Announcements

- Solution to Assignment 1 is posted. (One fix for coffee.py)
- Assignment 2 is available

Learning Objectives

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

- Explain how iterative deepening saves space
- Explain how and branch and bound can find optimal solutions with linear space
- Explain what search algorithm should be used for any problem.

Review: Searching

- A frontier is a set of paths
- Generic search algorithm: Repeatedly:
  - select a path from the frontier
  - stop of it is a path to a goal
  - otherwise expand it in all ways, and add the resulting paths to the frontier
- Frontier is a stack $\longrightarrow$ depth-firt search
- Frontier is a queue $\longrightarrow$ breadth-firt search
- Frontier is a priority queue ordered by path cost $\longrightarrow$ least-cost-first search
- Frontier is a priority queue ordered by $f(p) = cost(p) + h(p)$ $\longrightarrow$ A^* search
- Cycle pruning prunes paths that loop back on themselves
- Multiple-path pruning prunes paths to nodes that have already been expanded.

Clicker Question

With a heuristic that does not satisfy the monotone restriction, how might A^* search with multiple-path pruning not be admissible?

- A it might not expand a path on frontier with lowest f-value
- B it might not return a lowest-cost path
- C it is always admissible, even without the monotone restriction
- D it only considers the heuristic value and not both path cost and heuristic cost
- E it might use space exponential in the path length instead of linear

Clicker Question

With of the following is false:

A With multiple-path pruning, we don't need cycle pruning

B With multiple path pruning all search algorithms halt on finite graphs

C All algorithms have exponential space with multiple-path pruning

D Cycle pruning without multiple-path pruning makes A^* no longer admissible

Summary of Search Strategies

Strategy	Complete	Halts	Space
Depth-first	No	No	Linear
Depth-first with cycle pruning	No	Yes	Linear
Depth-first with MPP	No	Yes	Exp
Breadth-first with MPP	Yes	Yes	Exp
Lowest-cost-first with MPP	Yes	Yes	Exp
Best-first with MPP (min $h(p)$)	No	Yes	Exp
A^* without cycle or MP pruning	Yes	No	Exp
A^* with cycle pruning	Yes	Yes	Exp
A^* with MPP	Yes	Yes	Exp

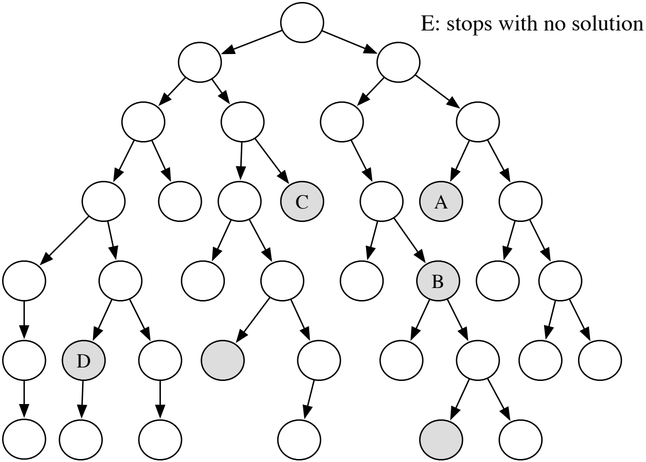
Complete — if there a path to a goal, it can find one, even on infinite graphs.
Halts — on finite graph (perhaps with cycles).
Space — as a function of the length of current or longest path
(Assume the graph and heustic follow assumptions of A^* proof)

Bounded Depth-first search

- A bounded depth-first search takes a bound (cost or depth) and does not expand paths that exceed the bound.
  - explores part of the search graph
  - uses space linear in the depth of the search.

Which shaded goal will a depth-bounded search find first

when the depth bound is 012345?



Iterative-deepening search

- Iterative-deepening search:
  - Start with a bound $b = 0$.
  - Do a bounded depth-first search with length bound b
  - If a solution is found return that solution
  - Otherwise increment b by 1 and repeat.
- This will find the same first solution as what other method?
- How much space is used?
- What happens if there is no path to a goal?
- Surely recomputing paths is wasteful!!!

Iterative Deepening Complexity

Complexity with solution at depth k & branching factor b:

level	breadth-first	iterative deepening	$\#$ nodes
1	1	k	b
2	1	$k-1$	b^2
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$k-1$	1	2	b^{k-1}
k	1	1	b^k
total	$\geq b^k$	$\leq b^k \left(\frac{b}{b-1}\right)^2$	

Depth-first Branch-and-Bound

- combines depth-first search with heuristic information.
- finds optimal solution.
- most useful when there are multiple solutions, and we want an optimal one.
- uses the space of depth-first search.

Depth-first Branch-and-Bound

Suppose we want to find a single optimal solution.

- Suppose *bound* is the cost of the lowest-cost path found to a goal so far.
- What if the search encounters a path p such that $cost(p) + h(p) \geq bound$?
  — p can be pruned.
- What can we do if a non-pruned path to a goal is found?
  bound can be set to the cost of p, and p can be remembered as the best solution so far.
- What can be guaranteed when the search completes?
  It has found an optimal solution if there is a solution with cost less than bound.
- Why should this use a depth-first search?
  Uses linear space.

Depth-first Branch-and-Bound

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

$frontier := \{\langle s \rangle : s \text{ is a start node}\}$
$best := \bot$
$expanded := \{\}$
while $frontier$ is not empty:
  select and **remove** path $\langle n_0, \ldots, n_k \rangle$ from $frontier$
  if $cost(\langle n_0, \ldots, n_k \rangle) + h(n_k) < bound$
    if $goal(n_k)$:
      $best := \langle n_0, \ldots, n_k \rangle$
      $bound := cost(\langle n_0, \ldots, n_k \rangle)$
    else
      $Frontier := Frontier \cup \{\langle n_0, \ldots, n_k, n \rangle : \langle n_k, n \rangle \in A\}$
return best

Depth-first Branch-and-Bound: Initializing Bound

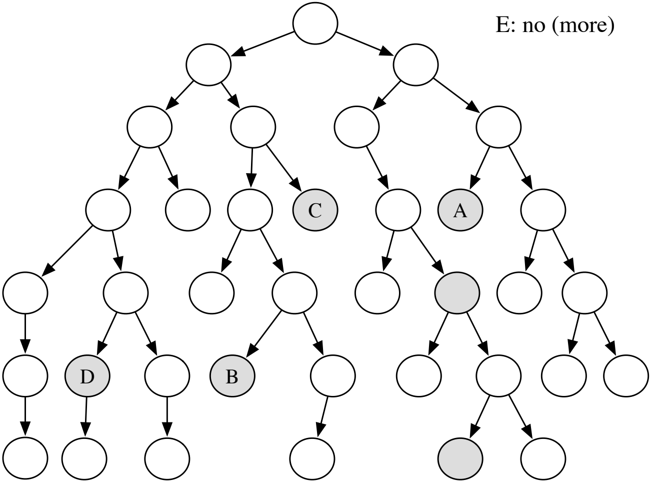
How should *bound* be initialized?

- The bound can be initialized to ∞.
- The bound can be set to an estimate of the optimal path cost. After depth-bounded depth-first search terminates either:
  - A solution was found.
    – this is an optimal solution
  - No solution was found, and no path was pruned.
    – there is no solution
  - No solution was found, and a path was pruned.
    – the bound can be increased, and search started again
    $\longrightarrow$ Depth-first Branch-and-Bound with Iterative Deepening

The efficiency is very sensitive to the bound (and how it is increased).

Which shaded goals will be best solutions

Suppose bound is initially set to 6 (or greater). Expand nodes from left to right.
Which shaded goal will be found first?second?third?



E: no (more)

Summary of Search Strategies

Strategy	Complete	Halts	Space
Depth-first	No	No	Linear
Depth-first with cycle pruning	No	Yes	Linear
Depth-first with MPP	No	Yes	Exp
A^* without cycle or MP pruning	Yes	No	Exp
A^* with MPP	Yes	Yes	Exp
DFBnB with inf bound	No	No	Linear
DFBnB + ID + MP pruning	Yes	Yes	Exp
DFBnB + ID + cycle pruning	Yes	Yes	Linear

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current or longest path

ID = iterative deepening

(Assume the graph and heustic follow assumptions of A^* proof)

What search algorithm should we use?

if we need an optimal solution:
    if we know the goal (before knowing start start):
        use DP to improve heuristic
    if there is enough space to store the graph:
        use A* with MPP
    else:
        use depth-first BnB + ID (with cycle pruning)
else if there is space:
        use A* with a non-admisible heursitic
    else:
        use use depth-bounded depth-first search