Reminders

- I have help hours Monday from 4-5:30pm
- Lepe'i's help hours: Sundays 6-8pm
- Lyra's help hours: Wednesdays 2-4pm
- Reading for next Tuesday: YLLATAILY 3-4
Recap
Defining A Search Problem

**States:** a representation of physical configuration

**Nodes:** a data structure representing:

\[ \langle \text{state}, \text{parent-node}, \text{children}, \text{action}, \text{path-cost}, \text{depth} \rangle \]

**Goal:** the state(s) we're trying to reach

**Start State:** initial starting point

**Solution:** a sequence of states that take us from the start state to the goal state.

**Optimal Solution:** the shortest solution
Graph Search vs Tree Search

**function** \( \text{TREE-SEARCH}(\text{problem}) \) **returns** a solution, or failure
initialize the frontier using the initial state of \( \text{problem} \)

**loop** do
  if the frontier is empty **then return** failure
  choose a leaf node and remove it from the frontier
  if the node contains a goal state **then return** the corresponding solution
  expand the chosen node, adding the resulting nodes to the frontier

**function** \( \text{GRAPH-SEARCH}(\text{problem}) \) **returns** a solution, or failure
initialize the frontier using the initial state of \( \text{problem} \)
*initialize the explored set to be empty*

**loop** do
  if the frontier is empty **then return** failure
  choose a leaf node and remove it from the frontier
  if the node contains a goal state **then return** the corresponding solution
  *add node to the explored set*
  expand the chosen node, adding the resulting nodes to the frontier
  *only if not in the frontier* *explored set*
Search Strategies

Review: *Strategy* = order of tree expansion

- Implemented by different queue structures (LIFO, FIFO, priority)

Dimensions for evaluation

- **Completeness** - always find the solution?
- **Optimality** - finds a least cost solution (lowest path cost) first?
- **Time complexity** - # of nodes generated *(worst case)*
- **Space complexity** - # of nodes simultaneously in memory *(worst case)*

Time/space complexity variables

- $b$, *maximum branching factor* of search tree
- $d$, *depth* of the shallowest goal node
- $m$, maximum length of any path in the state space (potentially $\infty$)
Uninformed Search

Uses only information available in problem definition

Informally:

**Uninformed search:** All non-goal nodes in frontier look equally good

**Informed search:** Some non-goal nodes can be ranked above others.
Breadth-first search

Idea:
  • Expand *shallowest* unexpanded node

Implementation:
  • *frontier* is FIFO (First-In-First-Out) Queue:
    • Put successors at the *end* of *frontier* successor list.
Properties of breadth-first search

Complete?  Yes (if $f$ is finite)
Optimal?  Yes! (assuming we measure cost as # of steps)

Time Complexity?  $O(b^d)$
Space Complexity?  $O(b^d)$
Exponential Space (and time) Is Not Good...

- Exponential complexity uninformed search problems cannot be solved for any but the smallest instances.
- Memory requirements are a bigger problem than execution time.

<table>
<thead>
<tr>
<th>DEPTH</th>
<th>NODES</th>
<th>TIME</th>
<th>MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>110</td>
<td>0.11 milliseconds</td>
<td>107 kilobytes</td>
</tr>
<tr>
<td>4</td>
<td>11110</td>
<td>11 milliseconds</td>
<td>10.6 megabytes</td>
</tr>
<tr>
<td>6</td>
<td>10^6</td>
<td>1.1 seconds</td>
<td>1 gigabyte</td>
</tr>
<tr>
<td>8</td>
<td>10^8</td>
<td>2 minutes</td>
<td>103 gigabytes</td>
</tr>
<tr>
<td>10</td>
<td>10^10</td>
<td>3 hours</td>
<td>10 terabytes</td>
</tr>
<tr>
<td>12</td>
<td>10^12</td>
<td>13 days</td>
<td>1 petabyte</td>
</tr>
<tr>
<td>14</td>
<td>10^14</td>
<td>3.5 years</td>
<td>99 petabytes</td>
</tr>
</tbody>
</table>

Assumes b=10, 1M nodes/sec, 1000 bytes/node
Depth-First Search
Depth-first search

Idea:
- Expand *deepest* unexpanded node

Implementation:
- *frontier* is LIFO (Last-In-First-Out) Queue:
  - Put successors at the *front* of *frontier* successor list.
1 \ x = \text{start vertex(1)}
2 \ \text{dfs}(x)
3
4 \ \text{def dfs}(x):
5 \ \quad \text{mark} \ x \ \text{as visited}
6 \ \quad \text{for each} \ y \ \text{in} \ x \ \text{connections:}
7 \ \quad \quad \text{if} \ y \ \text{not visited then}
8 \ \quad \quad \quad \text{dfs}(y)
Properties of depth-first search

Complete? Yes if tree is finite
Optimal? No
Time Complexity? $O(b^m)$
Space Complexity? $O(b \times m)$

$m =$ maximum depth of search space
Depth-first vs Breadth-first

Use depth-first if

- *Space is restricted*
- There are many possible solutions with long paths and wrong paths are usually terminated quickly
- Search can be fine-tuned quickly

Use breadth-first if

- *Possible infinite paths*
- Some solutions have short paths
- Can quickly discard unlikely paths

Slides adapted from Chris Callison-Burch
Search Conundrum

Breadth-first

☑ Complete,
☑ Optimal
☒ *but* uses $O(b^d)$ space

Depth-first

☒ Not complete *unless m is bounded*
☒ Not optimal
☒ Uses $O(b^m)$ time; terrible if $m >> d$
☑ *but* only uses $O(b^*m)$ space
Depth-limited search: A building block

Depth-First search *but with depth limit l.*
- i.e. nodes at depth $l$ *have no successors.*
- No infinite-path problem!

If $l = d$ (by luck!), then optimal
- But:
  - If $l < d$ then incomplete 😞
  - If $l > d$ then not optimal 😞

Time complexity: $O(b^l)$
Space complexity: $O(bl)$ 😊
## Summary of algorithms

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Breadth-First</th>
<th>Depth-First</th>
<th>Depth-limited</th>
<th>Iterative deepening</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Time</td>
<td>$b^d$</td>
<td>$b^m$</td>
<td>$b^l$</td>
<td>$b^d$</td>
</tr>
<tr>
<td>Space</td>
<td>$b^d$</td>
<td>$b^m$</td>
<td>$b^l$</td>
<td>$b^d$</td>
</tr>
<tr>
<td>Optimal?</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

Slides adapted from Chris Callison-Burch