Learning Objectives

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

- justify why depth-bounded search is useful
- demonstrate how iterative-deepening works for a particular problem
- demonstrate how depth-first branch-and-bound works for a particular problem

Summary of Search Strategies

Strategy	Frontier	Complete	Halts	Space
Depth-first w/o CP	Last added	No	No	Linear
Depth-first w CP	Last added	No	Yes	Linear
Depth-first w MPP	Last added	No	Yes	Exp
Breadth-first w/o MPP	First added	Yes	No	Exp
Breadth-first w MPP	First added	Yes	Yes	Exp
Best-first w/o MPP	Min $h(p)$	No	No	Exp
Best-first w MPP	Min $h(p)$	No	Yes	Exp
A^* w/o MPP	Min $f(p)$	Yes	No	Exp
A^* w MPP	Min $f(p)$	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).

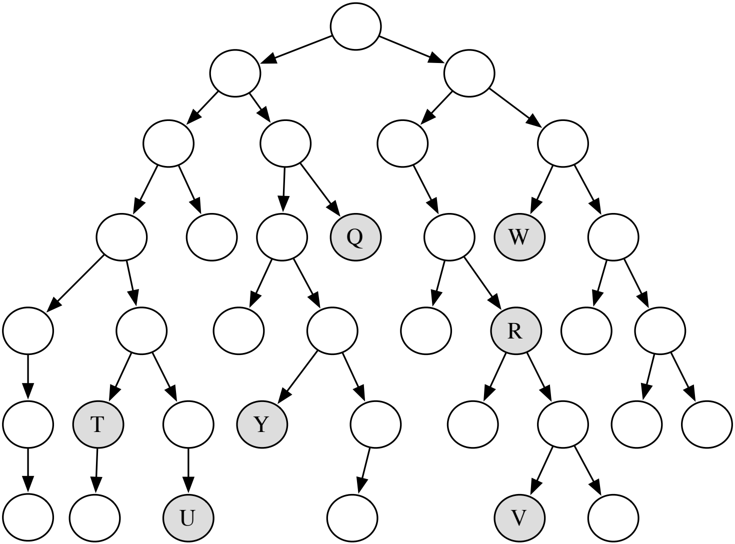
Space — as a function of the length of current path

Assume graph satisfies the assumptions of A^* proof + montonicity

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.
- How does this relate to other searches?
- How can this be extended to be complete?

Which shaded goal will a depth-bounded search find first?



Iterative-deepening search

- Iterative-deepening search:
  - Start with a bound $b = 0$.
  - Do a bounded depth-first search with bound b
  - If a solution is found return that solution
  - Otherwise increment b 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.
- Why should this use a depth-first search?
  Uses linear space.
- What can be guaranteed when the search completes?
  It has found an optimal solution.
- How should the bound be initialized?

Depth-first Branch-and-Bound: Initializing Bound

- The bound can be initialized to ∞.
- The bound can be set to an estimate of the optimal path cost. After depth-first search terminates either:
  - A solution was found.
  - No solution was found, and no path was pruned
  - No solution was found, and a path was pruned.

Which shaded goals will be best solutions so far?