Uninformed Search Algorithms and Their Computational Complexity

Instructor: Wei Xu
Ohio State University

[These slides were adapted from CS188 Intro to AI at UC Berkeley.]
Pac-Man and AI

https://www.youtube.com/watch?v=w5kFmdkr1uY

https://www.youtube.com/watch?v=zQyWMHFjewU
Today

- Agents that Plan Ahead
- Search Problems
- Uninformed Search Methods
  - Depth-First Search
  - Breadth-First Search
  - Uniform-Cost Search
Agents that Plan
Reflex Agents

- Reflex agents:
  - Choose action based on current percept (and maybe memory)
  - May have memory or a model of the world’s current state
  - Do not consider the future consequences of their actions
  - Consider how the world IS

- Can a reflex agent be rational?
Video of Demo Reflex Optimal
Video of Demo Reflex Optimal
Video of Demo Reflex Odd
Video of Demo Reflex Odd
Planning Agents

- Planning agents:
  - Ask “what if”
  - Decisions based on (hypothesized) consequences of actions
  - Must have a model of how the world evolves in response to actions
  - Must formulate a goal (test)
  - Consider how the world WOULD BE

- Optimal vs. complete planning

- Planning vs. replanning

[Demo: replanning (L2D3)]
[Demo: mastermind (L2D4)]
Video of Demo Replanning
Video of Demo Mastermind
Video of Demo Mastermind
Search Problems
A search problem consists of:
Search Problems

- A search problem consists of:
  - A state space

![Diagram of state space progression](image-url)
A search problem consists of:

- A state space
- A successor function (with actions, costs)
A search problem consists of:

- A state space

- A successor function (with actions, costs)
A search problem consists of:

- A state space
- A successor function (with actions, costs)
- A start state and a goal test
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.
Search Problems Are Models
Search Problems Are Models
Example: Traveling in Romania
Example: Traveling in Romania

- State space:
  - Cities
Example: Traveling in Romania

- **State space:**
  - Cities

- **Successor function:**
  - Roads: Go to adjacent city with cost = distance
Example: Traveling in Romania

- **State space:**
  - Cities

- **Successor function:**
  - Roads: Go to adjacent city with cost = distance

- **Start state:**
  - Arad
Example: Traveling in Romania

- State space:
  - Cities

- Successor function:
  - Roads: Go to adjacent city with cost = distance

- Start state:
  - Arad

- Goal test:
  - Is state == Bucharest?
Example: Traveling in Romania

- **State space:**
  - Cities

- **Successor function:**
  - Roads: Go to adjacent city with cost = distance

- **Start state:**
  - Arad

- **Goal test:**
  - Is state == Bucharest?

- **Solution?**
What’s in a State Space?

The world state includes every last detail of the environment.
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).
What’s in a State Space?

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

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

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

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

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

- World state:
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW
State Space Sizes?

- **World state:**
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW

- **How many**
State Space Sizes?

- **World state:**
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW

- **How many**
  - World states?
State Space Sizes?

- **World state:**
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW

- **How many**
  - World states?
    \[ 120 \times (2^{30}) \times (12^2) \times 4 \]
State Space Sizes?

- **World state:**
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW

- **How many**
  - World states?
    - $120 \times (2^{30}) \times (12^2) \times 4$
  - States for pathing?
State Space Sizes?

- **World state:**
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW

- **How many**
  - World states?
    \[ 120 \times (2^{30}) \times (12^2) \times 4 \]
  - States for pathing?
    120
State Space Sizes?

- **World state:**
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW

- **How many**
  - World states?
    - $120 \times (2^{30}) \times (12^2) \times 4$
  - States for pathing?
    - 120
  - States for eat-all-dots?
State Space Sizes?

- **World state:**
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW

- **How many**
  - World states?
    - $120 \times (2^{30}) \times (12^2) \times 4$
  - States for pathing?
    - 120
  - States for eat-all-dots?
    - $120 \times (2^{30})$
Quiz: Safe Passage

- Problem: eat all dots while keeping the ghosts perma-scared
Problem: eat all dots while keeping the ghosts perma-scared
What does the state space have to specify?
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)
State Space Graphs and Search Trees
State Space Graphs

- State space graph: A mathematical representation of a search problem
State Space Graphs

- State space graph: A mathematical representation of a search problem
State Space Graphs

- State space graph: A mathematical representation of a search problem
  - Nodes are (abstracted) world configurations
State space graph: A mathematical representation of a search problem
- Nodes are (abstracted) world configurations
- Arcs represent successors (action results)
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)
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!
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
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 search 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
Search Trees
Search Trees

This is now / start
Search Trees

Possible futures

This is now / start

“N”, 1.0  “E”, 1.0
Search Trees

“This is now / start

Possible futures

“N”, 1.0

“E”, 1.0
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

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

Each NODE in the search tree is an entire PATH in the state space graph.
Consider this 4-state graph:
Consider this 4-state graph: How big is its search tree (from S)?
Consider this 4-state graph: How big is its search tree (from S)?
Consider this 4-state graph:

How big is its search tree (from S)?

Important: Lots of repeated structure in the search tree!
Tree Search
Search Example: Romania
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
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
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?
Example: Tree Search
Depth-First Search
**Depth-First Search**

**Strategy:** expand a deepest node first

**Implementation:**
Fringe is a LIFO stack
**Depth-First Search**

**Strategy:** expand a deepest node first

**Implementation:**
Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation:
Fringe is a LIFO stack
**Depth-First Search**

**Strategy:** expand a deepest node first

**Implementation:**
Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation:
Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation:
Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation:
Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation:
Fringe is a LIFO stack
Depth-First Search

**Strategy:** expand a deepest node first

**Implementation:**
Fringe is a LIFO stack
Depth-First Search

Strategy: expand a deepest node first

Implementation:
Fringe is a LIFO stack
Search Algorithm Properties
Search Algorithm Properties

- Complete: Guaranteed to find a solution if one exists?
Search Algorithm Properties

- Complete: Guaranteed to find a solution if one exists?
- Optimal: Guaranteed to find the least cost path?
Search Algorithm Properties

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

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

\[ b \]

1 node
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
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

1 node
$b$ nodes
$b^2$ nodes
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

\[ \text{m tiers} \]

1 node
$b$ nodes
$b^2$ nodes
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
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

\[ b \text{ nodes} \]
\[ b^2 \text{ nodes} \]
\[ b^m \text{ nodes} \]
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

[Diagram of a search tree with nodes and tiers labeled: 1 node, $b$ nodes, $b^2$ nodes, $b^m$ nodes, and $m$ tiers]
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?
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?

![Diagram of DFS properties]

- 1 node
- \( b \) nodes
- \( b^2 \) nodes
- \( b^m \) nodes

\( m \) tiers
Depth-First Search (DFS) Properties

- What nodes DFS expand?

Diagram:
- 1 node
- $b$ nodes
- $b^2$ nodes
- $b^m$ nodes
- $m$ tiers
Depth-First Search (DFS) Properties

- What nodes DFS expand?

![Diagram showing DFS properties with m tiers, b nodes, b^2 nodes, and b^m nodes.](image)
Depth-First Search (DFS) Properties

- What nodes DFS expand?
Depth-First Search (DFS) Properties

- What nodes DFS expand?
  - Some left prefix of the tree.
Depth-First Search (DFS) Properties

- What nodes DFS expand?
  - Some left prefix of the tree.
  - Could process the whole tree!
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 \( O(b^m) \)
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?

![Diagram](image-url)
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 \( O(bm) \)
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?
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)
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?**
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

Strategy: expand a shallowest node first

Implementation: Fringe is a FIFO queue
Breadth-First Search

Strategy: expand a shallowest node first

Implementation: Fringe is a FIFO queue
Breadth-First Search

Strategy: expand a shallowest node first

Implementation: Fringe is a FIFO queue
Breadth-First Search

Strategy: expand a shallowest node first

Implementation: Fringe is a FIFO queue
**Breadth-First Search**

**Strategy:** expand a shallowest node first

**Implementation:**
Fringe is a FIFO queue
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?

![Diagram showing BFS expansion]

1 node
b nodes
b^2 nodes
b^m nodes
Breadth-First Search (BFS) Properties

- What nodes does BFS expand?

![Diagram showing BFS expansion with nodes labeled: 1 node, b nodes, b^2 nodes, and b^m nodes.](image-url)
What nodes does BFS expand?
What nodes does BFS expand?

- 1 node
- $b$ nodes
- $b^2$ nodes
- $b^m$ nodes
Breadth-First Search (BFS) Properties

- What nodes does BFS expand?

![Diagram showing BFS expansion]

- 1 node
- $b$ nodes
- $b^2$ nodes
- $b^m$ nodes
Breadth-First Search (BFS) Properties

- What nodes does BFS expand?
  - Processes all nodes above shallowest solution

![Diagram showing BFS properties]

- 1 node
- $b$ nodes
- $b^2$ nodes
- $b^m$ nodes
Breadth-First Search (BFS) Properties

- What nodes does BFS expand?
  - Processes all nodes above shallowest solution
  - Let depth of shallowest solution be $s$

![Diagram showing nodes and tiers](image-url)
Breadth-First Search (BFS) Properties

- What nodes does BFS expand?
  - Processes all nodes above shallowest solution
  - Let depth of shallowest solution be $s$

$$\begin{align*}
&b^1\text{ node} \\
&b\text{ nodes} \\
&b^2\text{ nodes} \\
&b^s\text{ nodes} \\
&b^m\text{ nodes}
\end{align*}$$

Diagram:
- Nodes are organized in tiers.
- Depth of shallowest solution is $s$ tiers.
- Each tier has $b$ times the nodes of the previous tier.
- The nodes are expanded in a breadth-first manner, starting from the top.
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 $O(b^s)$
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?**
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 $O(b^s)$
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?
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!
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?**
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)
Video of Demo Maze Water DFS/BFS (part 1)
Video of Demo Maze Water DFS/BFS (part 1)
Video of Demo Maze Water DFS/BFS (part 1)
Video of Demo Maze Water DFS/BFS (part 1)
Iterative Deepening

- Idea: get DFS’s space advantage with BFS’s time / shallow-solution advantages
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...
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...
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. .....
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. .....
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?
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!
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
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.
Uniform Cost Search
Uniform Cost Search

Strategy: expand the cheapest node first.

Fringe is a priority queue (priority: cumulative cost)
Uniform Cost Search

Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)
Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)
Uniform Cost Search

Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)
Uniform Cost Search

Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)
Uniform Cost Search

Strategy: expand a cheapest node first:
Fringe is a priority queue (priority: cumulative cost)
Strategy: expand the cheapest node first:

Fringe is a priority queue (priority: cumulative cost)
Uniform Cost Search

Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)
Uniform Cost Search

Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)
Uniform Cost Search

Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)
Uniform Cost Search

Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)
Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)
Uniform Cost Search (UCS) Properties

- What nodes does UCS expand?
Uniform Cost Search (UCS) Properties

- What nodes does UCS expand?
What nodes does UCS expand?
Uniform Cost Search (UCS) Properties

- What nodes does UCS expand?
Uniform Cost Search (UCS) Properties

- What nodes does UCS expand?
  - Processes all nodes with cost less than cheapest solution!
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 $\varepsilon$, then the “effective depth” is roughly $C^*/\varepsilon$
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 $\varepsilon$, then the “effective depth” is roughly $C^*/\varepsilon$
  - Takes time $O(b^{C^*/\varepsilon})$ (exponential in effective depth)
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 $\varepsilon$, then the “effective depth” is roughly $C^*/\varepsilon$
  - Takes time $O(b^{C^*/\varepsilon})$ (exponential in effective depth)

- How much space does the fringe take?
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 $\epsilon$, then the “effective depth” is roughly $C^* / \epsilon$
  - Takes time $O(b^{C^*/\epsilon})$ (exponential in effective depth)

- How much space does the fringe take?
  - Has roughly the last tier, so $O(b^{C^*/\epsilon})$
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 $\varepsilon$, then the “effective depth” is roughly $C^*/\varepsilon$
  - 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?
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 $\varepsilon$, then the “effective depth” is roughly $C^*/\varepsilon$
  - 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!
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 $\varepsilon$, then the “effective depth” is roughly $C^*/\varepsilon$
  - 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?
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 \( \varepsilon \), then the “effective depth” is roughly \( C^*/\varepsilon \)
  - 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!
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

\[c \leq 1\]
\[c \leq 2\]
\[c \leq 3\]

[Demo: empty grid UCS (L2D5)]
[Demo: maze with deep/shallow water DFS/BFS/UCS (L2D7)]
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

[Demo: empty grid UCS (L2D5)]
[Demo: maze with deep/shallow water DFS/BFS/UCS (L2D7)]
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!
Video of Demo Maze with Deep/Shallow Water --DFS, BFS, or UCS? (part 1)
Video of Demo Maze with Deep/Shallow Water -- DFS, BFS, or UCS? (part 1)
Video of Demo Maze with Deep/Shallow Water --DFS, BFS, or UCS? (part 1)
Video of Demo Maze with Deep/Shallow Water -- DFS, BFS, or UCS? (part 2)
Video of Demo Maze with Deep/Shallow Water -- DFS, BFS, or UCS? (part 2)
Video of Demo Maze with Deep/Shallow Water -- DFS, BFS, or UCS? (part 2)
Video of Demo Maze with Deep/Shallow Water -- DFS, BFS, or UCS? (part 3)
Video of Demo Maze with Deep/Shallow Water -- DFS, BFS, or UCS? (part 3)
Video of Demo Maze with Deep/Shallow Water -- DFS, BFS, or UCS? (part 3)
Search and Models

- Search operates over models of the world
  - The agent doesn’t actually try all the plans out in the real world!
  - Planning is all “in simulation”
  - Your search is only as good as your models...
Search and Models

- Search operates over models of the world
  - The agent doesn’t actually try all the plans out in the real world!
  - Planning is all “in simulation”
  - Your search is only as good as your models…
Search Gone Wrong?
Search Gone Wrong?
Search Gone Wrong?