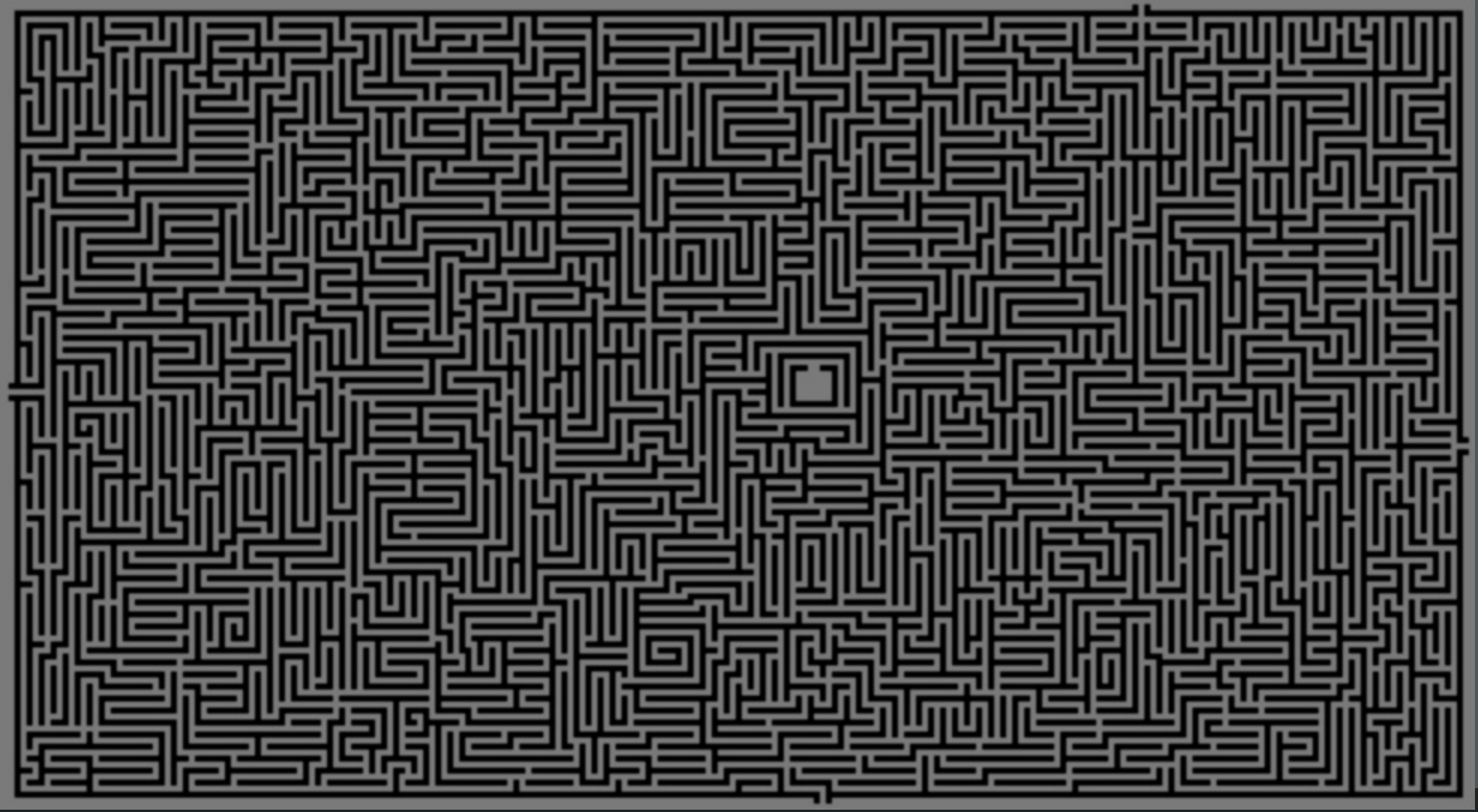


Search

searching for solutions

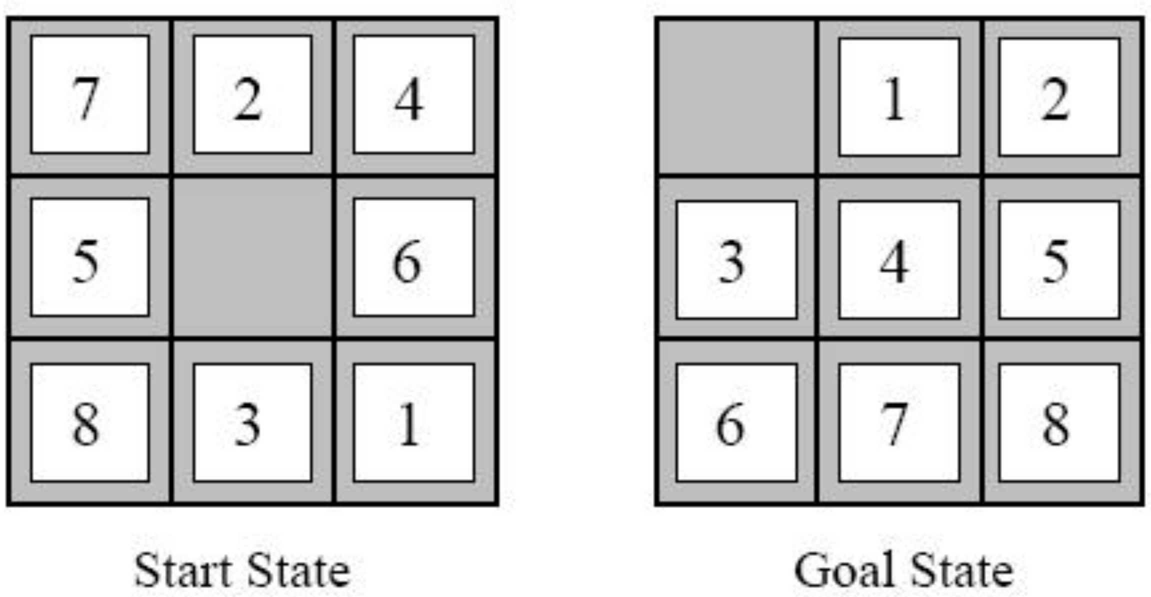
CS 4100/5100

Foundations of AI

Framework for Search Problems

- Problem formulation
  - A set of discrete states (usually finite, usually huge)
  - Start state
  - Subset of states designated as goals (usually)
  - A set of actions the agent can perform
- Objective
  - Find a sequence of operators that lead to the goal state
    - Optimal or least-cost would be nice!

Example: 8 Puzzle



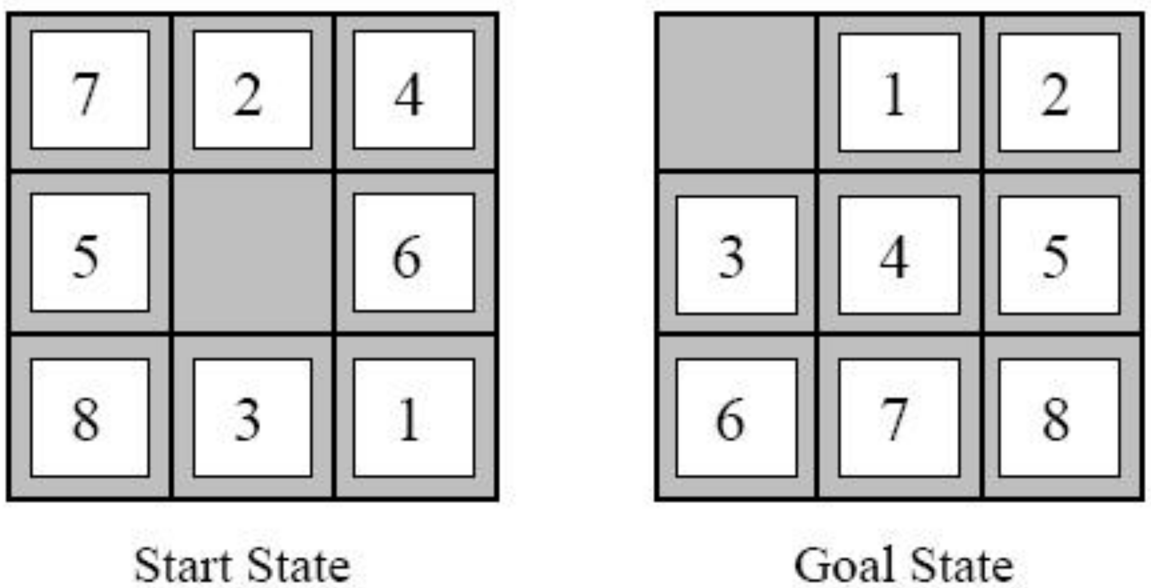
States

Actions

Goal test

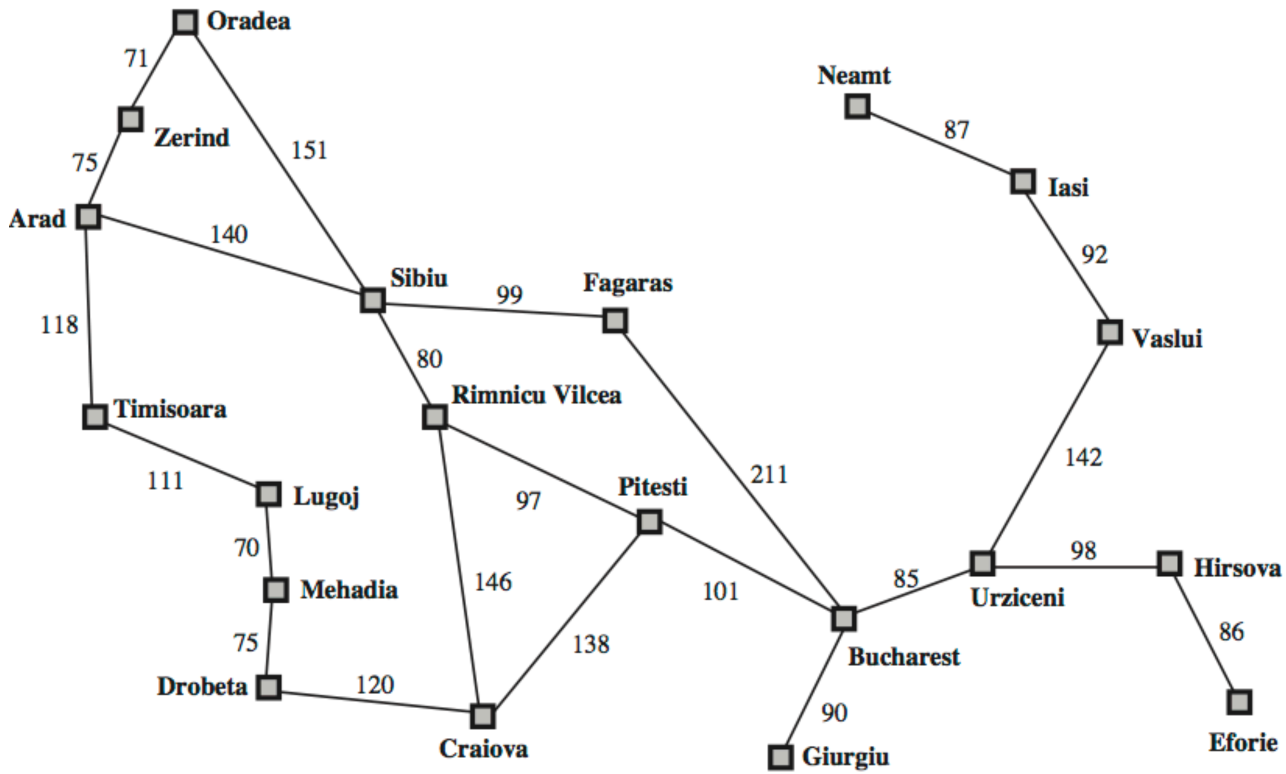
Path cost

Example: 8 Puzzle



States	locations of tiles + blanks
Actions	move blank left, right, up, down
Goal test	= goal state (given)
Path cost	1 per move

Example: Pathfinding



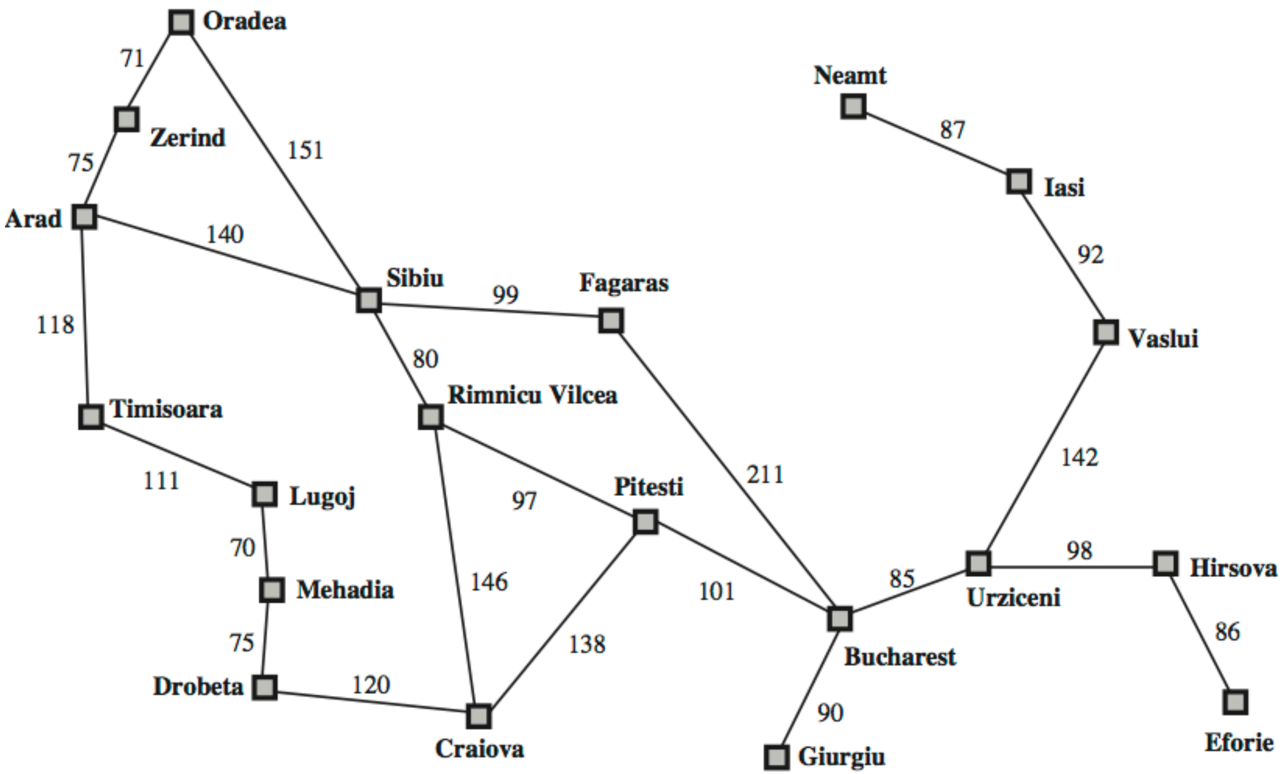
States

Actions

Goal test

Path cost

Example: Pathfinding



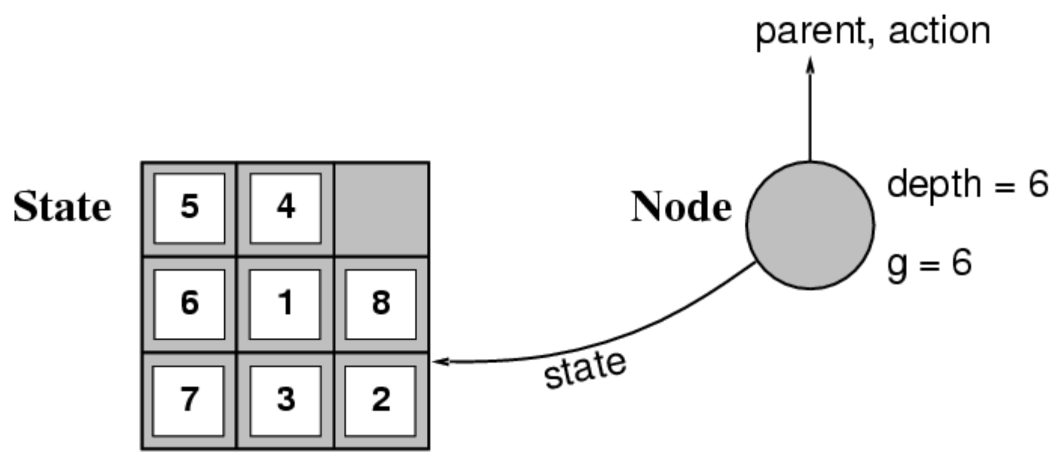
States	position of agent
Actions	moving along path
Goal test	at specified goal state
Path cost	length of path between cities

Search Tree

- State space forms a tree structure
  - Root = start state
  - Each node represents a state
  - Actions are branches, children are all possible next-states
- Search involves expanding a **frontier** of potential next states

Implementation: States vs. Nodes

- A state is a configuration of the world
- A node is a data structure constituting the search tree, including:
  - State
  - Parent node
  - Action taken (edge)
  - Path cost
  - Depth



- More than one node can correspond to a single world state!

Evaluating a Search Strategy

- Search strategy is defined by picking the **order of node expansion**
- Evaluation criteria
  - Completeness
  - Time complexity
  - Space Complexity
  - Optimality
- Time and space complexity measured in
  - b – maximum branching factor for the search tree
  - d – depth of least-cost solution
  - m – maximum depth of the state space

Kinds of Search

- Uninformed
- Informed
- Optimization
- Adversarial

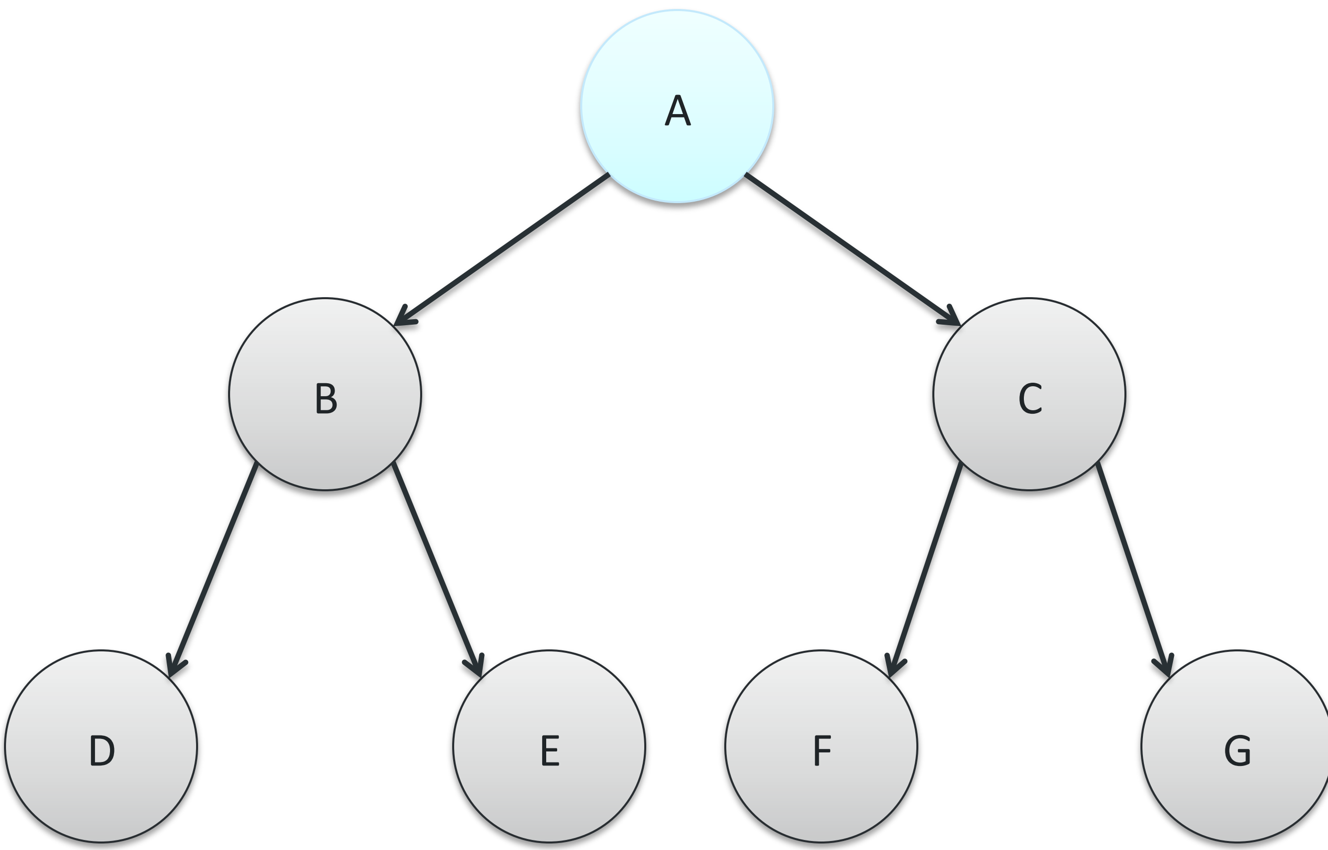
UNINFORMED SEARCH

Uninformed Search Strategies

- Breadth-First Search
- Depth-First Search
- Limited Depth Search
- Iterative Deepening

Breadth-First Search

- Order of node expansion?
  - **Shallowest** unexpanded node

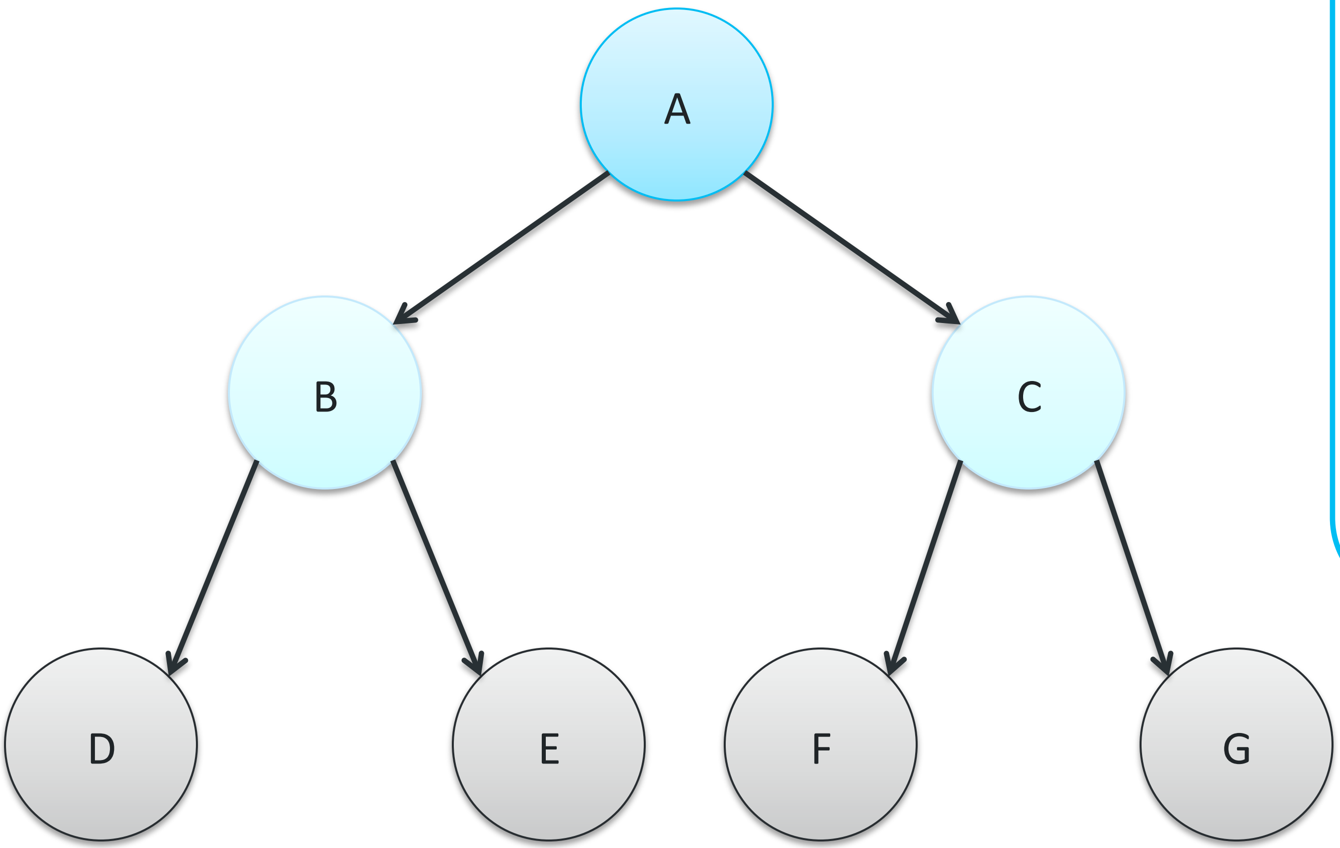


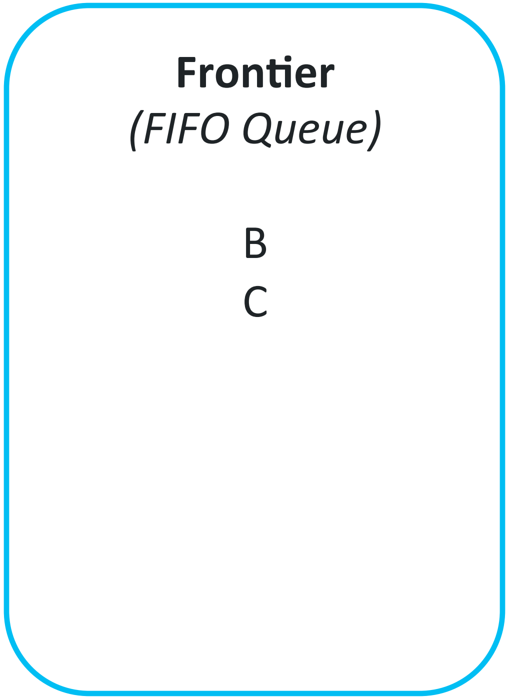
Frontier
(FIFO Queue)

A

Breadth-First Search

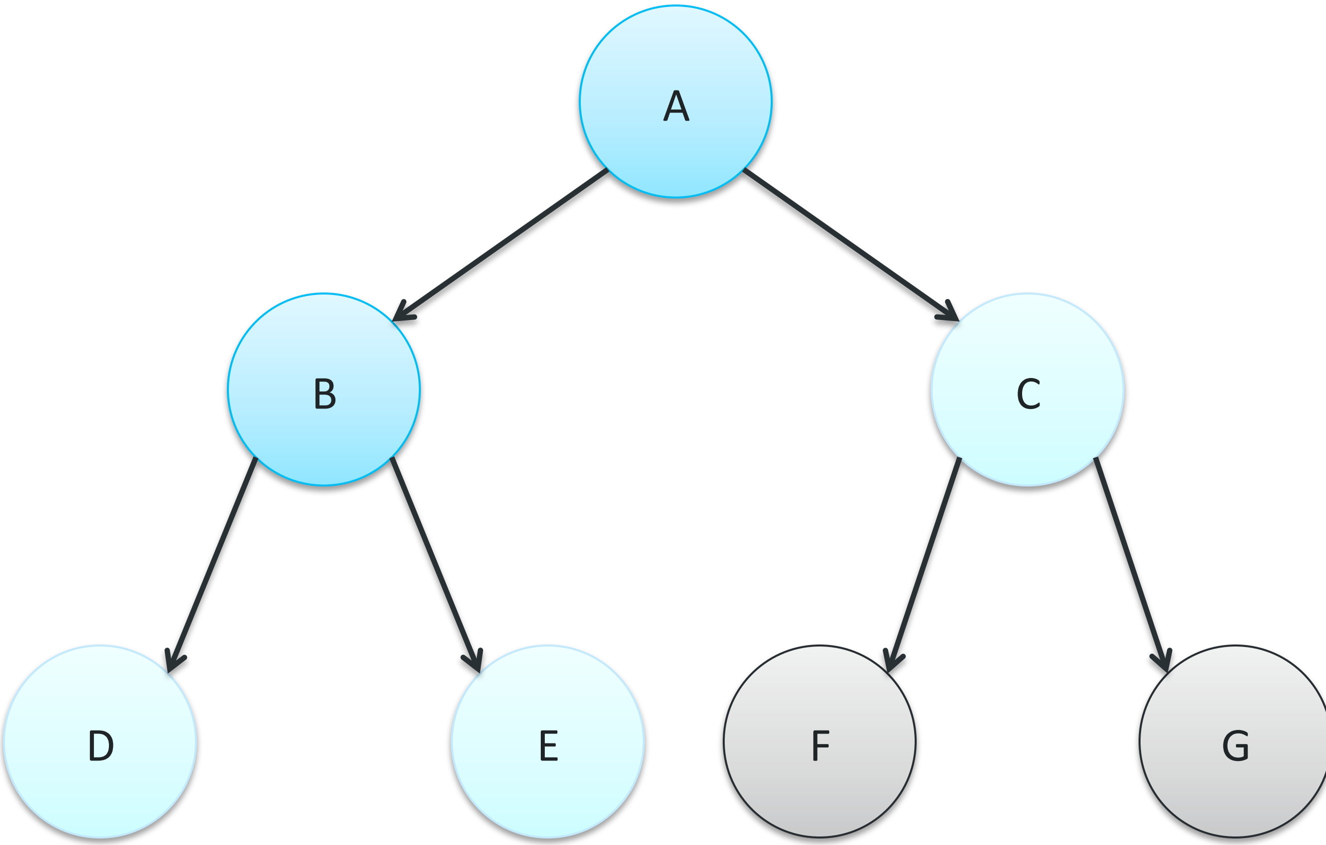
- Order of node expansion?
  - **Shallowest** unexpanded node





Breadth-First Search

- Order of node expansion?
  - **Shallowest** unexpanded node

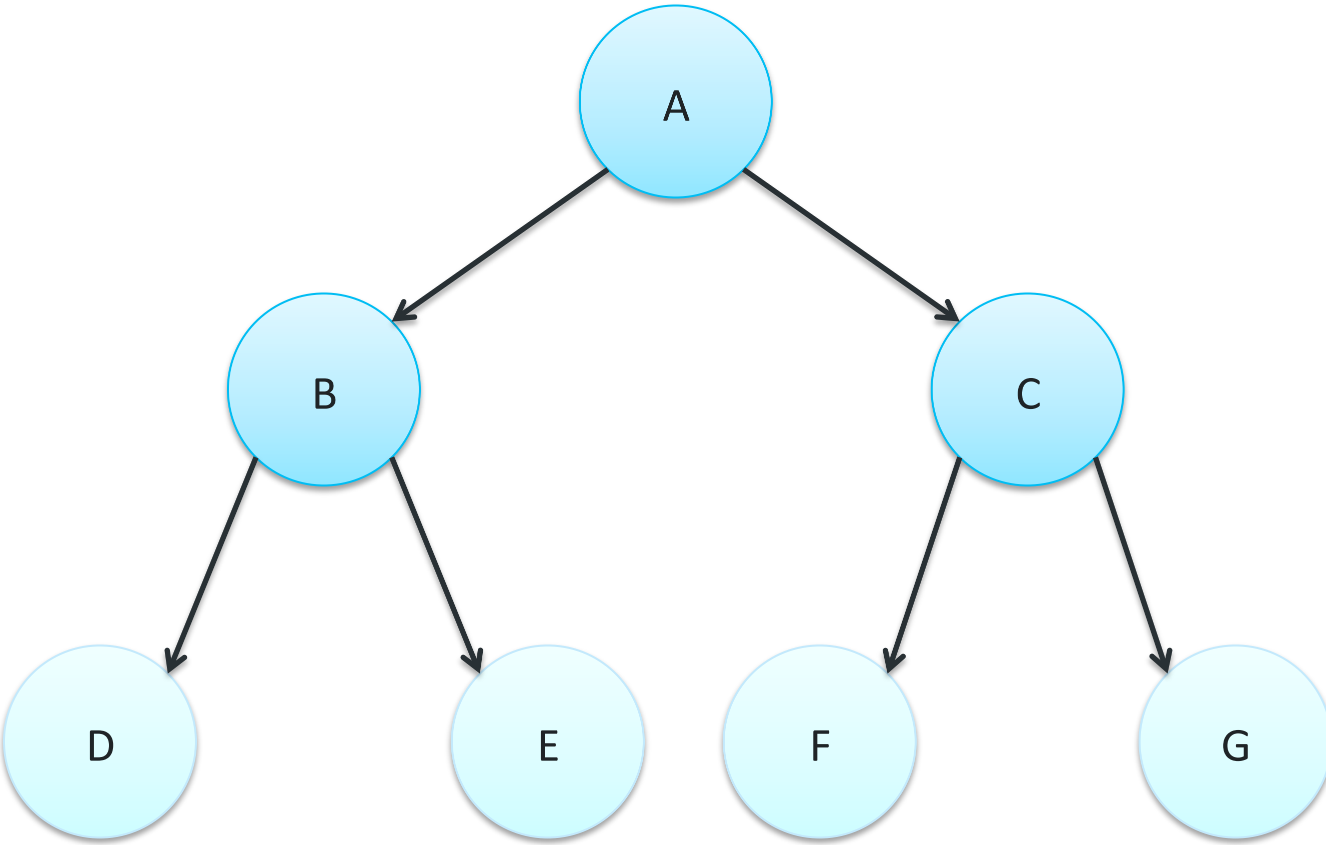


Frontier
(FIFO Queue)

C
D
E

Breadth-First Search

- Order of node expansion?
  - **Shallowest** unexpanded node



Frontier
(FIFO Queue)

D
E
F
G

Breadth-First Search

- Complete?
  -
- Optimal?
  -
- Time cost?
  -
- Space cost?
  -

Breadth-First Search

- Complete?
  - Yes! As long as *b* is finite
- Optimal?
  -
- Time cost?
  -
- Space cost?
  -

Breadth-First Search

- Complete?
  - Yes! As long as *b* is finite
- Optimal?
  - Yes! Assuming?
- Time cost?
  -
- Space cost?
  -

Breadth-First Search

- Complete?
  - Yes! As long as *b* is finite
- Optimal?
  - Yes! Assuming?
- Time cost?
  - $O(b^d)$
- Space cost?
  -

Breadth-First Search

- Complete?
  - Yes! As long as *b* is finite
- Optimal?
  - Yes! Assuming?
- Time cost?
  - $O(b^d)$
- Space cost?
  - $O(b^d)$

Breadth-First Search

- Complete?
  - Yes! As long as *b* is finite
- Optimal?
  - Yes! Assuming?
- Time cost?
  - $O(b^d)$
- **Space cost?**
  - $O(b^d)$

Suppose b=10, a node uses 1000 bytes, 1m nodes/sec can be generated

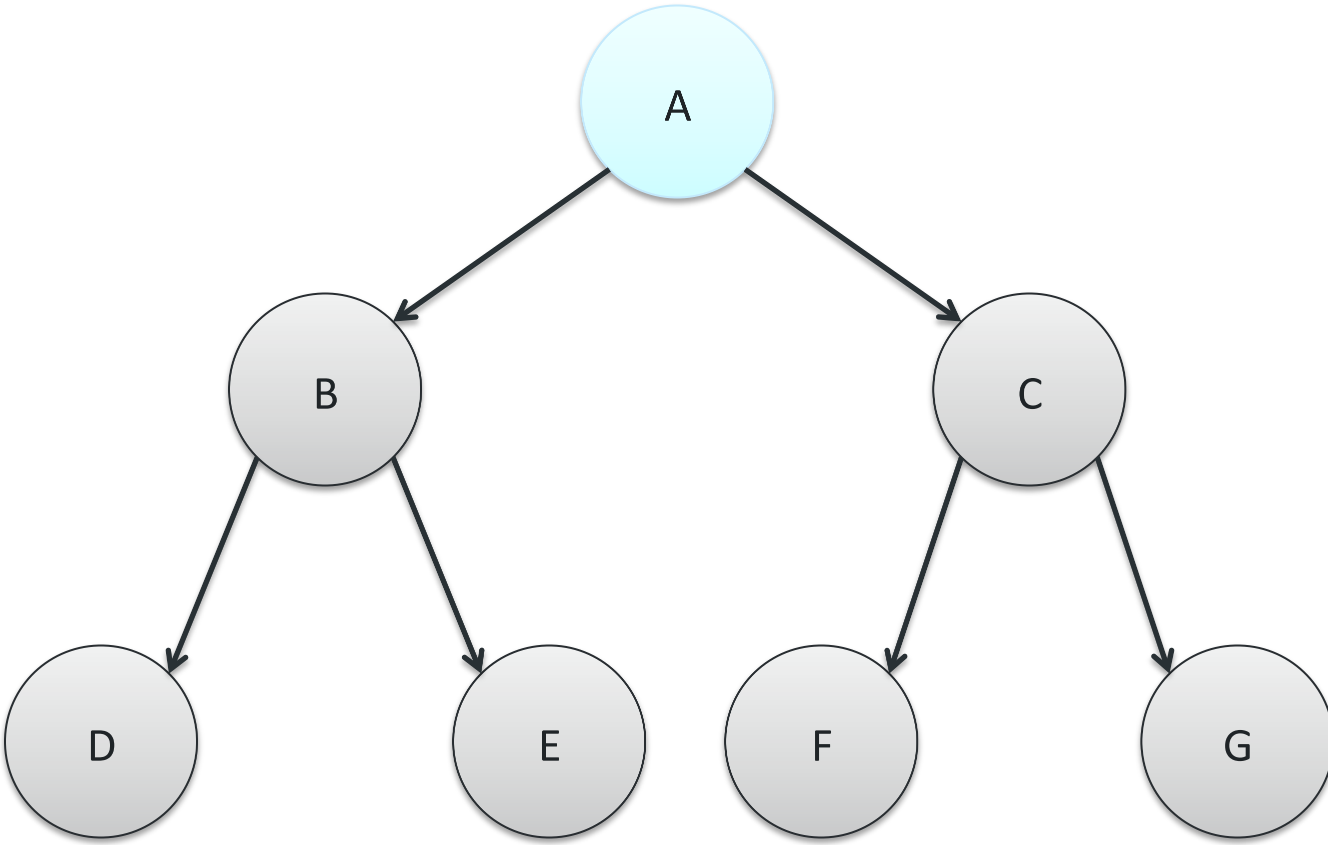
Depth 10: ~3 hours, ~10 terabytes

Depth 12: ~13 days, ~1 petabyte

Depth 14: ~3.5 years, ~99 petabytes

Depth-First Search

- Order of node expansion?
  - **Deepest** unexpanded node

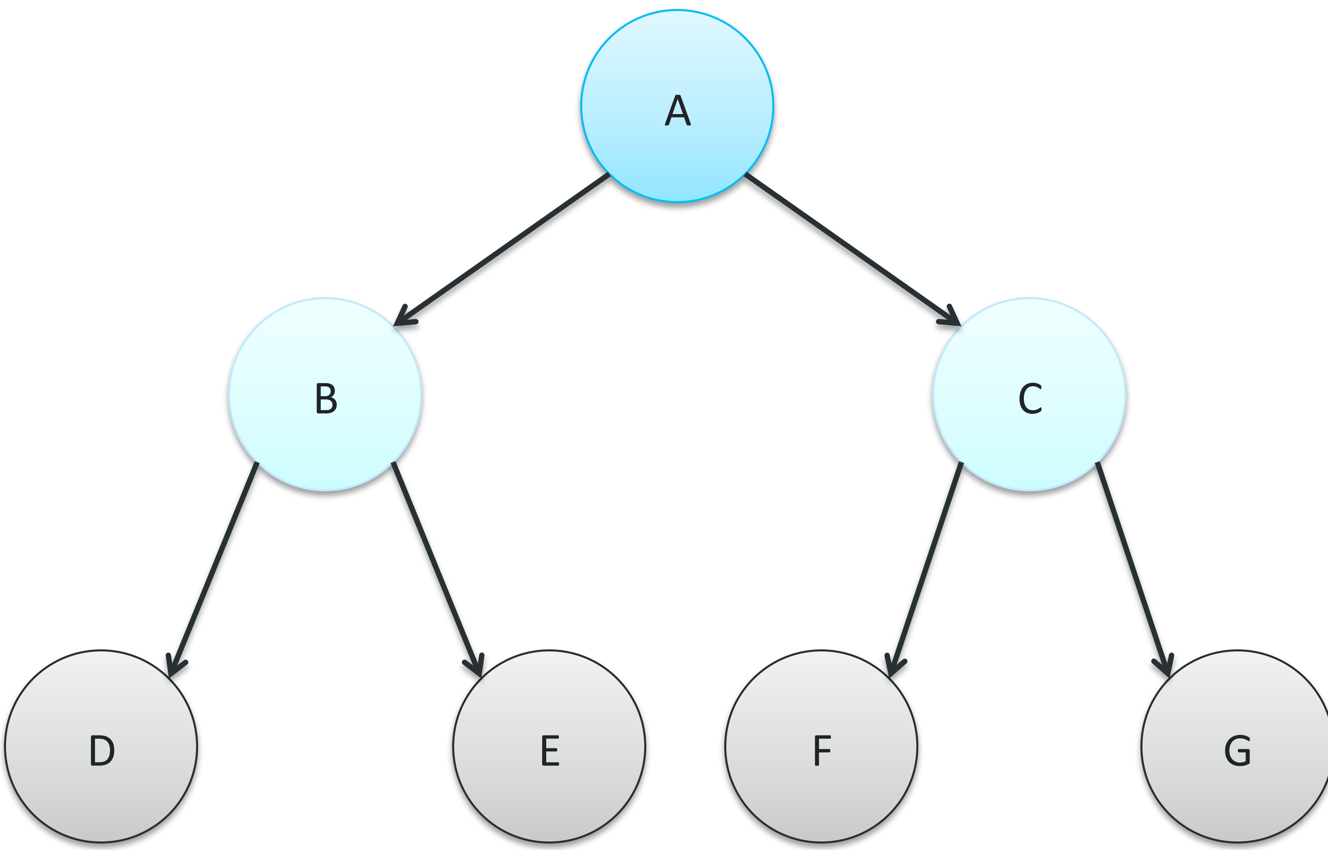


Frontier
(LIFO Queue)

A

Depth-First Search

- Order of node expansion?
  - **Deepest** unexpanded node

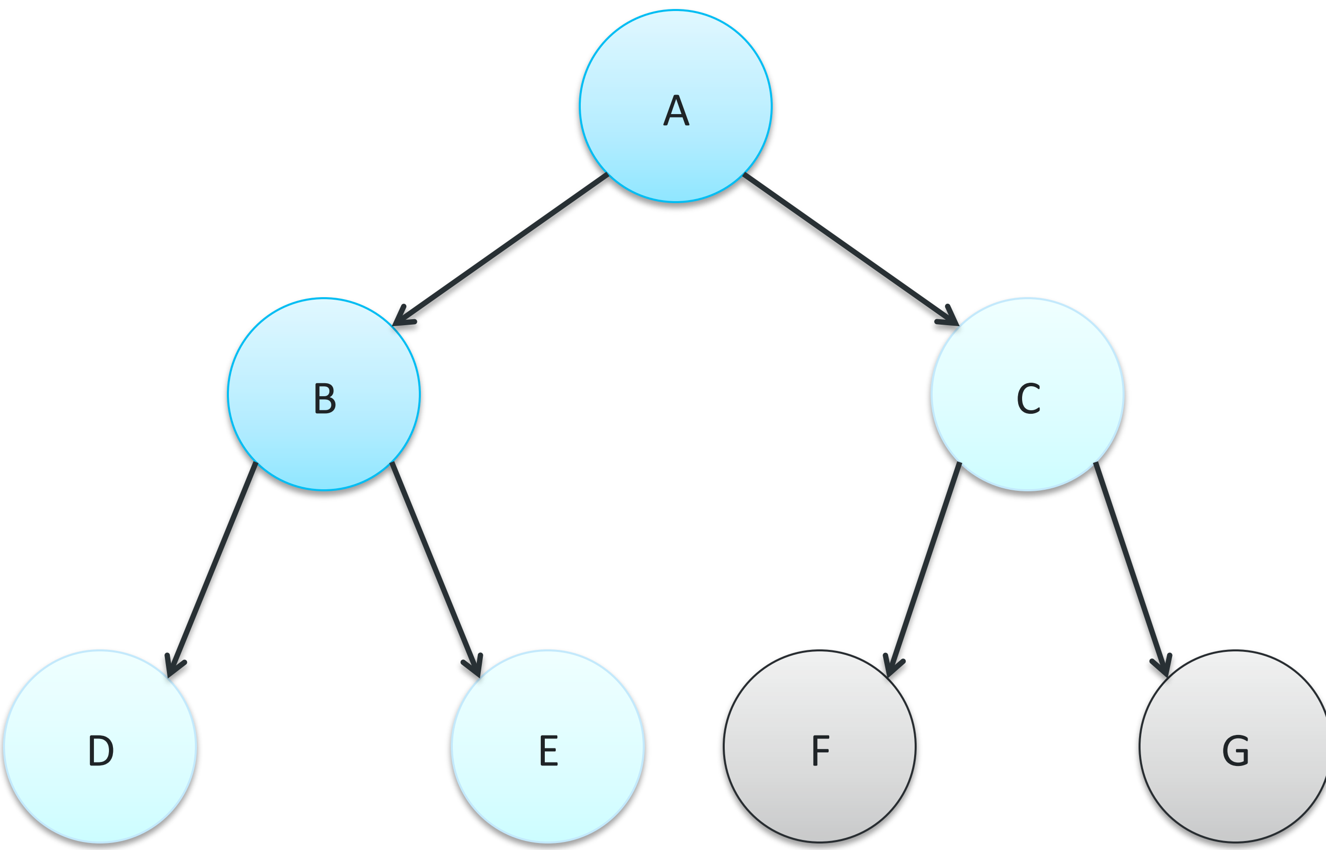


Frontier
(LIFO Queue)

C
B

Depth-First Search

- Order of node expansion?
  - **Deepest** unexpanded node

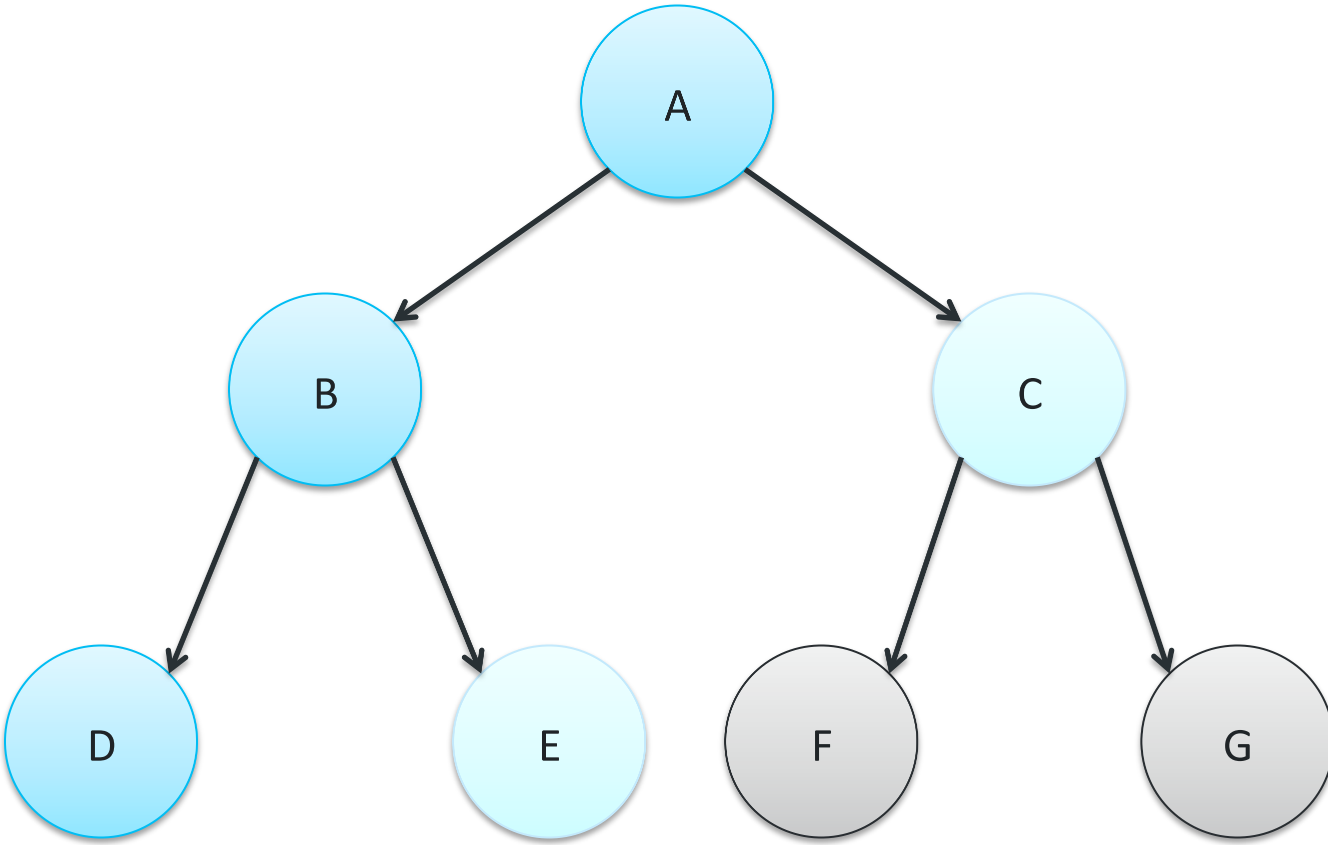


Frontier
(LIFO Queue)

C
E
D

Depth-First Search

- Order of node expansion?
  - **Deepest** unexpanded node

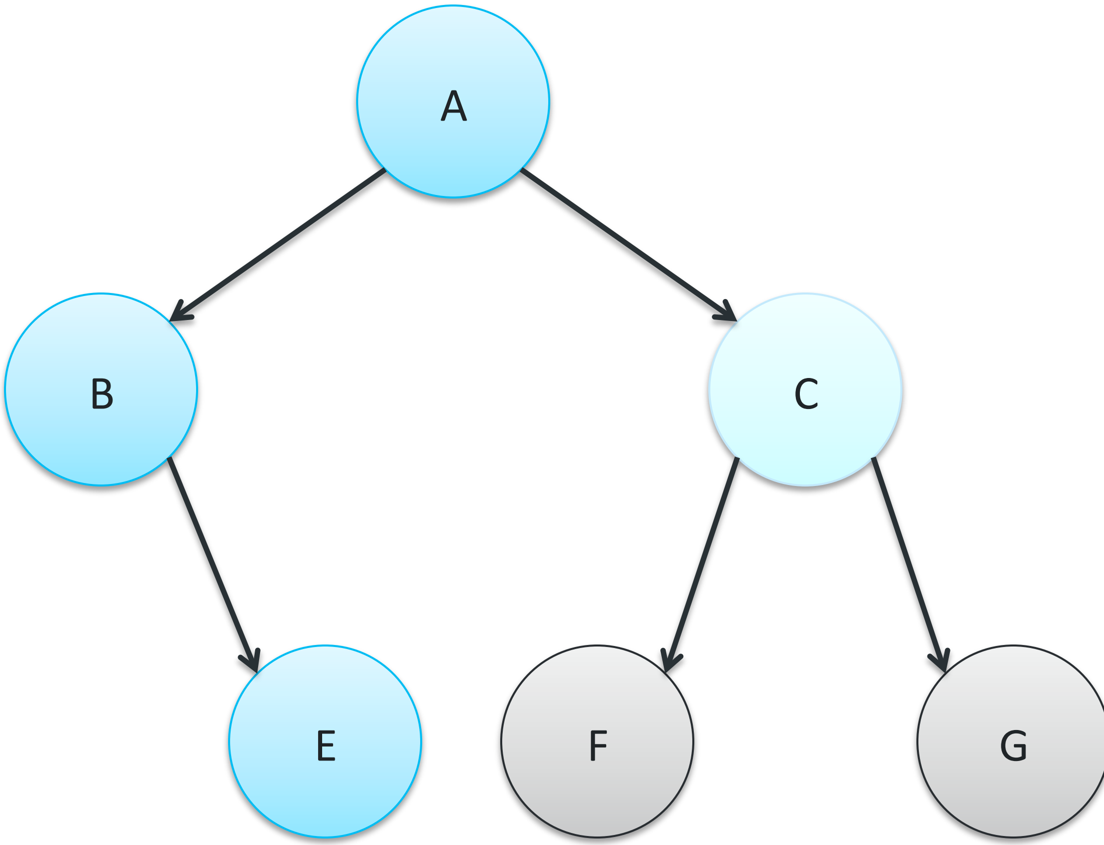


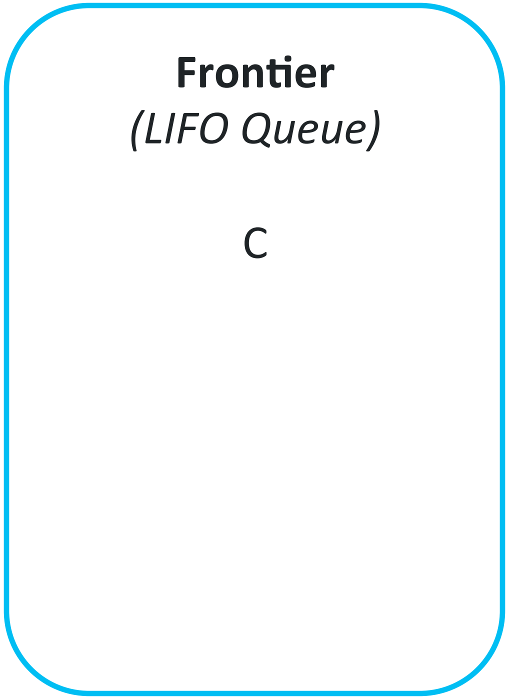
Frontier
(LIFO Queue)

C
E

Depth-First Search

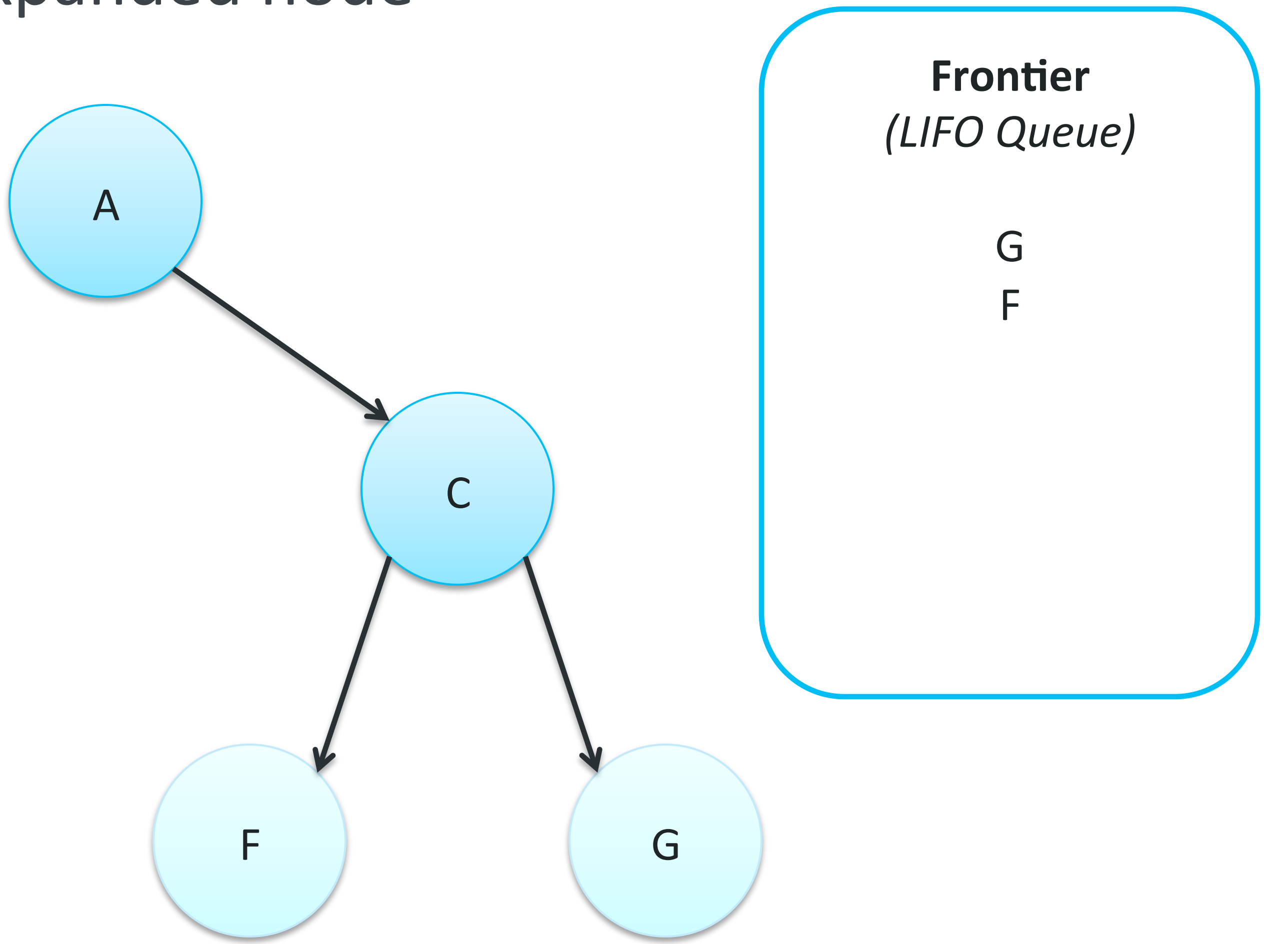
- Order of node expansion?
  - **Deepest** unexpanded node





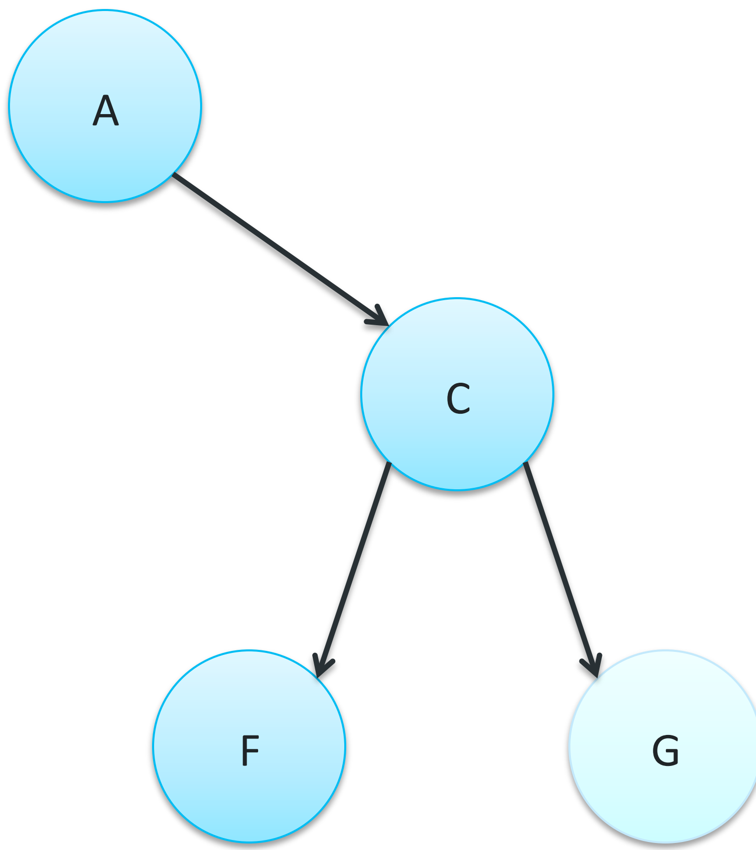
Depth-First Search

- Order of node expansion?
  - **Deepest** unexpanded node



Depth-First Search

- Order of node expansion?
  - **Deepest** unexpanded node

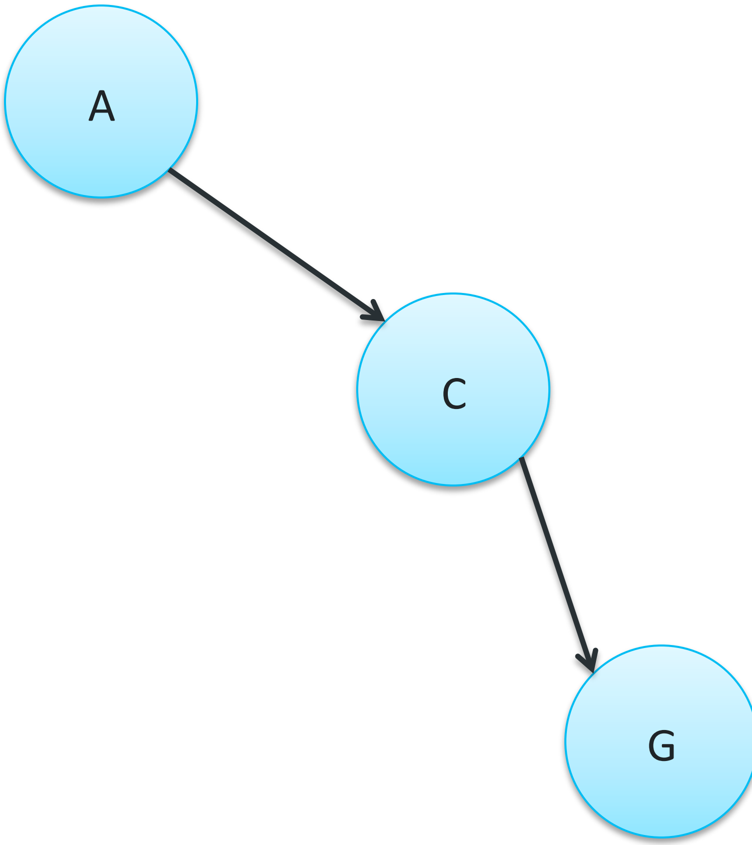


Frontier
(LIFO Queue)

G

Depth-First Search

- Order of node expansion?
  - **Deepest** unexpanded node



Frontier
(LIFO Queue)

Depth-First Search

- Complete?
  -
- Optimal?
  -
- Time cost?
  -
- Space cost?
  -

Depth-First Search

- Complete?
  - No! Why? Unless?
- Optimal?
  -
- Time cost?
  -
- Space cost?
  -

Depth-First Search

- Complete?
  - No! Why? Unless?
- Optimal?
  - No! Why?
- Time cost?
  -
- Space cost?
  -

Depth-First Search

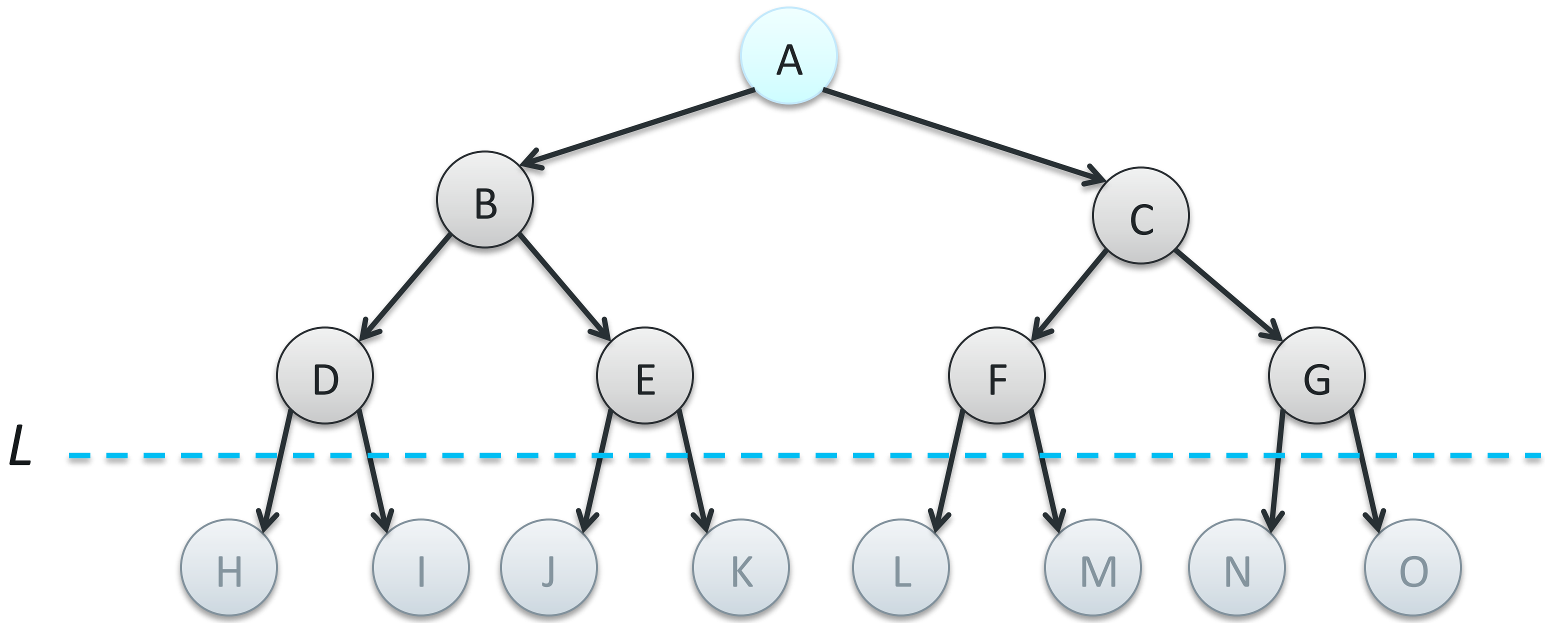
- Complete?
  - No! Why? Unless?
- Optimal?
  - No! Why?
- Time cost?
  - $O(b^m)$
- Space cost?
  -

Depth-First Search

- Complete?
  - No! Why? Unless?
- Optimal?
  - No! Why?
- Time cost?
  - $O(b^m)$
- Space cost?
  - $O(bm)$

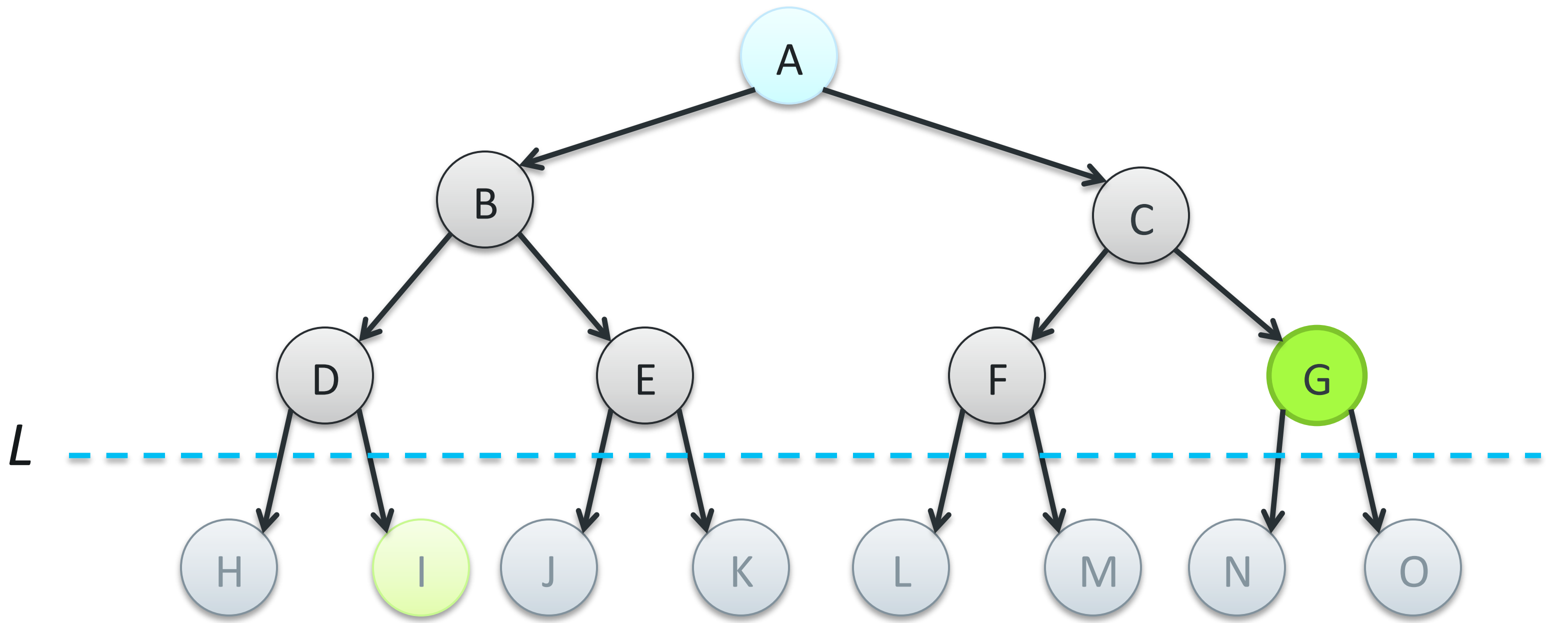
Depth-Limited Search

- Depth first search, but don't go past a pre-set limit (*L*) while searching



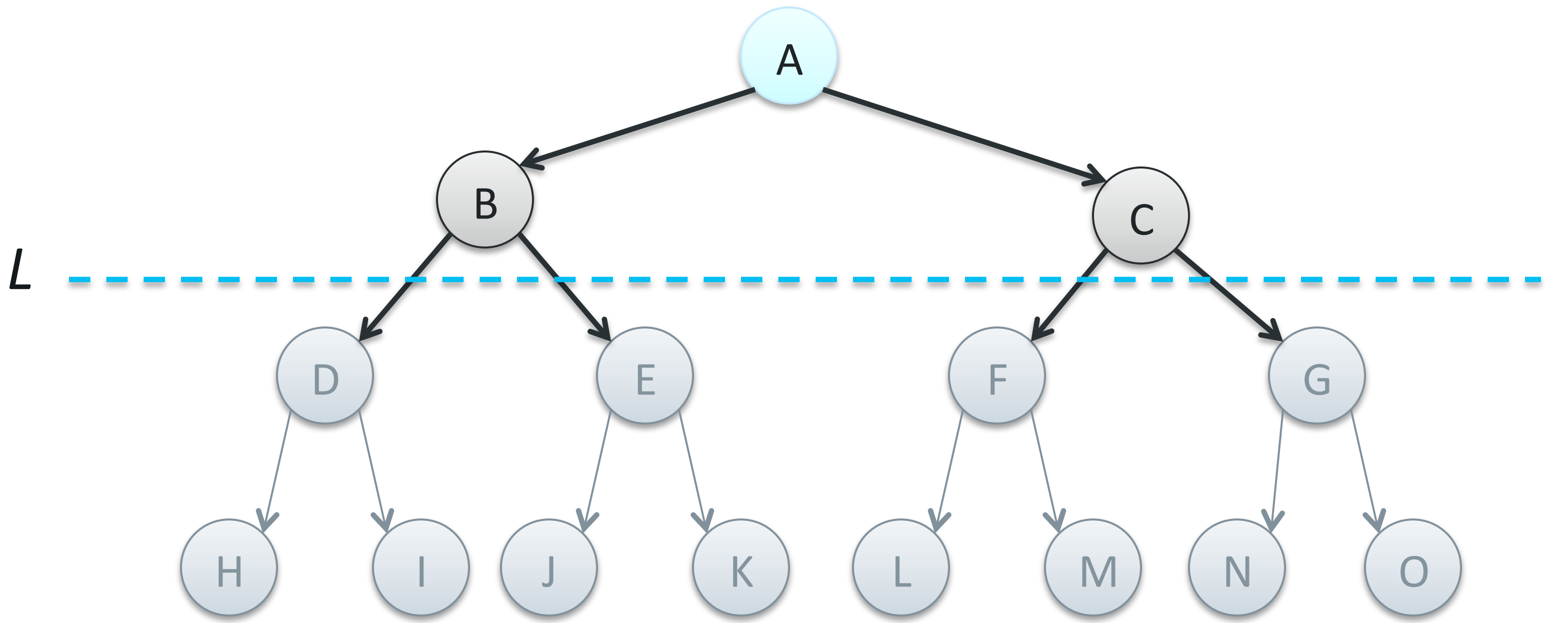
Depth-Limited Search

- Depth first search, but don't go past a pre-set limit (*L*) while searching



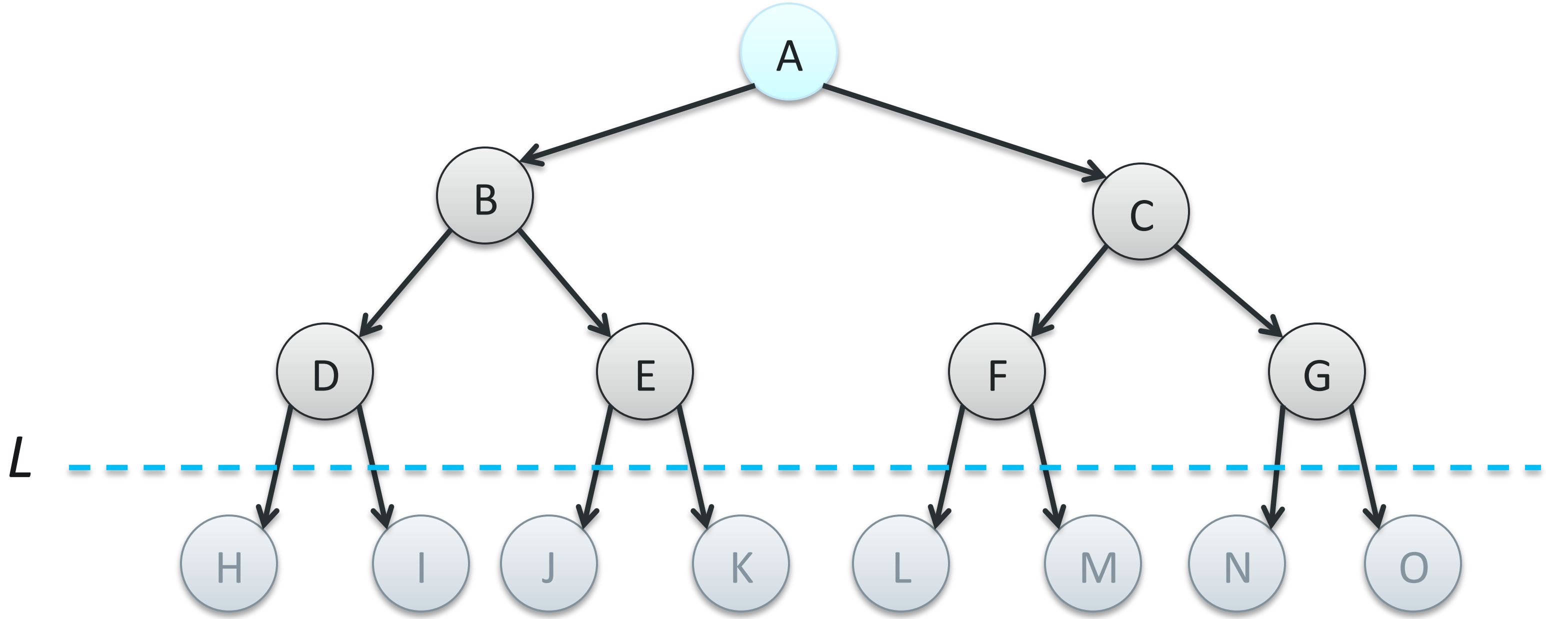
Iterative Deepening

- Limited-depth search, progressively lowering the limit



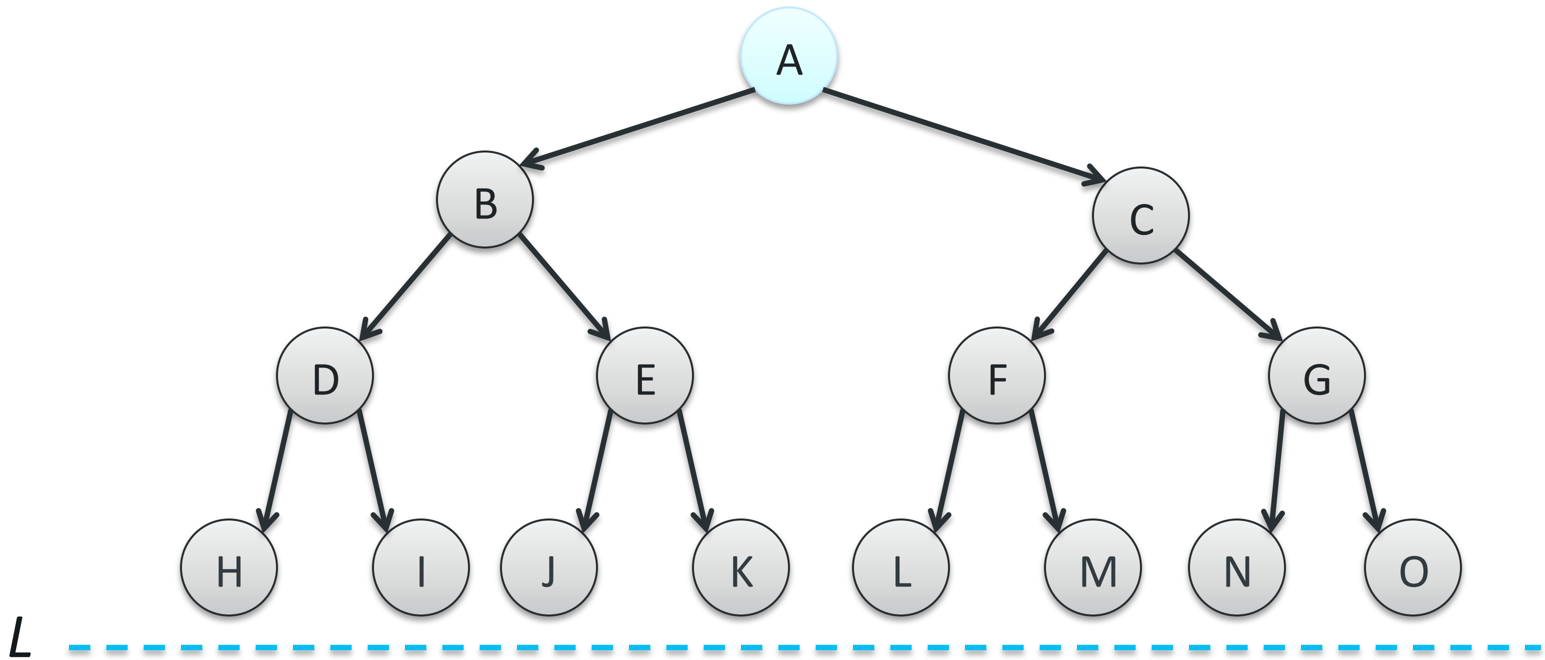
Iterative Deepening

- Limited-depth search, progressively lowering the limit



Iterative Deepening

- Limited-depth search, progressively lowering the limit



Limited Depth Search

- Combines benefits of depth- and breadth-first

- Number of nodes generated:
  - $(d+1)b^0 + (d)b^1 + (d-1)b^2 + \ldots + (2)b^{d-1} + (1)b^d = O(b^d)$

Limited Depth Search

- Combines benefits of depth- and breadth-first

- Number of nodes generated:
  - $(d+1)b^0 + (d)b^1 + (d-1)b^2 + \ldots + (2)b^{d-1} + (1)b^d =$ **$O(b^d)$**

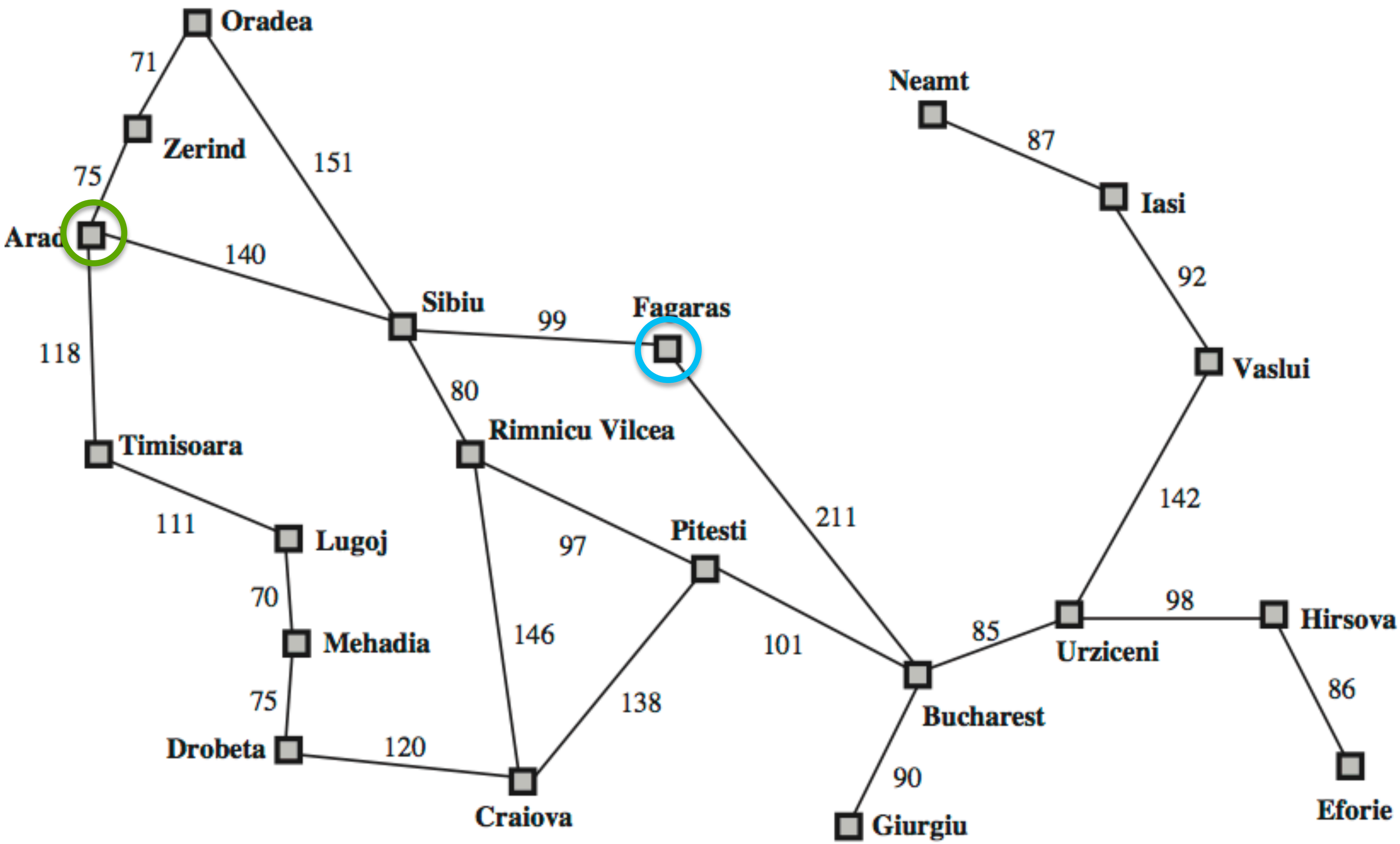
Iterative Deepening Search

- Combines benefits of depth- and breadth-first
- Number of nodes generated:
  - $(d+1)b^0 + (d)b^1 + (d-1)b^2 + \dots + (2)b^{d-1} + (1)b^d =$ $\mathbf{O(b^d)}$
- Spatial complexity: $O(bd)$
- **Comparison** to breadth-first search
  - Assume b = 10, d = 5
  - N(IDS) = 6 + 50 + 400 + 3,000 + 20,000 + 100,000 = 123,456
  - N(BFS) = 1 + 10 + 100 + 1,000 + 10,000 + 100,000 = 111,111
  - Overhead: (123,456 – 111,111)/111,111 = 11%

Summary

Search Type	Pros	Cons
Breadth-first search	Good for problems with expected shallow solutions	Takes up too much space
Depth-first search	Efficient space usage Good when lots of solutions	Gets stuck in loops (w/o modification) Can go too deep
Iterative deepening	Efficient space usage Does not get stuck on a bad path	Overhead of additional searching

In-Class Exercise



INFORMED SEARCH

Informed Search

- Use problem-specific knowledge beyond the problem definition to find solutions more efficiently

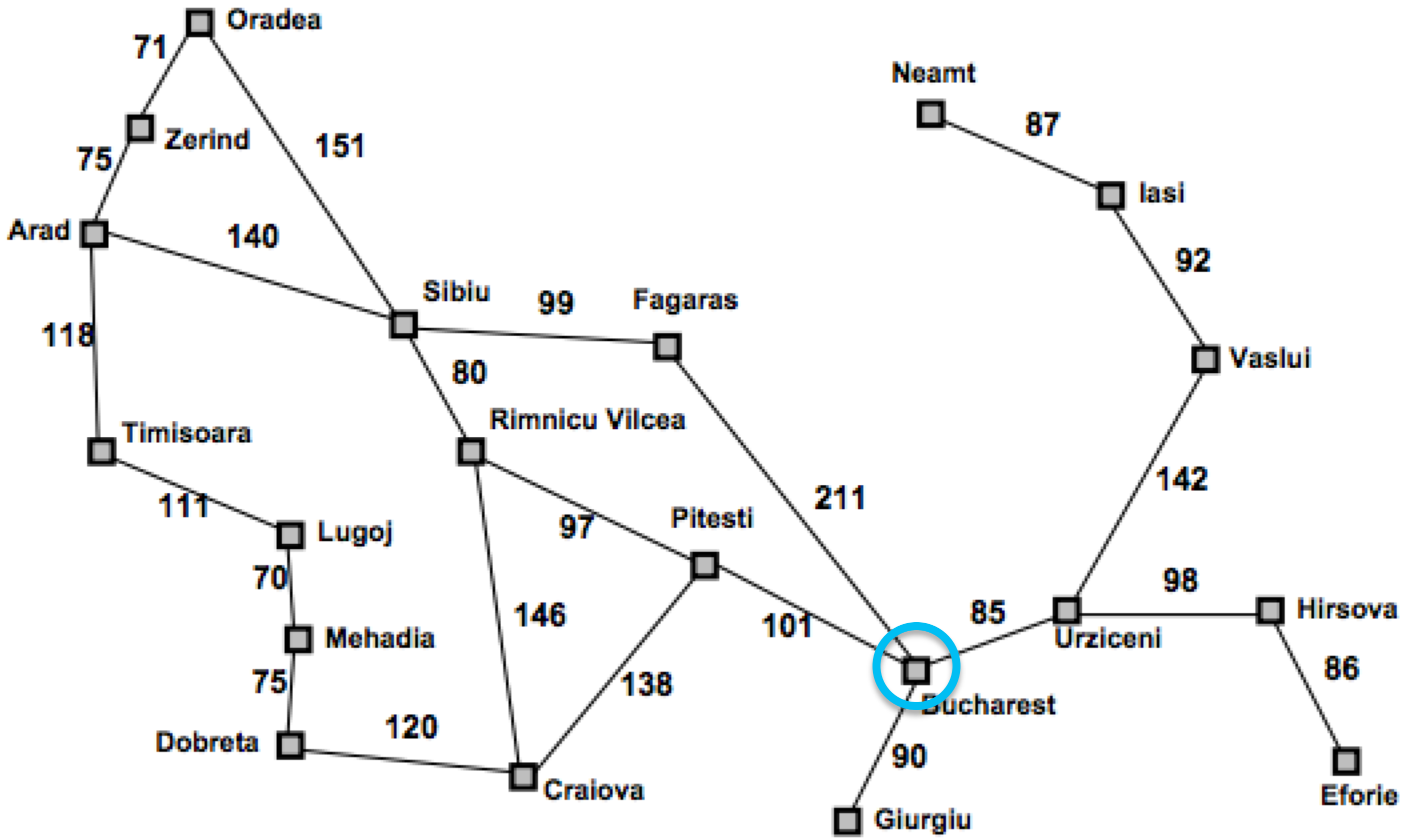
- **Heuristic**: which node should be expanded next?

Heuristic Search

- Use an evaluation function *f(n)* for each node
  - Estimate of “desirability”
  - Includes a heuristic function, *h(n)*

- Expand the **most desirable** unexpanded node

Example: Heuristic for Path-Finding



Straight−line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best-First Search

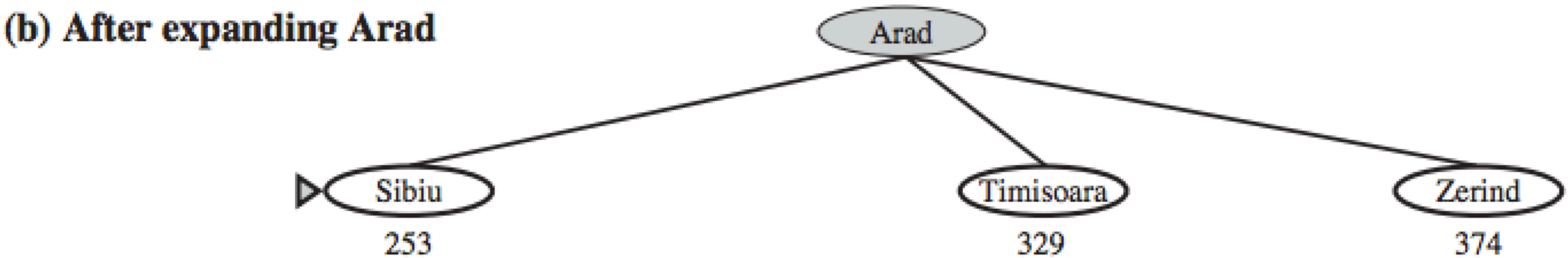
- Evaluation function $f(n) = h(n)$
  - E.g. $h(n)$ = straight line distance from n to goal
- Search strategy
  - Expand the node that **appears** closest to the goal

Greedy Best-First Search: Arad to Bucharest

(a) The initial state

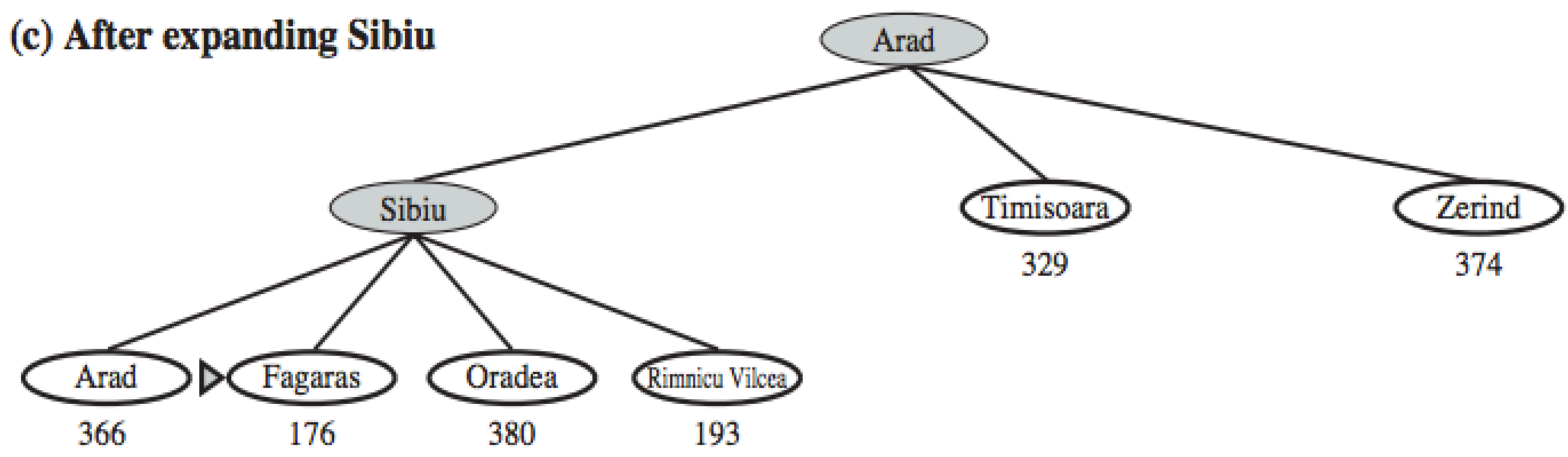
Arad	366	**Fagaras**	178	**Mehadia**	241	**Sibiu**	253
Bucharest	0	**Giurgiu**	77	**Neamt**	234	**Timisoara**	329
Craiova	160	**Hirsova**	151	**Oradea**	380	**Urziceni**	80
Dobreta	242	**Iasi**	226	**Pitesti**	98	**Vaslui**	199
Eforie	161	**Lugoj**	244	**Rimnicu Vilcea**	193	**Zerind**	374

Greedy Best-First Search: Arad to Bucharest



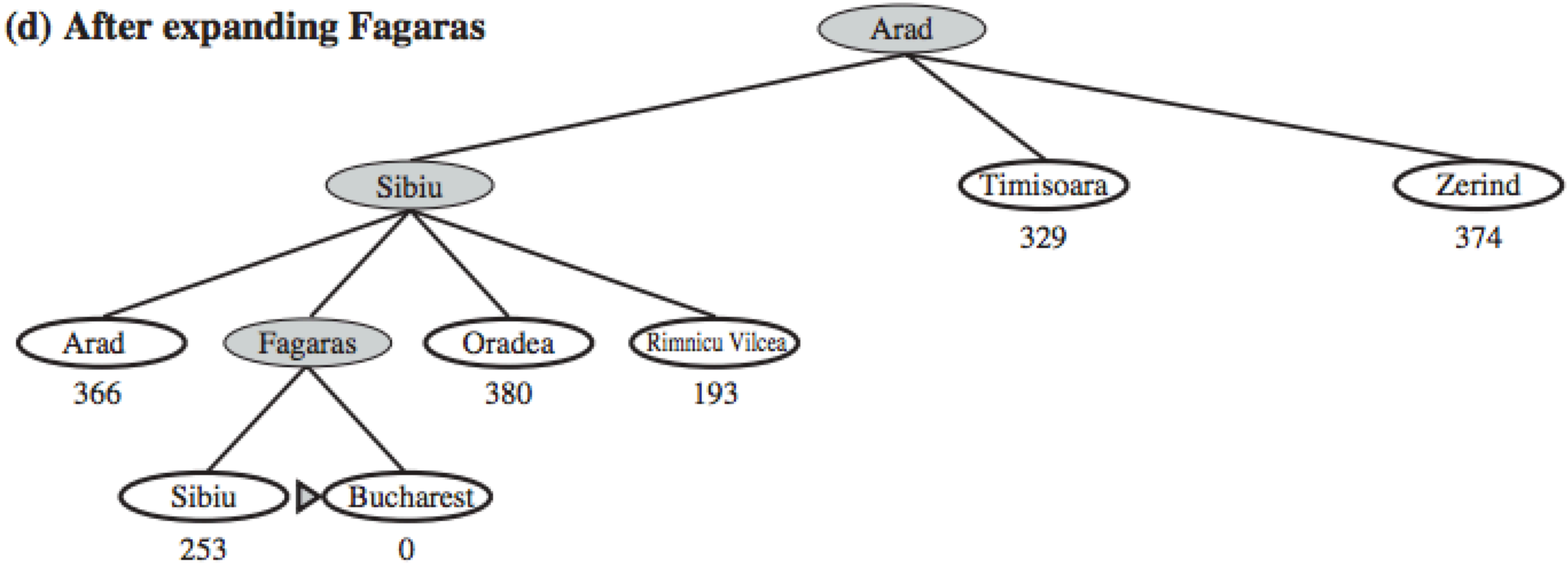
Arad	366	**Fagaras**	178	**Mehadia**	241	**Sibiu**	253
Bucharest	0	**Giurgiu**	77	**Neamt**	234	**Timisoara**	329
Craiova	160	**Hirsova**	151	**Oradea**	380	**Urziceni**	80
Dobreta	242	**Iasi**	226	**Pitesti**	98	**Vaslui**	199
Eforie	161	**Lugoj**	244	**Rimnicu Vilcea**	193	**Zerind**	374

Greedy Best-First Search: Arad to Bucharest



Arad	366	**Fagaras**	178	**Mehadia**	241	**Sibiu**	253
Bucharest	0	**Giurgiu**	77	**Neamt**	234	**Timisoara**	329
Craiova	160	**Hirsova**	151	**Oradea**	380	**Urziceni**	80
Dobreta	242	**Iasi**	226	**Pitesti**	98	**Vaslui**	199
Eforie	161	**Lugoj**	244	**Rimnicu Vilcea**	193	**Zerind**	374

Greedy Best-First Search: Arad to Bucharest



Arad	366	**Fagaras**	178	**Mehadia**	241	**Sibiu**	253
Bucharest	0	**Giurgiu**	77	**Neamt**	234	**Timisoara**	329
Craiova	160	**Hirsova**	151	**Oradea**	380	**Urziceni**	80
Dobreta	242	**Iasi**	226	**Pitesti**	98	**Vaslui**	199
Eforie	161	**Lugoj**	244	**Rimnicu Vilcea**	193	**Zerind**	374

Greedy Best-First Search

- Complete?
  -
- Optimal?
  -
- Time?
  -
- Space?
  -

Greedy Best-First Search

- Complete?
  - No – can get stuck in loops
- Optimal?
  -
- Time?
  -
- Space?
  -

Greedy Best-First Search

- Complete?
  - No – can get stuck in loops
- Optimal?
  - No. Why?
- Time?
  -
- Space?
  -

Greedy Best-First Search

- Complete?
  - No – can get stuck in loops
- Optimal?
  - No. Why?
- Time?
  - $O(b^m)$, but improves with great heuristic
- Space?
  -

Greedy Best-First Search

- Complete?
  - No – can get stuck in loops
- Optimal?
  - No. Why?
- Time?
  - $O(b^m)$, but improves with great heuristic
- Space?
  - $O(b^m)$, keeps all nodes in memory

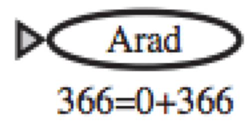
A* Search

- Avoid expanding paths that are already expensive

- Evaluation function: $f(n) = g(n) + h(n)$
  - g(n) = cost of path so far
  - h(n) = estimated cost from n to goal
  - f(n) = estimated **total cost** of path through n to goal

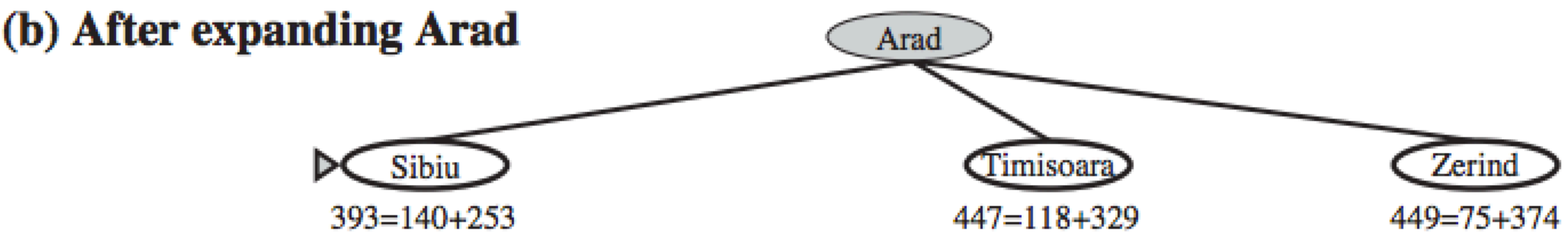
A* Search Example: Arad to Bucharest

(a) The initial state



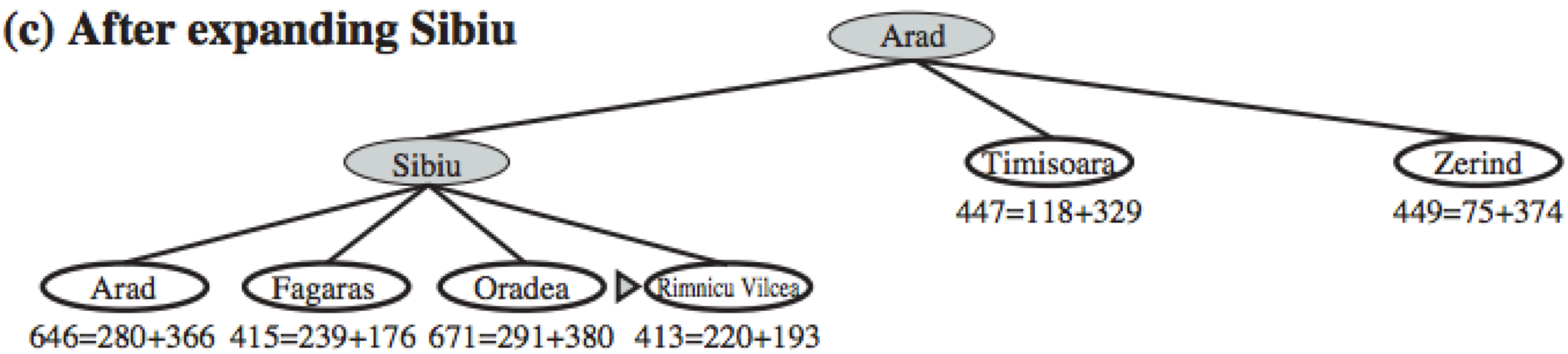
Arad	366	**Fagaras**	178	**Mehadia**	241	**Sibiu**	253
Bucharest	0	**Giurgiu**	77	**Neamt**	234	**Timisoara**	329
Craiova	160	**Hirsova**	151	**Oradea**	380	**Urziceni**	80
Dobreta	242	**Iasi**	226	**Pitesti**	98	**Vaslui**	199
Eforie	161	**Lugoj**	244	**Rimnicu Vilcea**	193	**Zerind**	374

A* Search Example: Arad to Bucharest



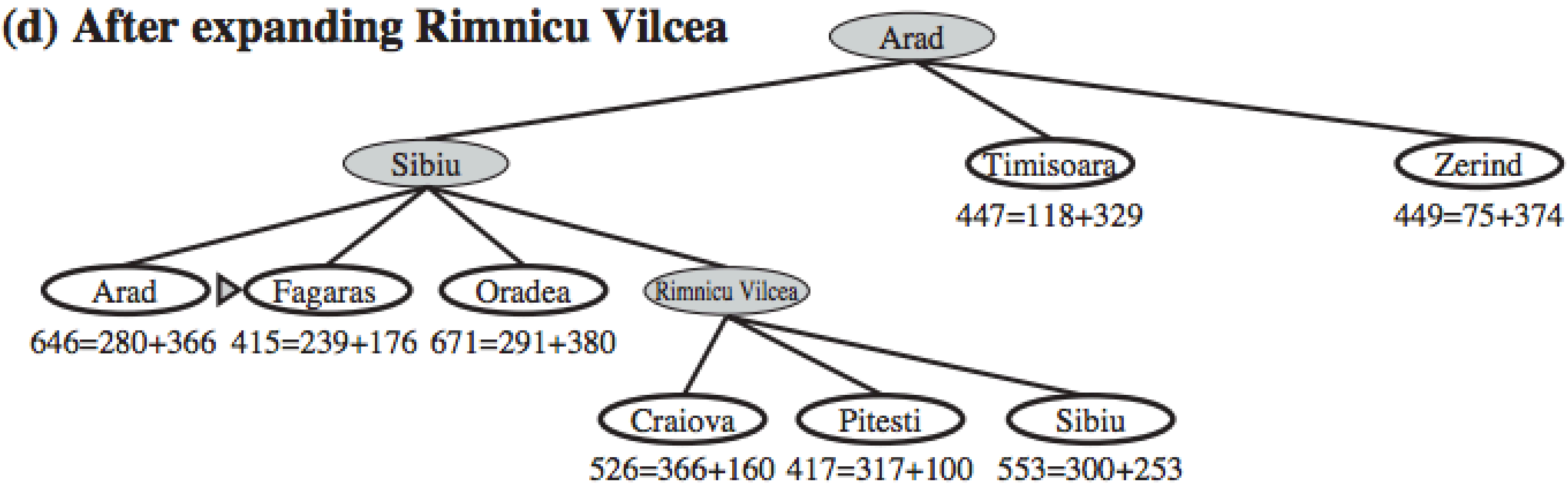
Arad	366	**Fagaras**	178	**Mehadia**	241	**Sibiu**	253
Bucharest	0	**Giurgiu**	77	**Neamt**	234	**Timisoara**	329
Craiova	160	**Hirsova**	151	**Oradea**	380	**Urziceni**	80
Dobreta	242	**Iasi**	226	**Pitesti**	98	**Vaslui**	199
Eforie	161	**Lugoj**	244	**Rimnicu Vilcea**	193	**Zerind**	374

A* Search Example: Arad to Bucharest



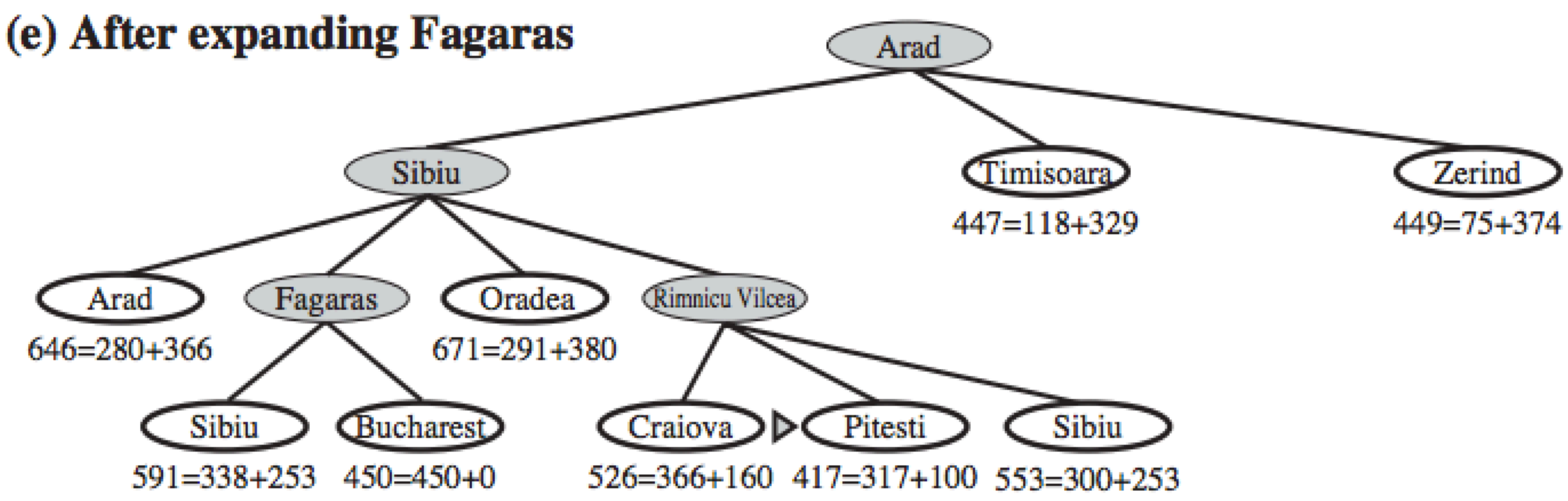
Arad	366	**Fagaras**	178	**Mehadia**	241	**Sibiu**	253
Bucharest	0	**Giurgiu**	77	**Neamt**	234	**Timisoara**	329
Craiova	160	**Hirsova**	151	**Oradea**	380	**Urziceni**	80
Dobreta	242	**Iasi**	226	**Pitesti**	98	**Vaslui**	199
Eforie	161	**Lugoj**	244	**Rimnicu Vilcea**	193	**Zerind**	374

A* Search Example: Arad to Bucharest



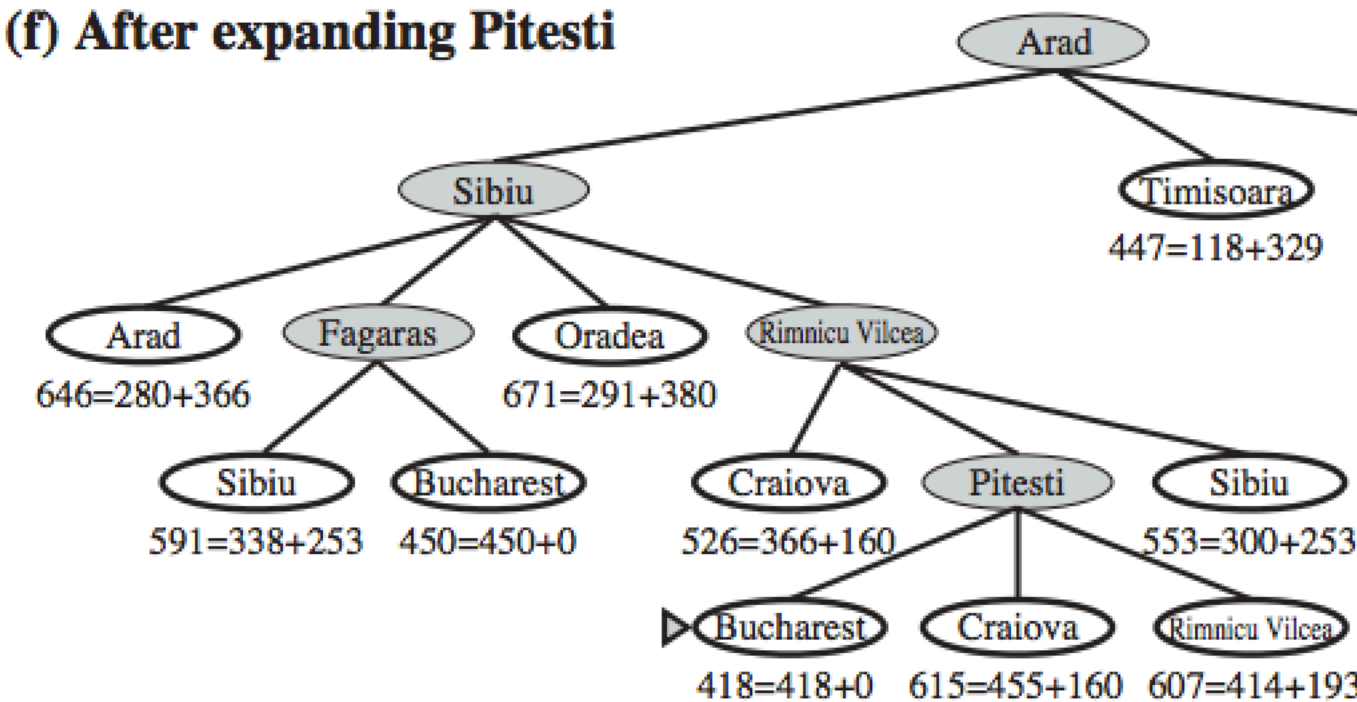
Arad	366	**Fagaras**	178	**Mehadia**	241	**Sibiu**	253
Bucharest	0	**Giurgiu**	77	**Neamt**	234	**Timisoara**	329
Craiova	160	**Hirsova**	151	**Oradea**	380	**Urziceni**	80
Dobreta	242	**Iasi**	226	**Pitesti**	98	**Vaslui**	199
Eforie	161	**Lugoj**	244	**Rimnicu Vilcea**	193	**Zerind**	374

A* Search Example: Arad to Bucharest



Arad	366	**Fagaras**	178	**Mehadia**	241	**Sibiu**	253
Bucharest	0	**Giurgiu**	77	**Neamt**	234	**Timisoara**	329
Craiova	160	**Hirsova**	151	**Oradea**	380	**Urziceni**	80
Dobreta	242	**Iasi**	226	**Pitesti**	98	**Vaslui**	199
Eforie	161	**Lugoj**	244	**Rimnicu Vilcea**	193	**Zerind**	374

A* Search Example: Arad to Bucharest



Arad	366	**Fagaras**	178	**Mehadia**	241	**Sibiu**	253
Bucharest	0	**Giurgiu**	77	**Neamt**	234	**Timisoara**	329
Craiova	160	**Hirsova**	151	**Oradea**	380	**Urziceni**	80
Dobreta	242	**Iasi**	226	**Pitesti**	98	**Vaslui**	199
Eforie	161	**Lugoj**	244	**Rimnicu Vilcea**	193	**Zerind**	374

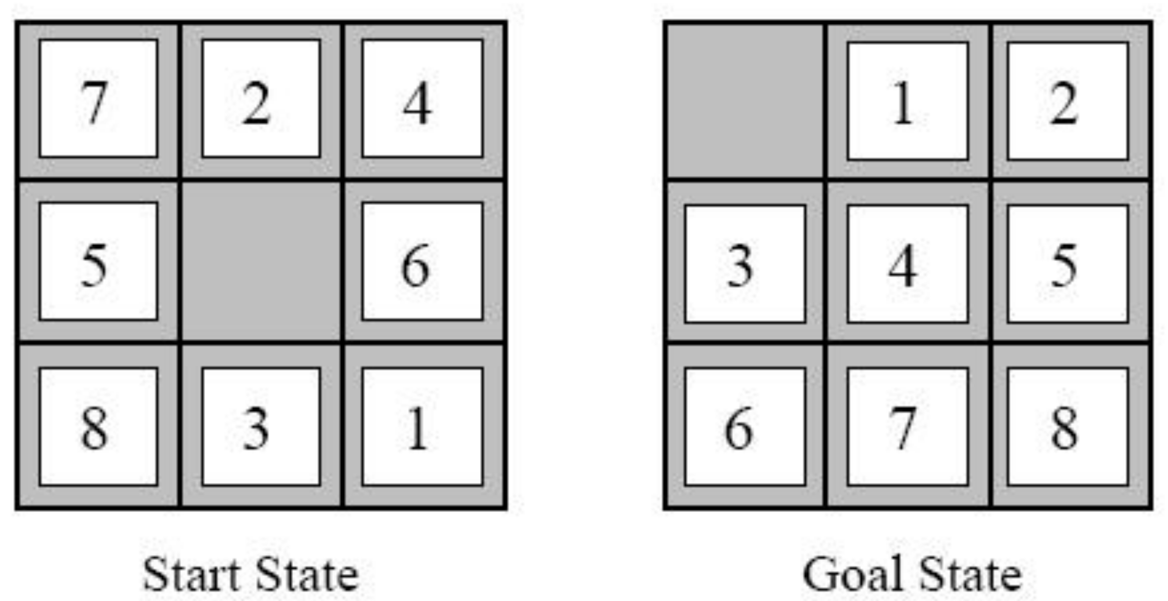
A* Search

- Complete?
  - Yes
- Optimal?
  - Yes!
- Time?
  - Exponential
- Space?
  - Keeps all nodes in memory

Admissible Heuristics

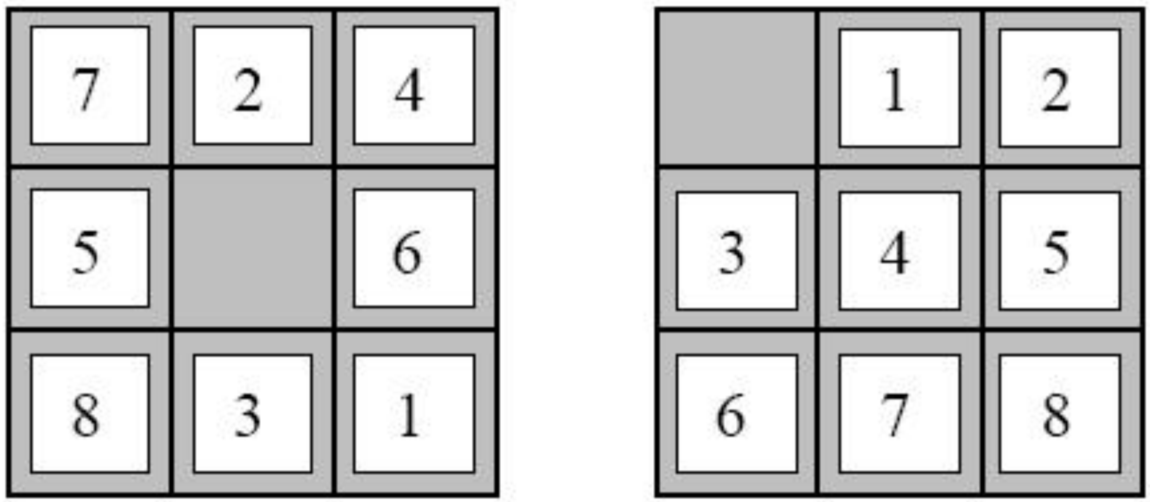
- A heuristic is admissible if, for every node n, $h(n) \leq h^*(n)$
  - $h^*(n)$: true cost to reach the goal from state n
- Example: path-finding heuristic never overestimates actual road distance
- Generate from a relaxed problem
  - Fewer restrictions than original problem

Class Activity



- What are some heuristics for the 8 puzzle?

Class Activity



Start State   Goal State

- Sketch out first few steps of a* search given the total number of misplaced tiles heuristic

OPTIMIZATION

Optimization Problems

- Path to goal state is irrelevant
- Don't know what the goal state is exactly, trying to find it!
- Local search algorithms
  - Keep single "current" state, hunt for solution

Example: *n*-queens

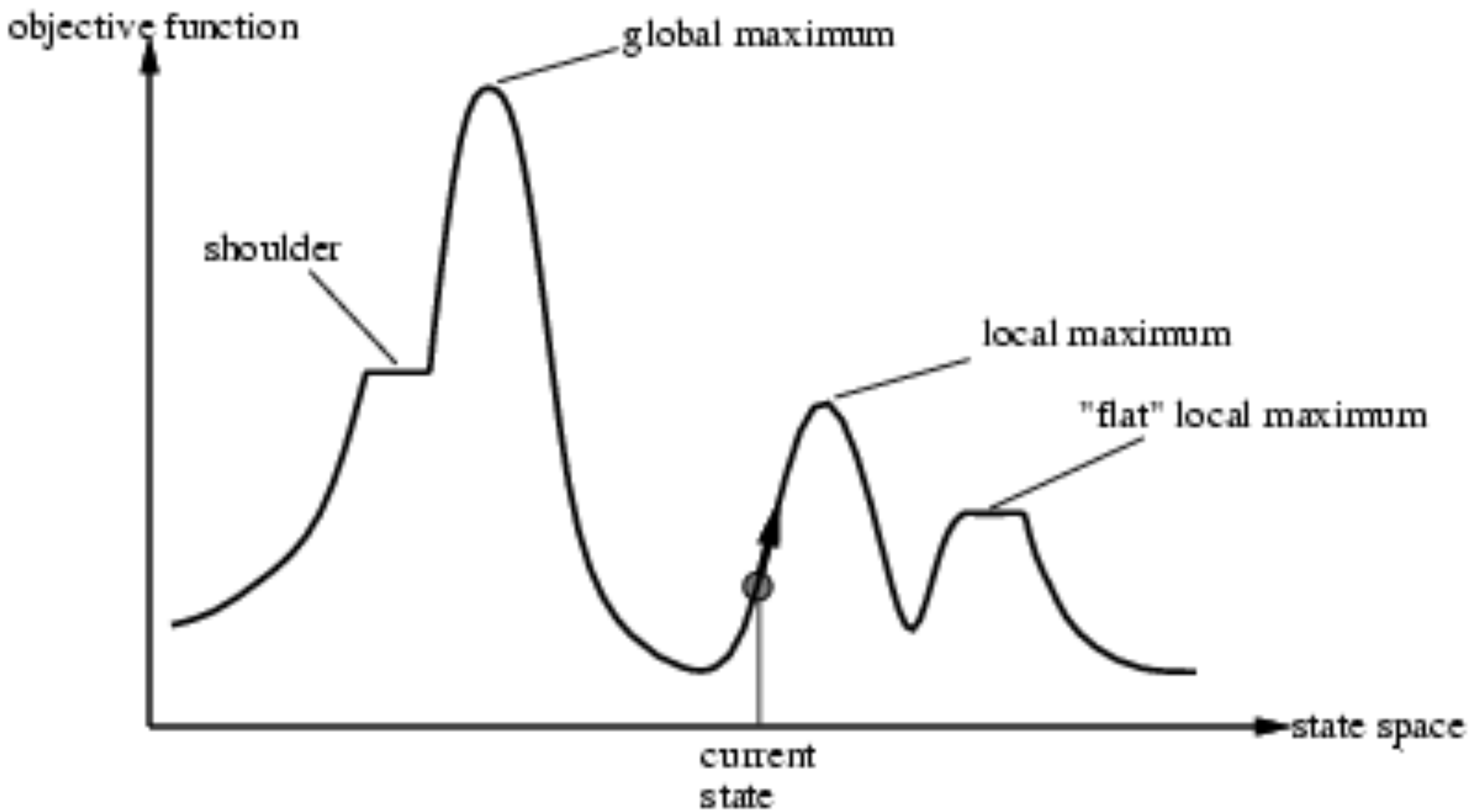
- Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal

Hill-Climbing

- "Like climbing Mount Everest in fog with amnesia"

- Set current state = initial state
- Loop
  - Pick the best neighboring state
  - If the neighboring state is better than the current state
    - Current state = neighboring state
  - Else
    - Current state is the best fit

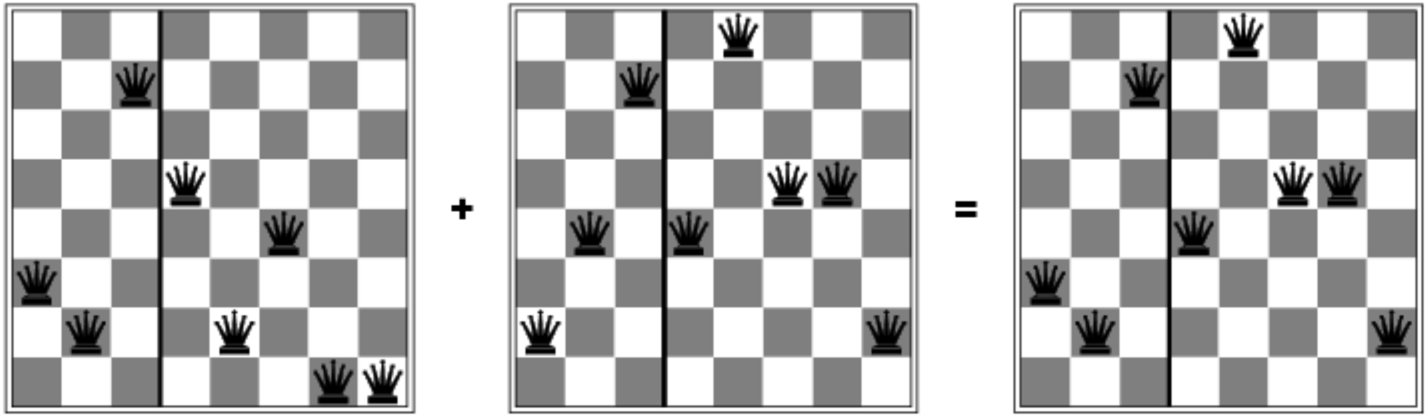
Hill-Climbing



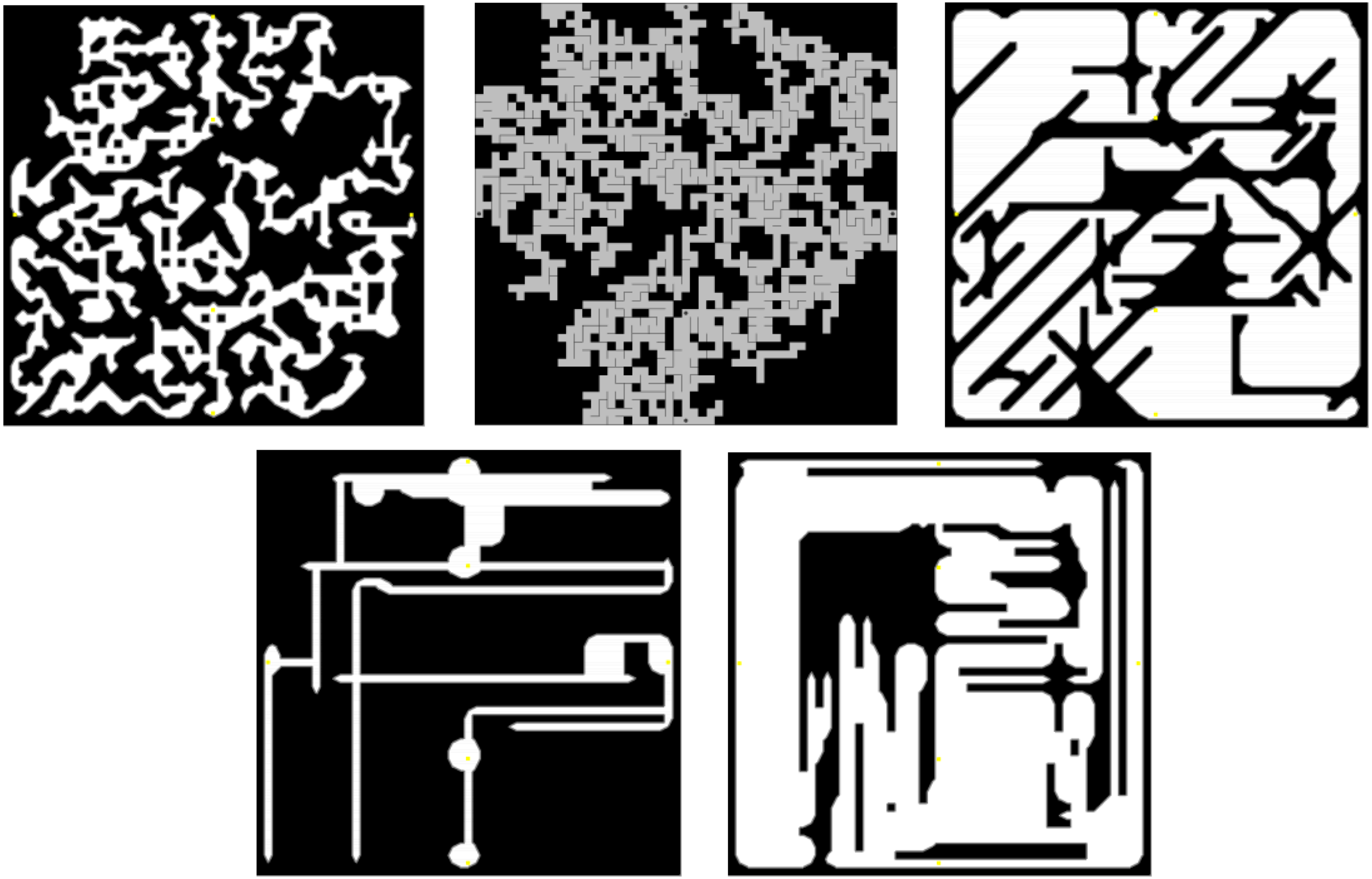
Genetic Algorithms

- Instead of keeping 1 state, keep a **population**

- Evolve population based on **fitness function**
  - Keep the best x% with some probability
  - Crossover the best y%
  - Mutate the remainder



Genetic Algorithms: Representation



Humans as Fitness Functions

[Picbreeder]

[Galactic Arms Race]

Evolving Virtual Creatures

Evolved Virtual Creatures Video

ADVERSARIAL SEARCH

Game Playing

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Game Playing

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Game Playing

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Game Playing

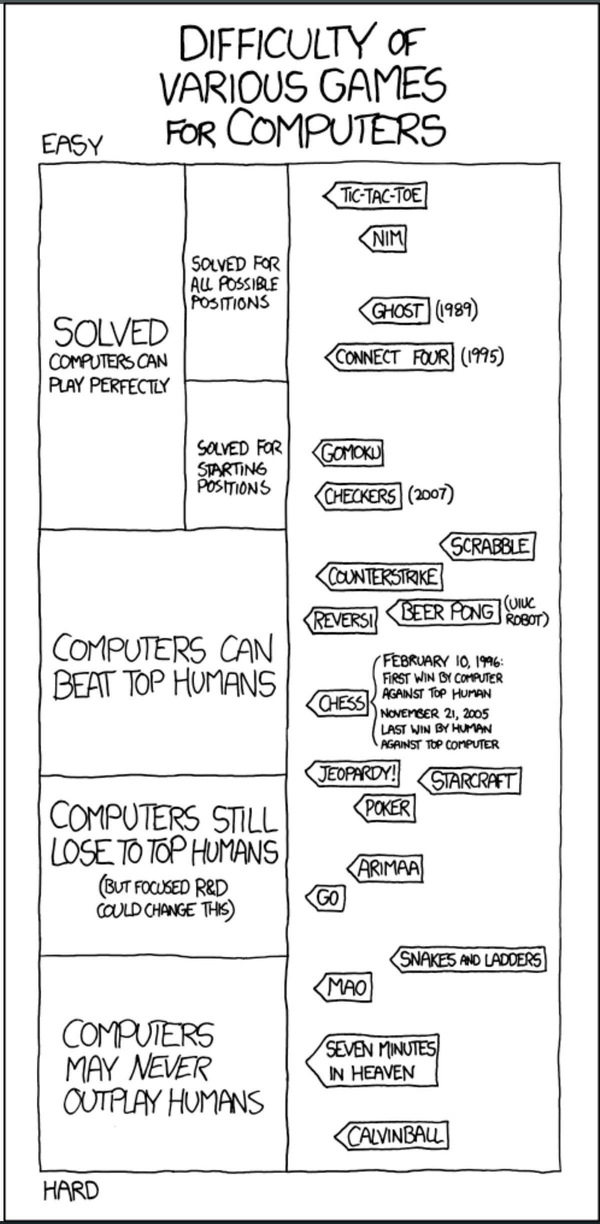
	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Game Playing

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

What are Games? (in the context of AI)

- Multi-agent environment
- Adversarial
- Planning a strategy
- High complexity + time limits

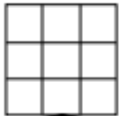


Minimax Algorithm

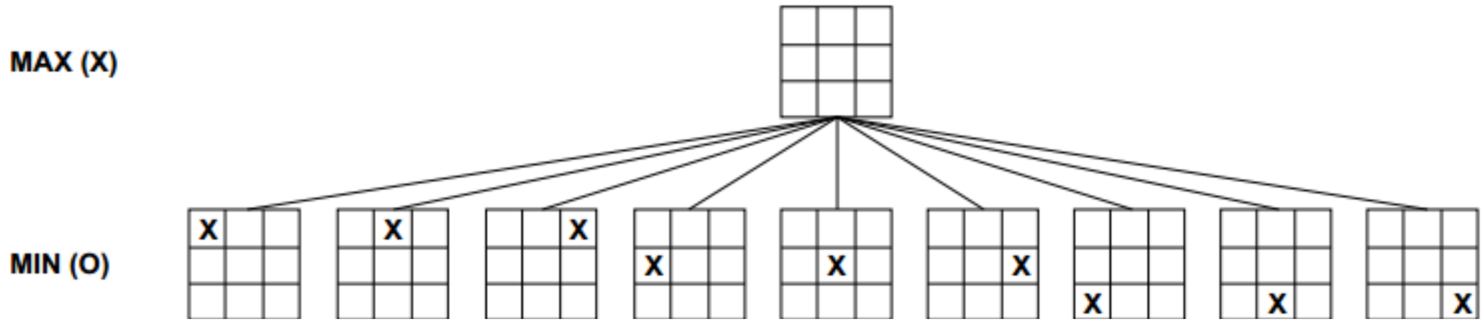
- Two player, adversarial, perfect information game
  - Tic Tac Toe
  - Nim
  - Chess
  - Go
- Two players: MAX and MIN
  - MAX moves first

Game Tree Setup

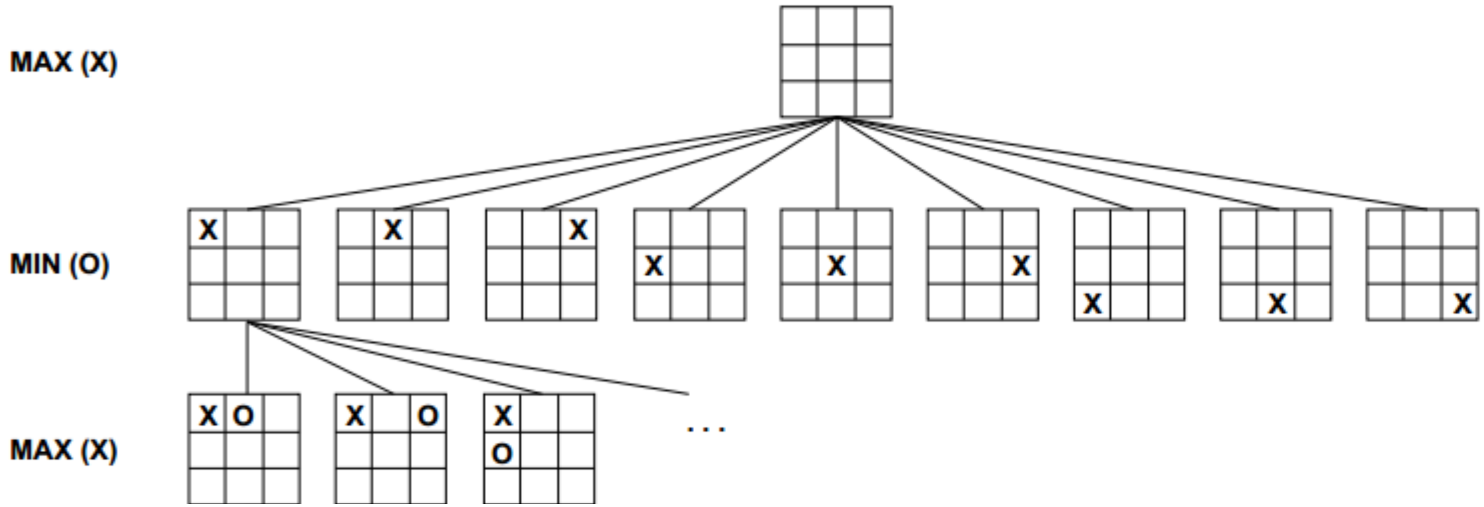
MAX (X)



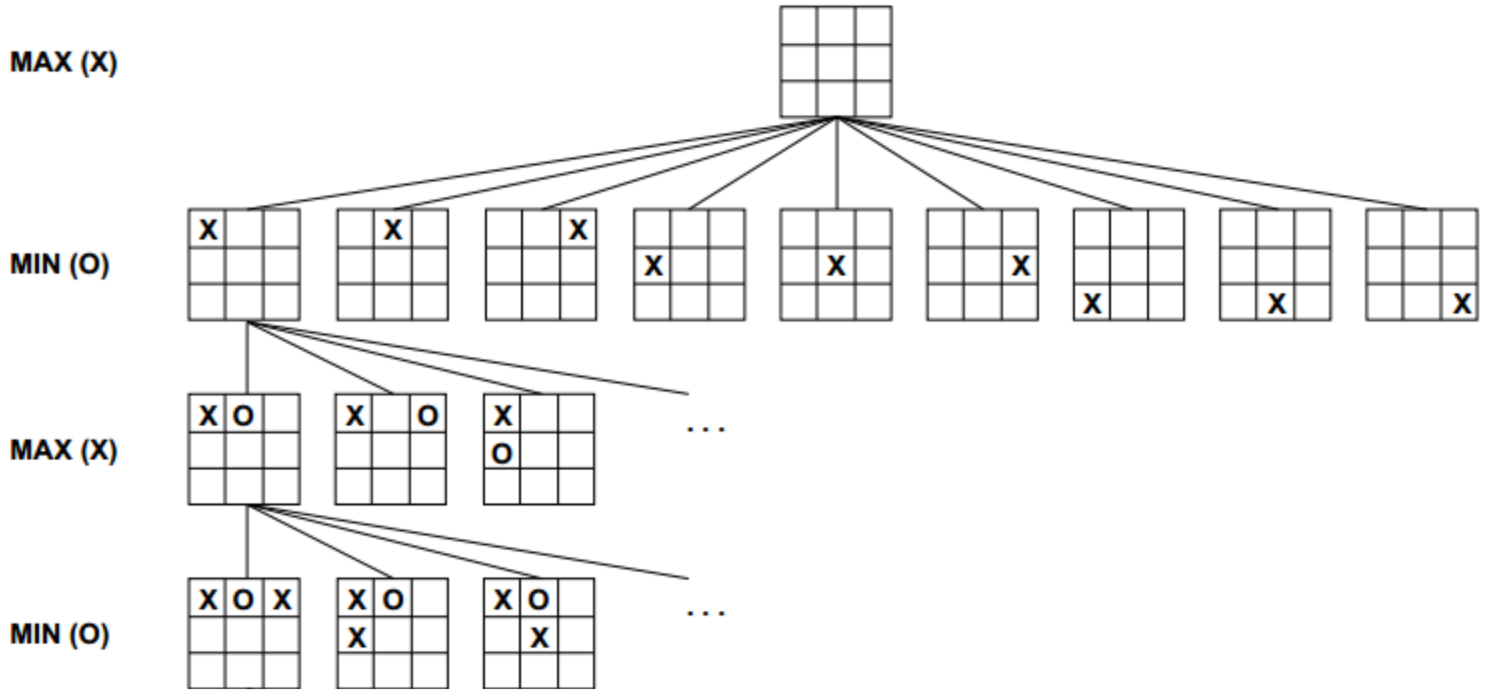
Game Tree Setup



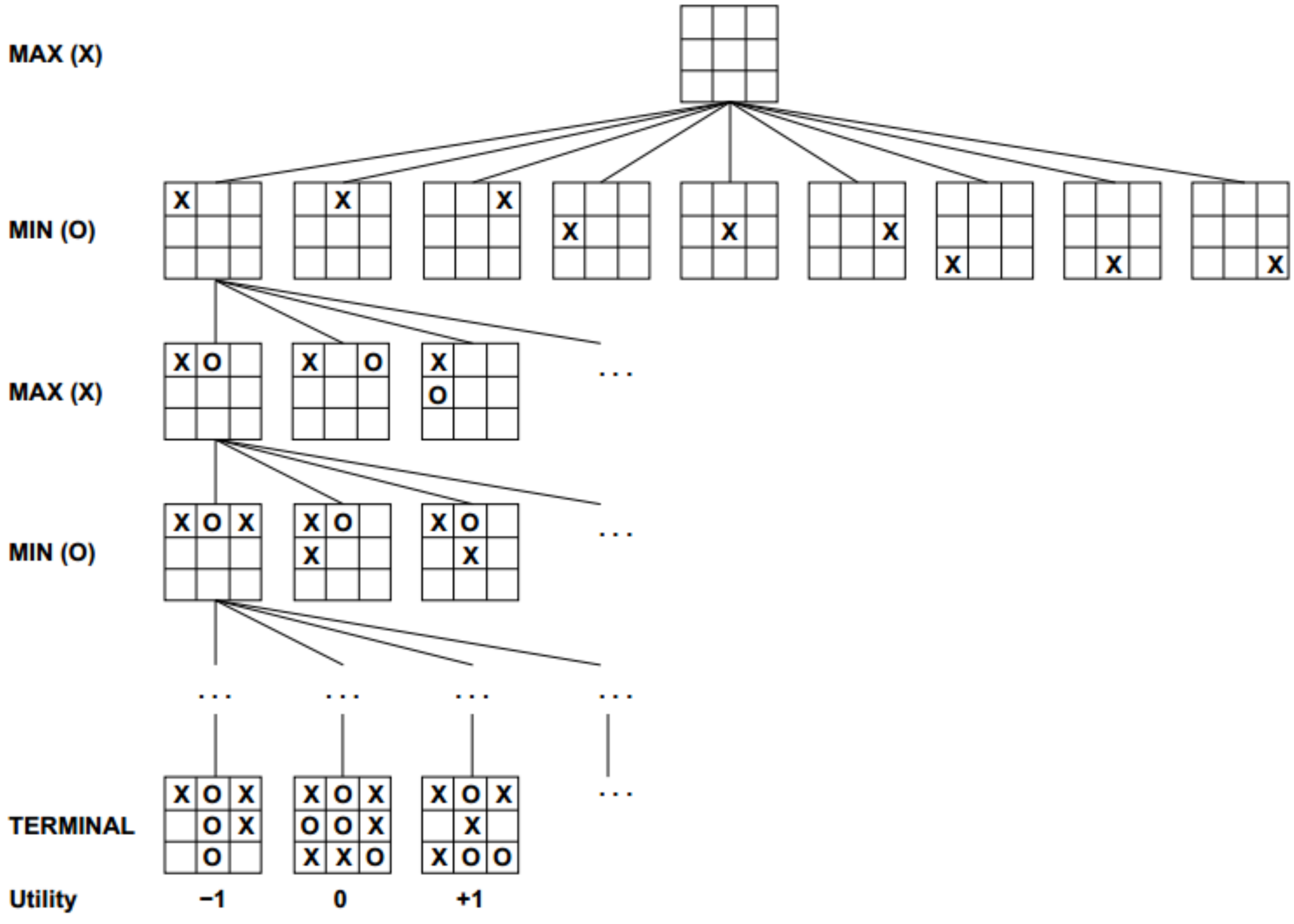
Game Tree Setup



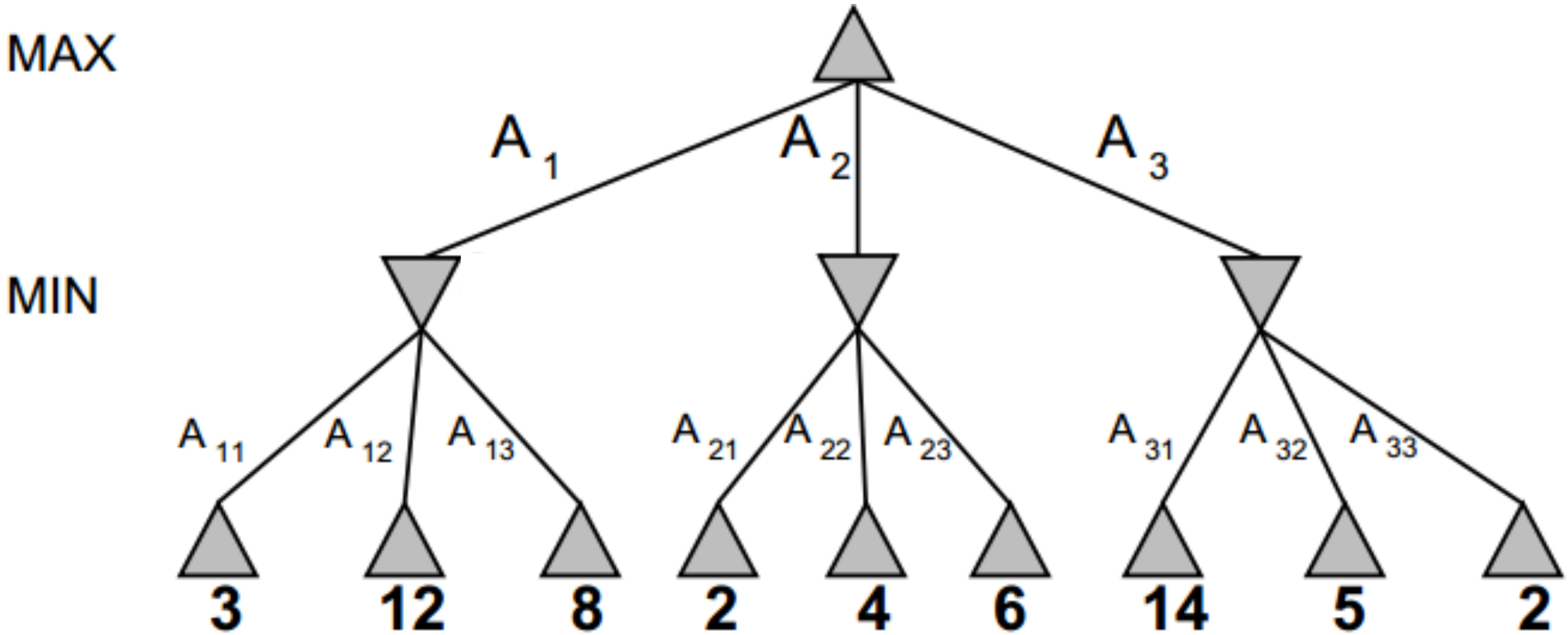
Game Tree Setup



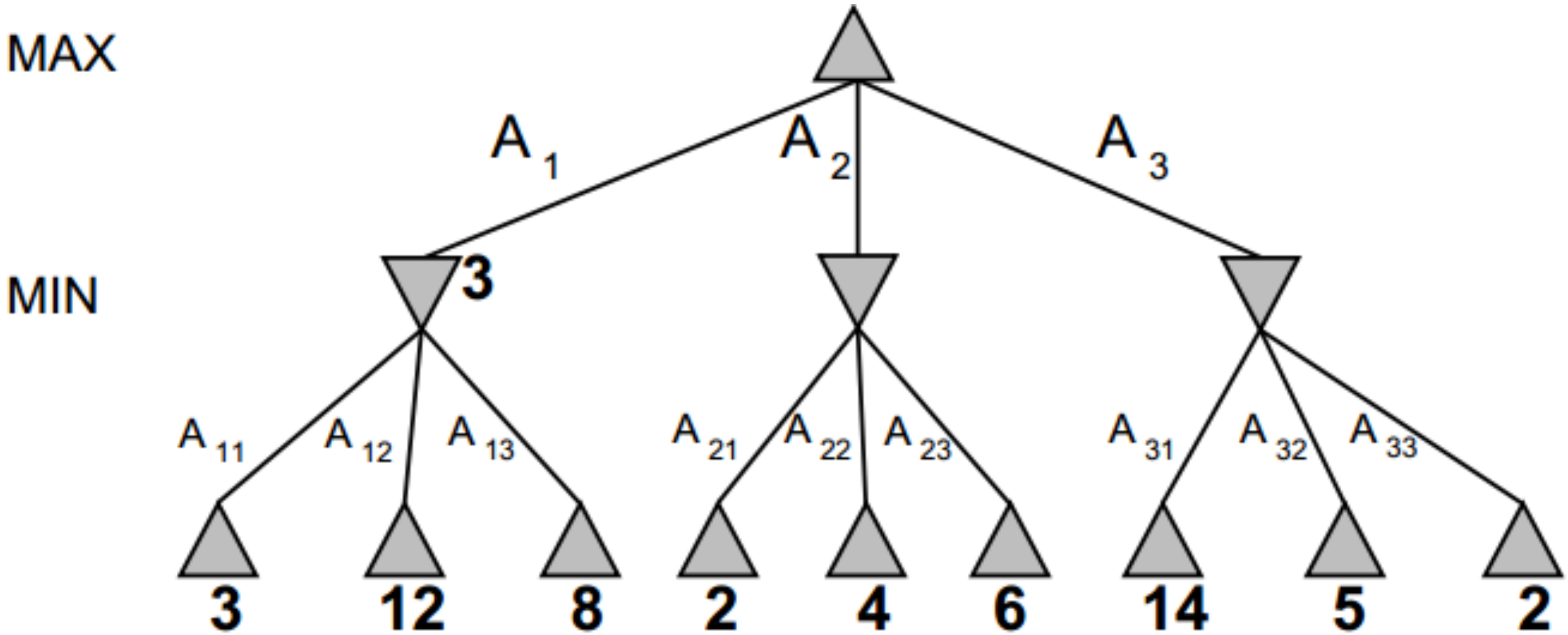
Game Tree Setup



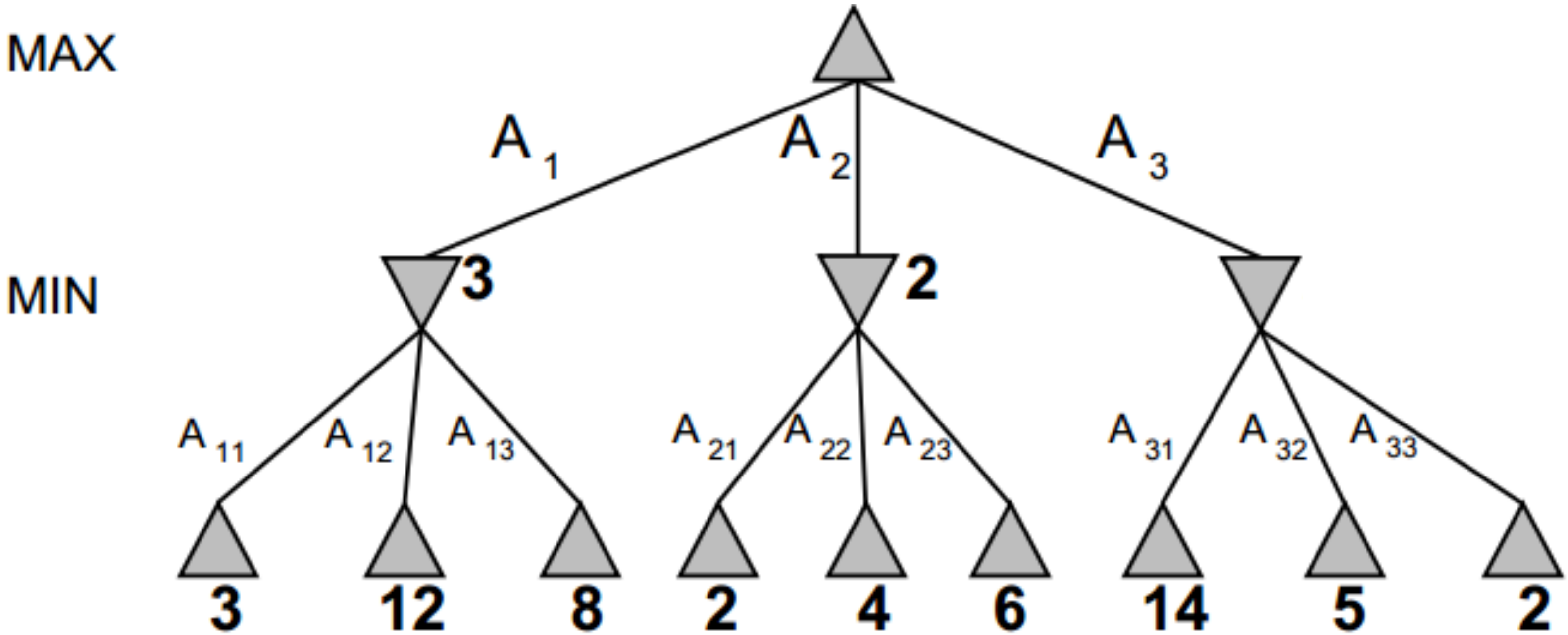
Minimax: Small Example



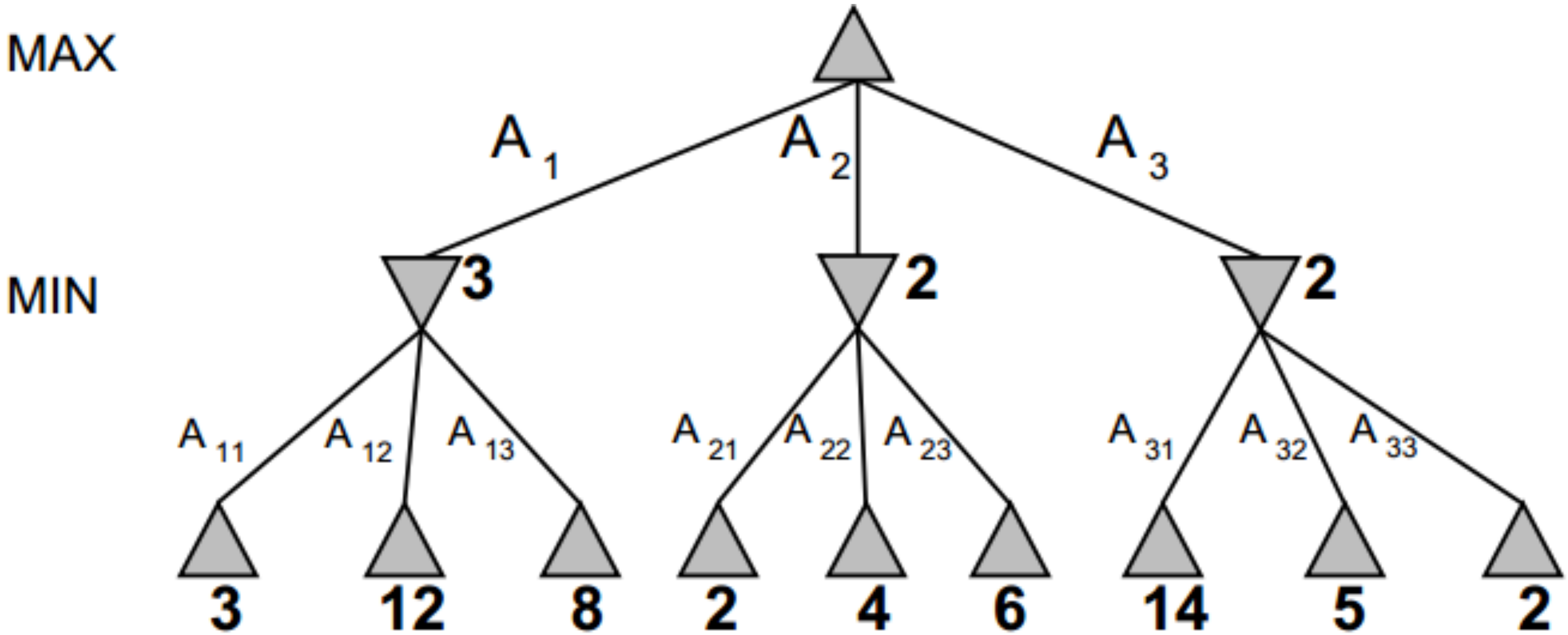
Minimax: Small Example



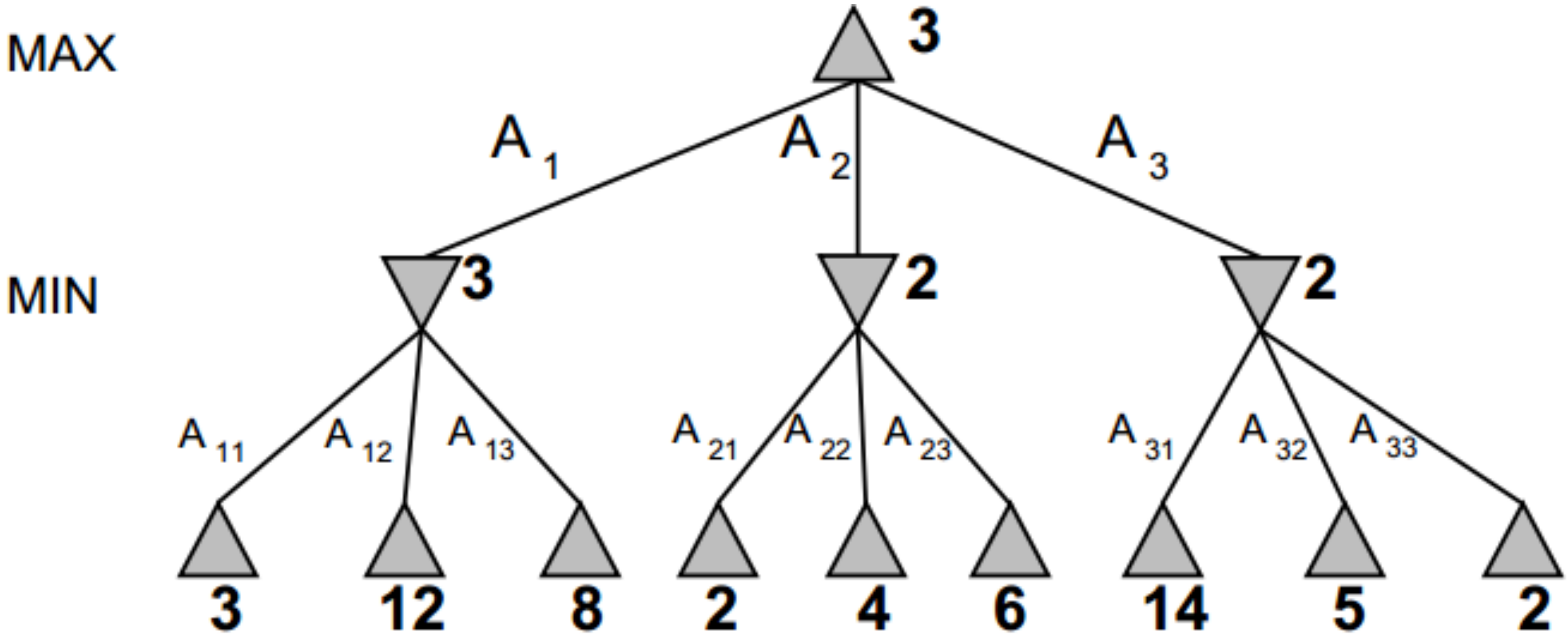
Minimax: Small Example



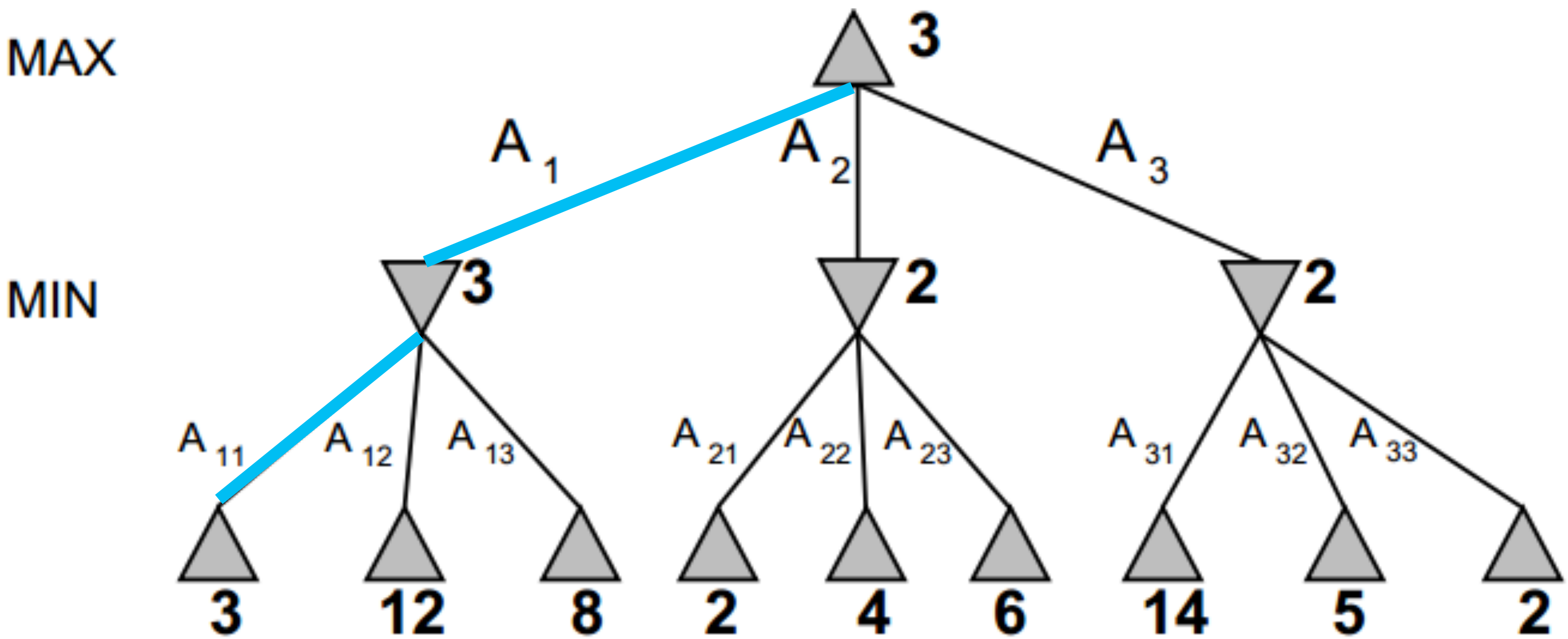
Minimax: Small Example



Minimax: Small Example



Minimax: Small Example



In-Class Exercise

- There are six spaces on a board that must be filled in order. Player 1 (MAX) uses ‘x’, and Player 2 (MIN) uses ‘o’. Each player can place at most 3 symbols per turn. Whoever places their symbol in the last slot on the board wins. Draw the minimax search tree.

1: X	4: X
2: O	5: X
3: O	**6: X**

In-Class Exercise

- There are six spaces on a board that must be filled in order. Player 1 (MAX) uses ‘x’, and Player 2 (MIN) uses ‘o’. Each player can place at most 3 symbols per turn. Whoever places their symbol in the last slot on the board wins. Draw the minimax search tree.

1: X	4:
2:	5:
3:	**6:**

In-Class Exercise

- There are six spaces on a board that must be filled in order. Player 1 (MAX) uses ‘x’, and Player 2 (MIN) uses ‘o’. Each player can place at most 3 symbols per turn. Whoever places their symbol in the last slot on the board wins. Draw the minimax search tree.

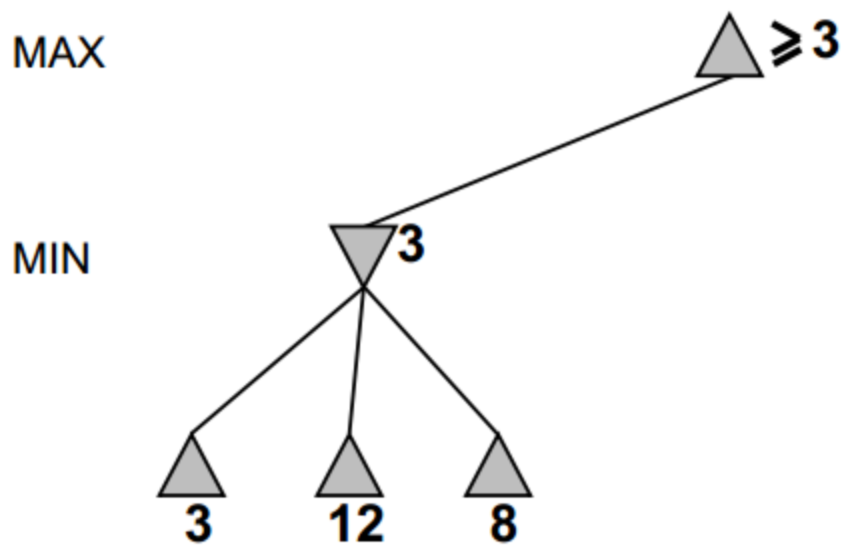
1: X	4:
2: O	5:
3: O	**6:**

In-Class Exercise

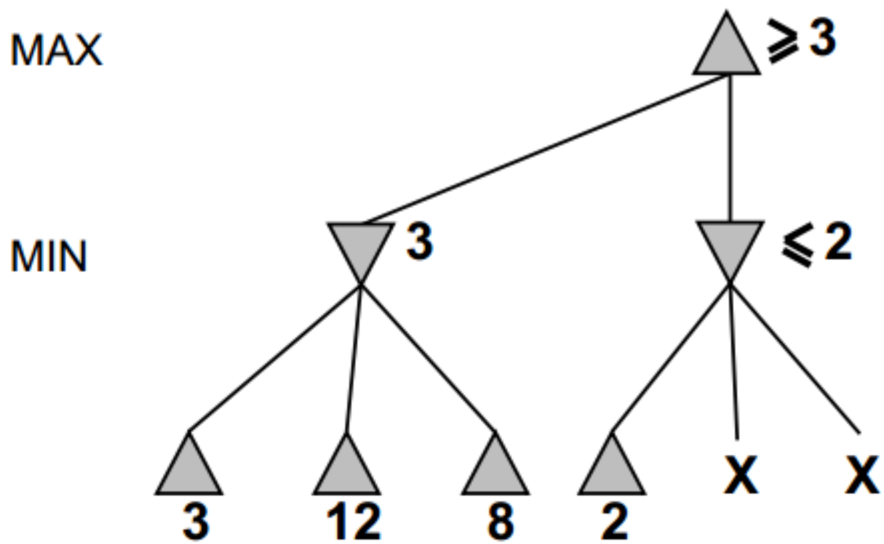
- There are six spaces on a board that must be filled in order. Player 1 (MAX) uses ‘x’, and Player 2 (MIN) uses ‘o’. Each player can place at most 3 symbols per turn. Whoever places their symbol in the last slot on the board wins. Draw the minimax search tree.

1: X	4: X
2: O	5: X
3: O	**6: X**

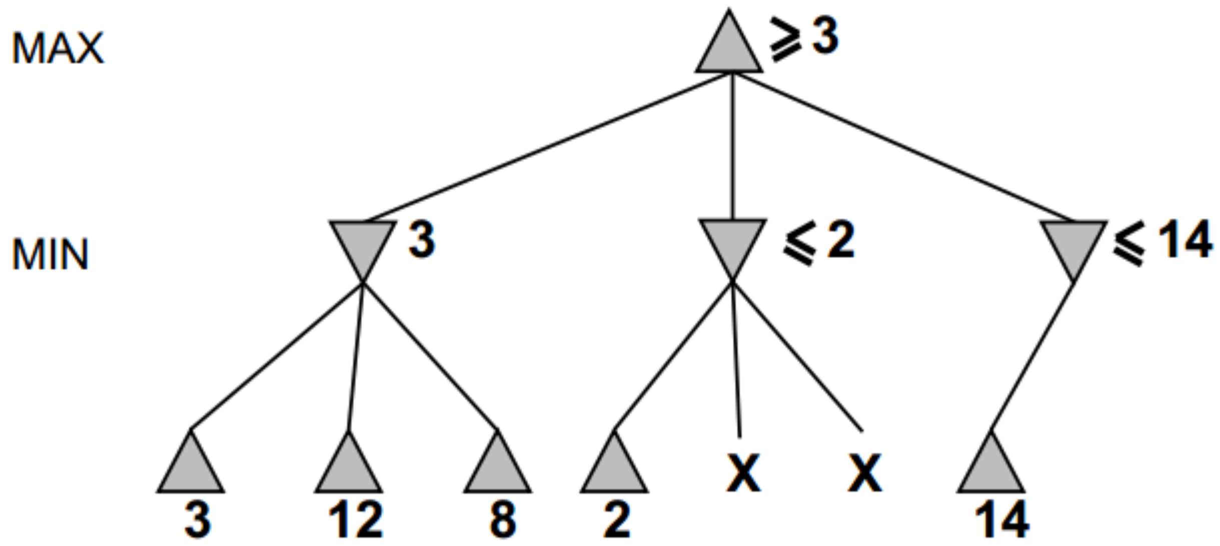
Alpha-Beta Pruning



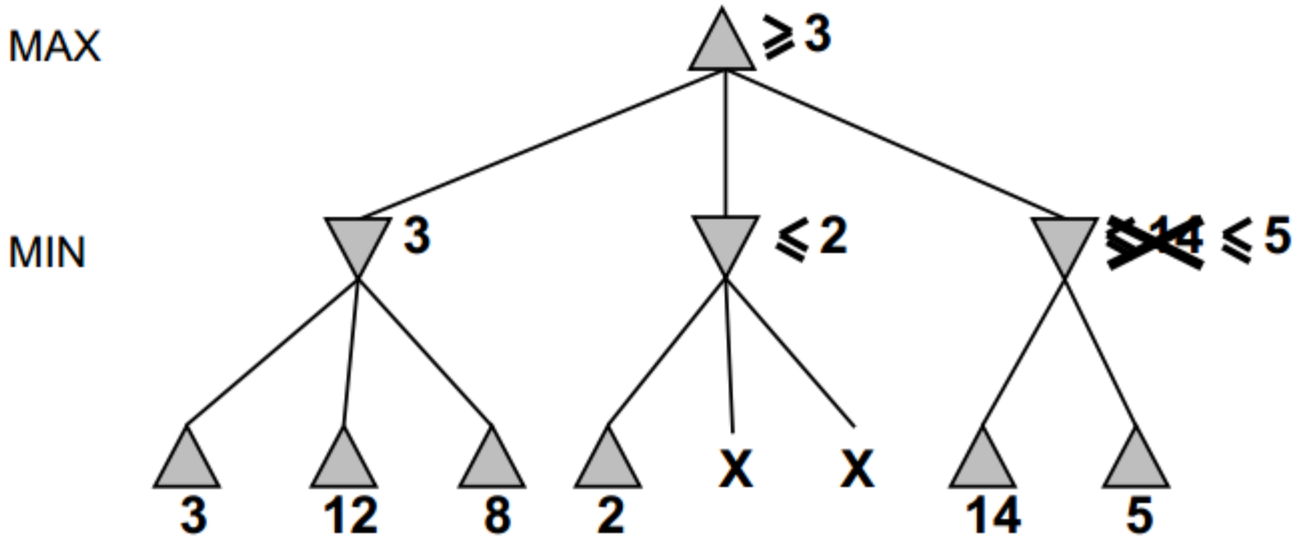
Alpha-Beta Pruning



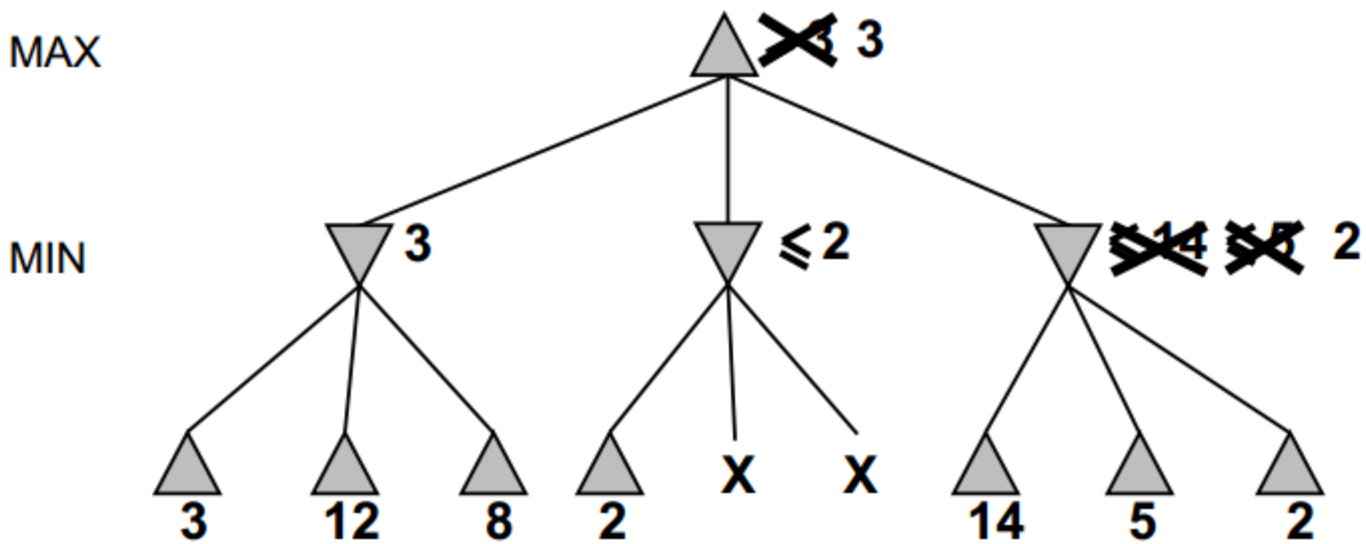
Alpha-Beta Pruning



Alpha-Beta Pruning



Alpha-Beta Pruning



What if the tree is too big?

- Heuristics!

- Quiescence search

MIDTERM REVIEW!

Grade Breakdown

- Overall mean: 84.42%
- Overall median: 82.5%

- Number of people who took late days: 5

- Average for sections:
  - What is AI? 90.57%
  - Propositional Logic 75.09%
  - First-Order Logic 85.63%
  - Constraint Solving 78.99%
  - Planning 78.46%

Regrading Policy

- If you think I made a mistake:
  - Write down your regrade request on the front of the exam
  - I reserve the right to **regrade the whole exam**
- Regrades are relatively low priority but will get done!

What is Plagiarism?

- Directly copying text or code from one source and claiming it as your own
- It can be accidental!
- It doesn't matter what the source is!
- Even copying a partial sentence counts!

Why is Plagiarism Bad?

- Ethical
  - Claiming someone else's work as your own is wrong
  - Damages your reputation (and your employer's...)

- Educational
  - Makes it unclear if you actually understand information or just have basic reading comprehension skills

Kinds of Plagiarism

- Copy-and-paste
- Close paraphrasing
- Failure to assign credit
- Self-plagiarism

Kinds of Plagiarism

- **Copy-and-paste**
- Close paraphrasing
- Failure to assign credit (citations)
- Self-plagiarism

Copy-and-Paste Plagiarism

- Source: “The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.” (AIMA, p48)

Copy-and-Paste Plagiarism

- Source: “The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.” (AIMA, p48)

- Bad: “These agents select actions on the basis of the current percept, ignoring the rest of the percept history.”

Copy-and-Paste Plagiarism

- Source: “The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.” (AIMA, p48)

- Also Bad: “Simple reflex agents select actions on the basis of current percept, ignoring rest of the percept history.”

Copy-and-Paste Plagiarism

- Source: “The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.” (AIMA, p48)

- Still bad: “Simple reflex agents select actions on the basis of the current percept.”

Kinds of Plagiarism

- Copy-and-paste
- **Close paraphrasing**
- Failure to assign credit (citations)
- Self-plagiarism

Close Paraphrasing

- Source: “The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.” (AIMA, p48)

Close Paraphrasing

- Source: "The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history." (AIMA, p48)

- Bad: "The simple reflex agent is the simplest kind of agent. It ignores percept history, and selects actions on the basis of the current percept."

Close Paraphrasing

- Source: “The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.” (AIMA, p48)

- Also Bad: “The simple reflex agent, the simplest kind of agent, chooses its actions based on what it currently knows, ignoring the rest of the percept history.”

Close Paraphrasing

- Source: “The **simplest kind of agent** is the simple reflex agent. These agents select actions on the *basis of the current percept*, **ignoring the rest of the percept history**.” (AIMA, p48)

- Also Bad: “The simple reflex agent, the **simplest kind of agent**, chooses its actions *based on what it currently knows*, **ignoring the rest of the percept history**.”

Close Paraphrasing

- Source: “The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.” (AIMA, p48)

- Okay: “A simple reflex agent works does not keep track of the history of its actions or what it has previously seen. It only knows how to act using simple rules that are triggered by the current percept. [AIMA, p48]”

Close Paraphrasing

- Source: “The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.” (AIMA, p48)

- Okay *but not good for an exam*: “According to Russell and Norvig, the simple reflex agent ‘select[s] actions on the basis of the correutn percept, ignoring the rest of the percept history’. [AIMA, p48]”

Kinds of Plagiarism

- Copy-and-paste
- Close paraphrasing
- **Failure to assign credit (citations)**
- Self-plagiarism

When and where do I cite something?

- If it is a direct quote – next to the quotation
- If it is a paraphrase – near the paraphrased paragraph
- Don't need to cite for common knowledge
  - The sky is blue
  - Boston is in Massachusetts
- If in doubt, cite it!

Common Knowledge?!

- Things that a reasonable person in your audience can be assumed to know
  - The sky is blue
  - Boston is in Massachusetts

- Never applies to data, statistics, or something you read somewhere that you didn't know before!
  - People who are not told that the sky is blue do not necessarily identify it as being blue [Radiolab, NPR, Season 10 Episode 13]
  - Boston's population is 589,141 as of the 2000 US Census, making it the largest city in Massachusetts. [US Census 2000]

What do I cite?

- Authors
- Name of paper/book/article
- Where that article was published
  - Newspaper?
  - Conference?
  - Website? (give url and date accessed)
- What date the article was published

Kinds of Plagiarism

- Copy-and-paste
- Close paraphrasing
- Failure to assign credit (citations)
- **Self-plagiarism**

Wait, how could I plagiarize *myself* ?!

- Reusing text that you've written in the past without attribution
- Submitting the same paper for two classes
  - Simultaneously
  - Re-use from prior class
- Academic dishonesty: less work for more credit without disclosure to instructor
- Academia: can't pad publication count without doing the extra work!

Plagiarism on Midterm Exam

- Frequent in short-answer questions
- Deducted half-credit where noticed
  - I could have brought them **all** to the Office of Student Conduct and Conflict Resolution
- There are no exceptions to this policy
- Please do not do this again!

Part 0: What is AI?

- Simple reflex agents vs. goal-based agents

- Frame problem + example

Part 1: Propositional Logic

- Converting to conjunctive normal form

Two sentences are logically equivalent iff true in same models:

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \land \beta) &\equiv (\beta \land \alpha) && \text{commutativity of } \land \\
(\alpha \lor \beta) &\equiv (\beta \lor \alpha) && \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) &\equiv (\alpha \land (\beta \land \gamma)) && \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) &\equiv (\alpha \lor (\beta \lor \gamma)) && \text{associativity of } \lor \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \lor \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \land \beta) &\equiv (\neg\alpha \lor \neg\beta) && \text{de Morgan} \\
\neg(\alpha \lor \beta) &\equiv (\neg\alpha \land \neg\beta) && \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) &\equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) && \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) &\equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) && \text{distributivity of } \lor \text{ over } \land
\end{aligned}
$$

Part 1: Does Jason Knit?

- Intent: use a combination of resolution and proof by contradiction
- Also received full credit for two proofs, one using resolution and one using contradiction

Part 2: First-Order Logic

- Literals are single predicates (that can be negated) that do not have any variables
- Unification
- Order of interpretation in prolog matters!

Part 3: Constraint Solving

- Most constrained variable: prunes search tree later by choosing the variable you're most likely to find conflicts with
- Least constraining value: stay flexible by ruling out the fewest remaining values
- AnsProlog!

Part 4: Planning

- Planning Richard’s Epic Babylon 5 Movie Party

- What’s the difference between a state and an operator?

Extra Credit

- Situated actions
  - Actions arise from circumstance
  - Plans are constructed after the fact, to explain what we've done
  - Handling situated behavior within plans (e.g. where does the plan stop and situated action begin?)

- Needed to give an example

Up Next: Assignment 4

- Evolutionary algorithms
- Using Processing
  - (optionally)
- Give yourself plenty of time! Relatively simple coding, long time waiting for solutions.

Mona Lisa, after ~162k generations

Genetic Algorithms

- Instead of keeping 1 state, keep a **population**
- Evolve population based on **fitness function**
  - Keep the best x% with some probability
  - Crossover the best y%
  - Mutate the remainder