Implementation: states vs. nodes

State

- (Representation of) a physical configuration

Node

- Data structure constituting part of a search tree
  - Includes *parent, children, depth, path cost* g(x)

States do not have parents, children, depth, or path cost!

Search strategies

A strategy is defined by picking the order of node expansion

Strategies are evaluated along the following dimensions:

- Completeness – does it always find a solution if one exists?
- Time complexity – number of nodes generated/expanded
- Space complexity – maximum nodes in memory
- Optimality – does it always find a least cost solution?

Time and space complexity are measured in terms of:

- b – maximum branching factor of the search tree
- d – depth of the least-cost solution
- m – maximum depth of the state space (may be infinite)

Uninformed Search Strategies

Uninformed strategies use only the information available in the problem definition

- Breadth-first search
- Uniform-cost search (related to breadth-first search; determines a path to the goal state that has the lowest weight)
- Depth-first search
- Depth-limited search
- Iterative deepening search (depth-first search is run repeatedly with increasing depth limits until the goal is found)

Breadth-first search

Expand shallowest unexpanded node

Implementation:

- *Fringe* is a FIFO queue, i.e., new successors go at end
- Execute first few expansions of Eight Puzzle using Breadth-first search

Properties of breadth-first search

Complete?? Yes (if b is finite)

Time?? $1 + b + b^2 + \ldots + b^d + b(b^d-1) = O(b^{d+1})$, i.e., exp in d

Space?? $O(b^{d+1})$ (keeps every node in memory)

Optimal?? If cost = 1 per step, not optimal in general

Space is the big problem; can easily generate nodes at 10 MB/s, so 24hrs = 860GB!

Uniform-cost search

Expand least-cost unexpanded node

Implementation:

- *Fringe* = queue ordered by path cost

Equivalent to breadth-first if…

Complete?? If step cost ≥ ε

Time?? # of nodes with g ≤ cost of optimal solution, $O(b^{C/\varepsilon})$, where C is the cost of the optimal solution

Space?? # of nodes with g ≤ cost of optimal solution, $O(b^{C/\varepsilon})$

Optimal?? Yes–nodes expanded in increasing order of *g(n)*

Execute first few expansions of Eight Puzzle using Uniform-first search

Depth-first search

Expand deepest unexpanded node

Implementation:

- Stack
- Execute first few expansions of Eight Puzzle game using Depth-first search

Depth-first search

Complete??

- No: fails in infinite-depth spaces, spaces with loops.
- Can be modified to avoid repeated states along path ➔ complete in finite spaces

Time??

- $O(b^m)$: terrible if m is much larger than d, but if solutions are dense, may be much faster than breadth-first

Space??

- *O(bm), i.e., linear space!*

Optimal??

- No

Iterative deepening search

```
function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution
  inputs: problem, a problem
  for depth ← 0 to ∞ do
    result ← DEPTH-LIMITED-SEARCH(problem, depth)
    if result ≠ cutoff then return result
  end
```

Summary

All tree searching techniques are more alike than different

Breadth-first has space issues, and possibly optimality issues

Uniform-cost has space issues

Depth-first has time and optimality issues, and possibly completeness issues

Depth-limited search has optimality and completeness issues

Iterative deepening is the best uninformed search we have explored

Next class we study informed searches