Search

Basic to problem solving:
- Problem can be in various states.
- Start in an initial state.
- Have some actions available.
- Each action has a cost.
- Want to reach some goal, minimizing cost.

Happens in simulation.
Not web search.

Formal Definition

Set of states $S$
Start state $s \in S$
Set of actions $A$ and action rules $a(s) \rightarrow s'$
Goal test $g(s) \rightarrow \{0, 1\}$
Cost function $C(s, a, s') \rightarrow \mathbb{R}^+$

So a search problem is specified by a tuple, $(S, s, A, g, C)$.

Example

Sudoku
States: all legal Sudoku boards.
Start state: a particular, partially filled-in, board.
Actions: inserting a number into the board.
Goal test: all cells filled and no collisions.
Cost function: 1 per number.
Example

- Flights - e.g., ITA Software.
- States: airports.
- Start state: RDU.
- Actions: available flights from each airport.
- Goal test: reached Tokyo.
- Cost function: time and/or money.

The Search Tree

Classical conceptualization of search.

Key to a search algorithm: which node to expand next?

Important quantities:
- breadth (branching factor)
- depth
- min soln depth $m$

$O(b^d)$ leaves in a search tree of breadth $b$, depth $d$. 
How to Expand?

*Uninformed strategy:* you know nothing about likely solutions in the tree.

What to do?
- Expand deepest node (depth-first search)
- Expand closest node (breadth-first search)

Properties
- Completeness
- Optimality
- Time Complexity (total number of nodes visited)
- Space Complexity (size of frontier)
**Depth-First Search**

- **Properties:**
  - **Completeness:** Only for finite trees.
  - **Optimality:** No.
  - **Time Complexity:** $O(b^{d+1})$
  - **Space Complexity:** $O(bd)$

Here $m$ is depth of found solution (not necessarily min solution depth).

The deepest node happens to be the one you most recently visited - easy to implement recursively OR manage frontier using LIFO queue.

---

**Breadth-First Search**
Breadth-First Search

Properties:
- Completeness: Yes.
- Optimality: Yes for constant cost.
- Time Complexity: $O(b^{m+1})$
- Space Complexity: $O(b^{m+1})$

Better than depth-first search in all respects except memory cost - must maintain a large frontier.

Manage frontier using FIFO queue.
Iterative Deepening Search

Combine these two strengths.

The core problems in DFS are a) not optimal, and b) not complete. In both cases, because it fails to explore other alternatives.

Iterative Deepening:
• Run DFS to a fixed depth $z$.
• Start at $z=1$. If no solution, increment $z$ and rerun.

IDS

Optimal for constant cost! Proof?

How can that be a good idea?

It duplicates work.

Sure but:
• Low memory requirement (equal to DFS).
• Not many more nodes expanded than BFS. (About twice as many for binary tree.)

IDS

Key Insight:
• Many more nodes at depth $m+1$ than at depth $m$.

MAGIC.

“In general, iterative deepening search is the preferred uninformed search method when the state space is large and the depth of the solution is unknown.” (R&N)

Friday

Informed searches … what if you know something about the the solution? What form should such knowledge take? How should you use it?

Ponder.