Uninformed Search

CPSC 322 Lecture 5

January 17, 2007
Textbook §2.4

Lecture Overview

1. Graph Search
2. Searching
3. Depth-First Search

Search

- What we want to be able to do:
  - find a solution when we are not given an algorithm to solve a problem, but only a specification of what a solution looks like
  - idea: search for a solution
- What we need:
  - A set of states
  - A start state
  - A goal state or set of goal states
    - or, equivalently, a goal test: a boolean function which tells us whether a given state is a goal state
  - A set of actions
  - An action function: a mapping from a state and an action to a new state

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

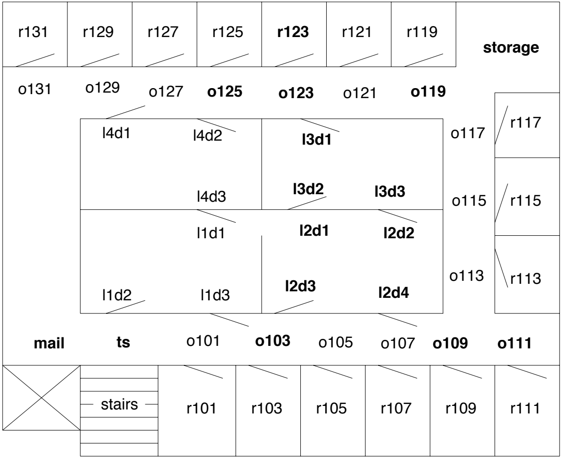
To make this more formal, we'll need to talk about graphs...

Search Graphs

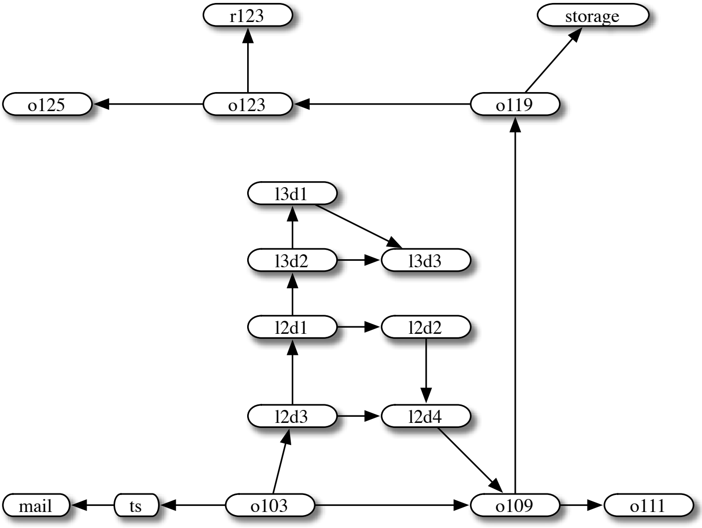
- A graph consists of
  - a set N of nodes;
  - a set A of ordered pairs of nodes, called arcs or edges.
- Node n_2 is a neighbor of n_1 if there is an arc from n_1 to n_2.
  - i.e., if $\langle n_1, n_2 \rangle \in A$
- A path is a sequence of nodes $\langle n_0, n_1, \ldots, n_k \rangle$ such that $\langle n_{i-1}, n_i \rangle \in A$.
- Given a start node and a set of goal nodes, a solution is a path from the start node to a goal node.

Example Domain for the Delivery Robot

The agent starts outside room 103,
and wants to end up inside room 123.



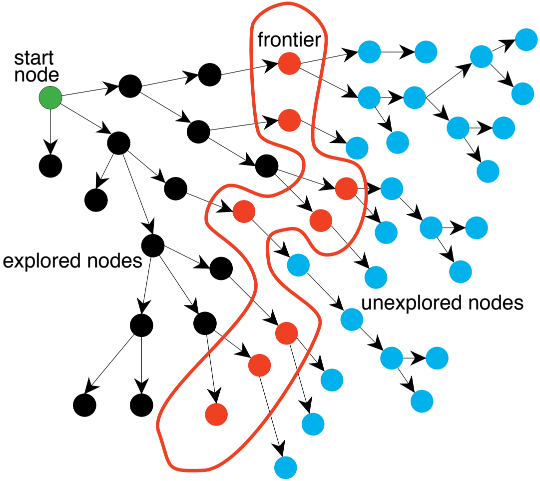
Example Graph for the Delivery Robot



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.

Problem Solving by Graph Searching



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.

Lecture Overview

1. Graph Search
2. Searching
3. Depth-First Search

Graph Search Algorithm

Input: a graph,
a set of start nodes,
Boolean procedure $goal(n)$ that tests if n is a goal node.
$frontier := \{\langle s \rangle : s$ 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

- After the algorithm returns, it can be asked for more answers and the procedure continues.
- Which value is selected from the frontier defines the search strategy.
- The *neighbor* relationship defines the graph.
- The *goal* function defines what is a solution.

Branching Factor

- The forward branching factor of a node is the number of arcs going out of that node
- The backward branching factor of a node is the number of arcs going into the node
- If the forward branching factor of every node is b and the graph is a tree, how many nodes are exactly n steps away from the start node?

Branching Factor

- The forward branching factor of a node is the number of arcs going out of that node
- The backward branching factor of a node is the number of arcs going into the node
- If the forward branching factor of every node is b and the graph is a tree, how many nodes are exactly n steps away from the start node?
  - b^n nodes.
- We'll assume that all branching factors are finite.

Comparing Algorithms

- Completeness
  - if at least one solution exists, the algorithm is guaranteed to find a solution within a finite amount of time
- Time Complexity
  - in terms of the maximum path length m, and the maximum branching factor b, what is the worst-case amount of time that the algorithm will take to run?
- Space Complexity
  - in terms of m and b, what is the worst-case amount of memory that the algorithm must use?

Lecture Overview

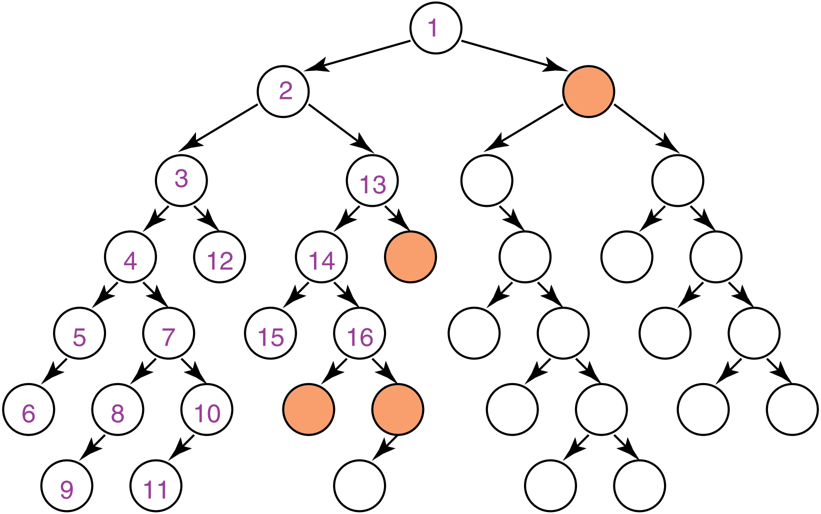
1. Graph Search
2. Searching
3. Depth-First Search

Depth-first Search

- Depth-first search treats the frontier as a stack
- It always selects one of the last elements added to the frontier.

- Example:
  - the frontier is $[p_1, p_2, \ldots, p_r]$
  - neighbours of p_1 are $\{n_1, \ldots, n_k\}$
- What happens?
  - p_1 is selected, and tested for being a goal.
  - Neighbours of p_1 replace p_1 at the beginning of the frontier.
  - Thus, the frontier is now $[n_1, \ldots, n_k, p_2, \ldots, p_r]$.
  - p_2 is only selected when all paths from p_1 have been explored.

Illustrative Graph — Depth-first Search



Analysis of Depth-first Search

- Is DFS complete?

Analysis of Depth-first Search

- Is DFS complete?
  - Depth-first search isn't guaranteed to halt on infinite graphs or on graphs with cycles.
  - However, DFS *is* complete for finite trees.

Analysis of Depth-first Search

- Is DFS complete?
  - Depth-first search isn't guaranteed to halt on infinite graphs or on graphs with cycles.
  - However, DFS *is* complete for finite trees.
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?

Analysis of Depth-first Search

- Is DFS complete?
  - Depth-first search isn't guaranteed to halt on infinite graphs or on graphs with cycles.
  - However, DFS *is* complete for finite trees.
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - Search is unconstrained by the goal until it happens to stumble on the goal.

Analysis of Depth-first Search

- Is DFS complete?
  - Depth-first search isn't guaranteed to halt on infinite graphs or on graphs with cycles.
  - However, DFS *is* complete for finite trees.
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - Search is unconstrained by the goal until it happens to stumble on the goal.
- What is the space complexity?

Analysis of Depth-first Search

- Is DFS complete?
  - Depth-first search isn't guaranteed to halt on infinite graphs or on graphs with cycles.
  - However, DFS *is* complete for finite trees.
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - Search is unconstrained by the goal until it happens to stumble on the goal.
- What is the space complexity?
  - Space complexity is $O(bm)$: the longest possible path is m, and for every node in that path must maintain a fringe of size b.

Using Depth-First Search

- When is DFS appropriate?

Using Depth-First Search

- When is DFS appropriate?
  - space is restricted
  - solutions tend to occur at the same depth in the tree
  - you know how to order nodes in the list of neighbours so that solutions will be found relatively quickly

Using Depth-First Search

- When is DFS appropriate?
  - space is restricted
  - solutions tend to occur at the same depth in the tree
  - you know how to order nodes in the list of neighbours so that solutions will be found relatively quickly

- When is DFS inappropriate?

Using Depth-First Search

- When is DFS appropriate?
  - space is restricted
  - solutions tend to occur at the same depth in the tree
  - you know how to order nodes in the list of neighbours so that solutions will be found relatively quickly

- When is DFS inappropriate?
  - some paths have infinite length
  - the graph contains cycles
  - some solutions are very deep, while others are very shallow