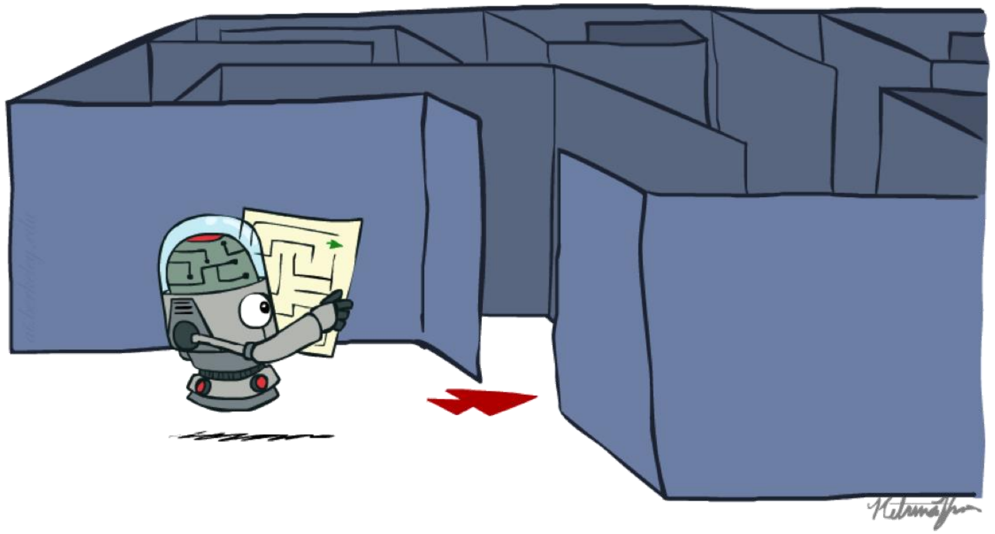
COMS W4701: Artificial Intelligence

Lecture 3: Uninformed Search



Instructor: Tony Dear

*Lecture materials derived from UC Berkeley's AI course at ai.berkeley.edu

Announcements

- TA office hours started this week
- Review sessions held on Fridays 1-2pm, 4-5pm

- HW0 submission for wait list students
- Course is still open for auditing
- 4701 will likely be offered again next semester

Last Time

- Characteristics of agents
- Characteristics of environments
- Search problem formalism

Search Problems

- State space: All possible descriptions of the agent in the world
- Actions: All possible actions an agent can take from current state
  - *Actions*(s), $s \in$ state space
- Transition model: Function mapping current state + action to a new state
  - *Results*: $(s_1, a) \rightarrow s_2,\ a \in$ *Actions*(s_1)
- Path costs
- Start state and goal test
- Solution: Sequence of actions going from start state to goal state

Example: Grid Path Finding

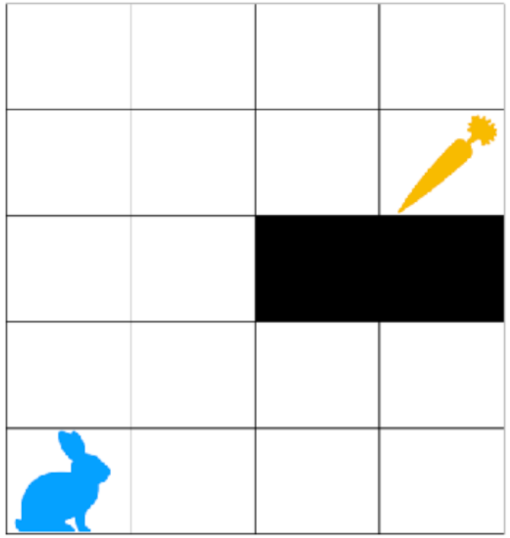
- State space: $\{(x, y) \mid 0 \leq x \leq 3, 0 \leq y \leq 4\}$ (what's the space size?)
- Initial state: $(0,0)$
- Actions: $Actions(x, y) =$

  {Up if $y < 4$ and $(x, y + 1)$ is not a wall,

  {Down if $y > 0$ and $(x, y - 1)$ is not a wall,

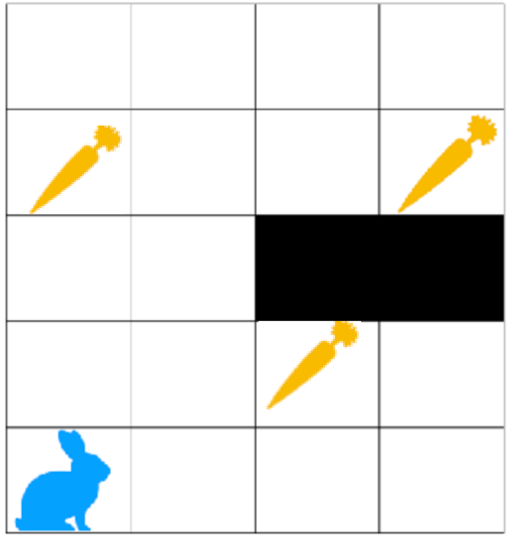
  {Left if $x > 0$ and $(x - 1, y)$ is not a wall,

  {Right if $x < 3$ and $(x + 1, y)$ is not a wall}



- Transition model: $Result((x, y), \text{Up}) = (x, y + 1)$, $Result((x, y), \text{Down}) = \cdots$
- Goal test: In($(3,3)$)?
- Path costs: Unit cost per step taken or per time step

Multiple Carrots?



- What has changed about the problem?
- Agent (rabbit) description is the same
- State space
  - Location of rabbit, Booleans indicating carrots
- Transition model
  - Update both rabbit location as well as carrot Boolean if locations match
- Goal test
  - Are all carrots eaten? Are all Boolean indicators switched?
- New state space size?

Search Problems Are Models

Today

- Graph and tree representations

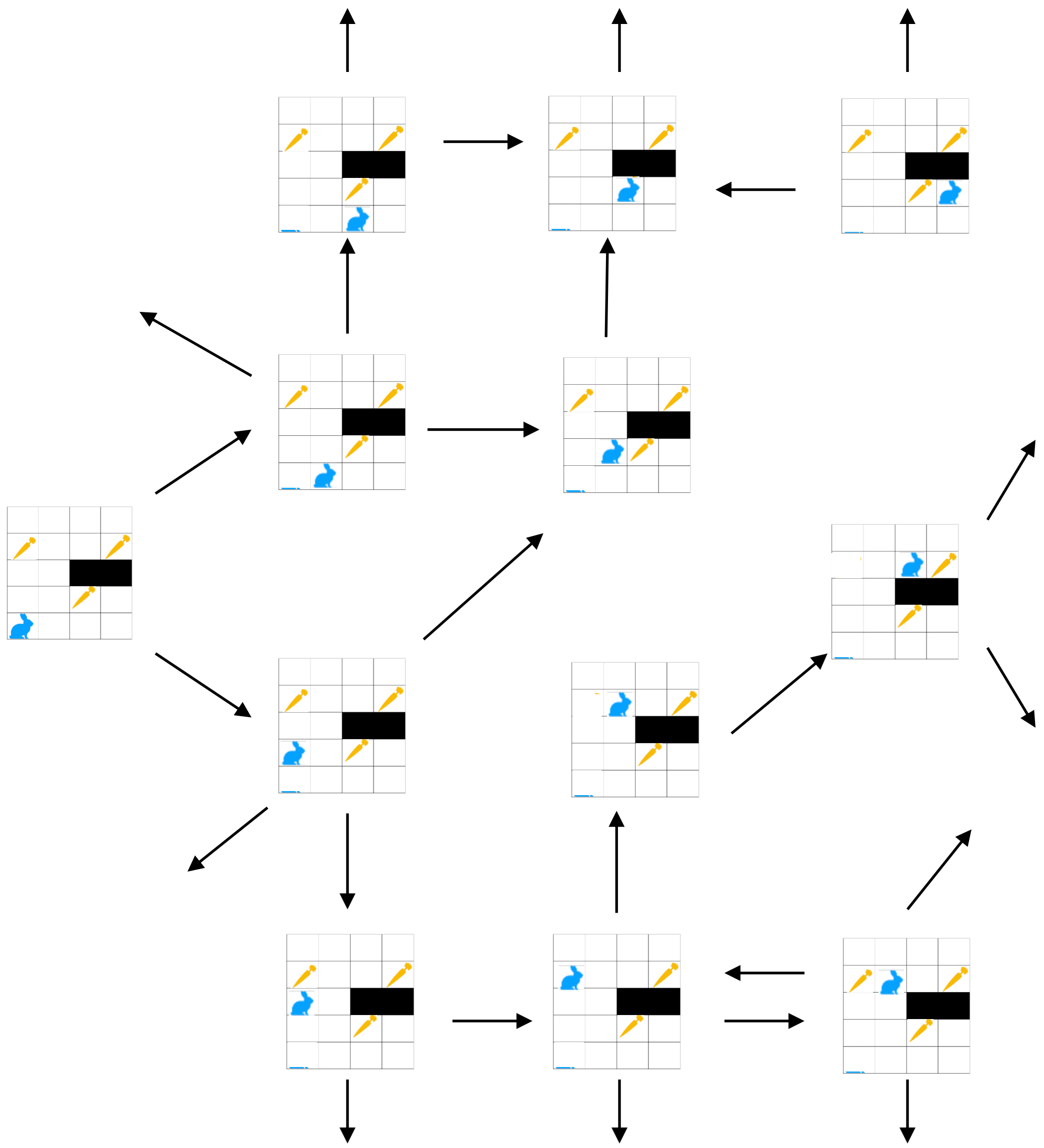
- Tree search strategies
  - Depth-first search
  - Breadth-first search
  - Uniform-cost search

State Space Graphs

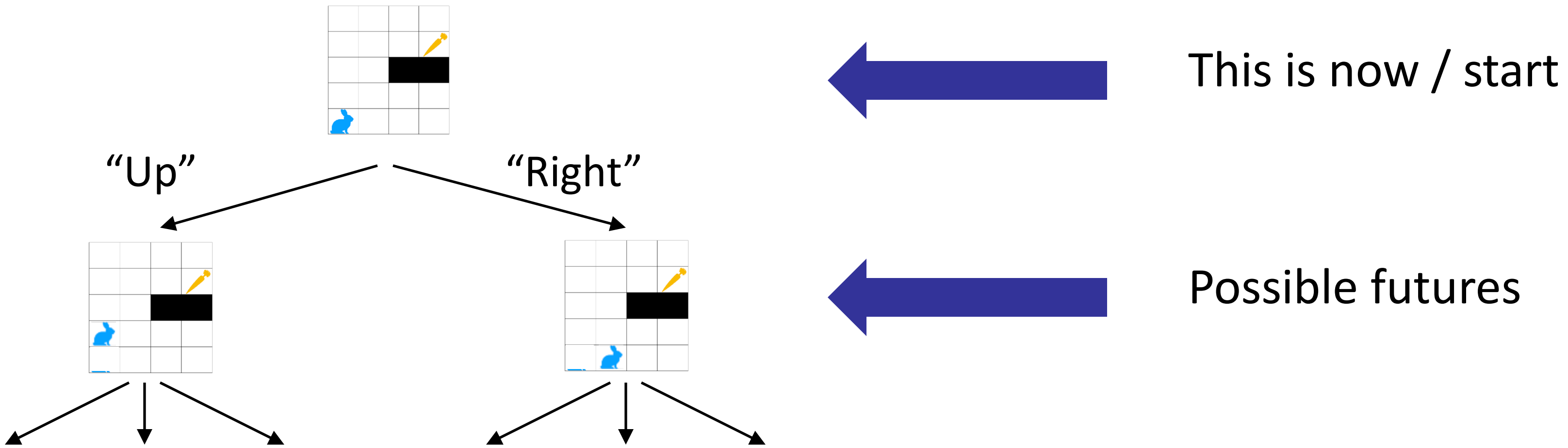
- State space graph: A mathematical representation of a search problem
  - Vertices are (abstracted) world configurations
  - Edges represent action results
  - The goal test is a set of goal nodes (maybe only one)

- Each state occurs only once!

- We can rarely build this full graph in memory (it's too big), but it's a useful idea



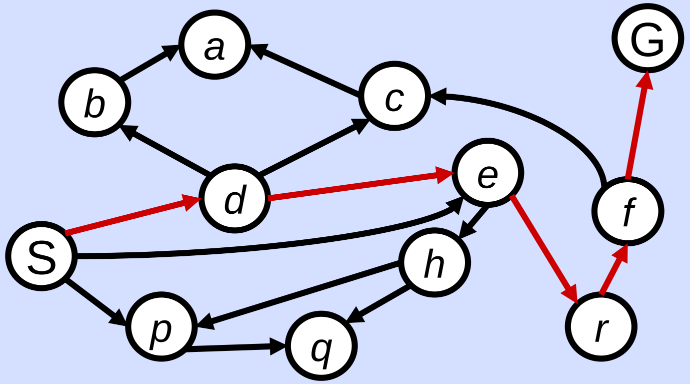
Search Trees



- A search tree of plans and outcomes:
  - The start state is the root node
  - Children correspond to successors
  - Nodes correspond to PLANS that achieve the states shown
  - For most problems, we can never actually build the whole tree

State Space Graphs vs. Search Trees

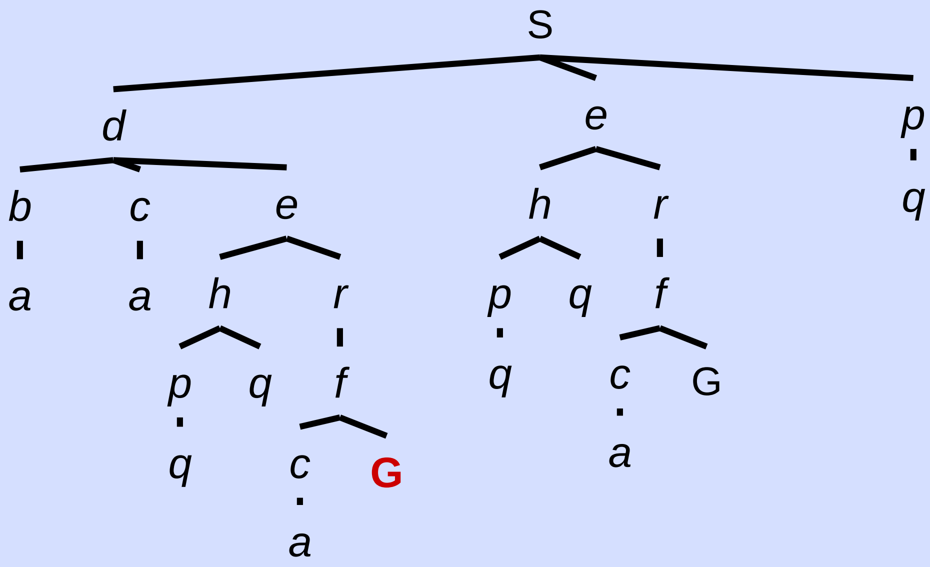
State Space Graph



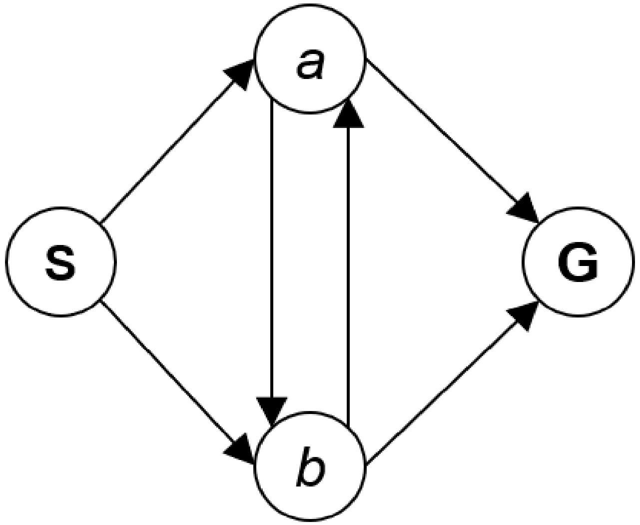
Each NODE in in the search tree is an entire PATH in the state space graph.

We construct both on demand – and we construct as little as possible.

Search Tree



How big is the search tree for this graph?



Two

Four

Five

Infinite

General Tree Search

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

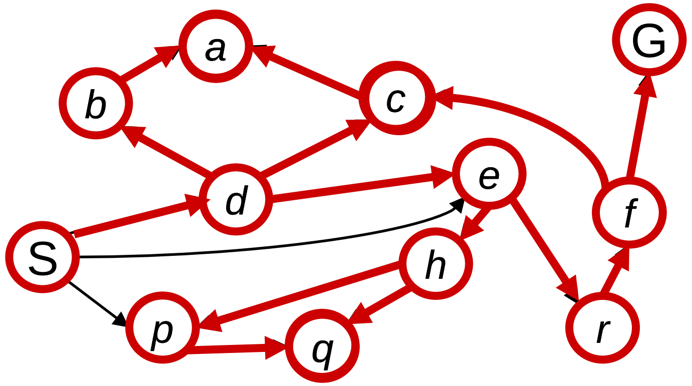
- Important ideas:
  - Fringe
  - Expansion
  - Exploration strategy
- Main question: which fringe nodes to explore?

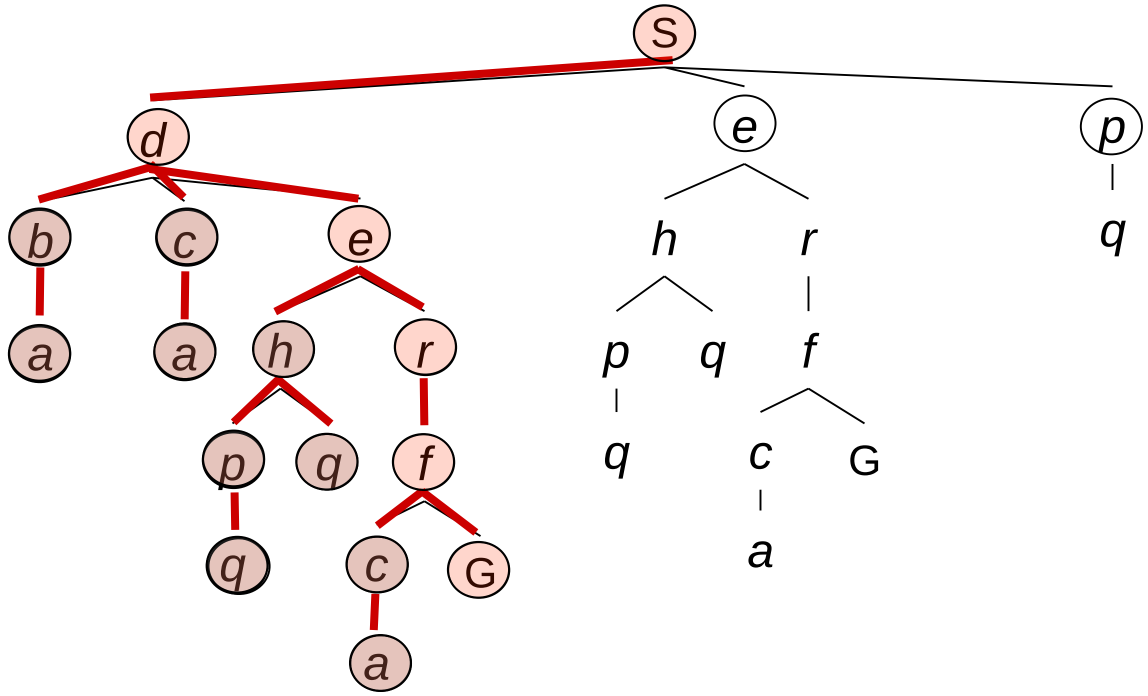
Depth-First Search

Depth-First Search

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack



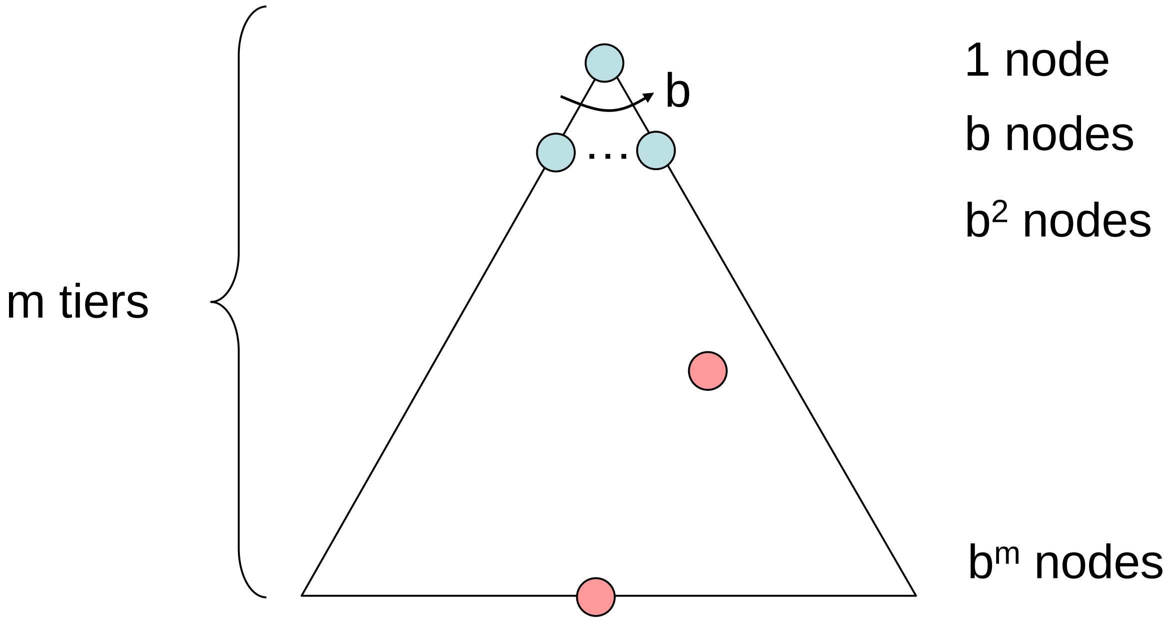


Search Algorithm Properties

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

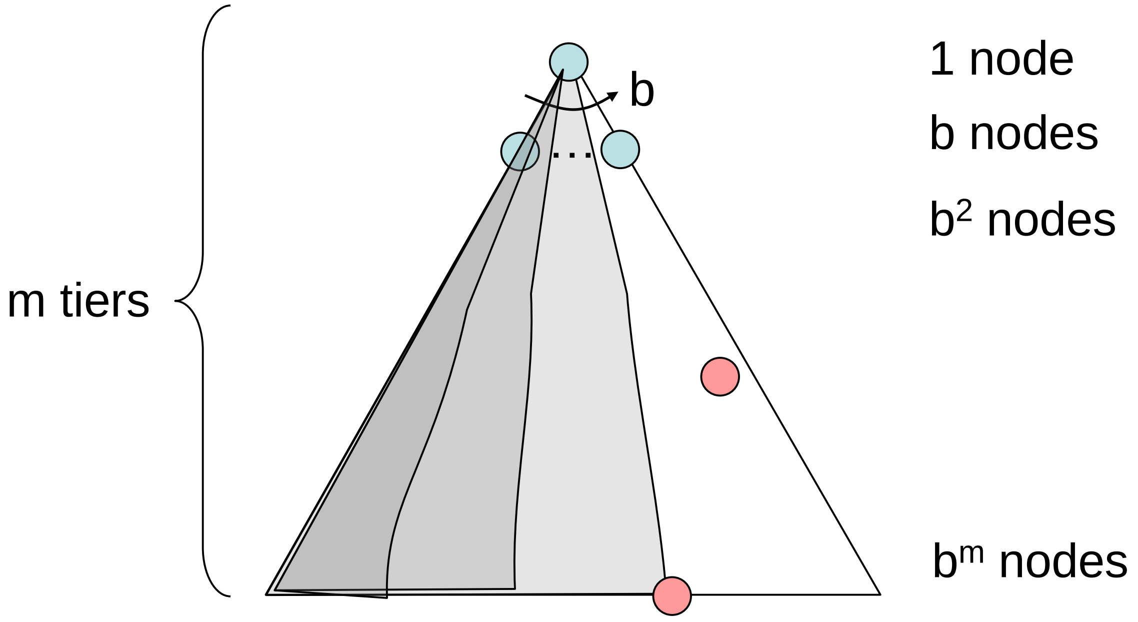
- Cartoon of search tree:
  - b is the branching factor
  - m is the maximum depth
  - solutions at various depths

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

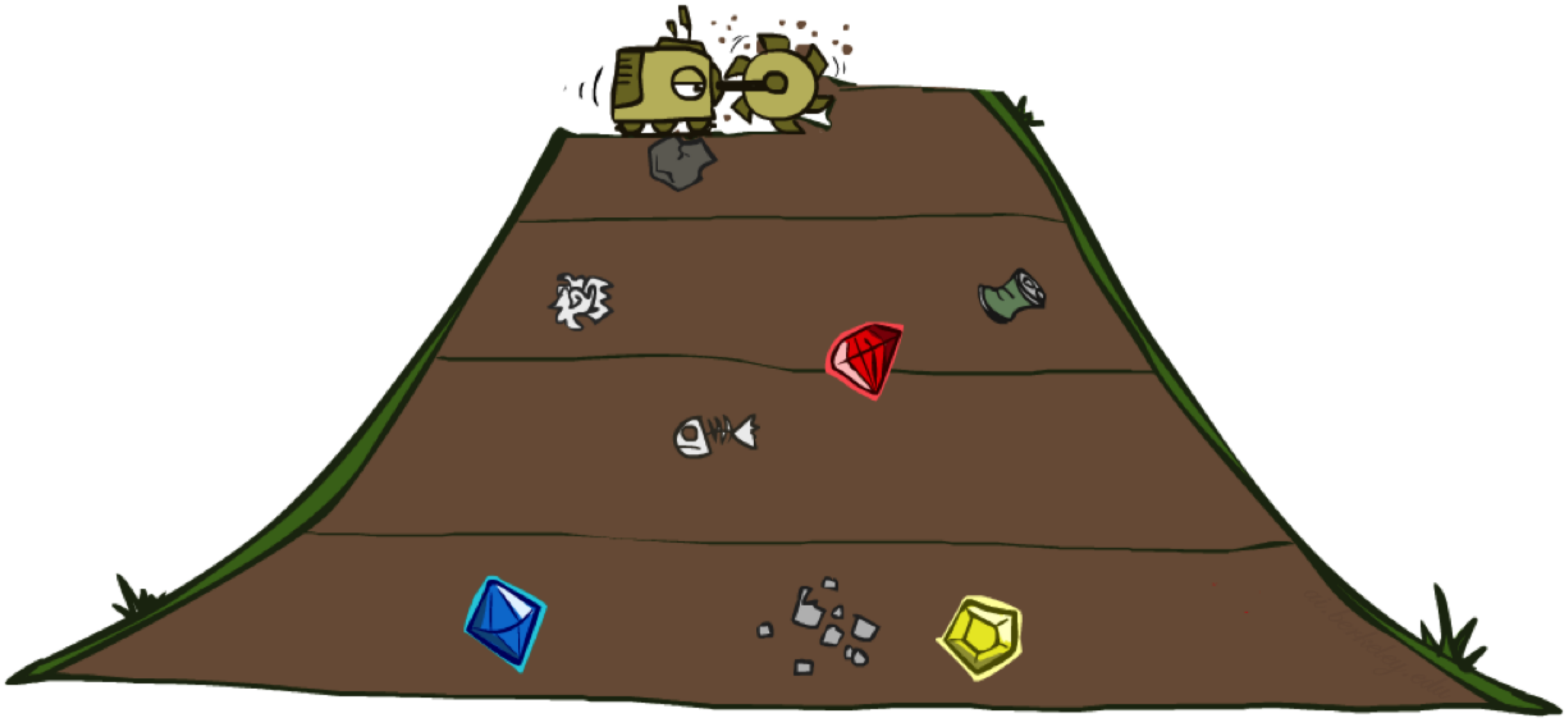


Depth-First Search (DFS) Properties

- 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



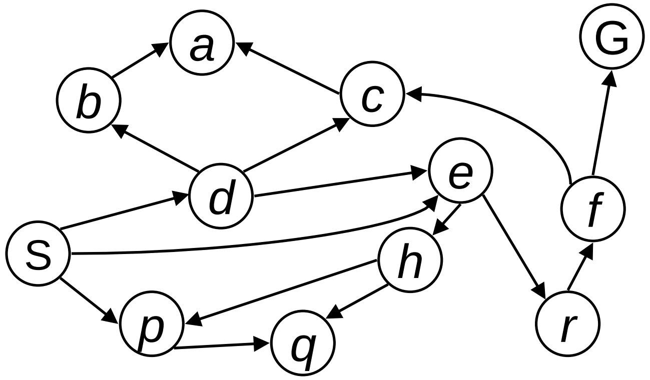
Breadth-First Search

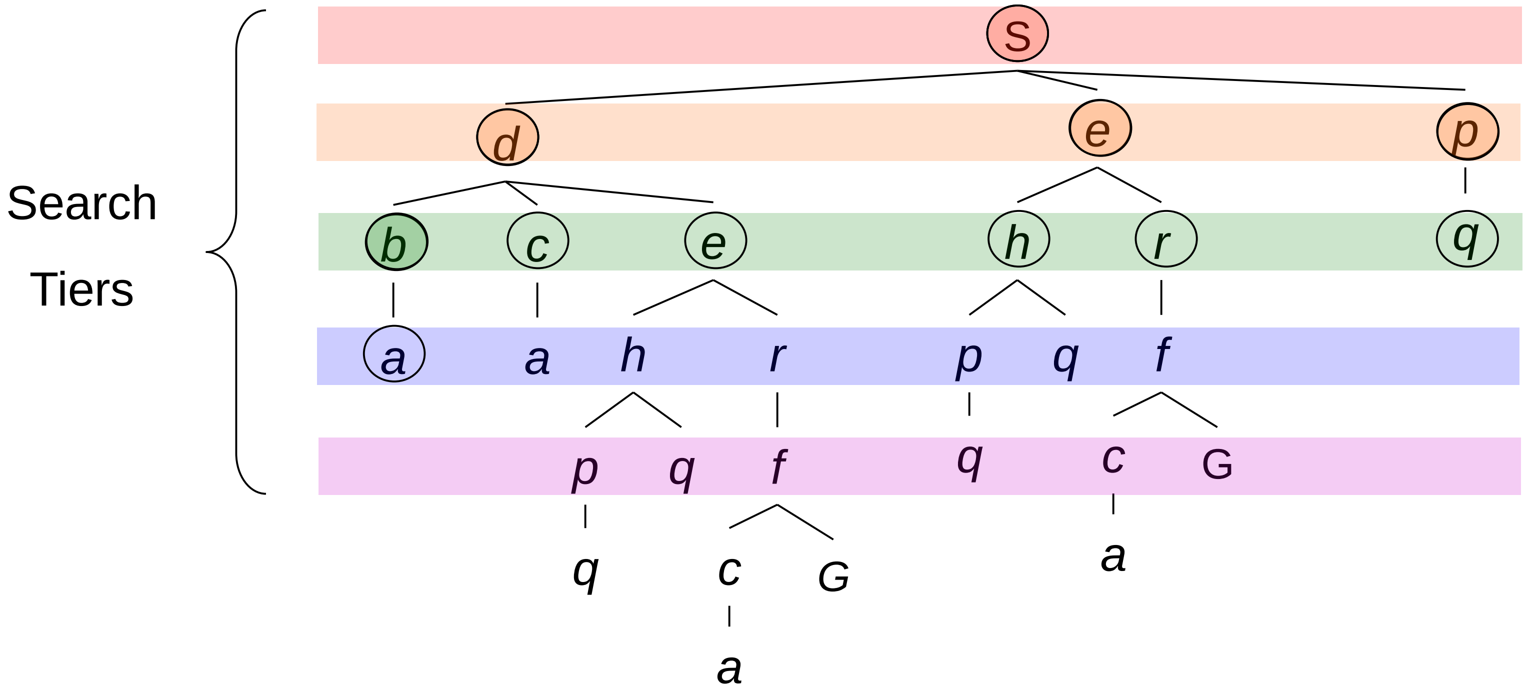


Breadth-First Search

Strategy: expand a shallowest node first

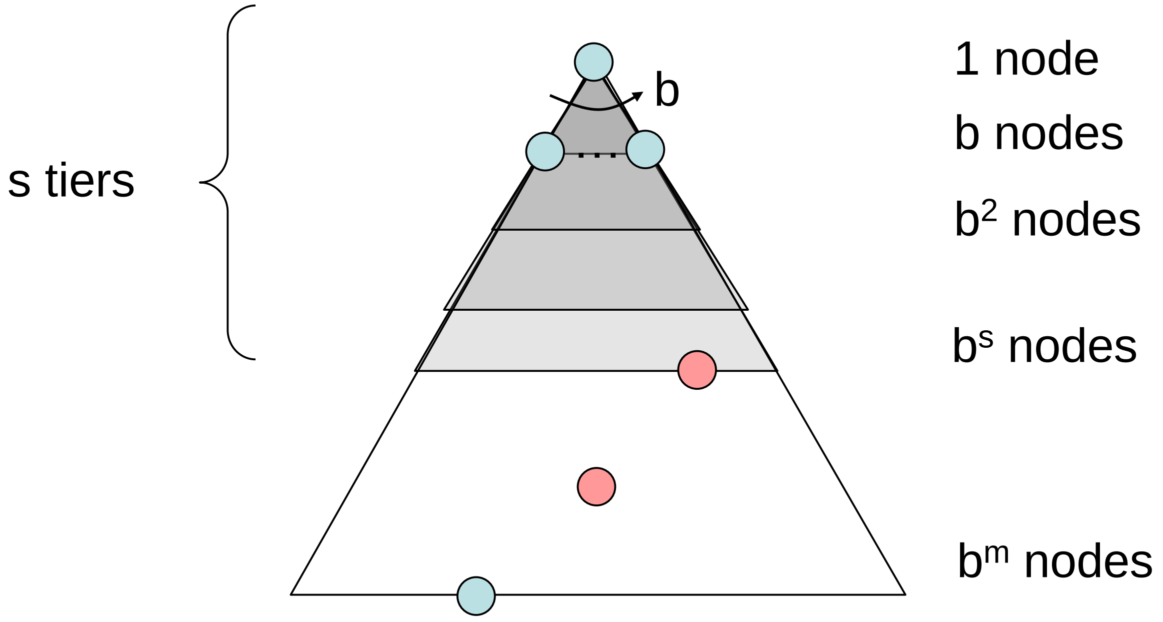
Implementation: Fringe is a FIFO queue



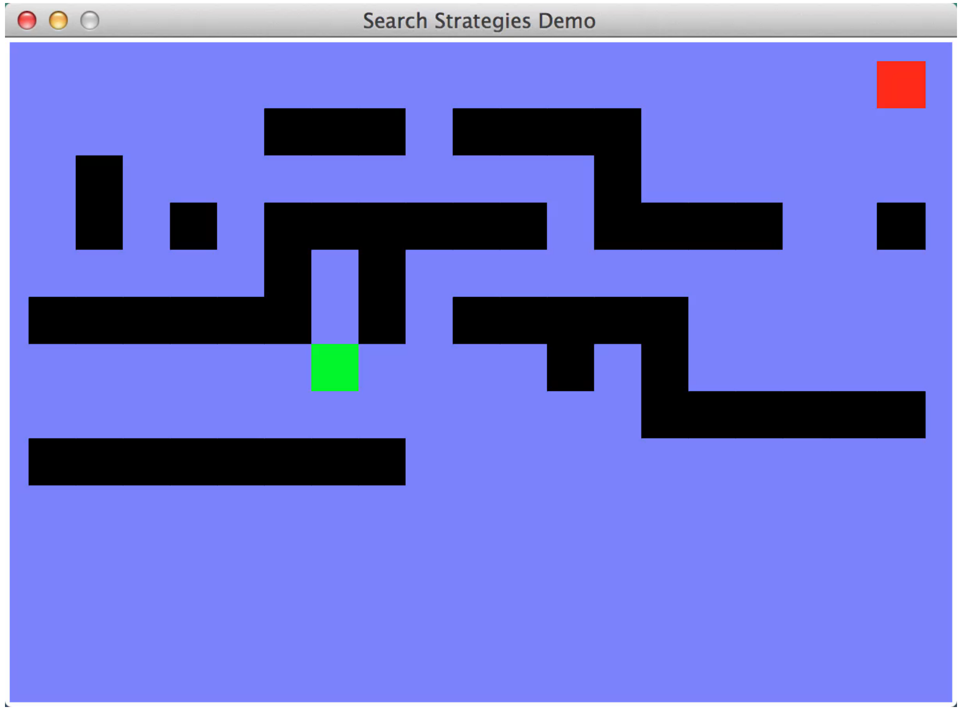


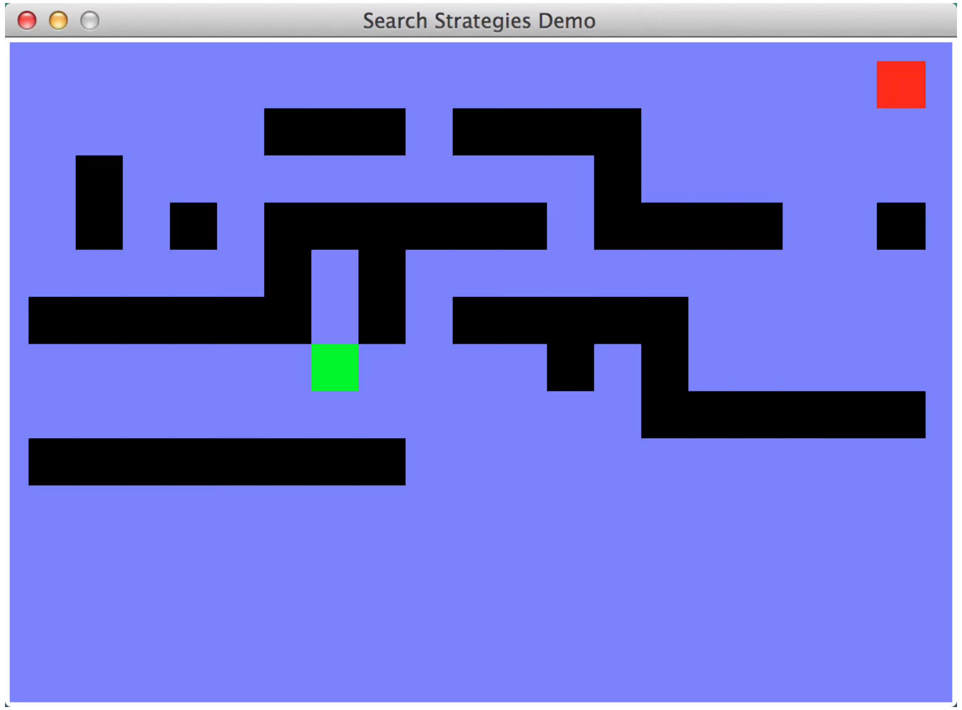
Breadth-First Search (BFS) Properties

- 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?
  - Has roughly the last tier, so $O(b^s)$
- 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

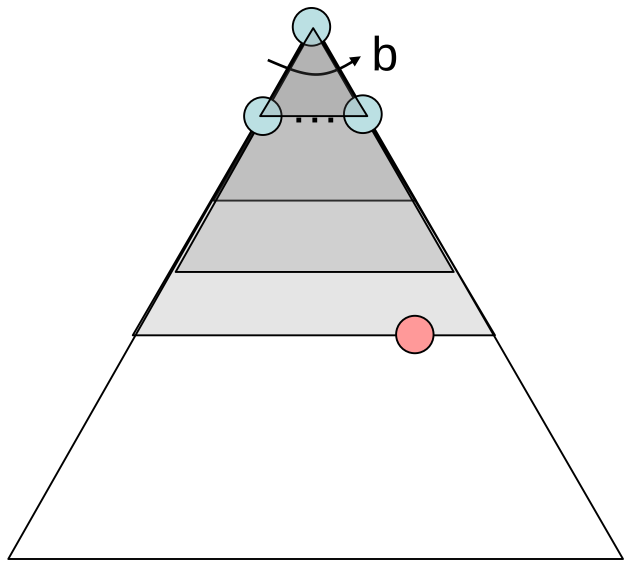




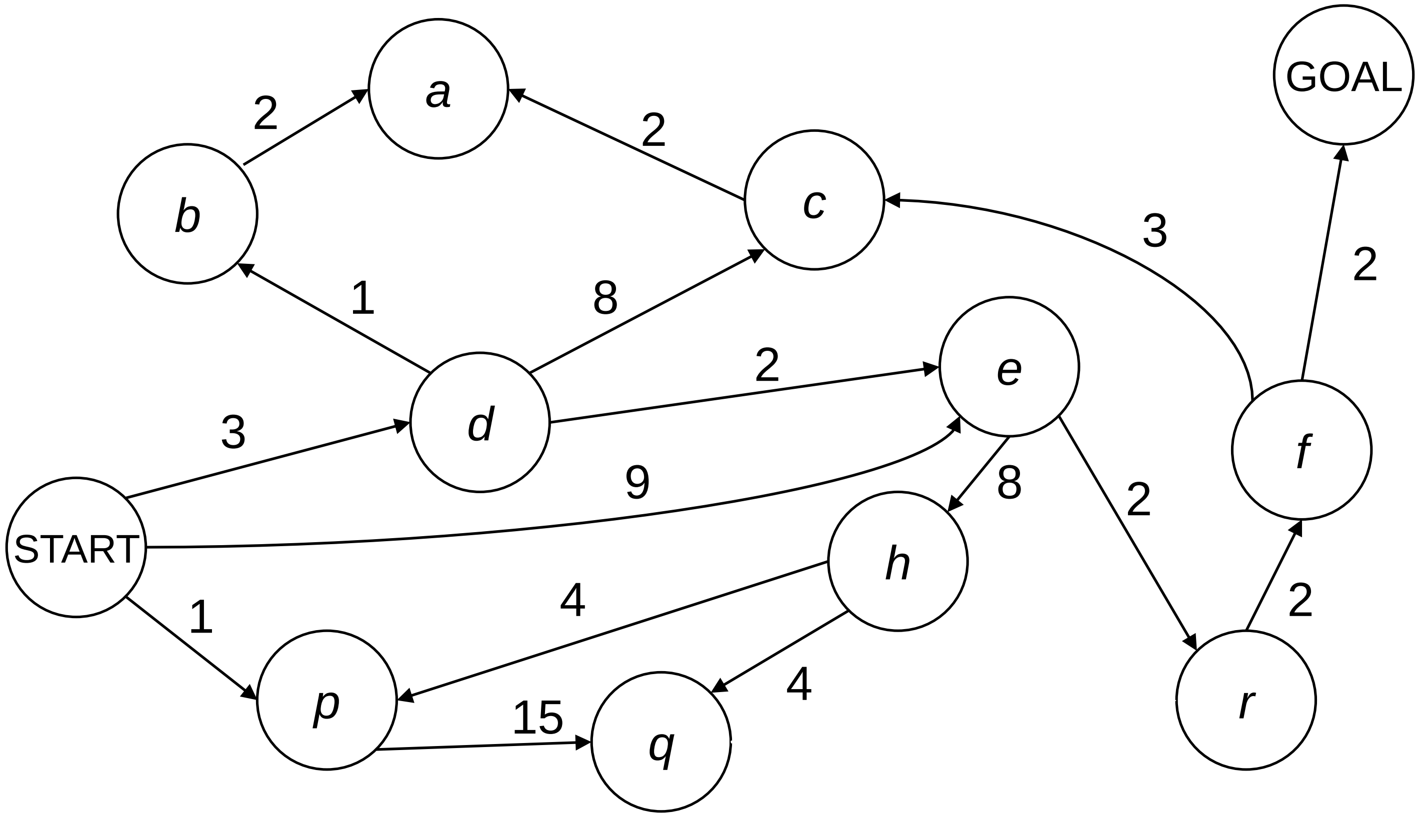
Iterative Deepening

- 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. …..

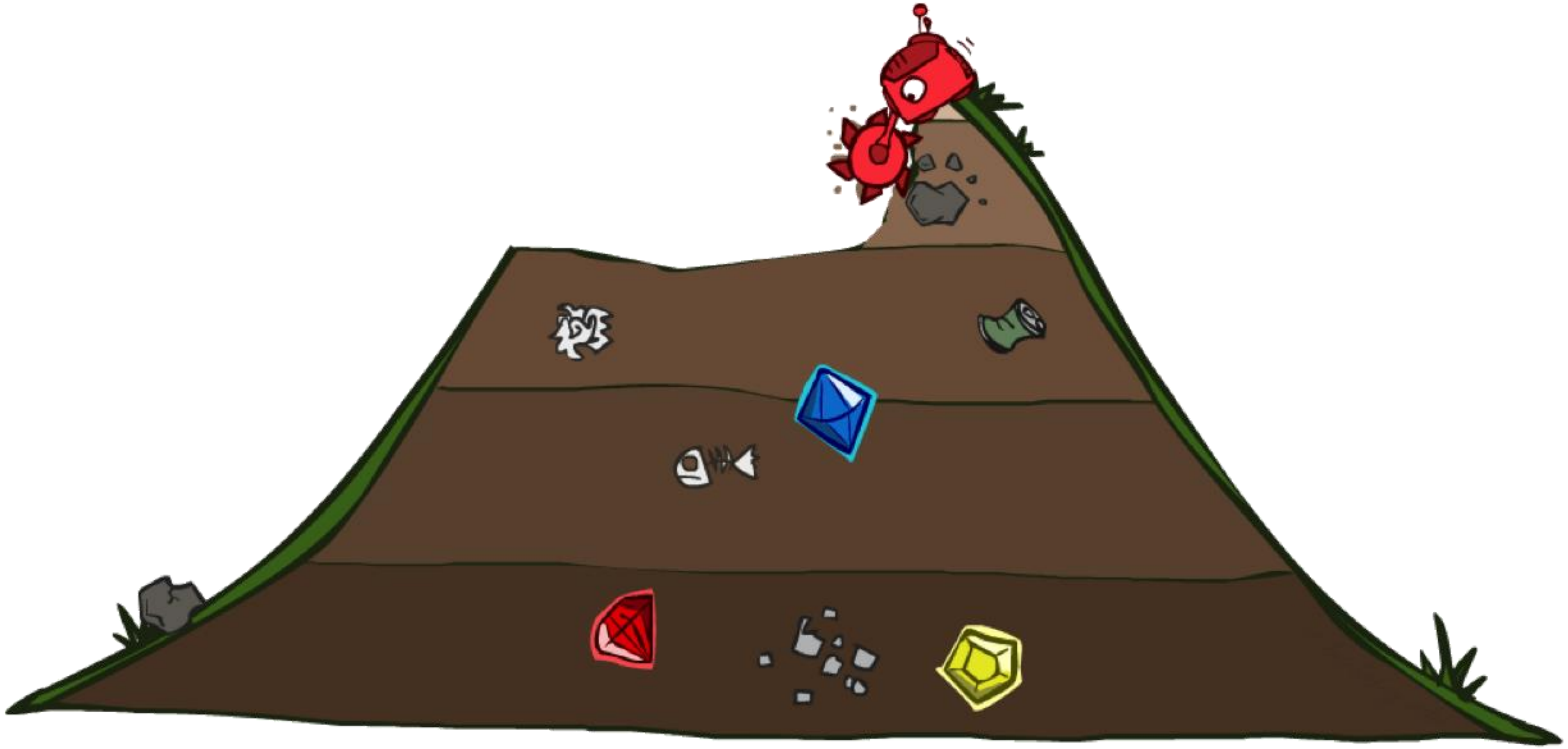
- Isn't that wastefully redundant?
  - Generally most work happens in the lowest level searched, so not so bad!



Cost-Sensitive Search



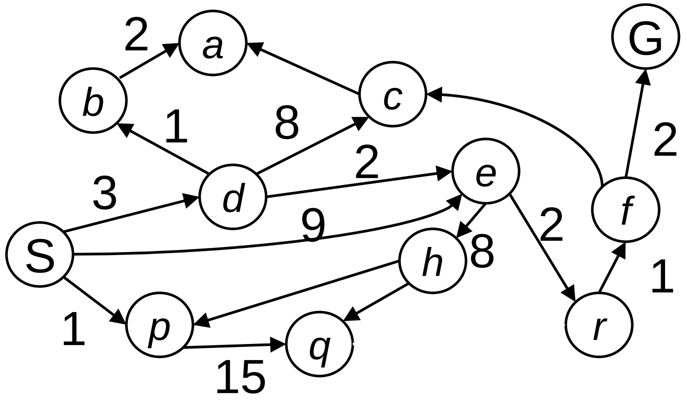
Uniform Cost Search

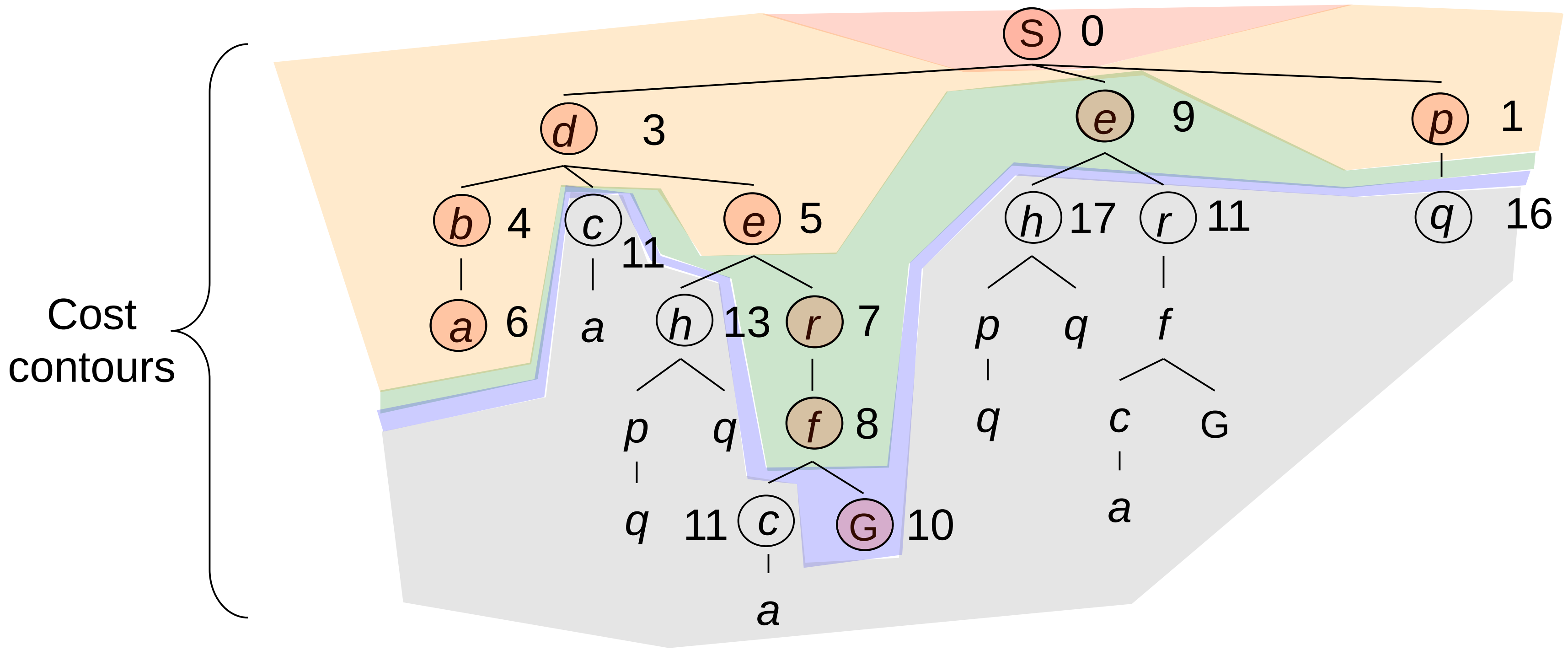


Uniform Cost Search

Strategy: expand a cheapest node first

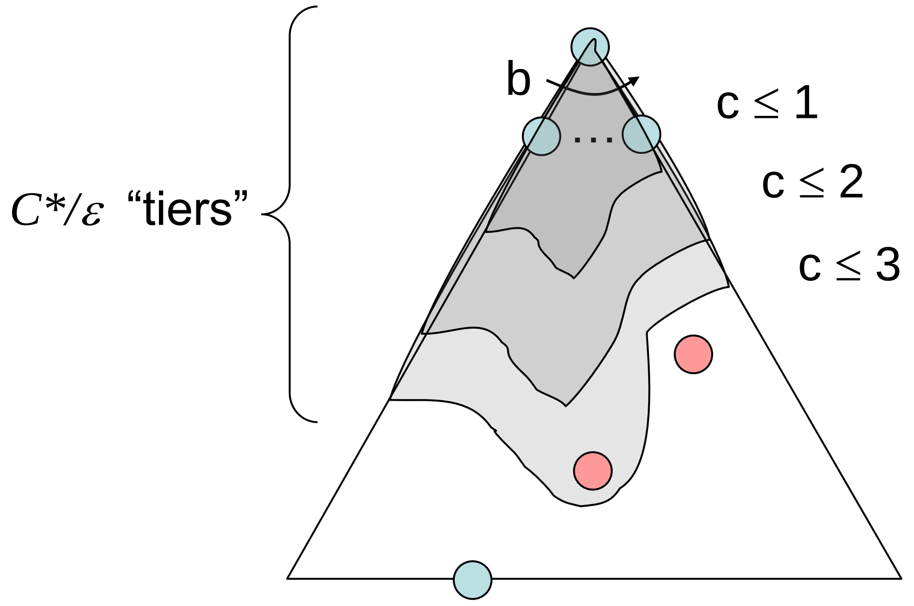
Fringe is a priority queue (priority: cumulative cost)





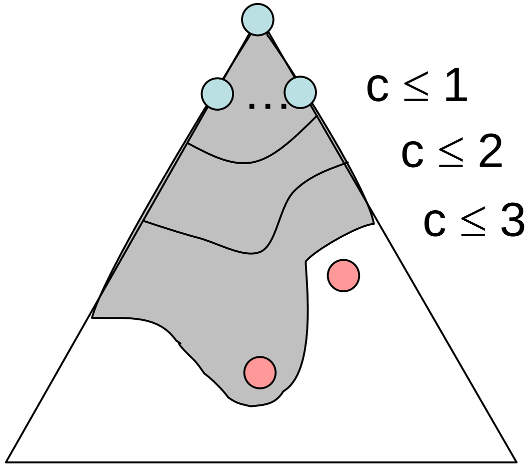
Uniform Cost Search (UCS) Properties

- 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! (Proof next lecture via A*)



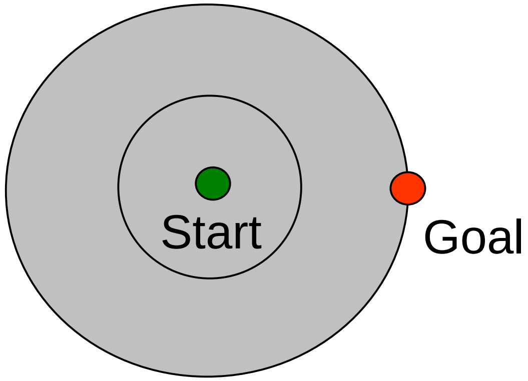
Uniform Cost Issues

- Remember: UCS explores increasing cost contours



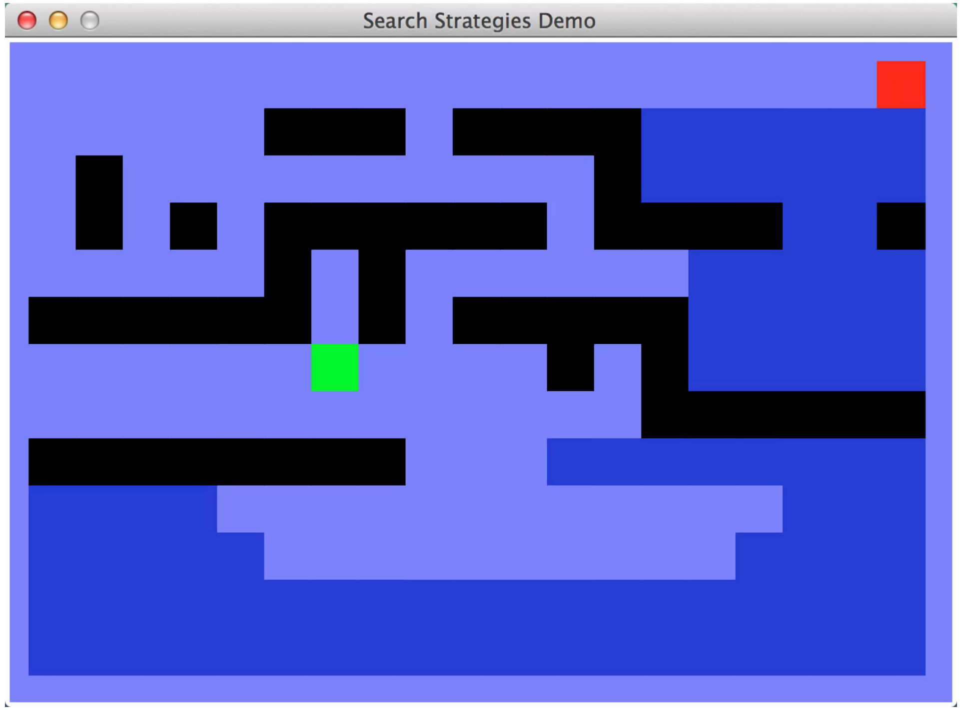
- The good: UCS is complete and optimal!

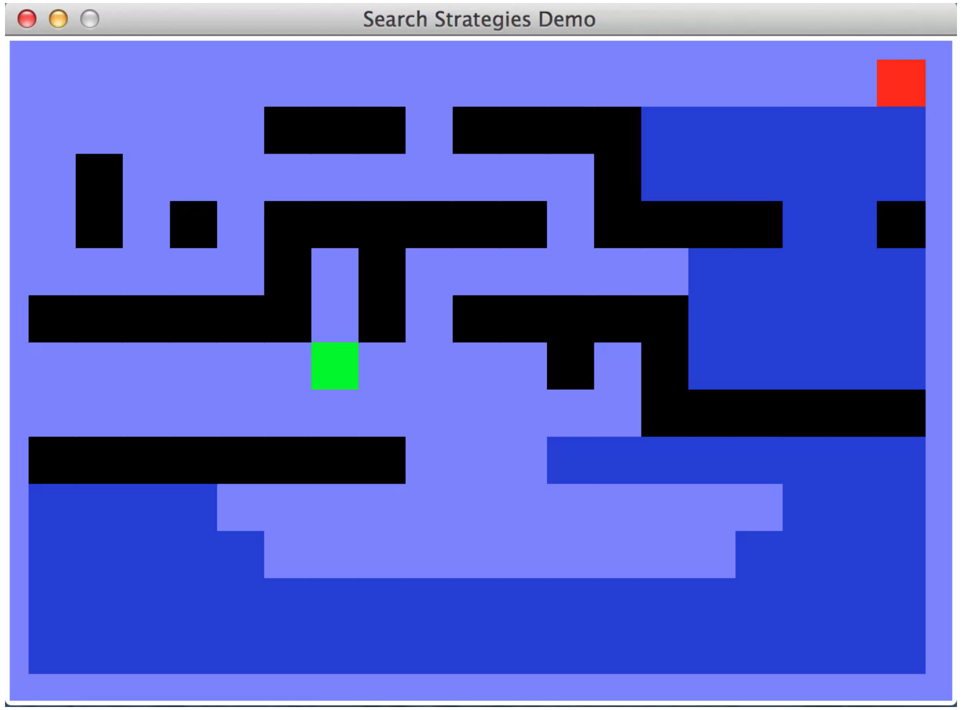
- The bad:
  - Explores options in every "direction"
  - No information about goal location



- We'll fix that soon!

UCS in Action





Recap

- Search problems represented as graphs and trees, which we construct as little as possible

- Tree search: Maintain a **fringe** of nodes to explore and expand
  - DFS: Neither complete nor optimal; exponential time, polynomial space
  - BFS: Complete but not optimal; exponential time, exponential space
  - Iterative deepening: Complete but not optimal; exponential time, polynomial space
  - UCS: Complete and optimal; exponential time, exponential space