Lecture 09
Search Problems

Song Chaoyang
Assistant Professor
Department of Mechanical and Energy Engineering
songcy@sustech.edu.cn

[Slides adapted from Sergey Levine & Stuart Russell, CS188, UCB]

SUSTech
Southern University of Science and Technology

AncoraSIR.com

Problem-Solving Agents

Problem Formulation

- A problem is defined by five components:
  - **Initial state**: e.g., “at Arad”
  - **Actions**: A description of the possible actions available to the agent.
  - **Transition model**: A function RESULT(s, a) that returns the state that results from doing action a in state s.
  - **Goal test**: Determines whether a given state is a goal state.
  - **Path cost** (additive): assigns a numeric cost to each path, e.g., sum of distances, number of actions executed, etc.

- **Solution**: a sequence of actions leading from the initial state to a goal state.

Problem-Solving Agents

A Simple Example

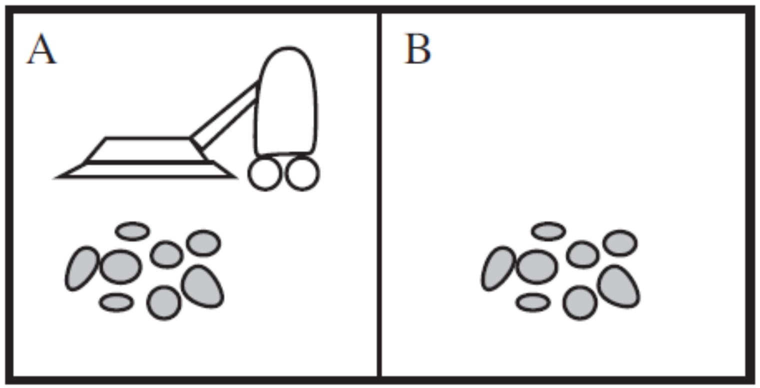
```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  persistent: seq, an action sequence, initially empty
              state, some description of the current world state
              goal, a goal, initially null
              problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then
     goal ← FORMULATE-GOAL(state)
     problem ← FORMULATE-PROBLEM(state, goal)
     seq ← SEARCH(problem)
     if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

AncoraSIR.com

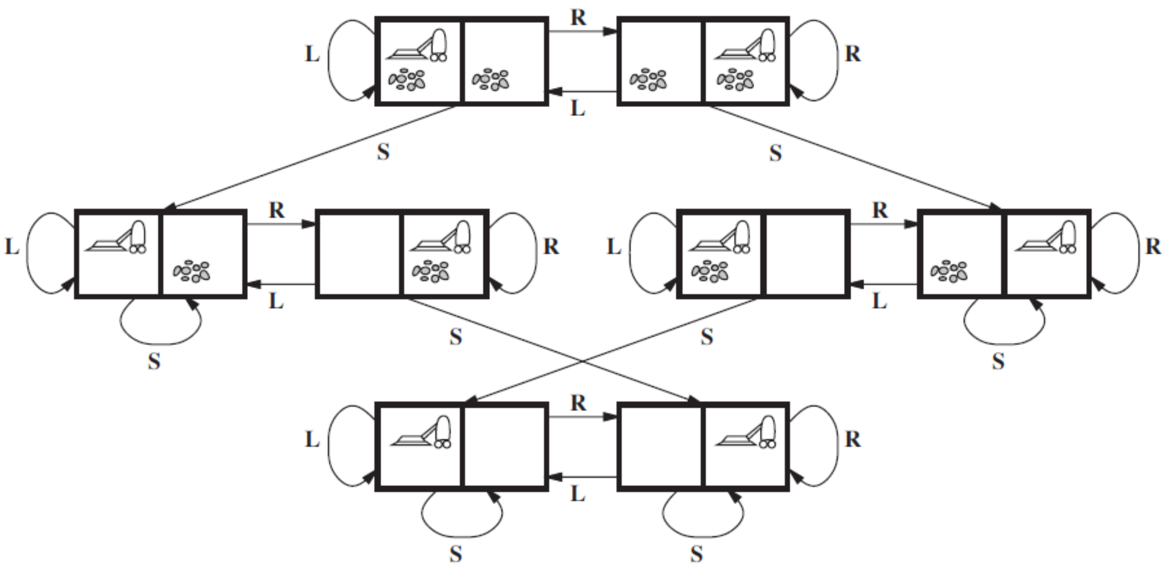
SUSTech Southern University of Science and Technology

Example of Vacuum World Problems

- **States**: The state is determined by both the agent location and the dirt locations. 8 possible world states.
- **Initial state**: Any state can be designated as the initial state.
- **Actions**: Each state has three actions: *Left*, *Right*, and *Suck*.
- **Transition model**: The actions have their expected effects.
- **Goal test**: This checks whether all the squares are clean.
- **Path cost**: Each step costs 1, so the path cost is the number of steps in the path.



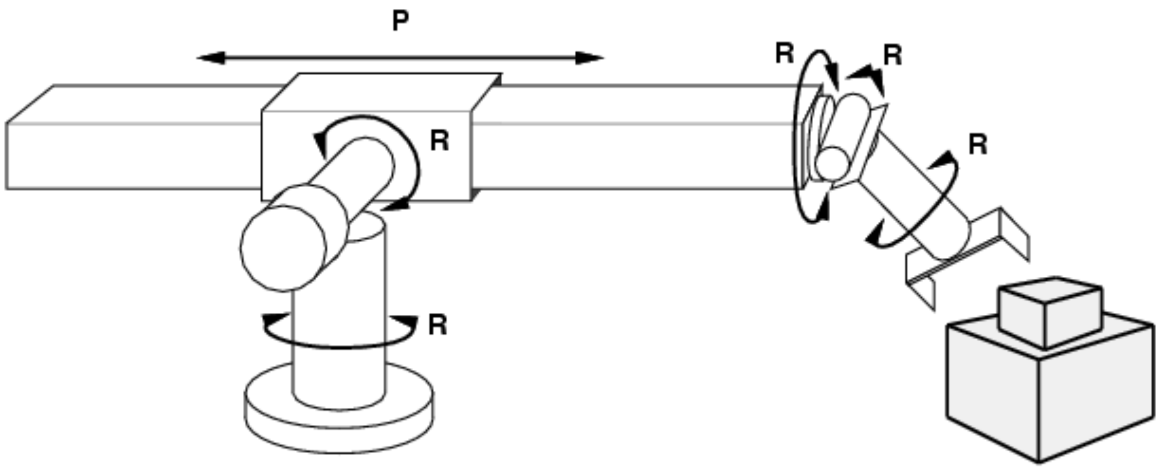
AncoraSIR.com



Example Problems

Robotic Assembly

- **States**: Real-valued coordinates of robot joint angles, parts of the object to be assembled
- **Initial state**: Any state can be designated as the initial state
- **Actions**: Continuous motions of robot joints
- **Transition model**: The actions have their expected effects.
- **Goal test**: Complete assembly
- **Path cost**: Time to execute

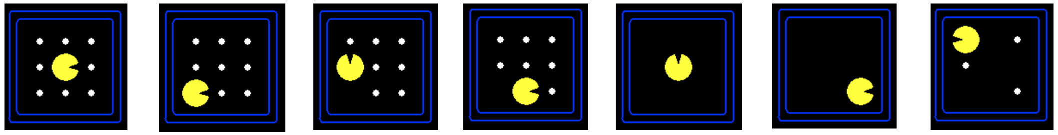


AncoraSIR.com

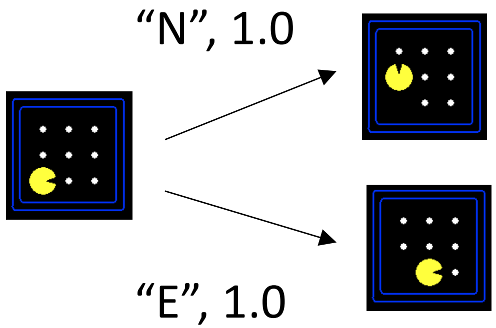
Search Problems

Search Problems Are Models

- A search problem consists of:
  - A state space



  - A successor function (with actions, costs)

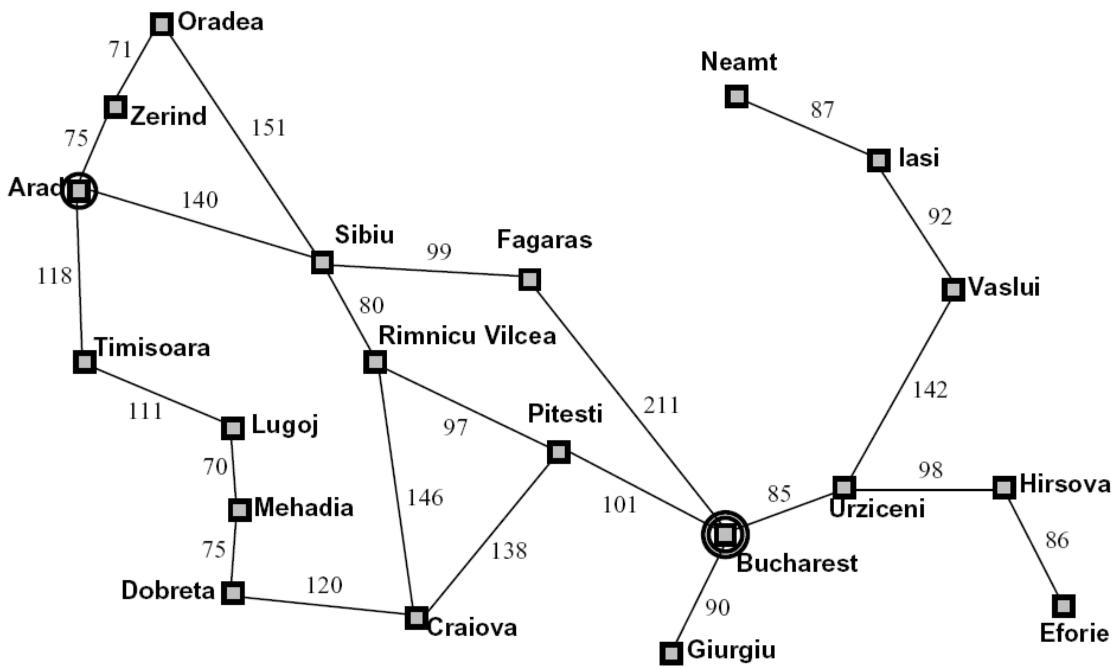


  - A start state and a goal test

- A solution is a sequence of actions (a plan) which transforms the start state to a goal state

Traveling in Romania

Example

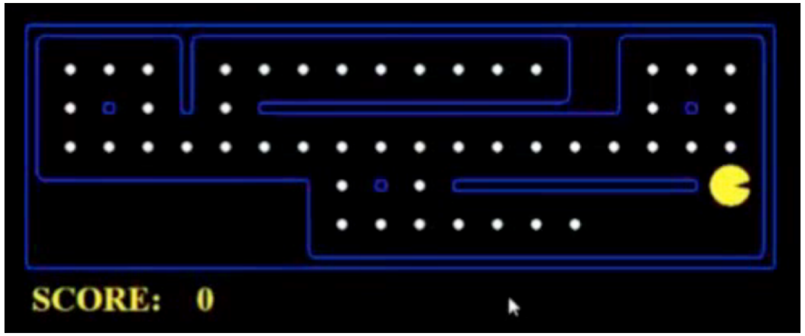


- State space:
  - Cities
- Successor function:
  - Roads: Go to adjacent city with cost = distance
- Start state:
  - Arad
- Goal test:
  - Is state == Bucharest?
- Solution?

AncoraSIR.com

What's in a State Space?

- The ***world state*** includes every last detail of the environment



- A ***search state*** keeps only the details needed for planning (abstraction)

Wrong example for "eat-all-dots": (x, y, dot count)

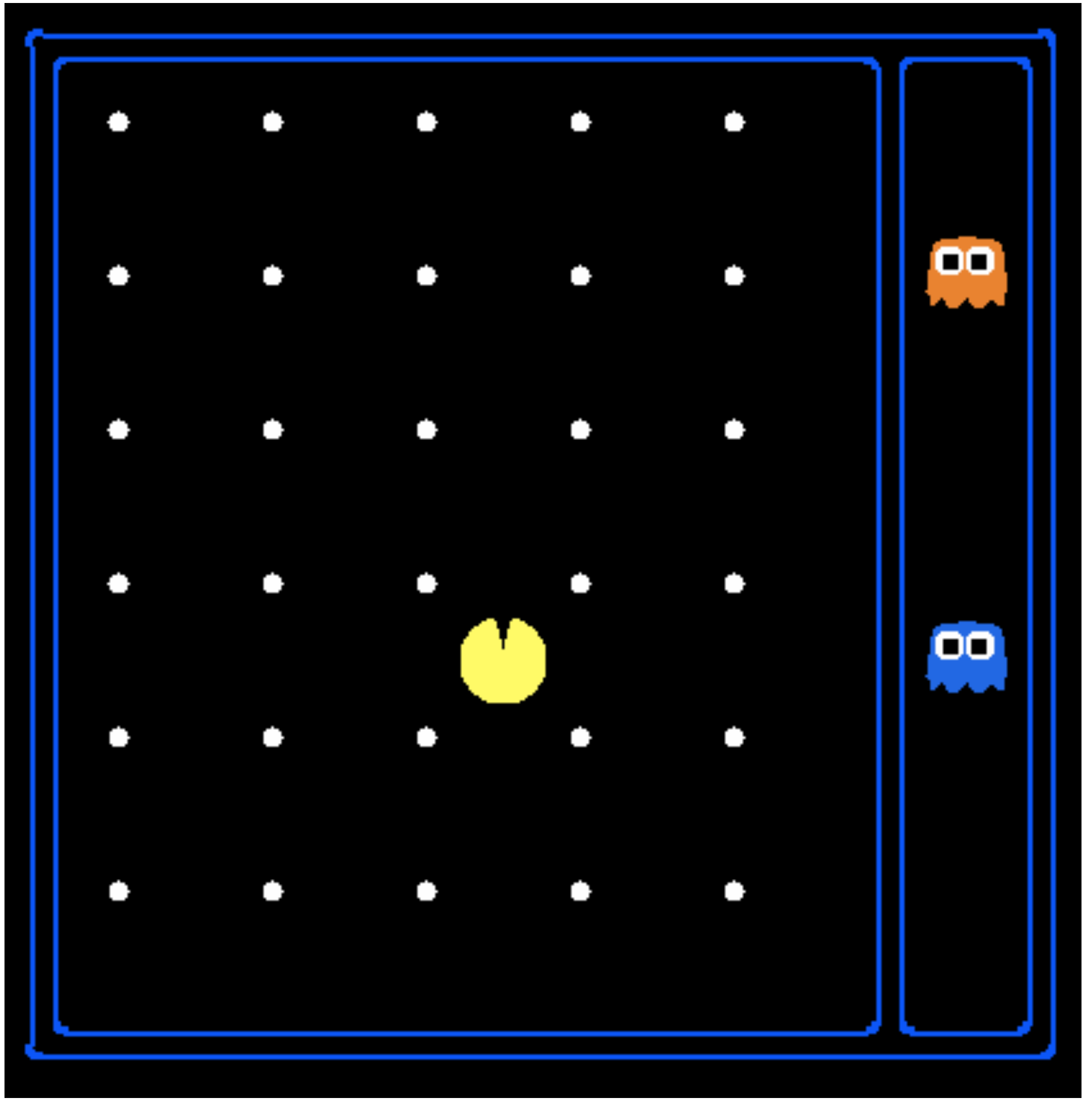
- **Problem: Pathing**
  - States: (x,y) location
  - Actions: NSEW
  - Successor: update location only
  - Goal test: is (x,y) = END

- **Problem: Eat-All-Dots**
  - States: {(x,y), dot booleans}
  - Actions: NSEW
  - Successor: update location and possibly a dot boolean
  - Goal test: dots all false

AncoraSIR.com

State Space Sizes?

- World state:
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW
- How many
  - World states?
    $120\text{x}(2^{30})\text{x}(12^2)\text{x}4$
  - States for pathing?
    120
  - States for eat-all-dots?
    $120\text{x}(2^{30})$

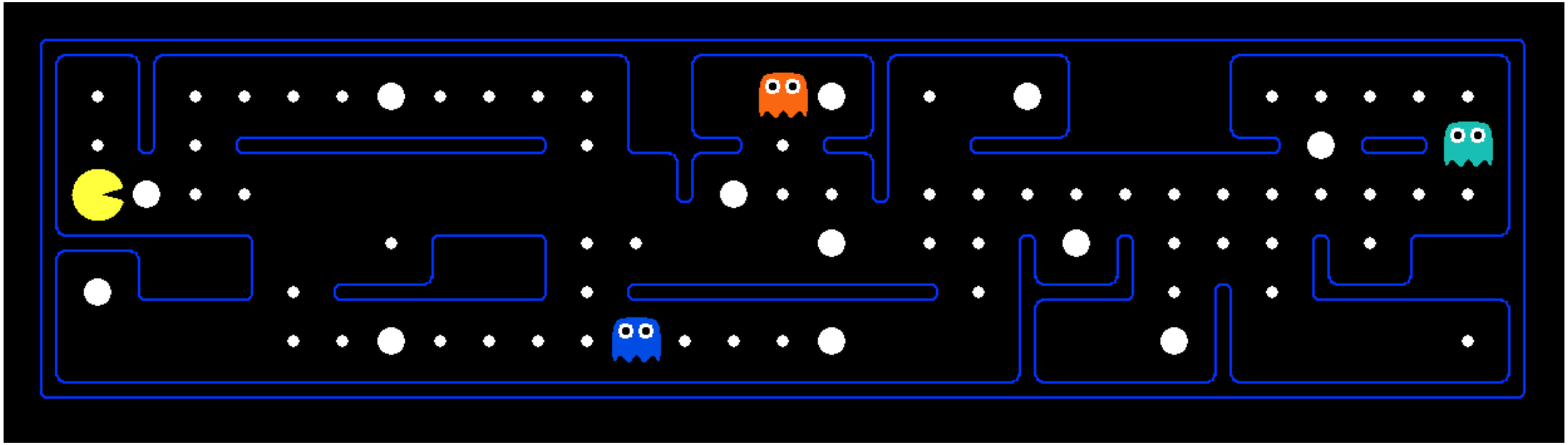


AncoraSIR.com

SUSTech Southern University of Science and Technology

Quiz: Safe Passage

- Problem: eat all dots while keeping the ghosts perma-scared
- What does the state space have to specify?
  - (agent position, dot booleans, power pellet booleans, remaining scared time)

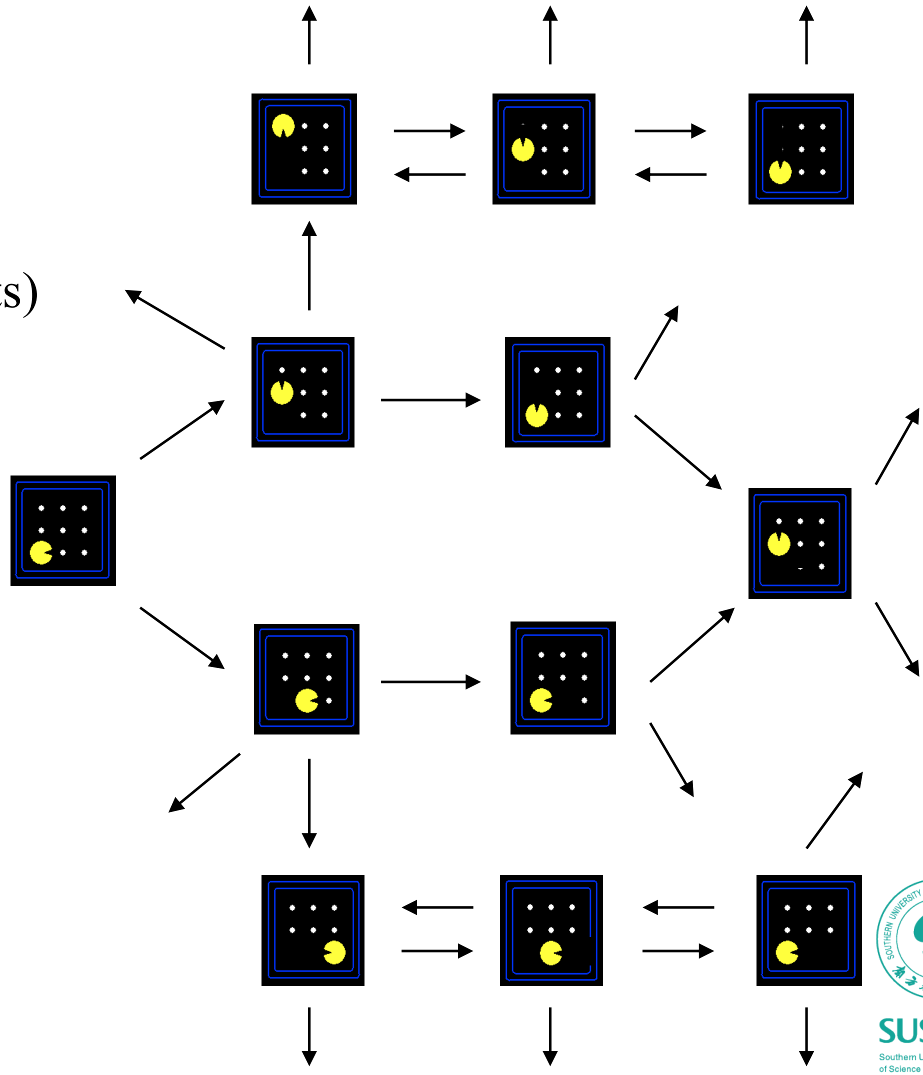


AncoraSIR.com

SUSTech Southern University of Science and Technology

State Space Graphs

- **State space graph**: A mathematical representation of a search problem
  - Nodes are (abstracted) world configurations
  - Arcs represent successors (action results)
  - The goal test is a set of goal nodes (maybe only one)
- In a state space graph, each state occurs only once!
- We can rarely build this full graph in memory (it's too big), but it's a useful idea



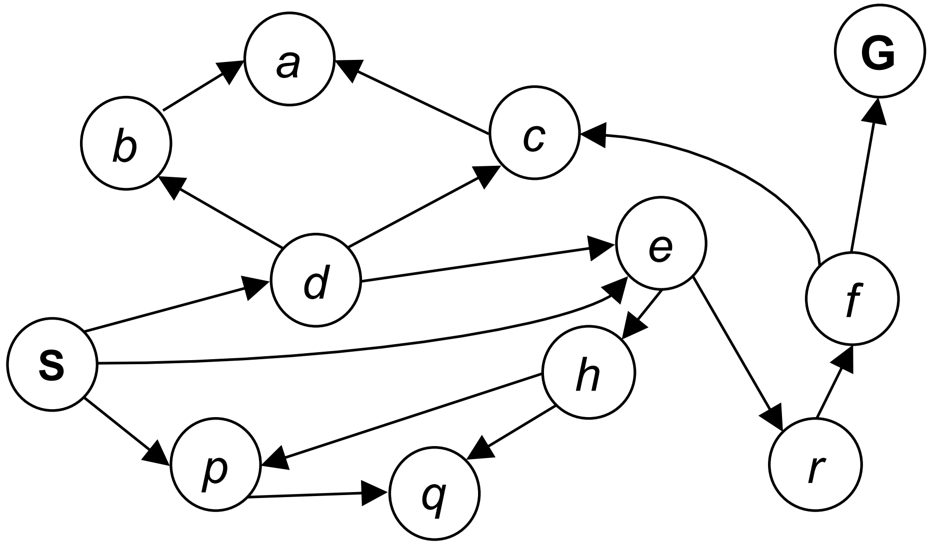
AncoraSIR.com

State Space Graphs

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

- In a state space graph, each state occurs only once!

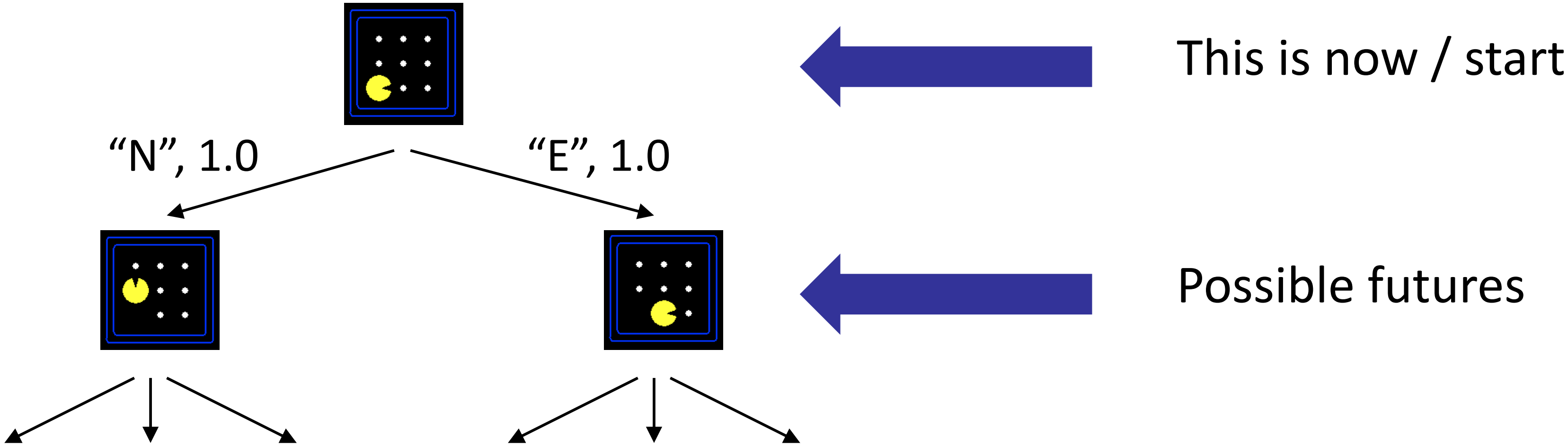
- We can rarely build this full graph in memory (it's too big), but it's a useful idea



Tiny state space graph for a tiny search problem

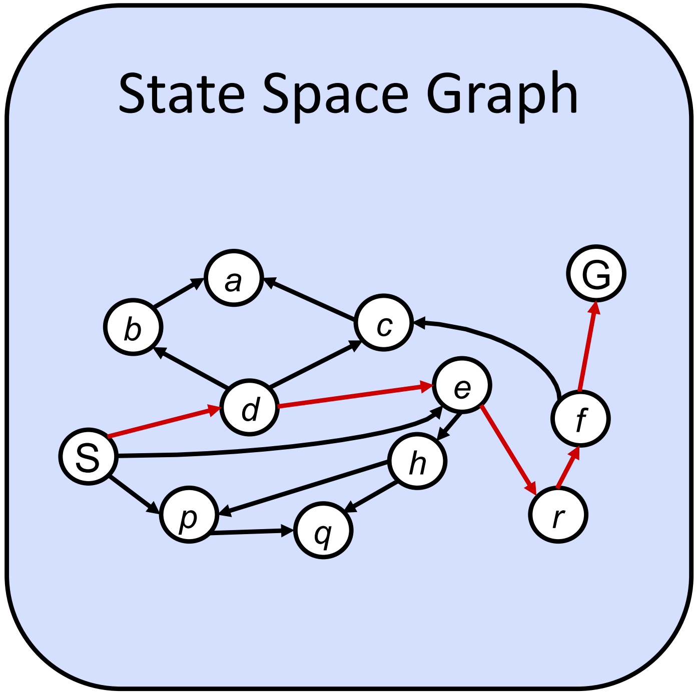
AncoraSIR.com

Search Trees



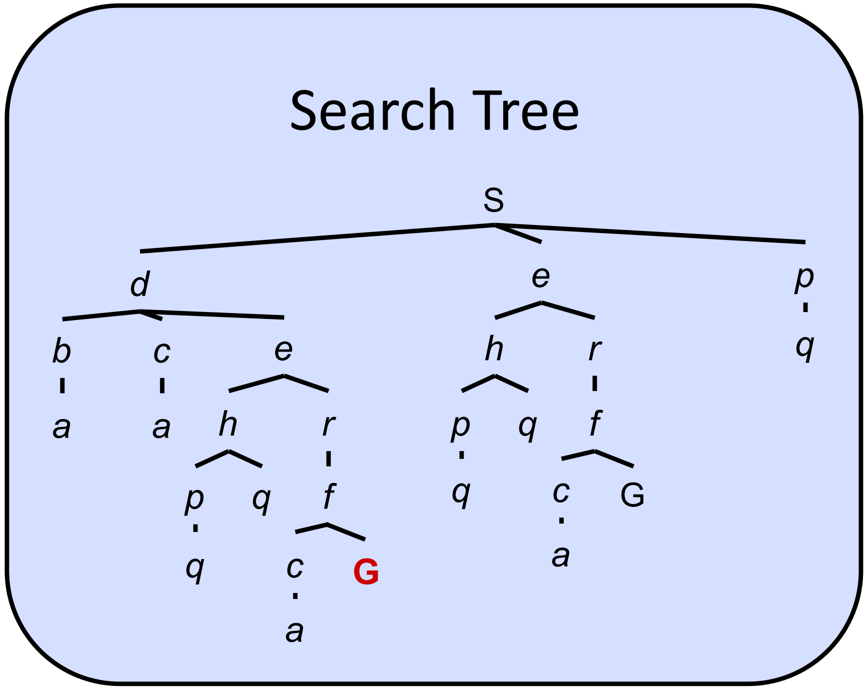
- A search tree:
  - A "what if" tree of plans and their outcomes
  - The start state is the root node
  - Children correspond to successors
  - Nodes show states, but correspond to PLANS that achieve those states
  - *For most problems, we can never actually build the whole tree*

State Space Graphs vs. Search Trees



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

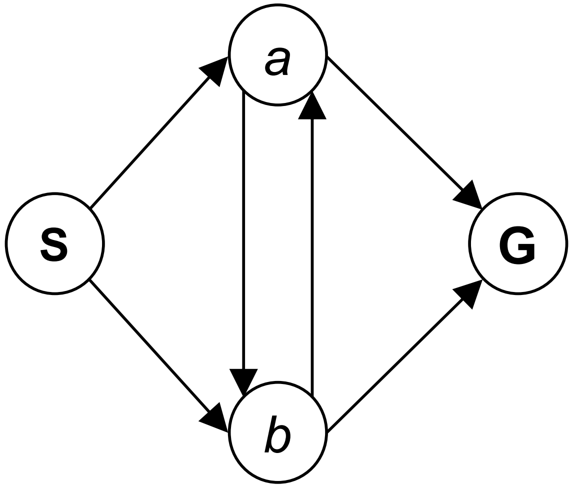
We construct both on demand – and we construct as little as possible.



AncoraSIR.com

Quiz: State Space Graphs vs. Search Trees

Consider this 4-state graph:



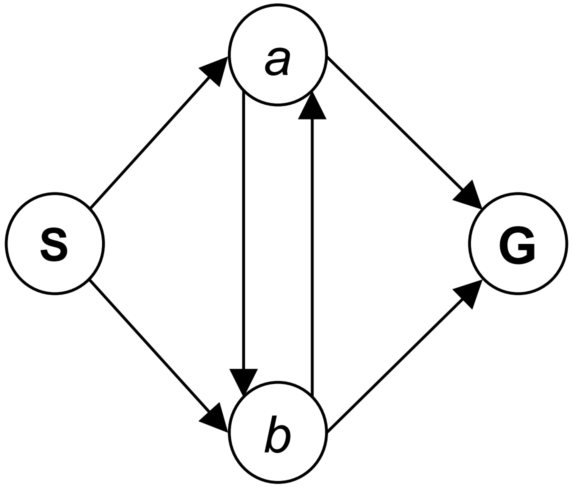
How big is its search tree (from S)?

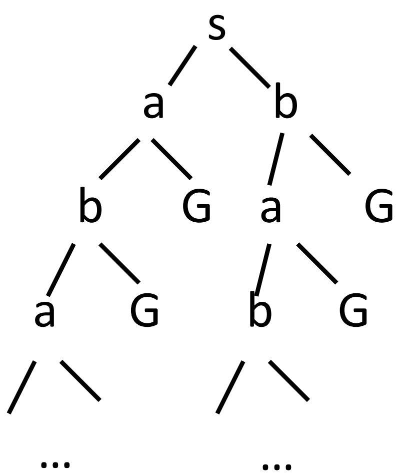
Important: Lots of repeated structure in the search tree!

Quiz: State Space Graphs vs. Search Trees

Consider this 4-state graph:

How big is its search tree (from S)?



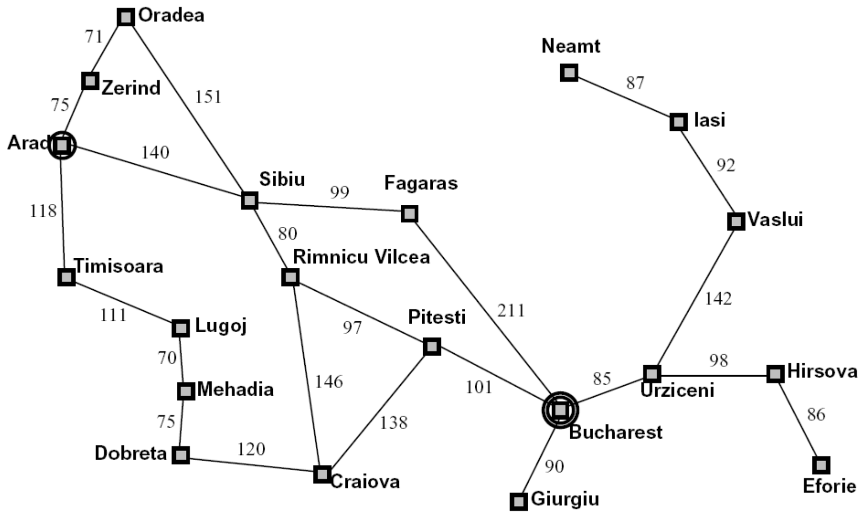


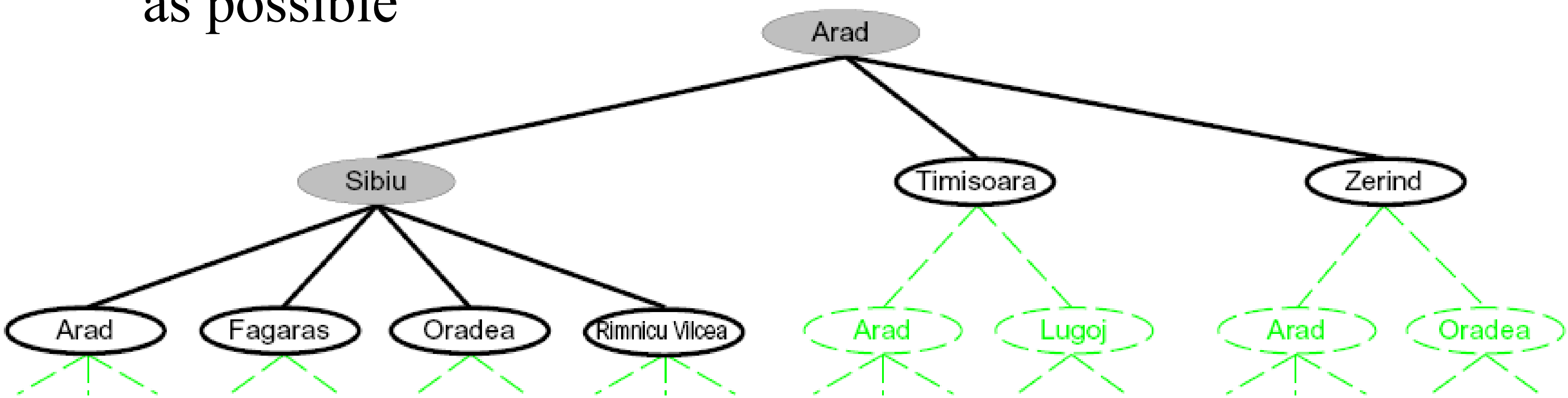
Important: Lots of repeated structure in the search tree!

AncoraSIR.com

Searching with a Search Tree

- Search:
  - Expand out potential plans (tree nodes)
  - Maintain a **fringe** of partial plans under consideration
  - Try to expand as few tree nodes as possible





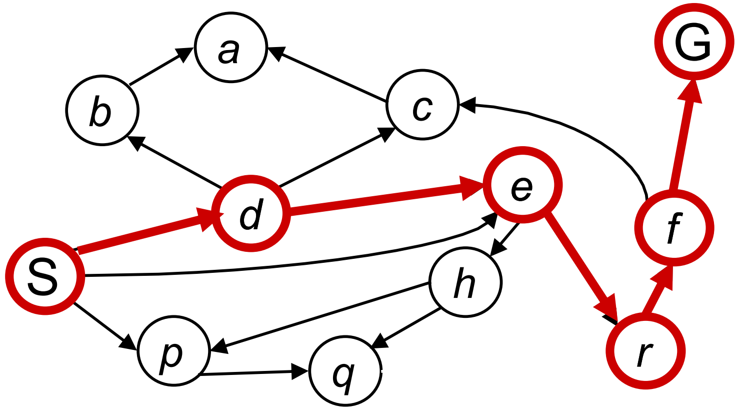
General Tree Search

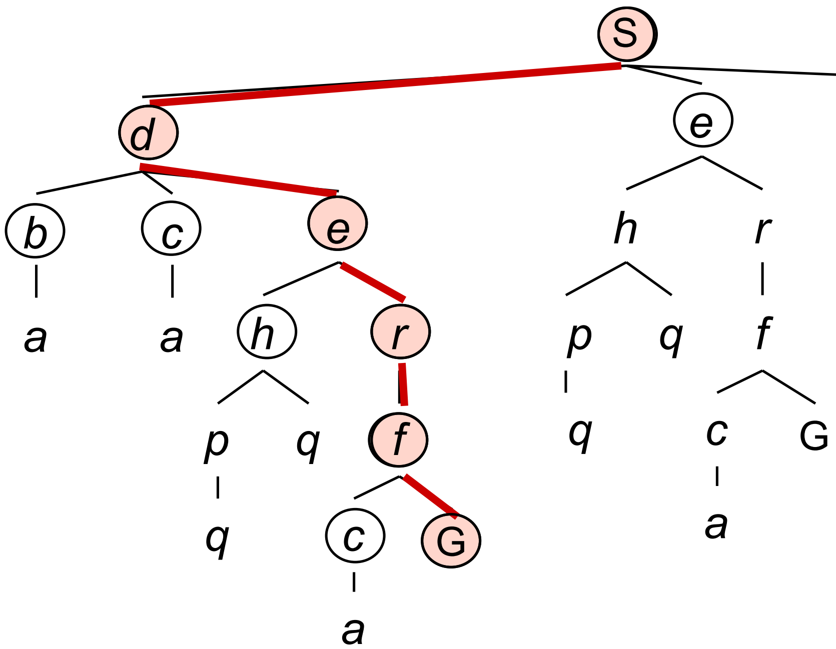
```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the 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?

AncoraSIR.com

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

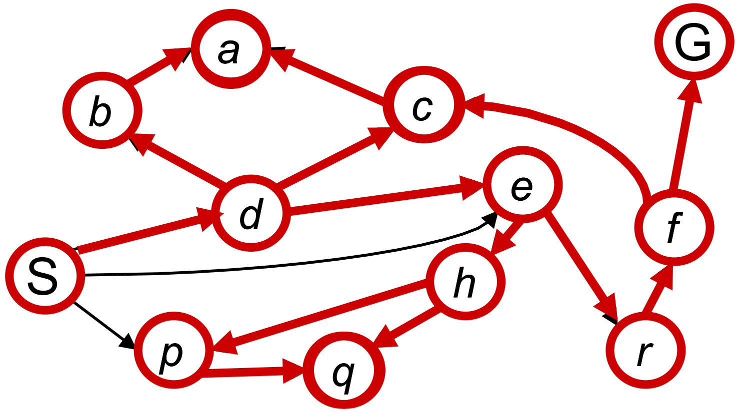
~~s → d → e → r → f → G~~

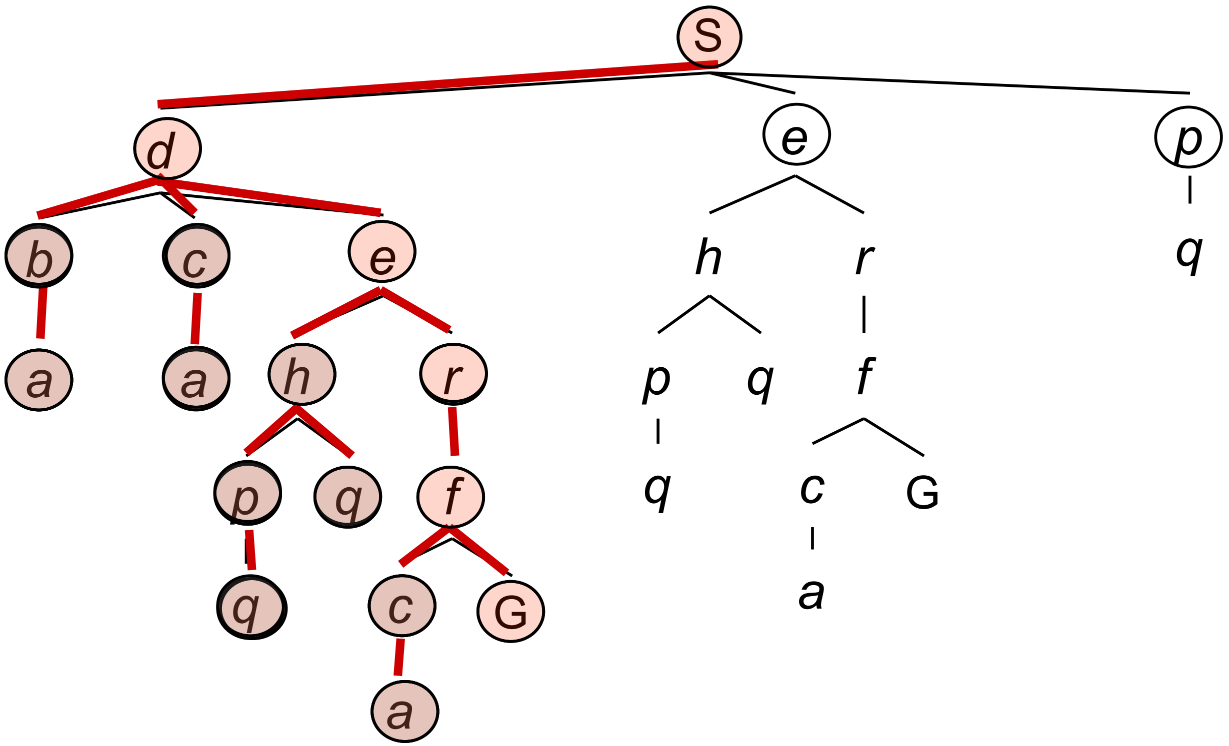
AncoraSIR.com

Depth-First Search

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack



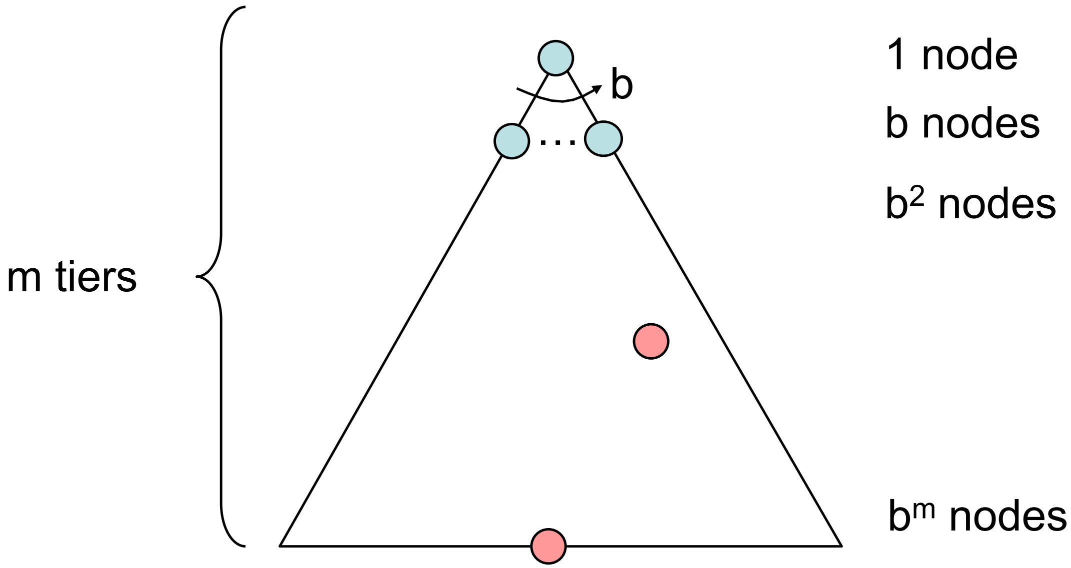


AncoraSIR.com

SUSTech
Southern University of Science and Technology

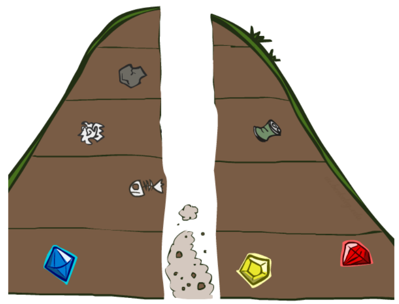
Search Algorithm Properties

- 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

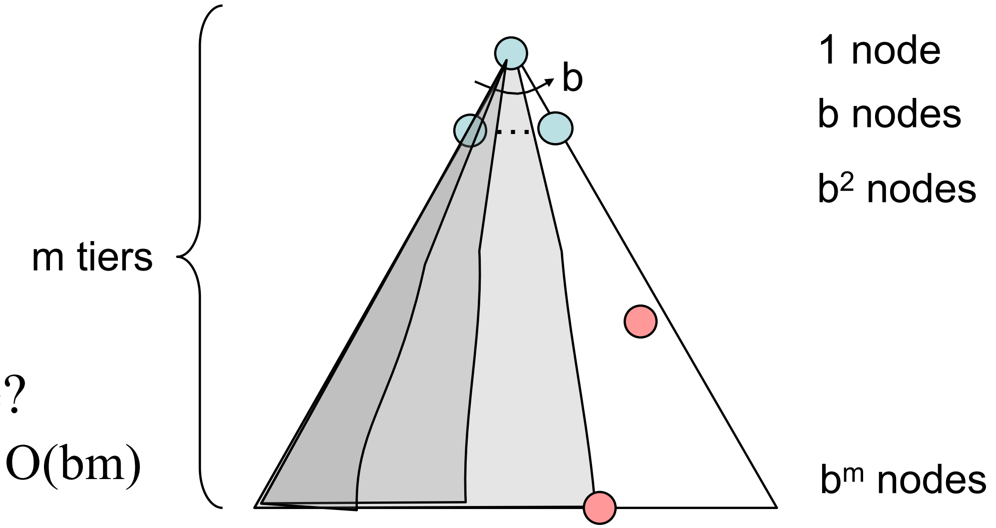
- Number of nodes in entire tree?
  - $1 + b + b^2 + \ldots . b^m = O(b^m)$



AncoraSIR.com

Depth-First Search (DFS) Properties

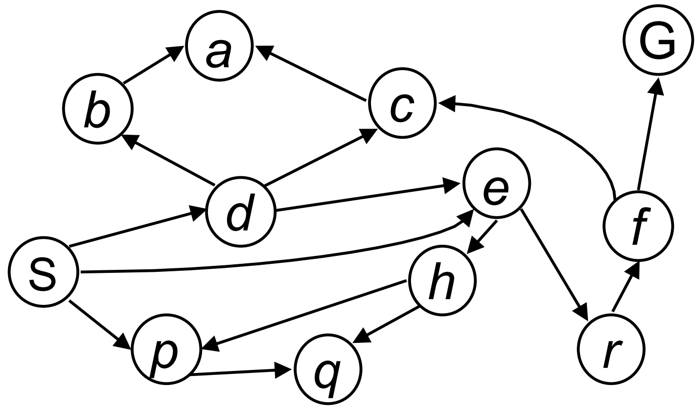
- What nodes DFS expand?
  - Some left prefix of the tree.
  - Could process the whole tree!
  - If m is finite, takes time O(bm)
- How much space does the fringe take?
  - Only has siblings on path to root, so O(bm)
- Is it complete?
  - m could be infinite, so only if we prevent cycles (more later)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

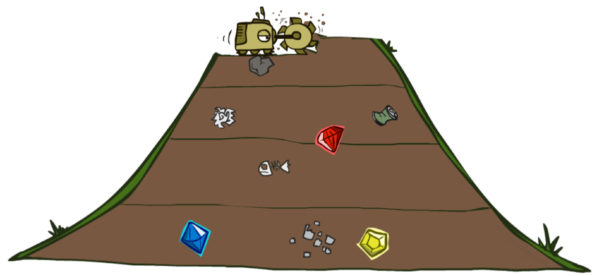


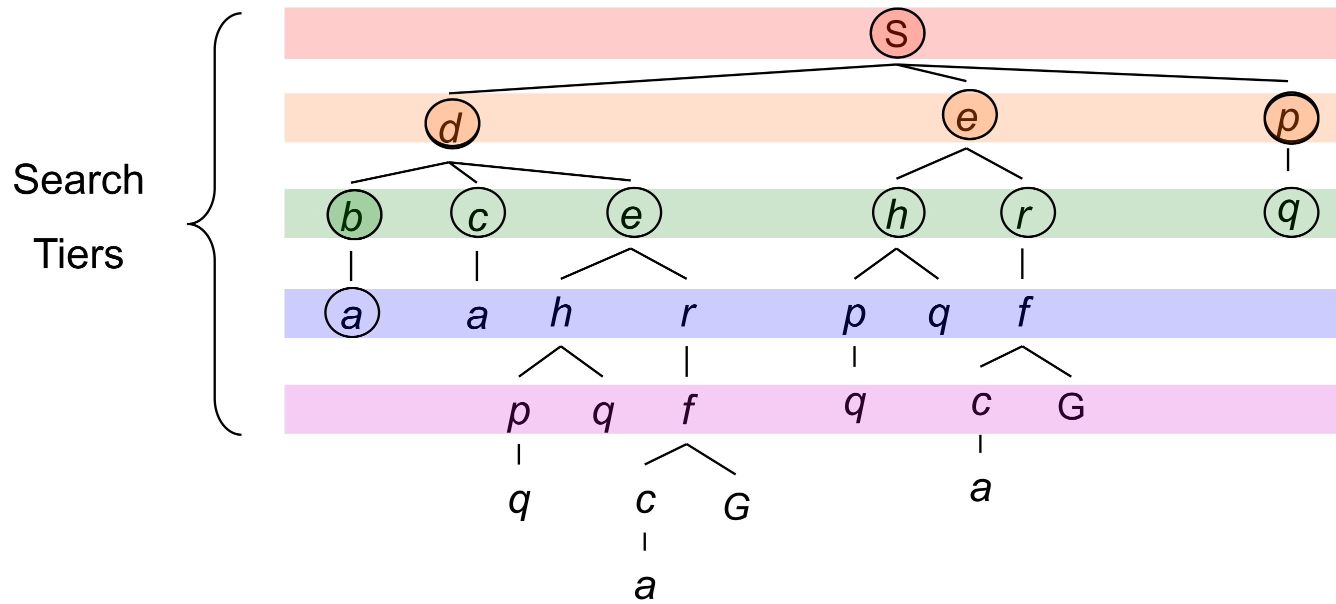
Breadth-First Search

Strategy: expand a shallowest node first

Implementation: Fringe is a FIFO queue



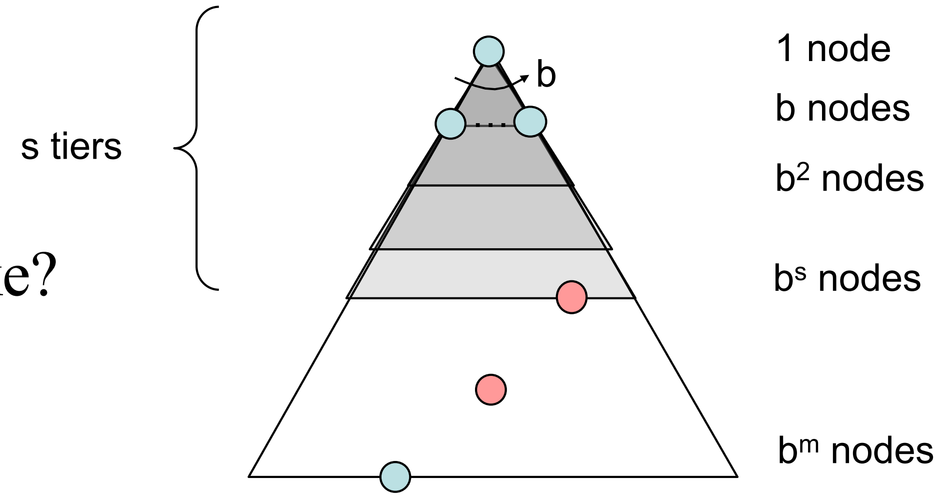




AncoraSIR.com

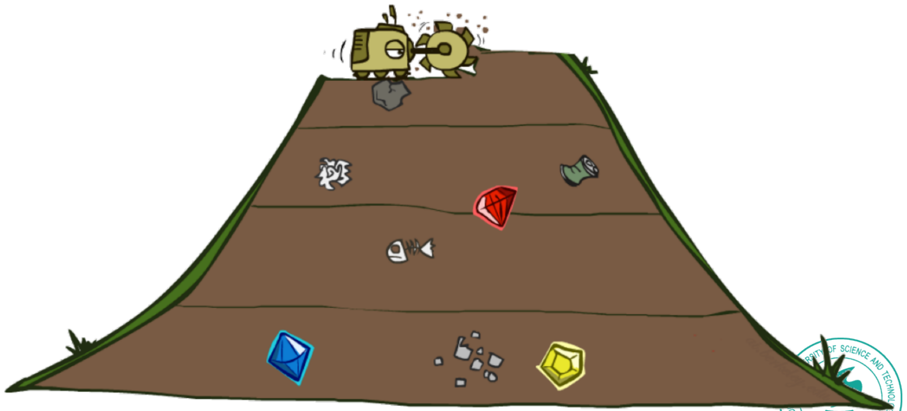
Breadth-First Search (BFS) Properties

- 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?
  - Is must be finite if a solution exists, so yes!
- Is it optimal?
  - Only if costs are all 1 (more on costs later)



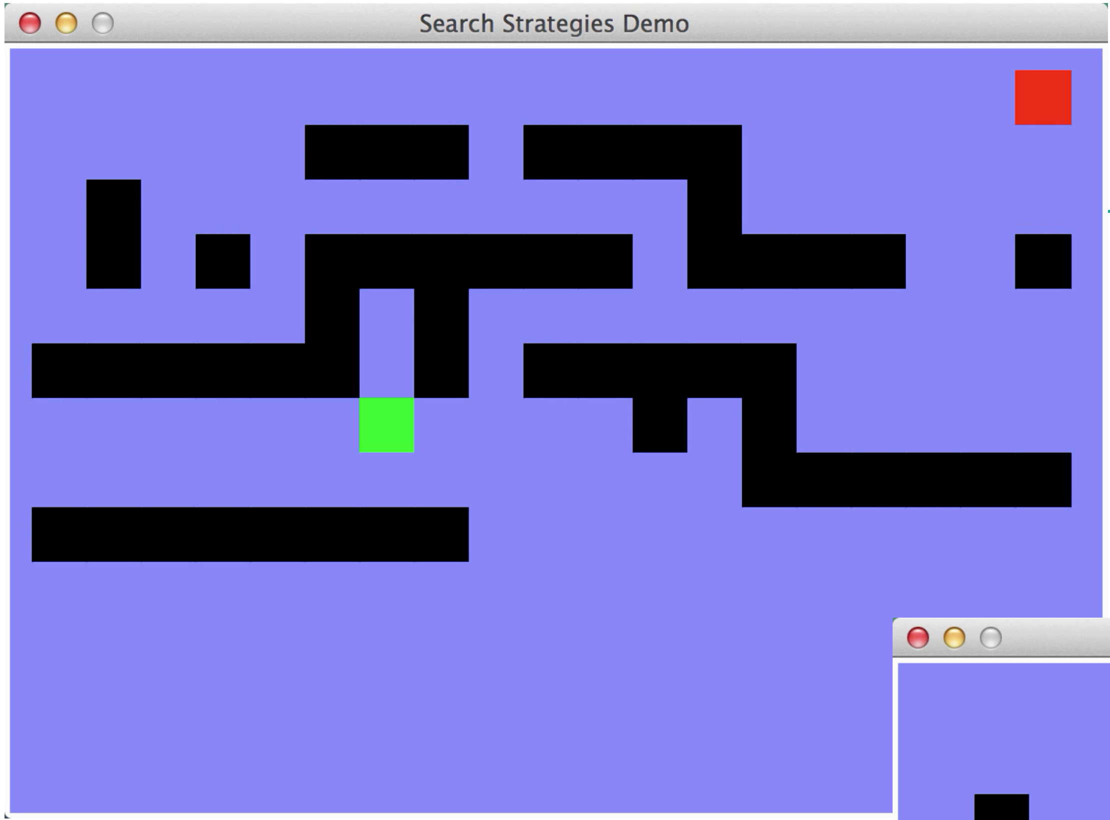
Quiz: DFS vs BFS

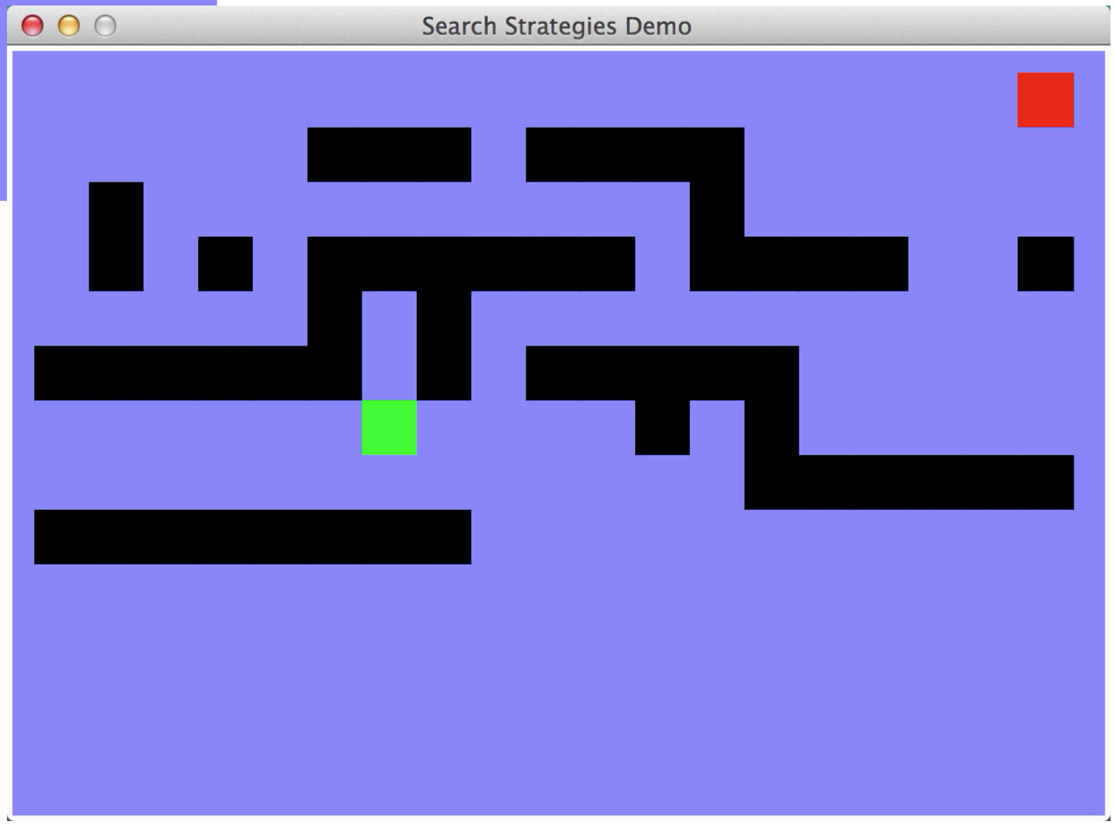
- When will BFS outperform DFS?
- When will DFS outperform BFS?



AncoraSIR.com

SUSTech
Southern University of Science and Technology



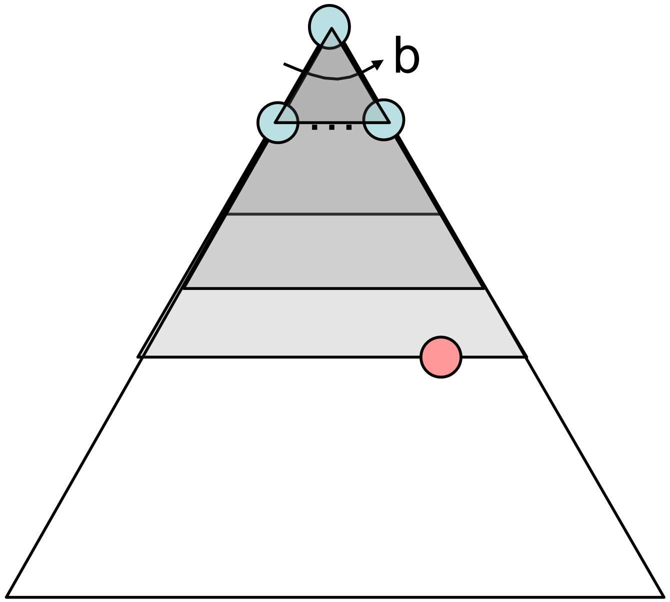


AncoraSIR.com

Iterative Deepening

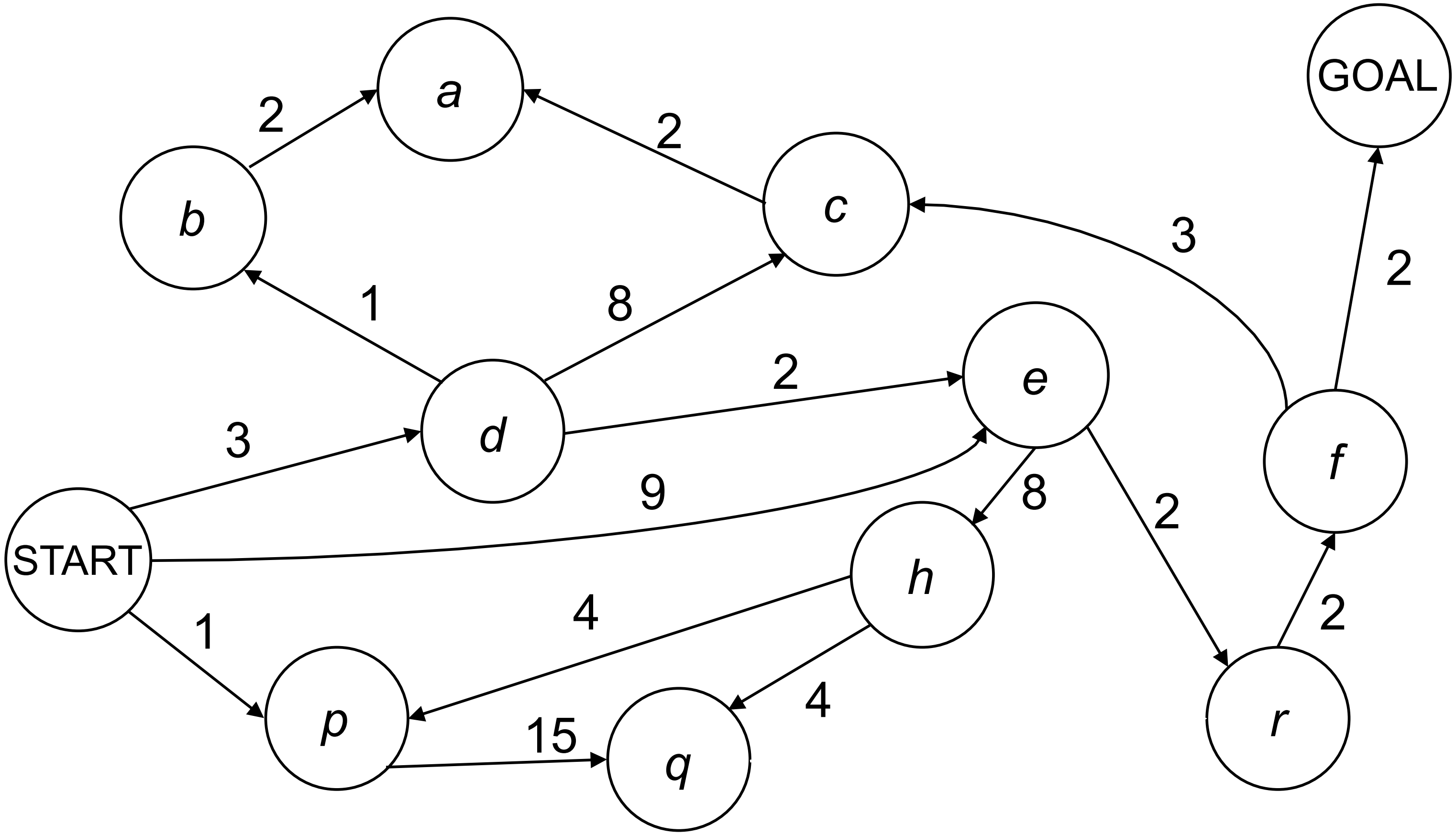
- Idea: get DFS's space advantage with BFS's time / shallow-solution advantages
  - Run a DFS with depth limit 1. If no solution…
  - Run a DFS with depth limit 2. If no solution…
  - Run a DFS with depth limit 3. …..

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



AncoraSIR.com

Cost-Sensitive Search



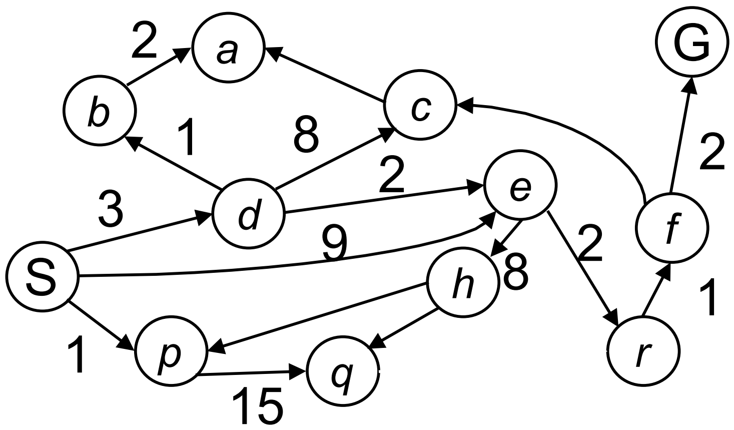
BFS finds the shortest path in terms of number of actions. It does not find the least-cost path. We will now cover a similar algorithm which does find the least-cost path.

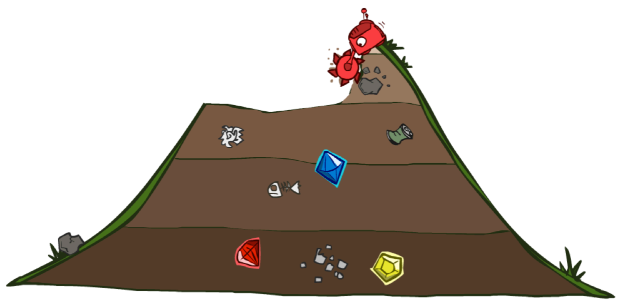
AncoraSIR.com

Uniform Cost Search

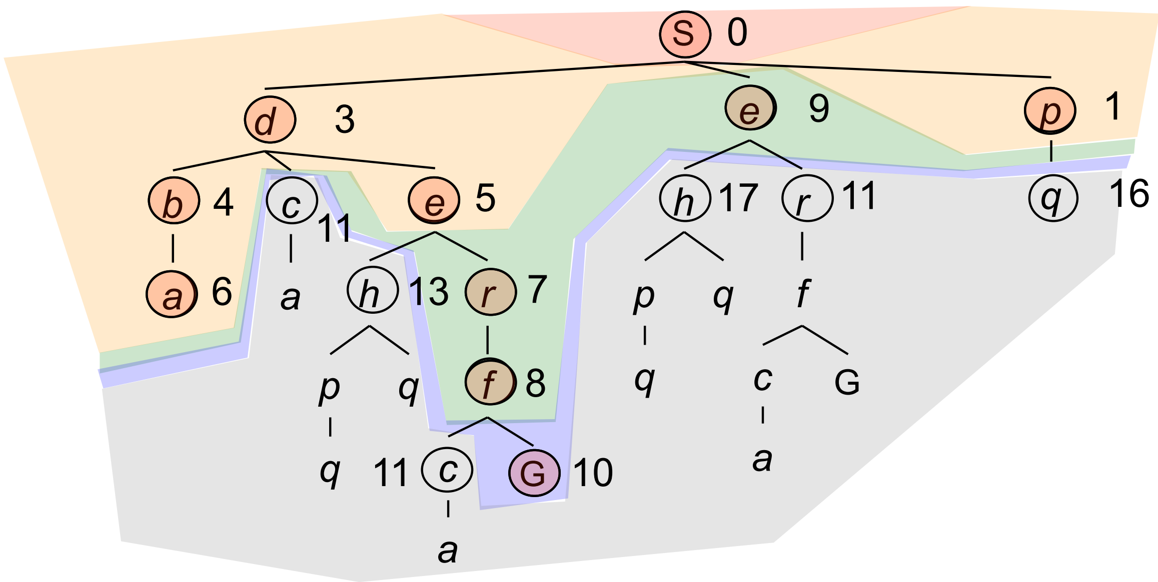
Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)





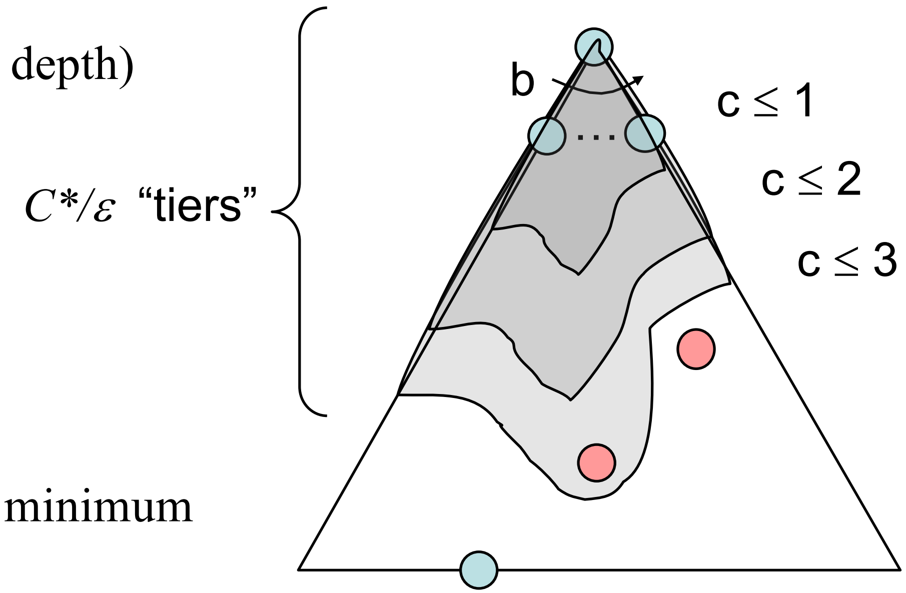
Cost contours



AncoraSIR.com

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)

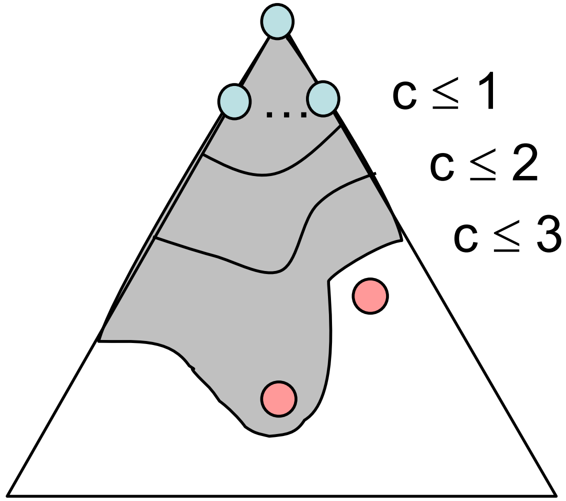


- How much space does the fringe take?
  - Has roughly the last tier, so $O(b^{C^*/\varepsilon})$
- Is it complete?
  - Assuming best solution has a finite cost and minimum arc cost is positive, yes!
- Is it optimal?
  - Yes!

AncoraSIR.com

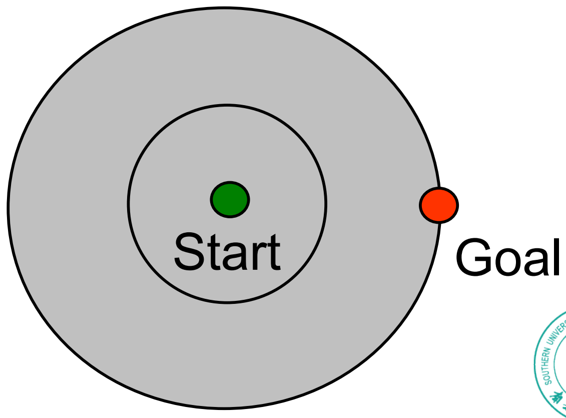
Uniform Cost Issues

- Remember: UCS explores increasing cost contours



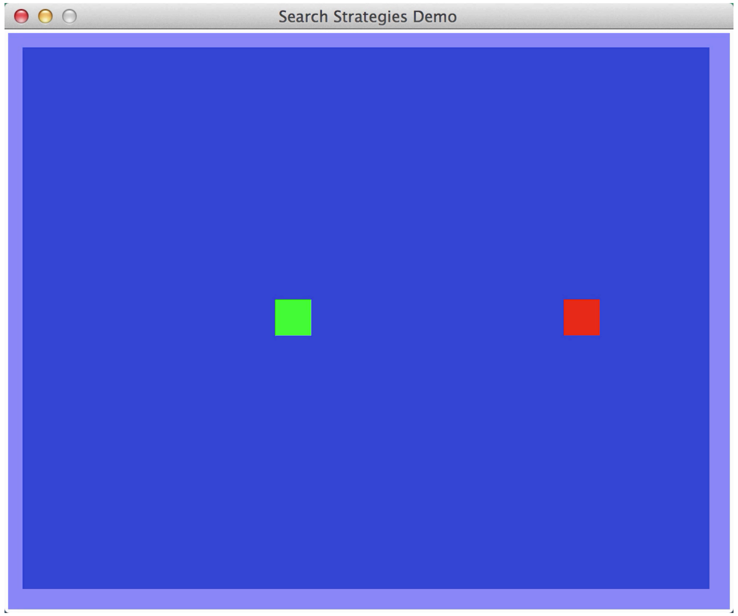
- The good: UCS is complete and optimal!

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

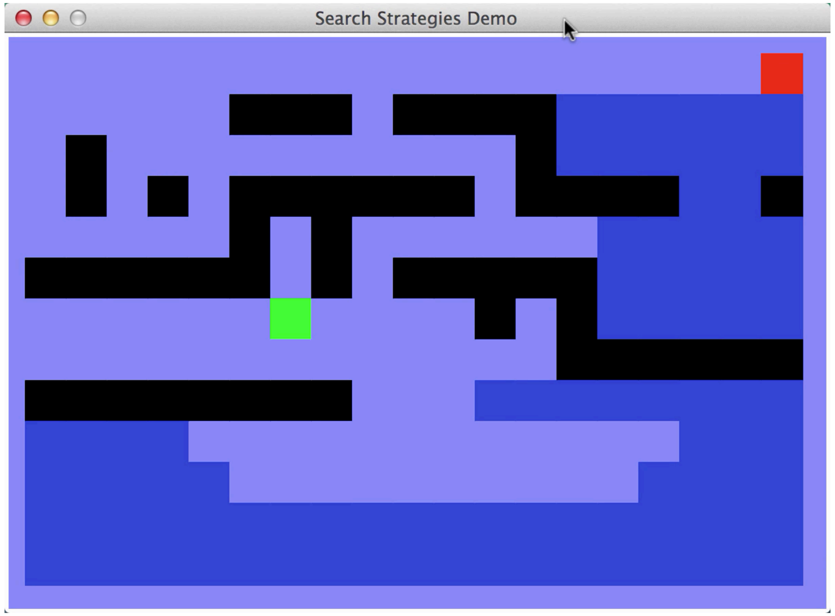


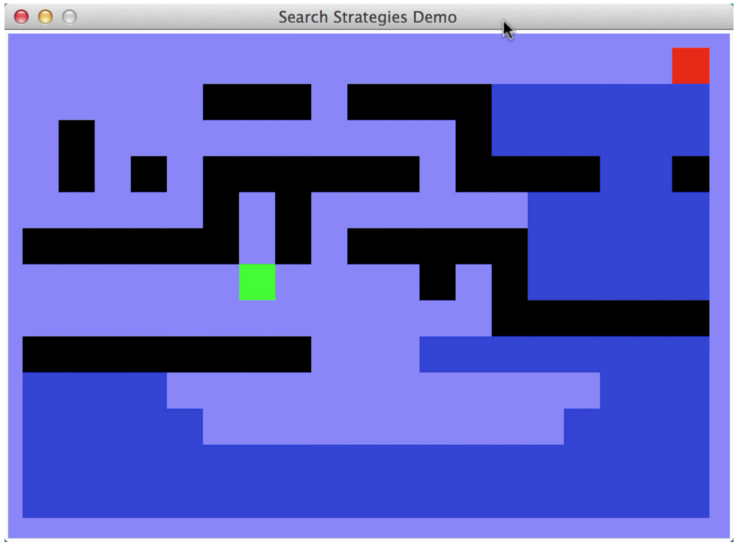
AncoraSIR.com

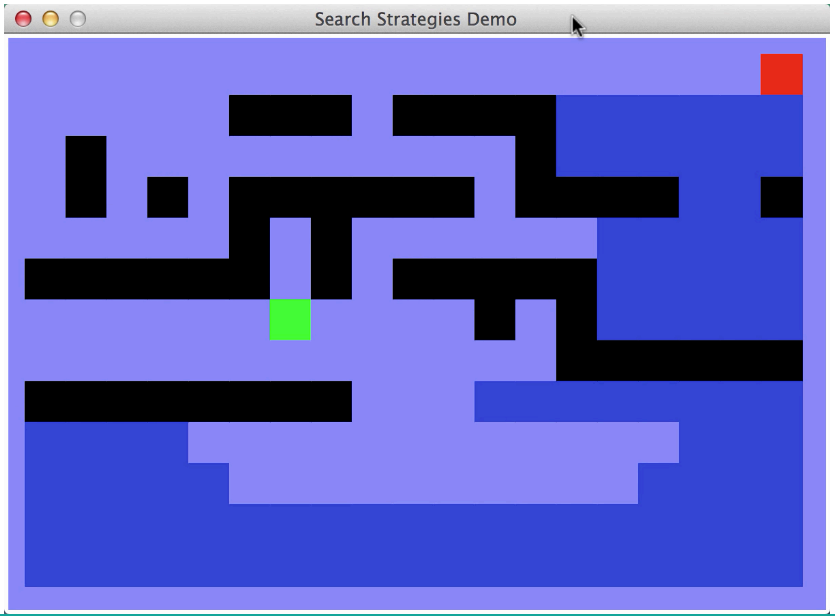
Empty UCS



Maze with Deep/Shallow Water DFS, BFS, or UCS?



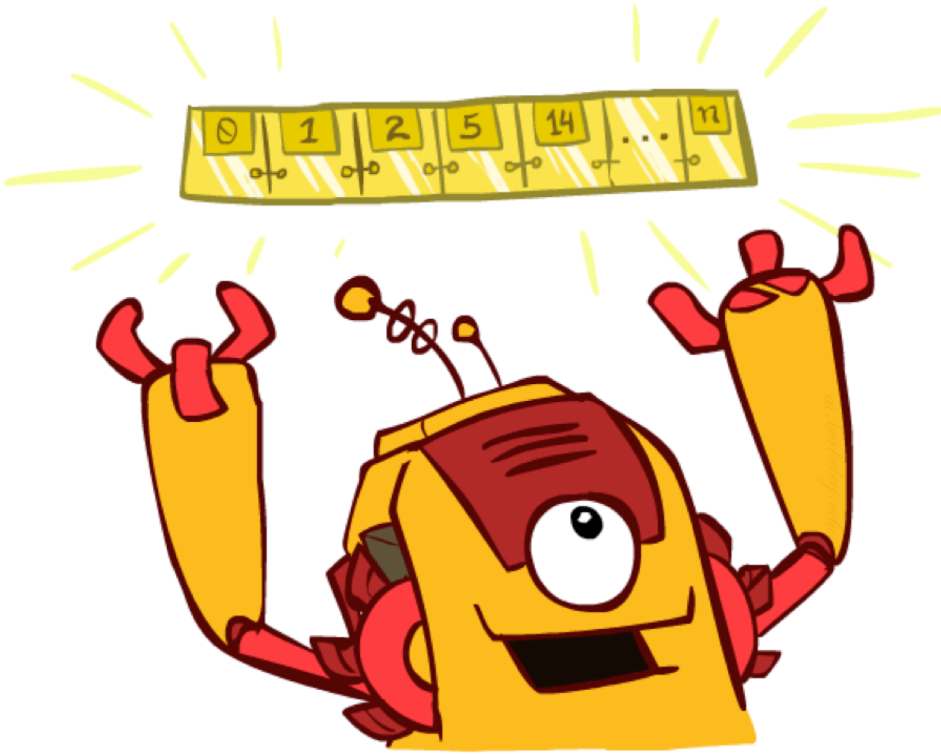




AncoraSIR.com

The One Queue

- All these search algorithms are the same except for fringe strategies
  - Conceptually, all fringes are priority queues (i.e. collections of nodes with attached priorities)
  - Practically, for DFS and BFS, you can avoid the log(n) overhead from an actual priority queue, by using stacks and queues
  - Can even code one implementation that takes a variable queuing object



AncoraSIR.com

Thank you!

Prof. Song Chaoyang

- Dr. Wan Fang (sophie.fwan@hotmail.com)

AncoraSIR.com