
CIS 471/571 (Fall 2023): Introduction to Artificial Intelligence

Lecture 3: Uninformed Search (Part 2)

Thanh H. Nguyen

Most slides are by Pieter Abbeel, Dan Klein, Luke Zettlemoyer, John DeNero,
Stuart Russell, Andrew Moore, or Daniel Lowd
Source: <http://ai.berkeley.edu/home.html>



Reminder

- Homework 1: Search
 - Deadline: Oct 11th, 2023
- Project 1: Search
 - Deadline: Oct 16th, 2023

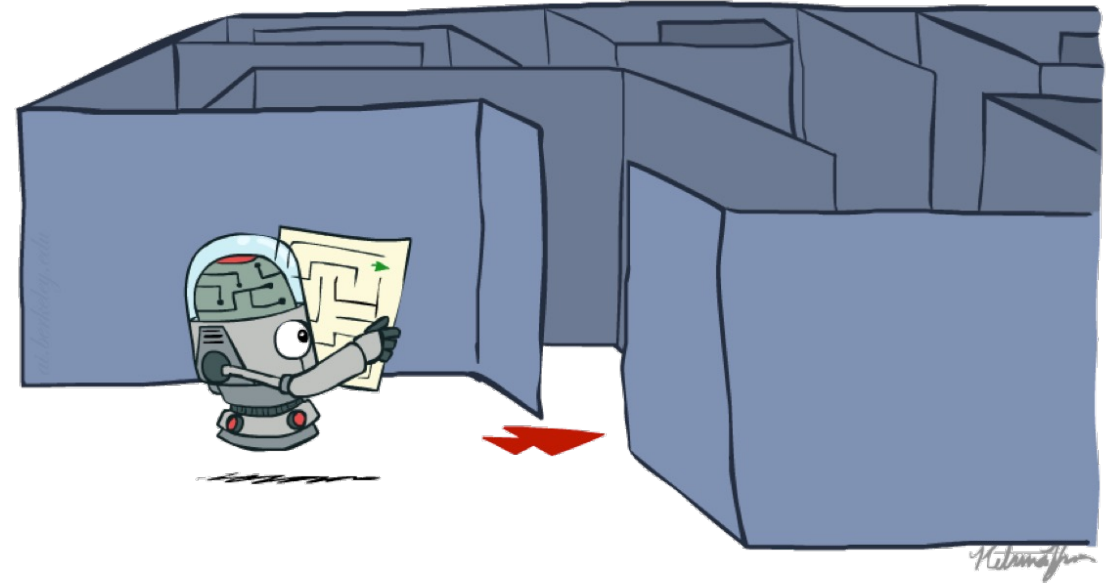
Today

- Uninformed Search
 - Depth-first Search
 - Breadth-first Search
 - Uniform Cost Search
- Informed Search
 - Heuristics
 - Greedy Search
 - A* Search



Recap: Search

- Search problem:
 - States (configurations of the world)
 - Actions and costs
 - Successor function (world dynamics)
 - Start state and goal test
- Search tree:
 - Nodes: represent plans for reaching states
 - Plans have costs (sum of action costs)
- Search algorithm:
 - Systematically builds a search tree
 - Chooses an ordering of the fringe (unexplored nodes)
 - Optimal: finds least-cost plans

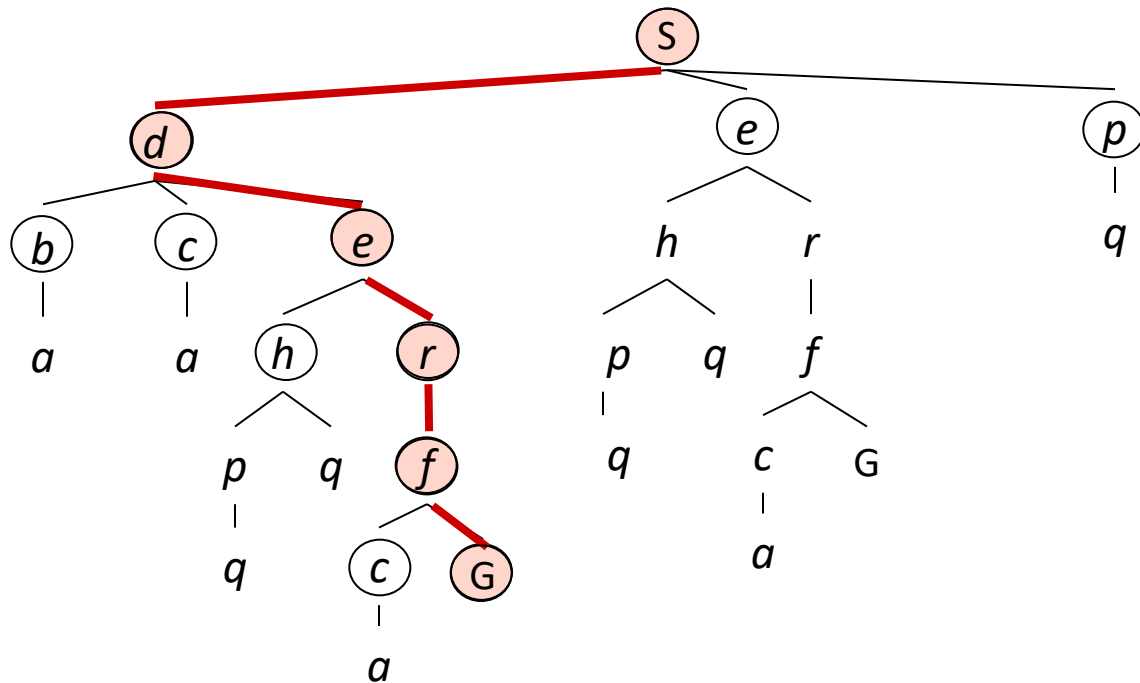
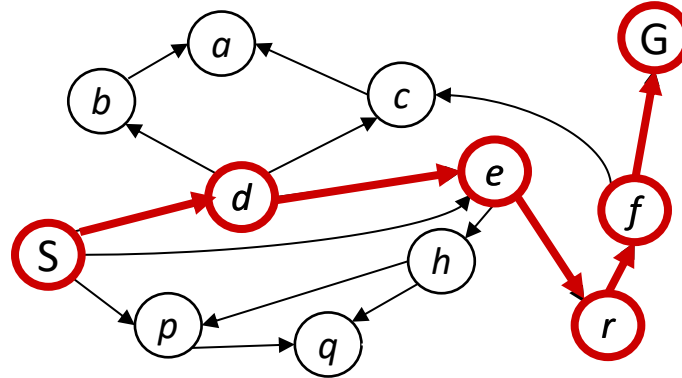


Tree Search Pseudo-Code

```
function TREE-SEARCH(problem, fringe) return a solution, or failure
  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node ← REMOVE-FRONT(fringe)
    if GOAL-TEST(problem, STATE[node]) then return node
    for child-node in EXPAND(STATE[node], problem) do
      fringe ← INSERT(child-node, fringe)
    end
  end
```



Example: Tree Search



~~s~~
~~s → d~~
s → e
s → p
s → d → b
s → d → c
~~s → d → e~~
s → d → e → h
~~s → d → e → r~~
~~s → d → e → r → f~~
s → d → e → r → f → c
~~s → d → e → r → f → G~~



Uninformed Search



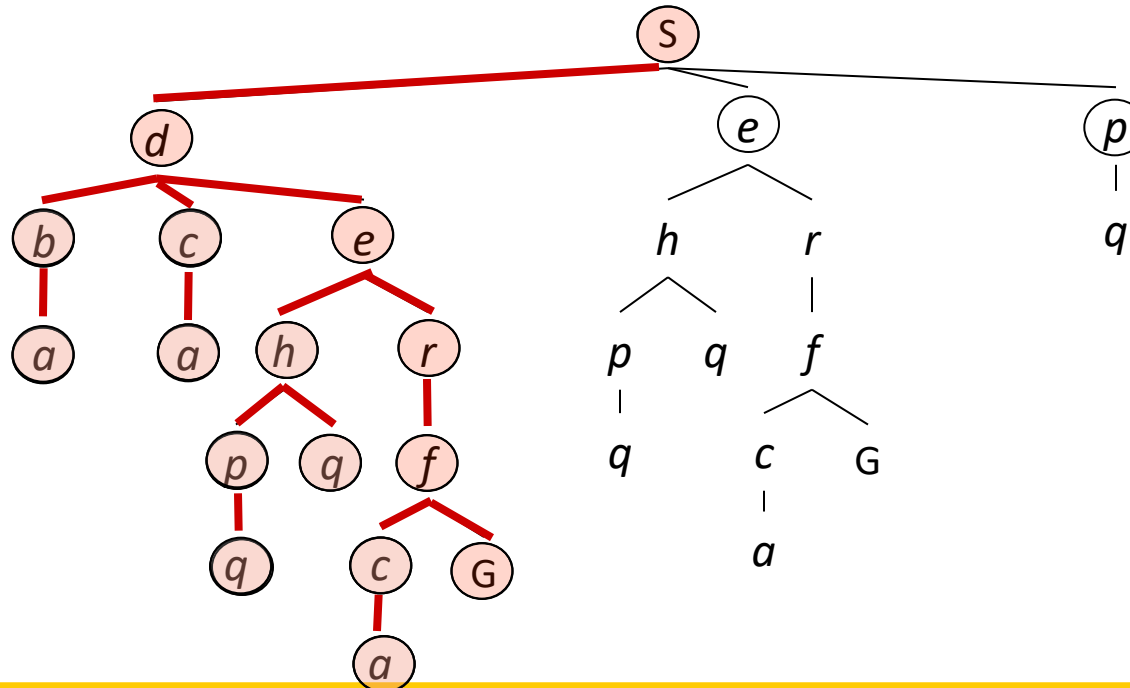
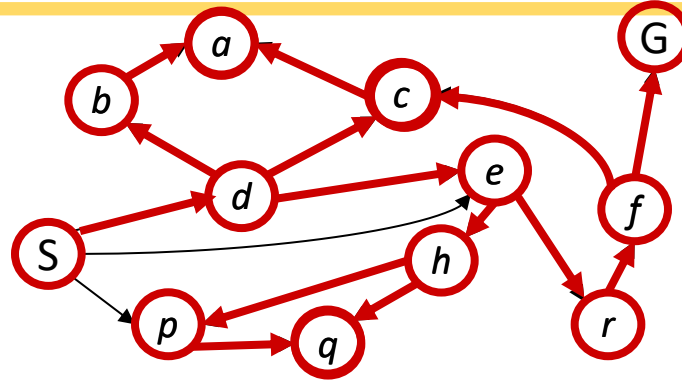
Depth-First Search (DFS)



Depth-First Search (DFS)

*Strategy: expand a
deepest node first*

*Implementation: Fringe
is a LIFO stack*

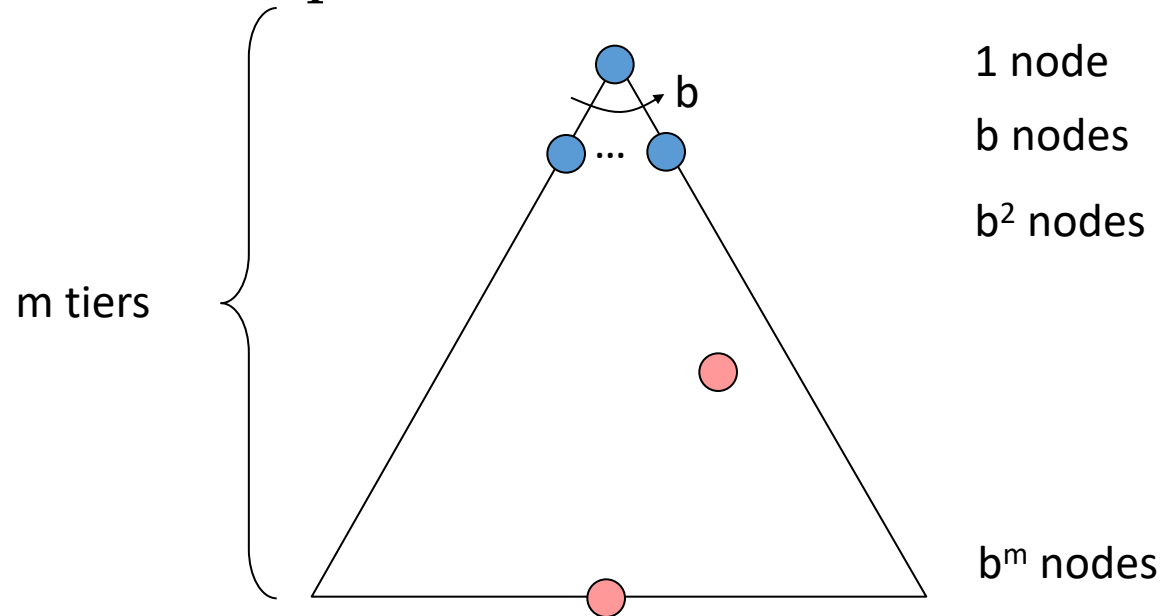


Search Algorithm Properties

- Complete: Guaranteed to find a solution if one exists?
- Optimal: Guaranteed to find the least cost path?
- Time complexity?
- Space complexity?

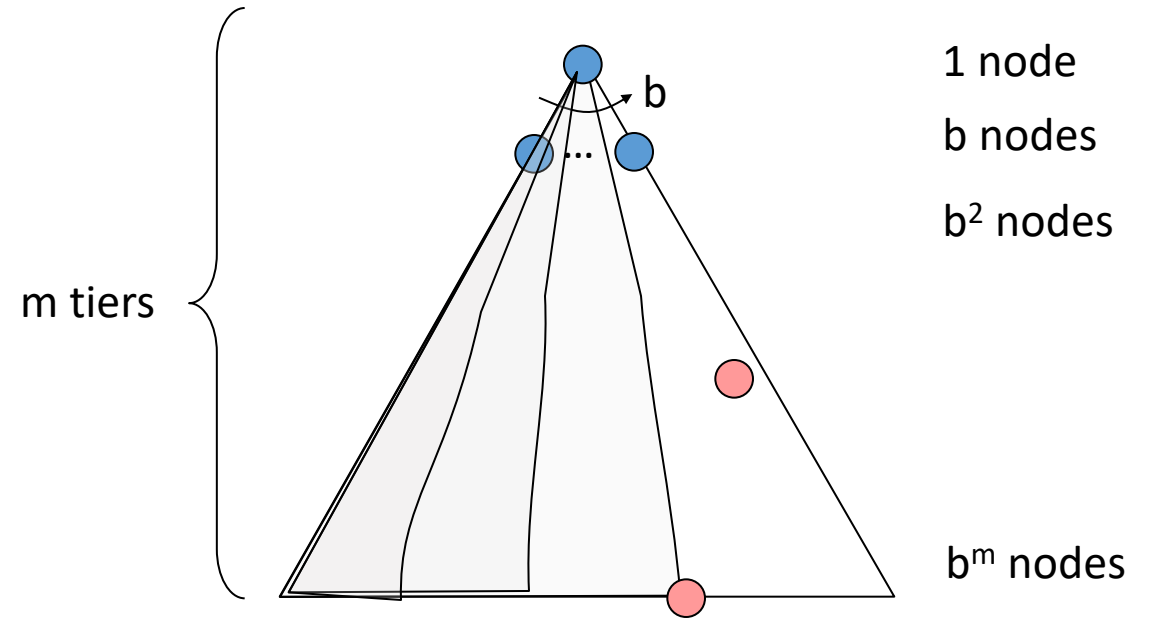
- Cartoon of search tree:
 - b is the branching factor
 - m is the maximum depth
 - solutions at various depths

- Number of nodes in entire tree?
 - $1 + b + b^2 + \dots + b^m = O(b^m)$

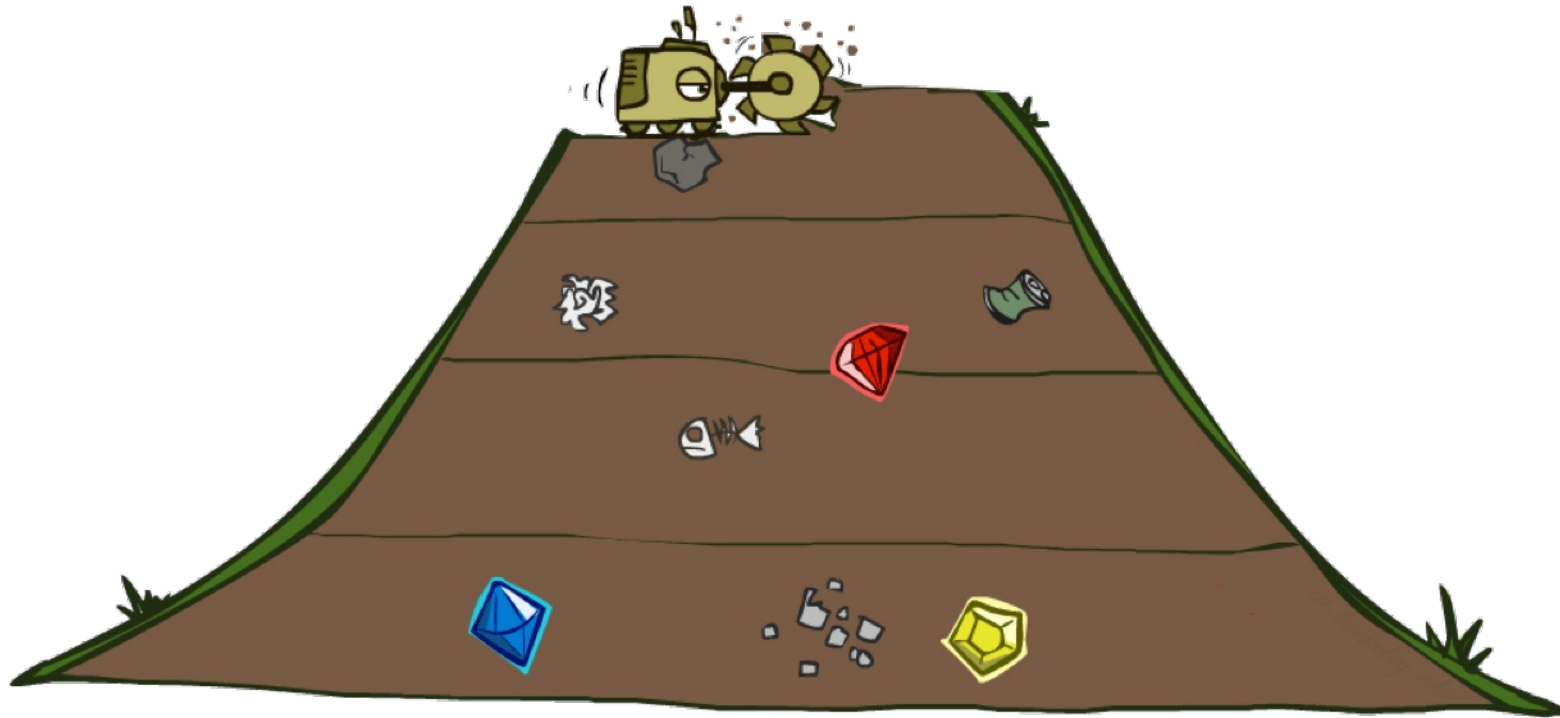


DFS Properties

- What nodes DFS expand?
 - Some left prefix of the tree.
 - Could process the whole tree!
 - If m is finite, takes time $O(b^m)$
- How much space does the fringe take?
 - Only has siblings on path to root, so $O(bm)$
- Is it complete?
 - m could be infinite, so only if we prevent cycles (more later)
- Is it optimal?
 - No, it finds the “leftmost” solution, regardless of depth or cost



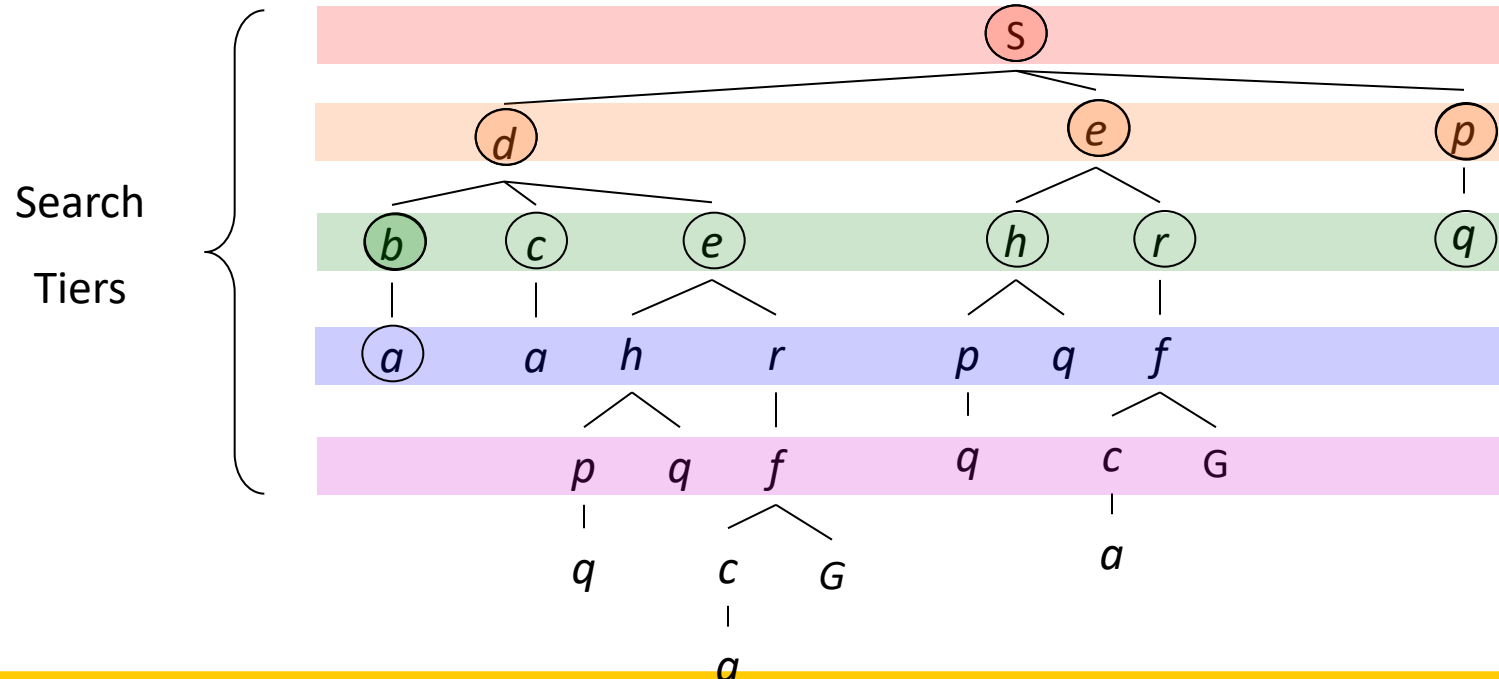
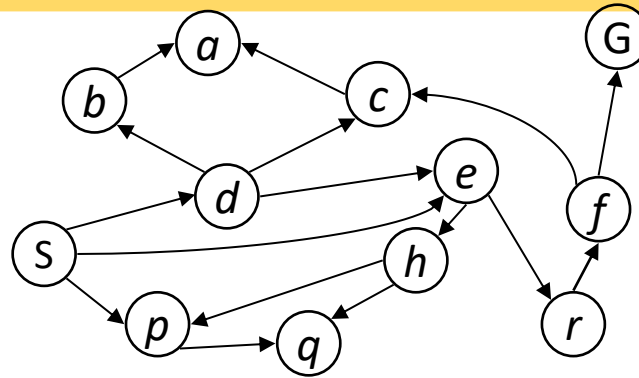
Breadth-First Search (BFS)



Breadth-First Search (BFS)

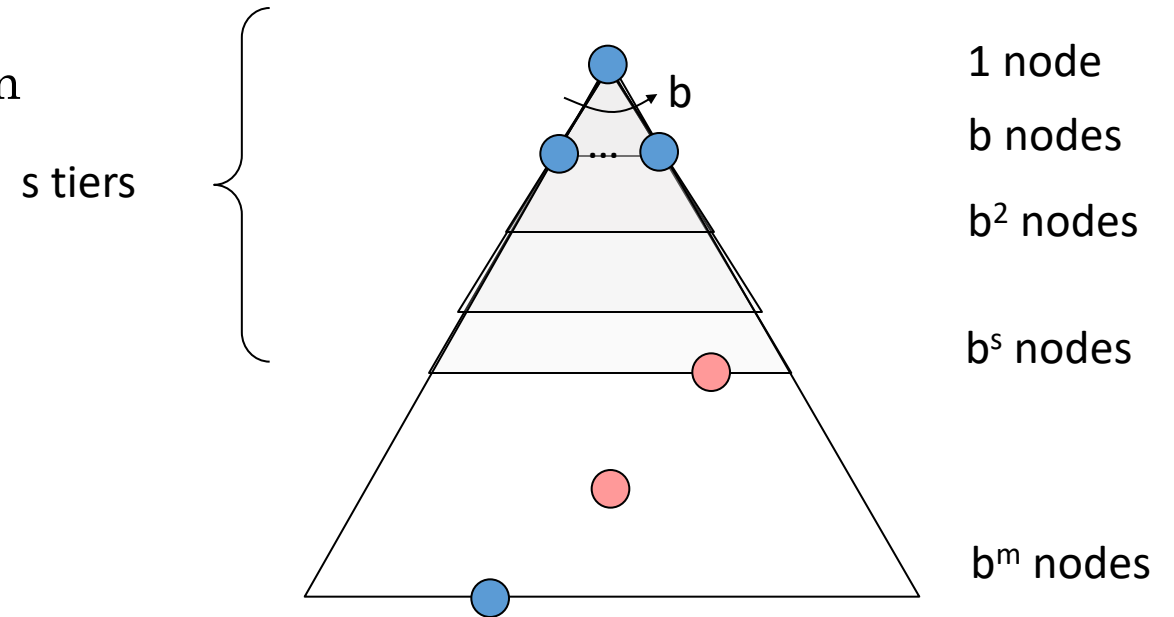
Strategy: expand a shallowest node first

*Implementation:
Fringe is a FIFO queue*



BFS Properties

- What nodes does BFS expand?
 - Processes all nodes above shallowest solution
 - Let depth of shallowest solution be s
 - Search takes time $O(b^s)$
- How much space does the fringe take?
 - $O(b^{s+1})$
- Is it complete?
 - s must be finite if a solution exists, so yes!
- Is it optimal?
 - Only if costs are all 1 (more on costs later)



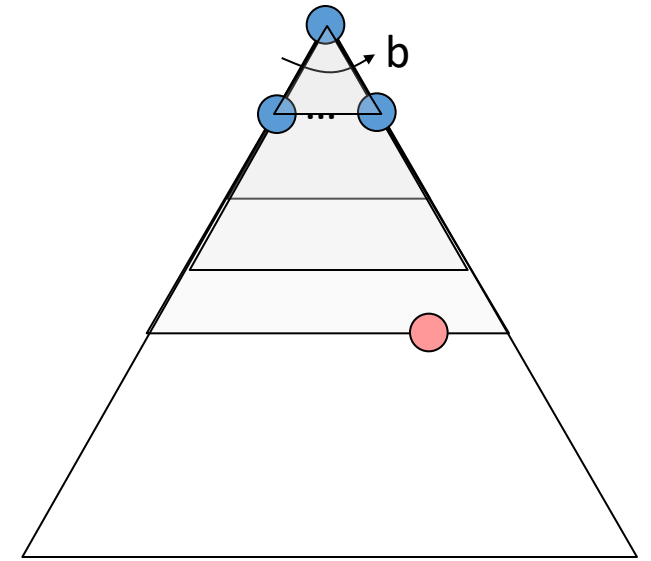
DFS vs BFS

- When will BFS outperform DFS?
- When will DFS outperform BFS?

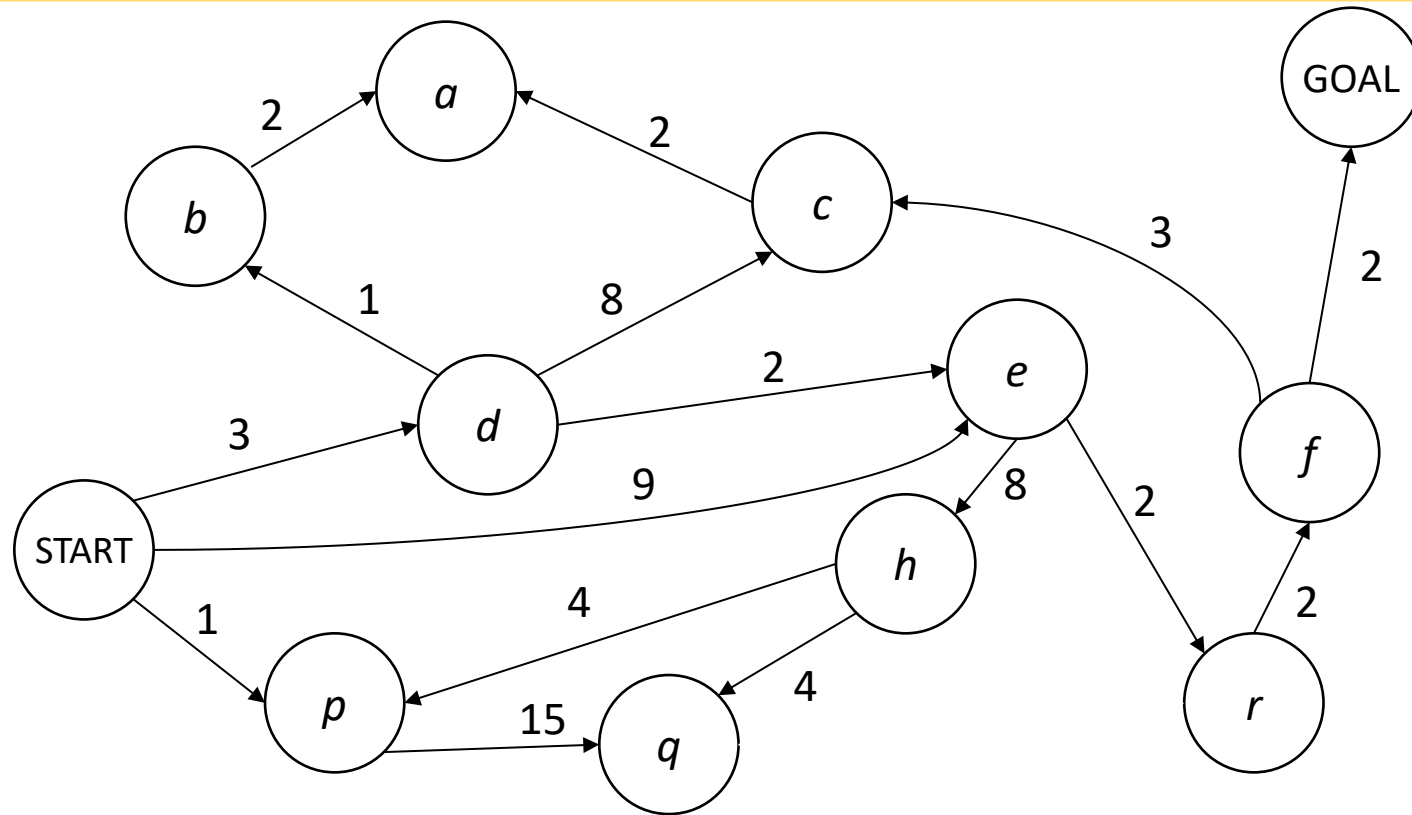


Iterative Deepening

- Idea: get DFS's space advantage with BFS's time / shallow-solution advantages
 - Run a DFS with depth limit 1. If no solution...
 - Run a DFS with depth limit 2. If no solution...
 - Run a DFS with depth limit 3.
- Isn't that wastefully redundant?
 - Generally most work happens in the lowest level searched, so not so bad!

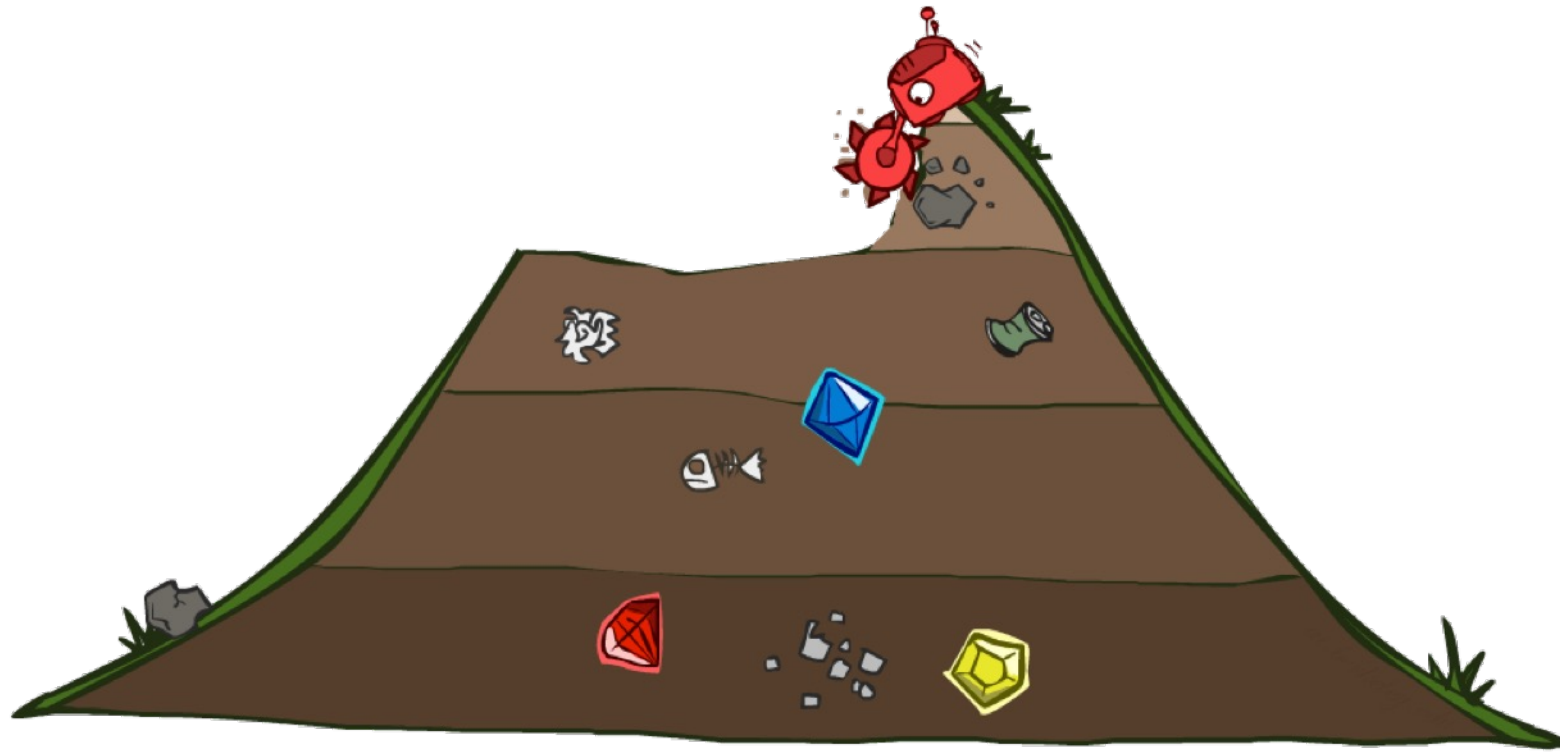


Cost on Actions



Note: BFS finds the shortest path in terms of number of actions. It does not find the least-cost path.

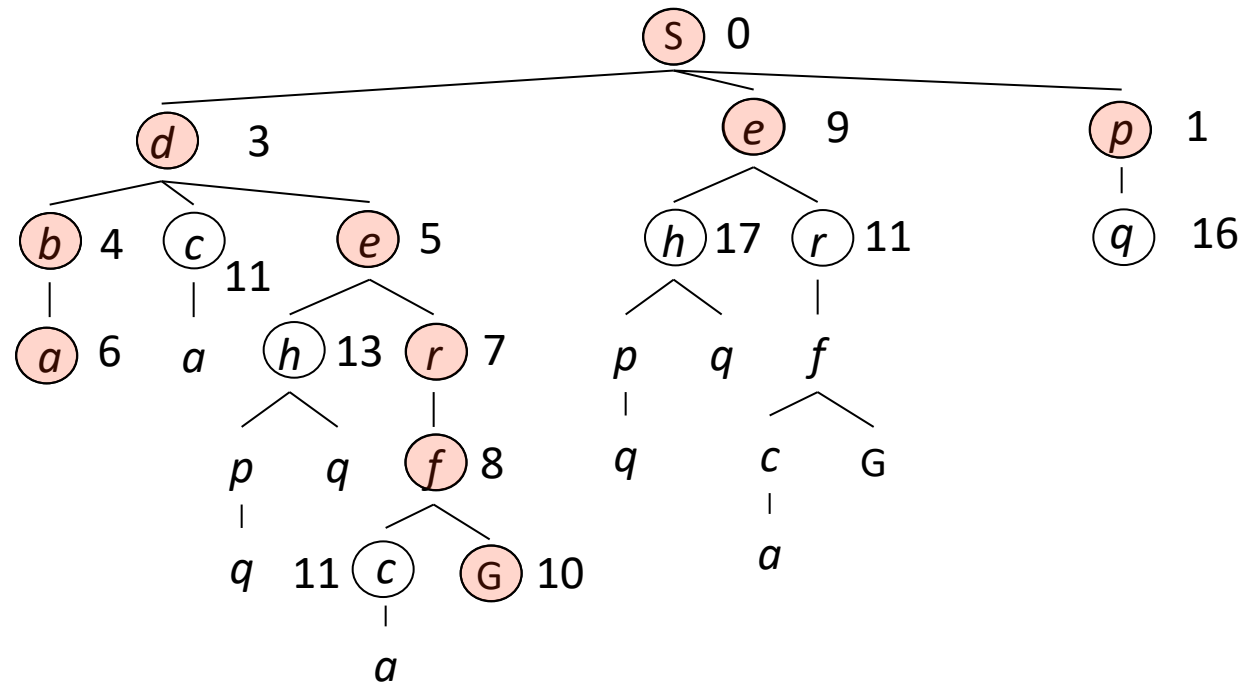
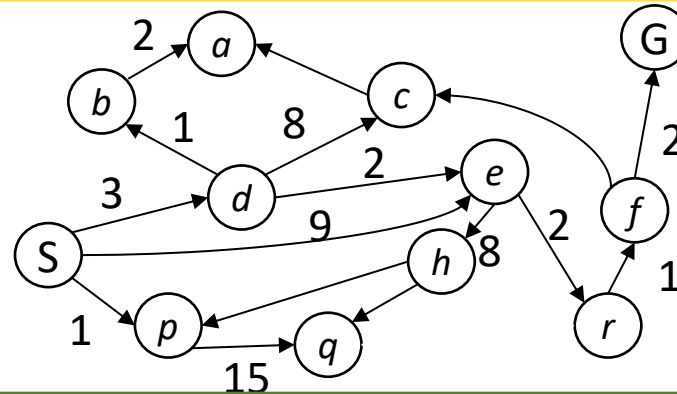
Uniform-Cost Search (UCS)



Uniform-Cost Search (UCS)

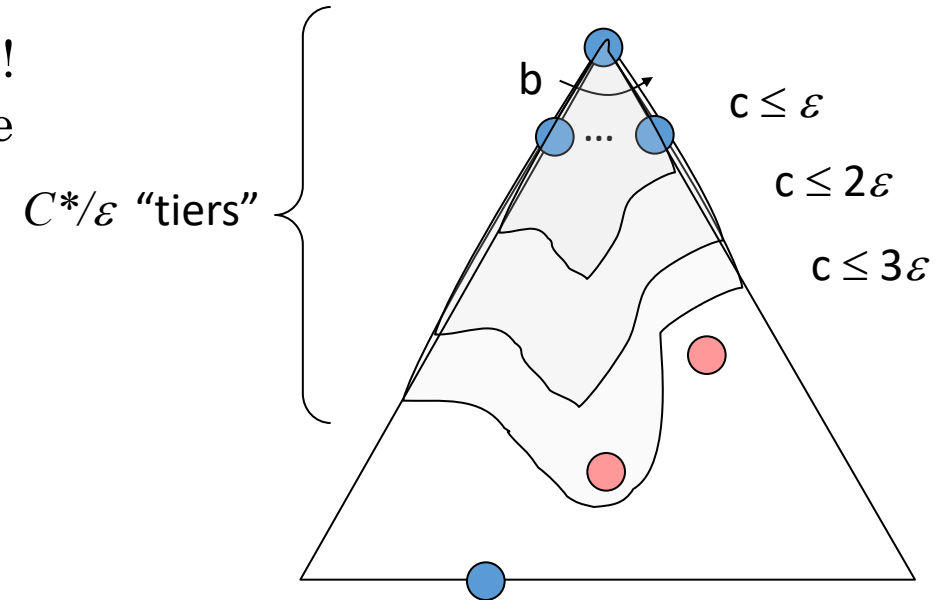
*Strategy: expand a
cheapest node first:*

*Fringe is a priority queue
(priority: cumulative cost)*



UCS Properties

- What nodes does UCS expand?
 - Processes all nodes with cost less than cheapest solution!
 - If that solution costs C^* and arcs cost at least ε , then the “effective depth” is roughly C^*/ε
 - Takes time $O(b^{C^*/\varepsilon})$ (exponential in effective depth)
- How much space does the fringe take?
 - Has roughly the last tier, so $O(b^{C^*/\varepsilon})$
- Is it complete?
 - Assuming best solution has a finite cost and minimum arc cost is positive, yes!
- Is it optimal?
 - Yes!



Uniform Cost Issues

- Remember: UCS explores increasing cost contours
- The good: UCS is complete and optimal!
- The bad:
 - Explores options in every “direction”
 - No information about goal location
- We'll fix that soon!

