean
  - move: move to other room
- Goal: have both rooms clean
Robot Cleaner

- 2 rooms, one cleaning robot
- rooms can be clean or dirty
- robot actions:
  - suck: makes the room that the robot is in clean
  - move: move to other room
- Goal: have both rooms clean
- How many states are there?
Grid game: Rob needs to collect coins $C_1, C_2, C_3, C_4$, without running out of fuel, and end up at location $(1, 1)$:
Partial Search Space for a Video Game

Grid game: Rob needs to collect coins $C_1$, $C_2$, $C_3$, $C_4$, without running out of fuel, and end up at location $(1, 1)$:

State:
$\langle X\text{-pos}, Y\text{-pos}, \text{Fuel}, C_1, C_2, C_3, C_4 \rangle$

Goal:
$\langle 1, 1, ?, t, t, t, t \rangle$
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.
Problem Solving by Graph Searching

- Start node
- Explored nodes
- Ends of paths on frontier
- Unexplored nodes

© D. Poole and A. Mackworth 2017
Graph Search Algorithm

**Input:** a graph,
   a set of start nodes,
   Boolean procedure $goal(n)$ that tests if $n$ is a goal node.

$frontier := \{\langle s \rangle : s \text{ is a start node}\}$

**while** $frontier$ is not empty:

**select** and **remove** path $\langle n_0, \ldots, n_k \rangle$ from $frontier$

**if** $goal(n_k)$

**return** $\langle n_0, \ldots, n_k \rangle$

**for every** neighbor $n$ of $n_k$

**add** $\langle n_0, \ldots, n_k, n \rangle$ to $frontier$

**end while**
Graph Search Algorithm

- Which value is selected from the frontier at each stage defines the search strategy.
- The neighbors define the graph.
- *goal* defines what is a solution.
- If more than one answer is required, the search can continue from the return.
Often we don’t want any solution, but the best solution or optimal solution.

Costs on arcs give costs on paths. We want the least-cost path to a goal.