Search, Part 1

Lecture 4

Jim Rehg

January 20, 2016

College of Computing
Georgia Tech

Problem Types

Assumptions for now:

- Observable state
- Static environment
- Discrete states/actions
- Deterministic actions

Goals for Today

Examples of search problem formulations

General tree search

Introduction to uninformed search

- Breadth First Search
- Uniform Cost Search
- Depth First Search

Problem Formulation for Search

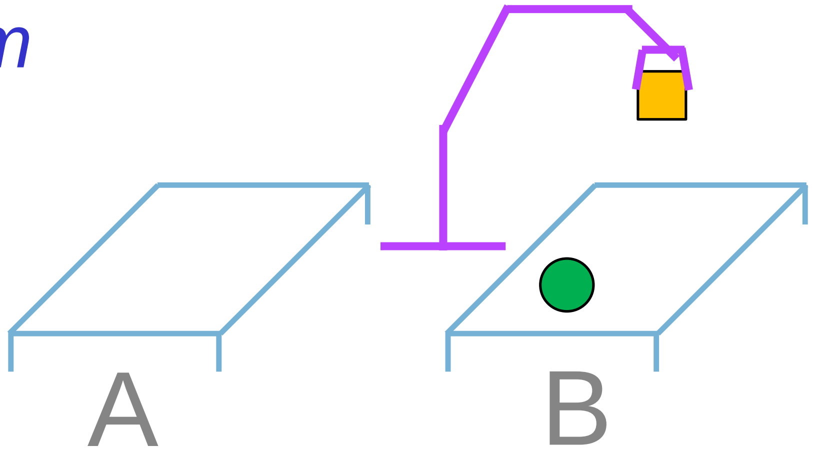
Initial State

Transition Model

Goal Test

Path Cost

Robot Arm Example



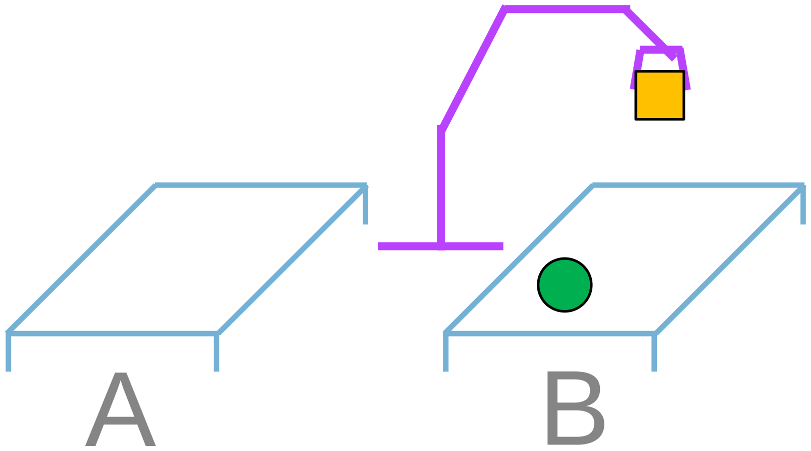
Problem Formulation for Search

Initial State

Transition Model

Goal Test

Path Cost



s = <Square-H, Ball-B, Hand-B>

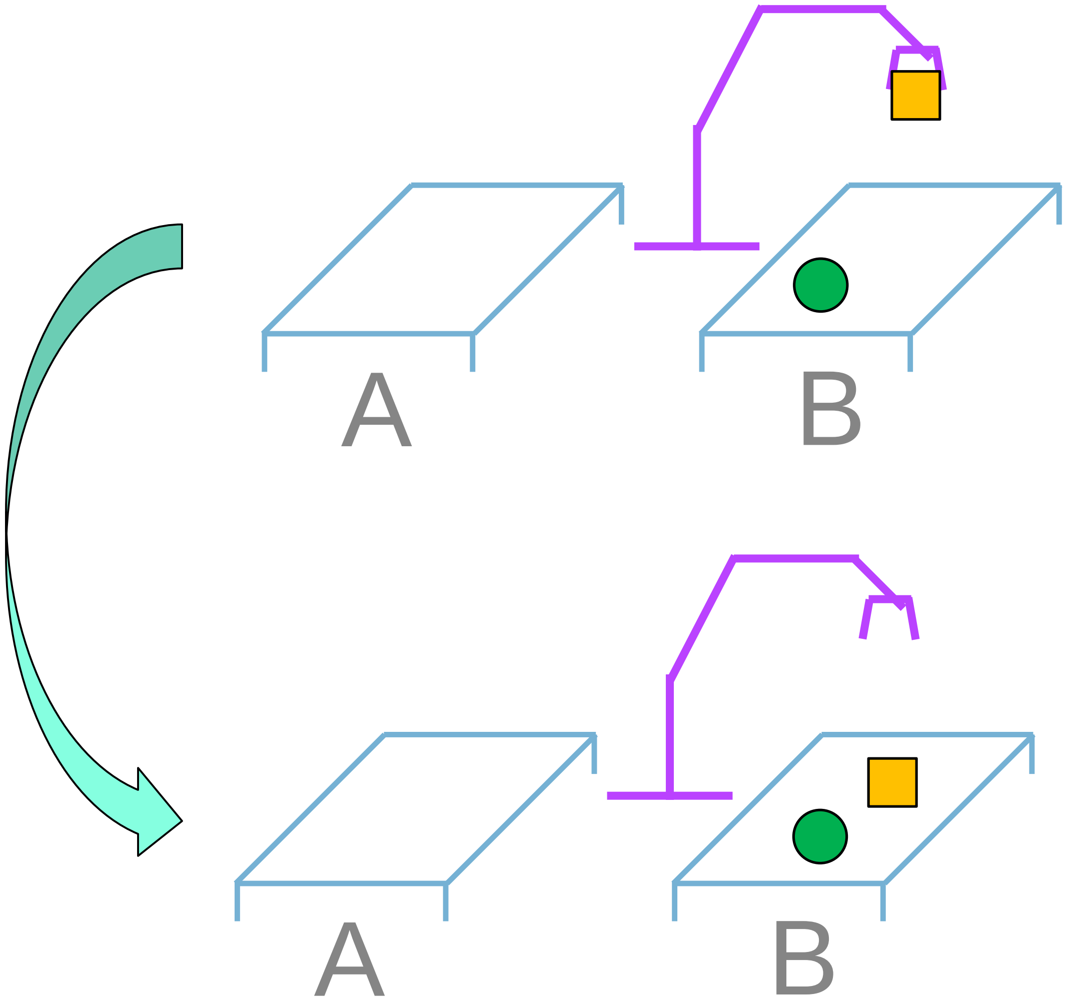
Problem Formulation for Search

Initial State

Transition Model

Goal Test

Path Cost



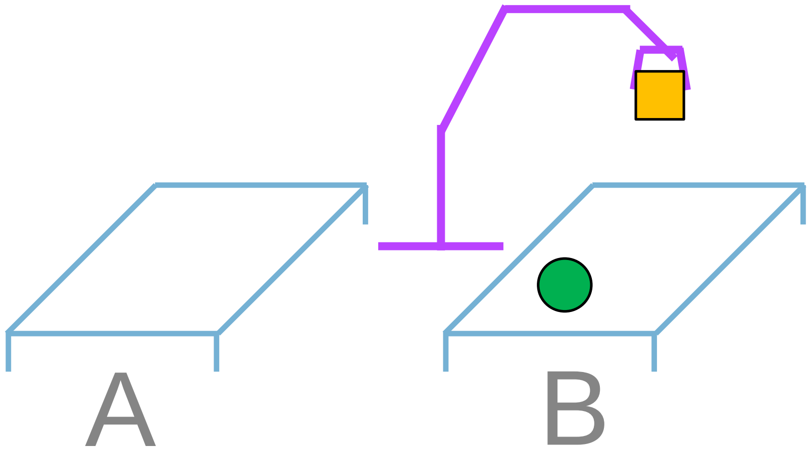
Problem Formulation for Search

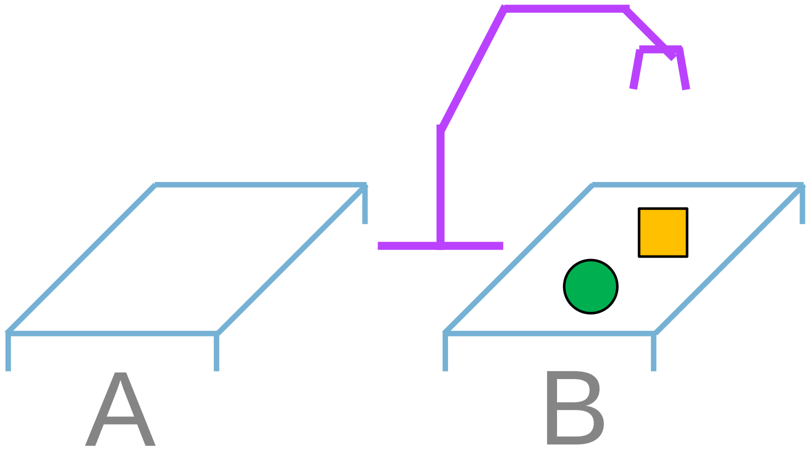
Initial State

Transition Model

Goal Test

Path Cost





goal = <Square-B, Ball-B, Hand-B>

Success!

Problem Formulation for Search

Initial State

Transition Model

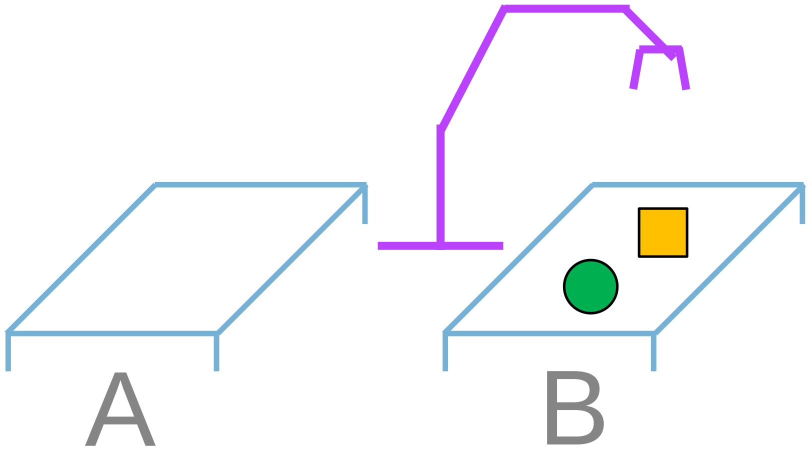
Goal Test

Path Cost = Sum of cost of each action in the sequence

e.g. Number of moves, total energy, total time, etc.

goal = <Square-B, Ball-B, Hand-B>

Success!



Problem Formulation for Search

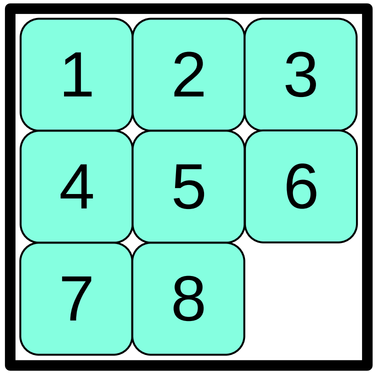
Initial State

Transition Model

Goal Test

Path Cost

Eight Puzzle



1	2	3
4	5	6
7	8	

What are the states?
What are the actions?

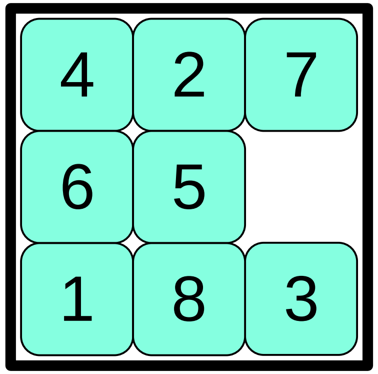
Problem Formulation for Search

Initial State

Transition Model

Goal Test

Path Cost



4	2	7
6	5	
1	8	3

Initial State

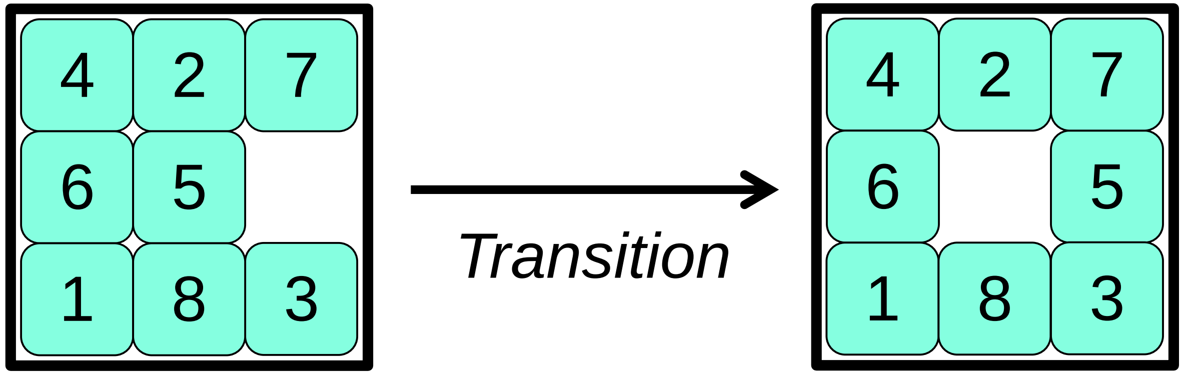
Problem Formulation for Search

Initial State

Transition Model

Goal Test

Path Cost



Initial State

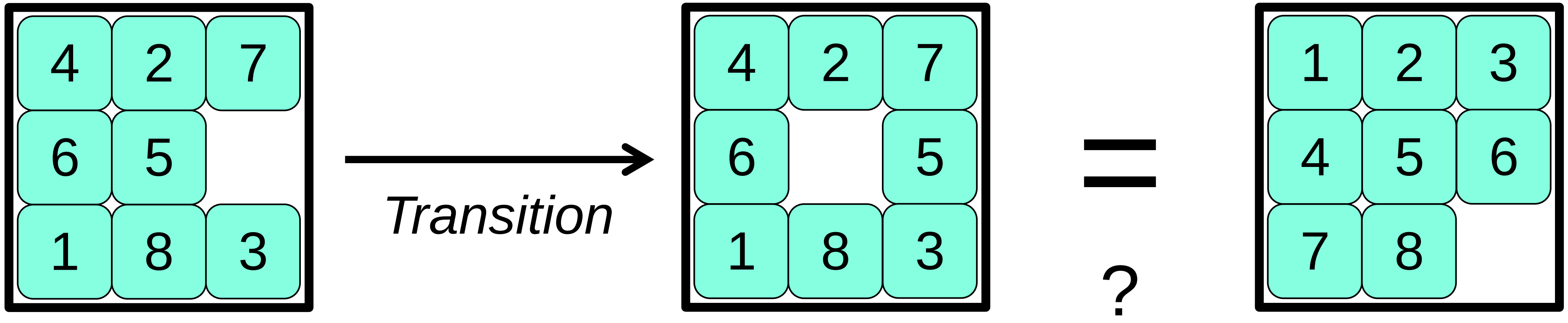
Problem Formulation for Search

Initial State

Transition Model

Goal Test

Path Cost



Initial State

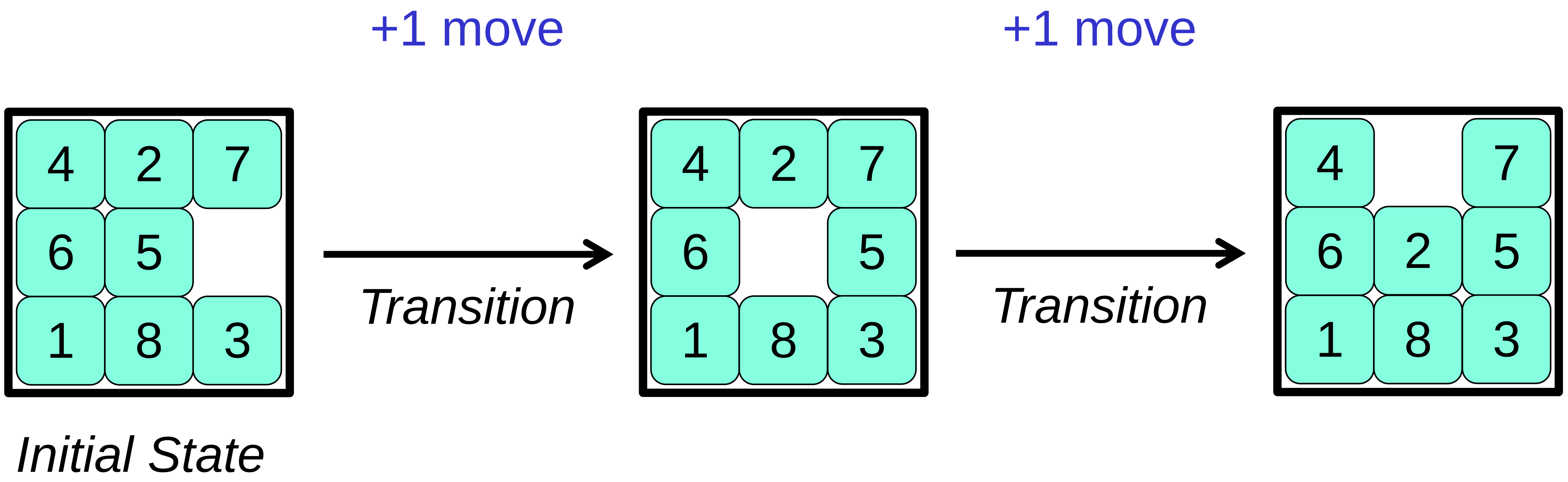
Problem Formulation for Search

Initial State

Transition Model

Goal Test

Path Cost



Goals for Today

Examples of search problem formulations

General tree search

Introduction to uninformed search

- Breadth First Search
- Uniform Cost Search
- Depth First Search

Generalized Tree Search

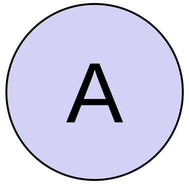
Initial State = Root Node

Transition Model = Node Expansion

Goal Test = Node Test

Path Cost = Path Cost in Tree

Initial State



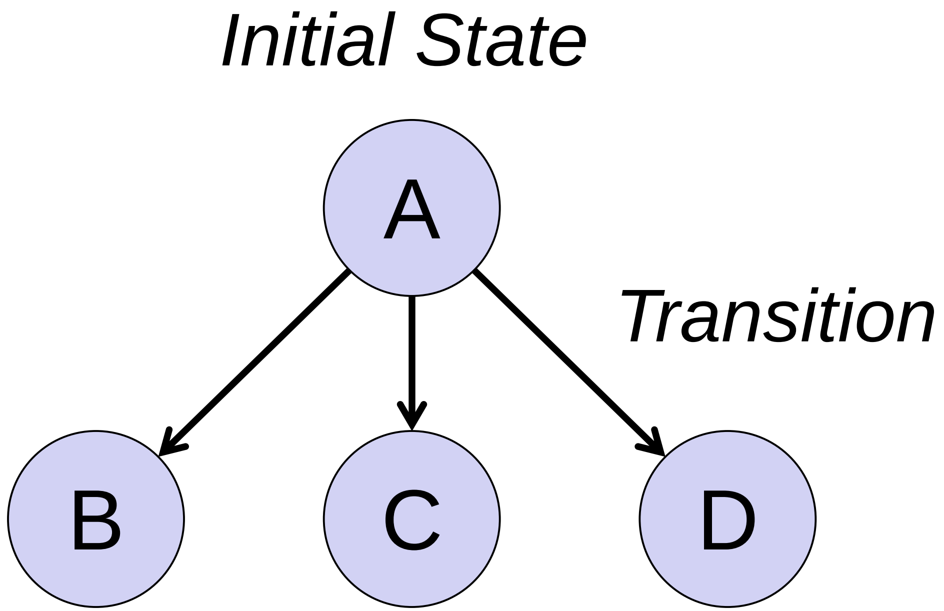
Generalized Tree Search

Initial State = Root Node

Transition Model = Node Expansion

Goal Test = Node Test

Path Cost = Path Cost in Tree



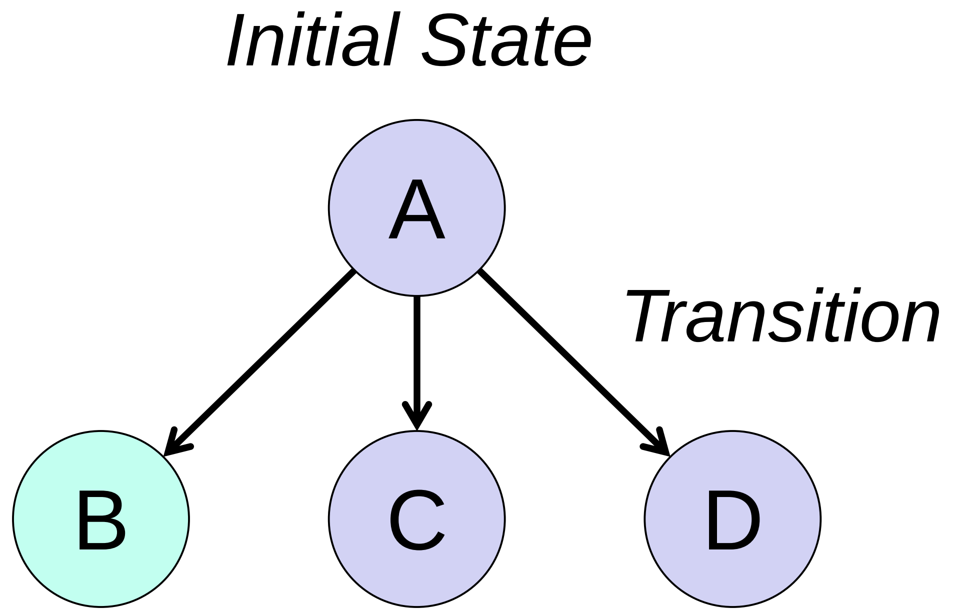
Generalized Tree Search

Initial State = Root Node

Transition Model = Node Expansion

Goal Test = Node Test

Path Cost = Path Cost in Tree



Goal = B ?

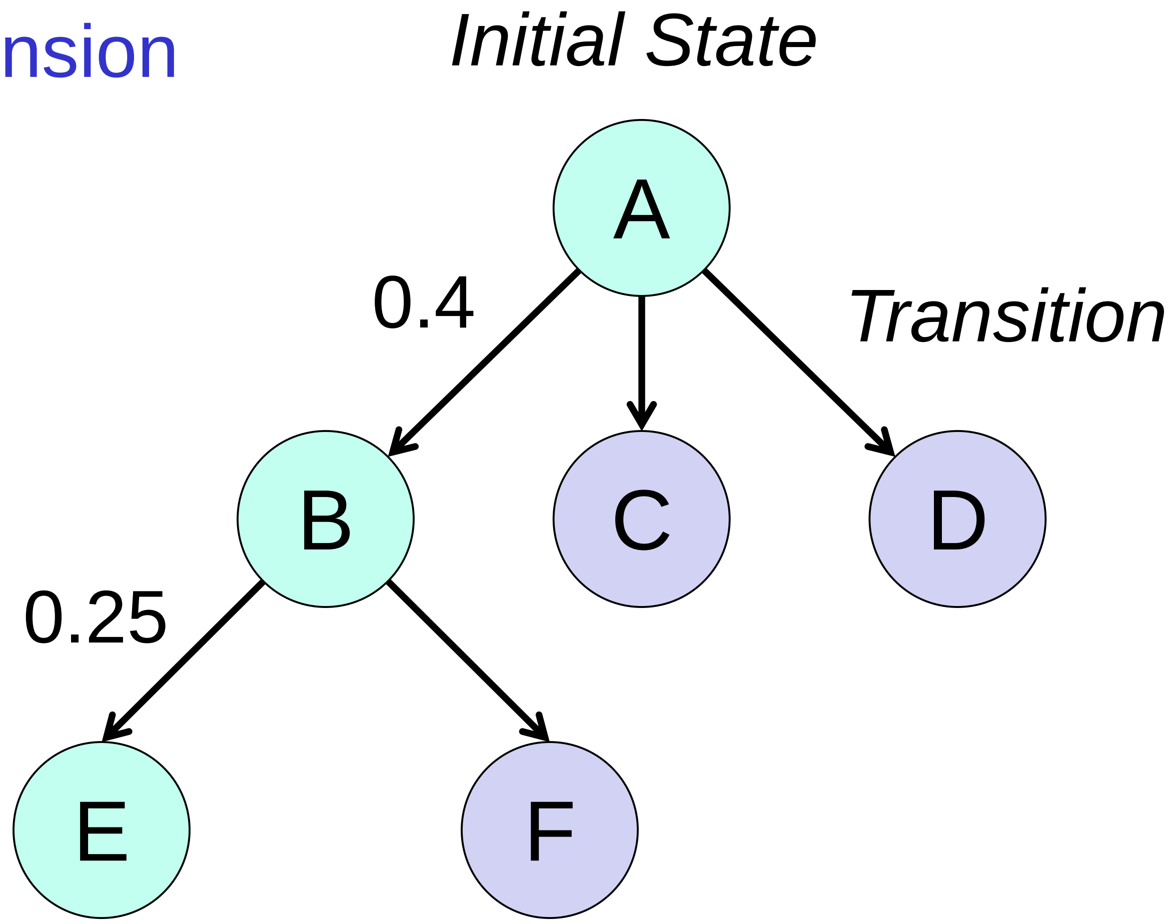
Generalized Tree Search

Initial State = Root Node

Transition Model = Node Expansion

Goal Test = Node Test

Path Cost = Path Cost in Tree



Path Cost = 0.65

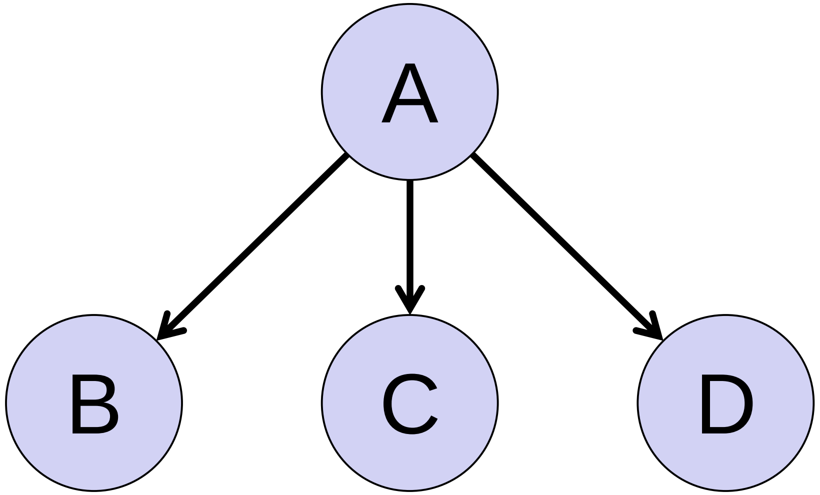
Generalized Tree Search

Initial State = Root Node

Transition Model = Node Expansion

Goal Test = Node Test

Path Cost = Path Cost in Tree



What node to expand next?

Node Representation in Tree Search

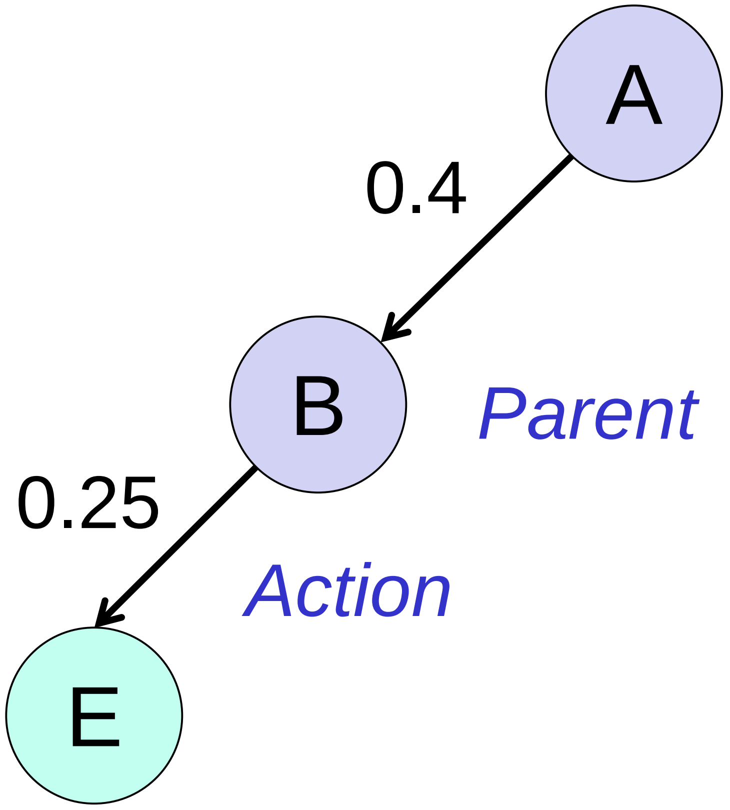
State this node represents

Parent node that generated this node

Action that generated this node

Cost of path from root to this node

Depth of path from root to this node



State = E
Depth = 2
Cost = 0.65

General Tree Search Algorithm

```
TreeSearch(problem) returns solution
  frontier = {problem.inital_state}
  loop:
     If empty(frontier) return failure
     Select leaf node, remove from frontier
     If node.contains(problem.goal)
          return node.solution
     node.expand(), add children to frontier
```

Frontier implemented as queue, how nodes get added is important

Evaluating Search Algorithms

Completeness = Guaranteed to find a solution if one exists

Optimality = Guaranteed to find the solution with lowest path cost

Time Complexity = Number of nodes generated

Space Complexity = Size of tree stored in memory

Goals for Today

Examples of search problem formulations

General tree search

Introduction to uninformed search

- Breadth First Search
- Uniform Cost Search
- Depth First Search

Uninformed Search

Breadth-first search

Uniform-cost search

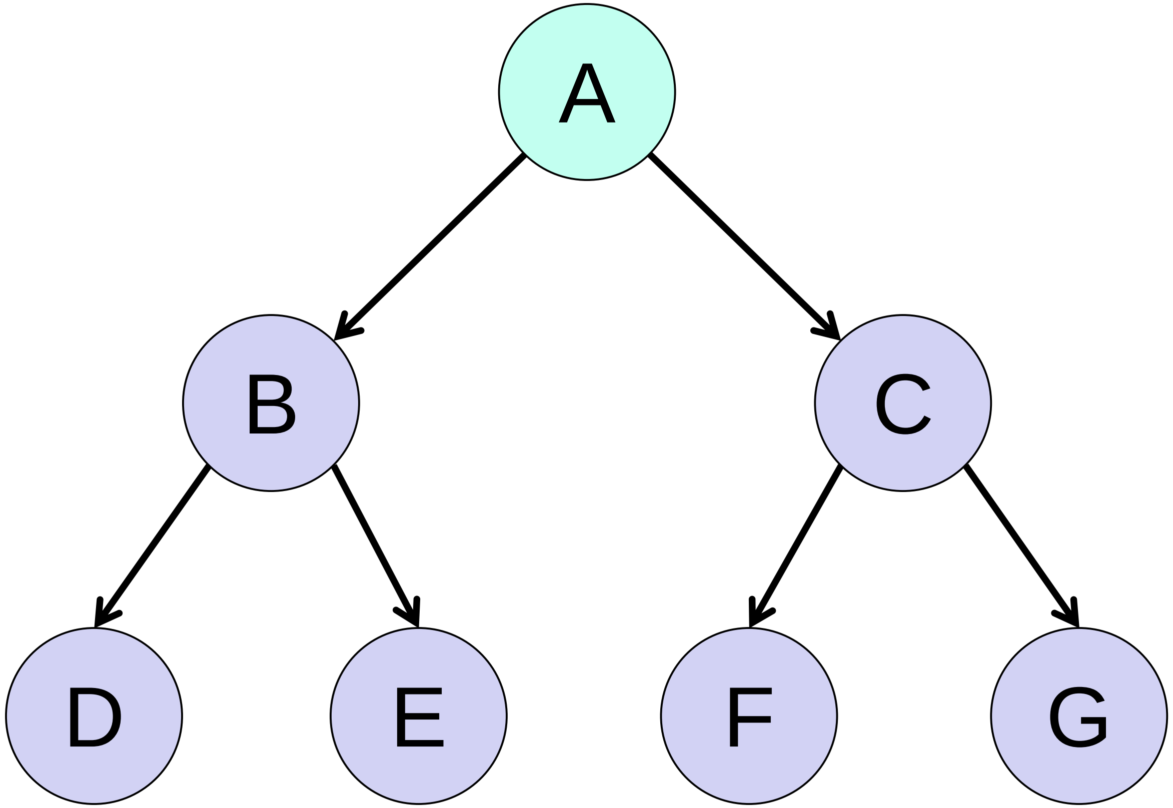
Depth-first search

Depth-limited search

Iterative-deepening search

Uninformed = Search method uses only information from problem

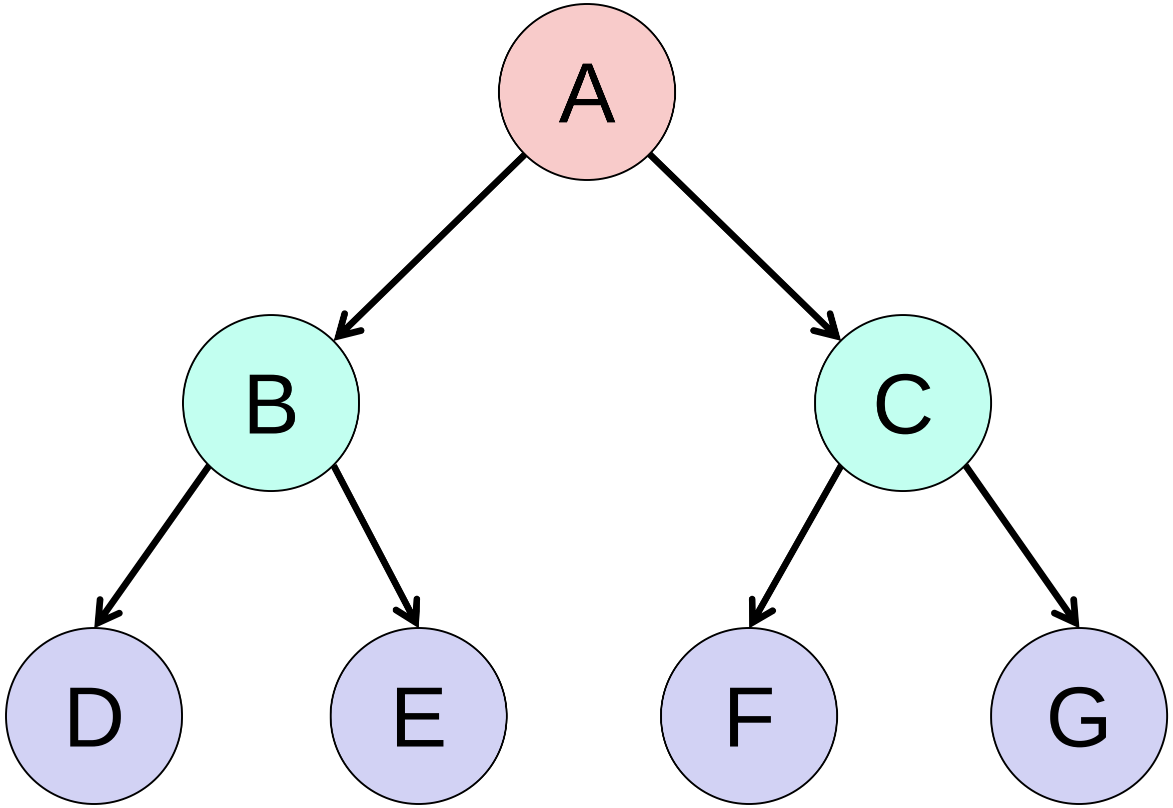
Breadth-First Search



Frontier (Queue)

1) A

Breadth-First Search

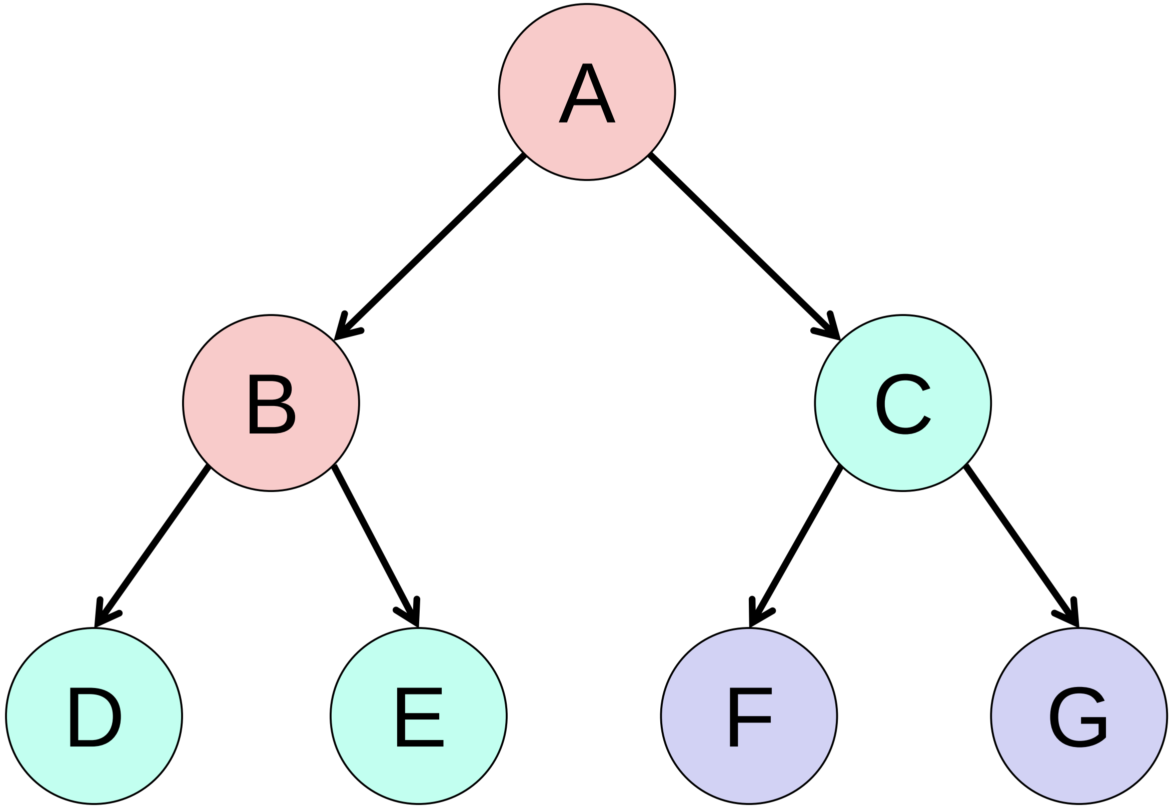


Frontier (Queue)

1) ~~A~~

2) B C

Breadth-First Search



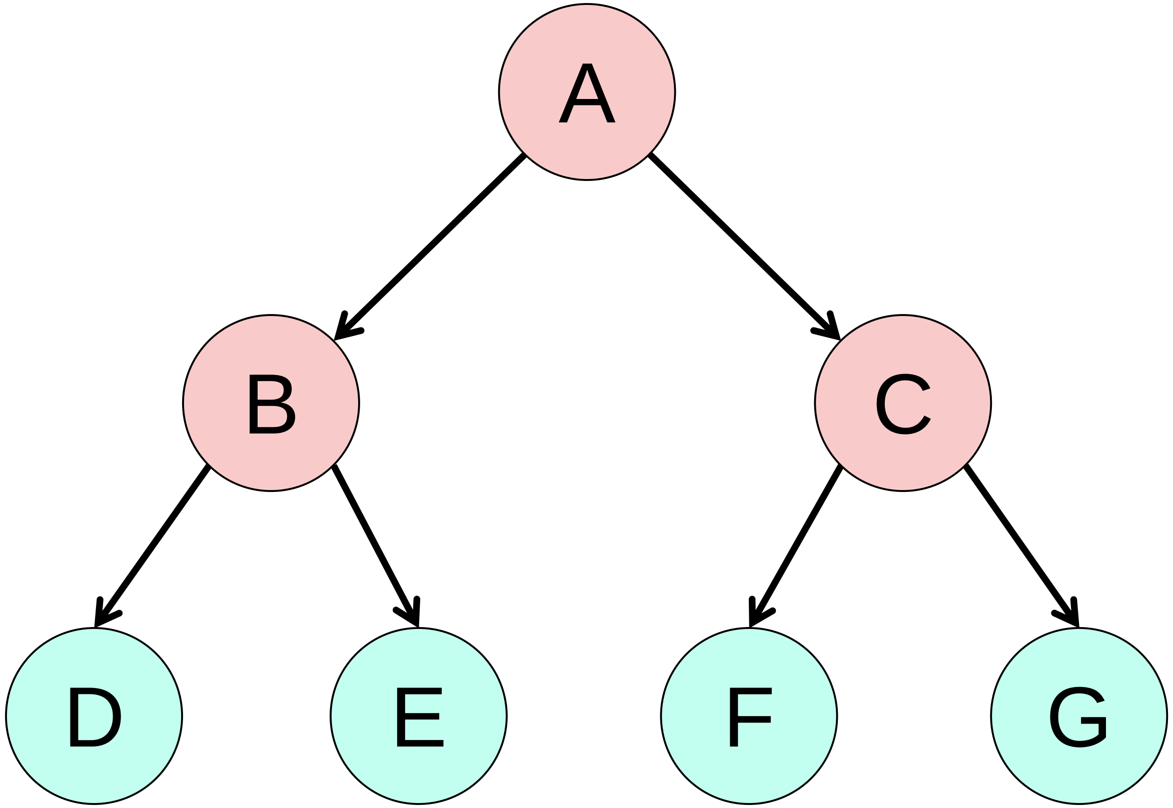
Frontier (Queue)

1) ~~A~~

2) ~~B~~ C

3) C D E

Breadth-First Search



Frontier (Queue)

1) ~~A~~

2) ~~B~~ C

3) ~~C~~ D E

4) D E F G

Analysis of Breadth-First Search

Implementation: FIFO queue

Complete? Yes – will find shallowest goal node

Optimal? Yes if all actions have the same cost, then shallowest node will have lowest path cost

Time Complexity

b = branching factor (2 in our example)

$b + b^2 + b^3 + \ldots = O(b^{d+1})$ for goal at depth d

Space Complexity

$O(b^d)$ since entire frontier kept in memory

Complexity Analysis

$O(b^{d+1})$ → time and space scale *exponentially* with the problem size d (the depth of the shallowest solution)

How bad is that?

If b = 10 (every parent has 10 children); and

If we can evaluate 100M nodes/second;

Then it would take 2.7 hours to find a solution 11 levels deep,

i.e. only 11 moves ahead in a game-playing application

Note: IBM Deep Blue (in 1997) could evaluate 200M positions/sec using special hardware, and looked 21 moves ahead

Uninformed Search

Breadth-first search

Uniform-cost search

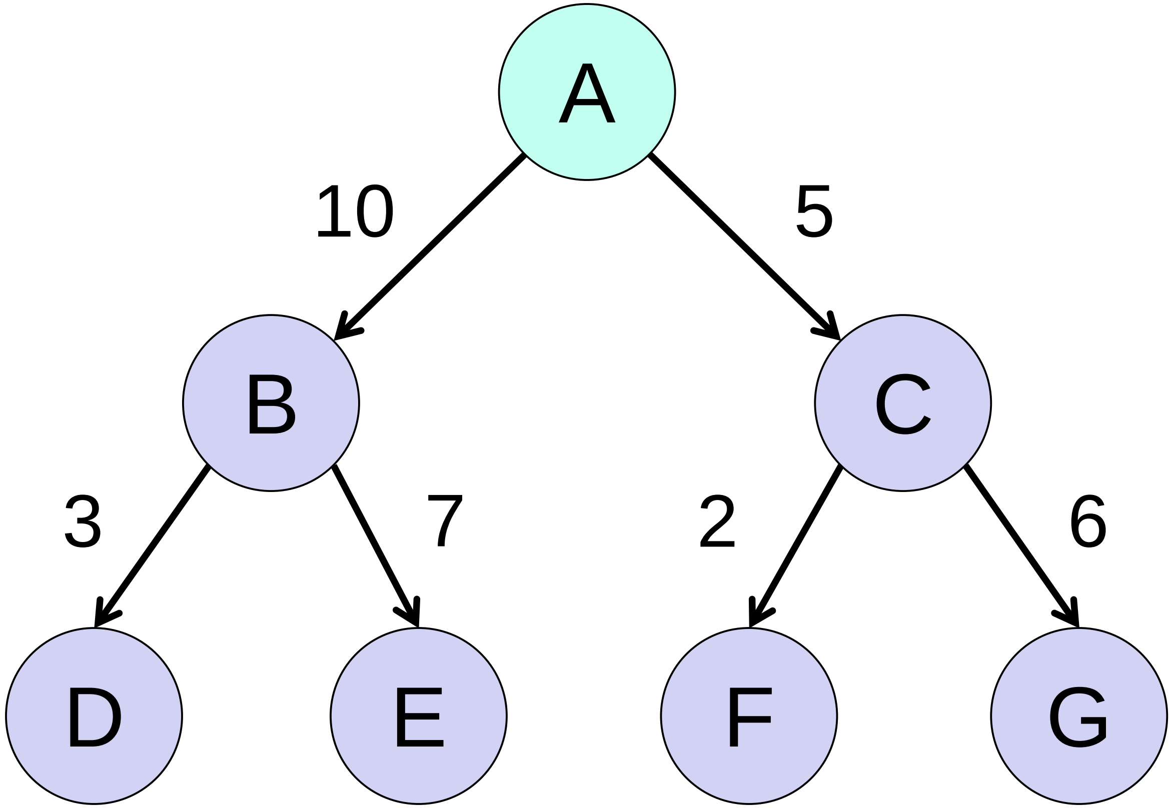
Depth-first search

Depth-limited search

Iterative-deepening search

Uninformed = Search method uses only information from problem

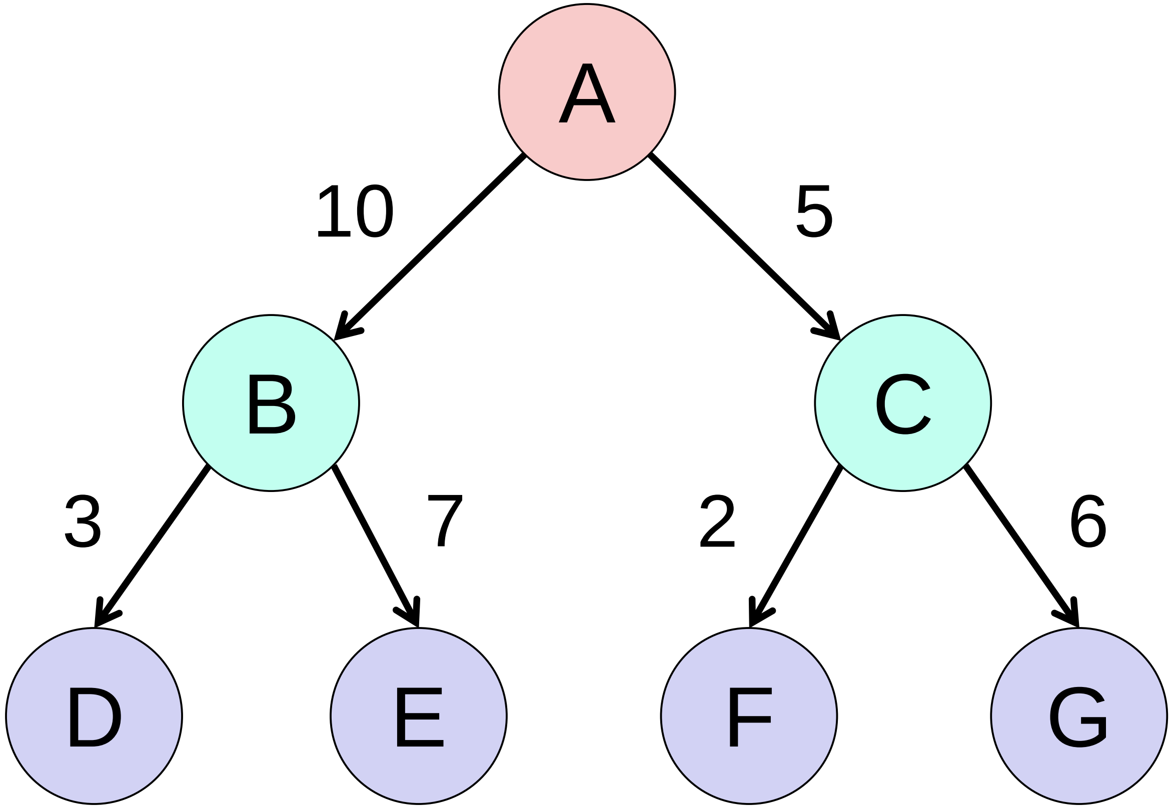
Uniform-Cost Search



Frontier (Queue)

1) A(0)

Uniform-Cost Search

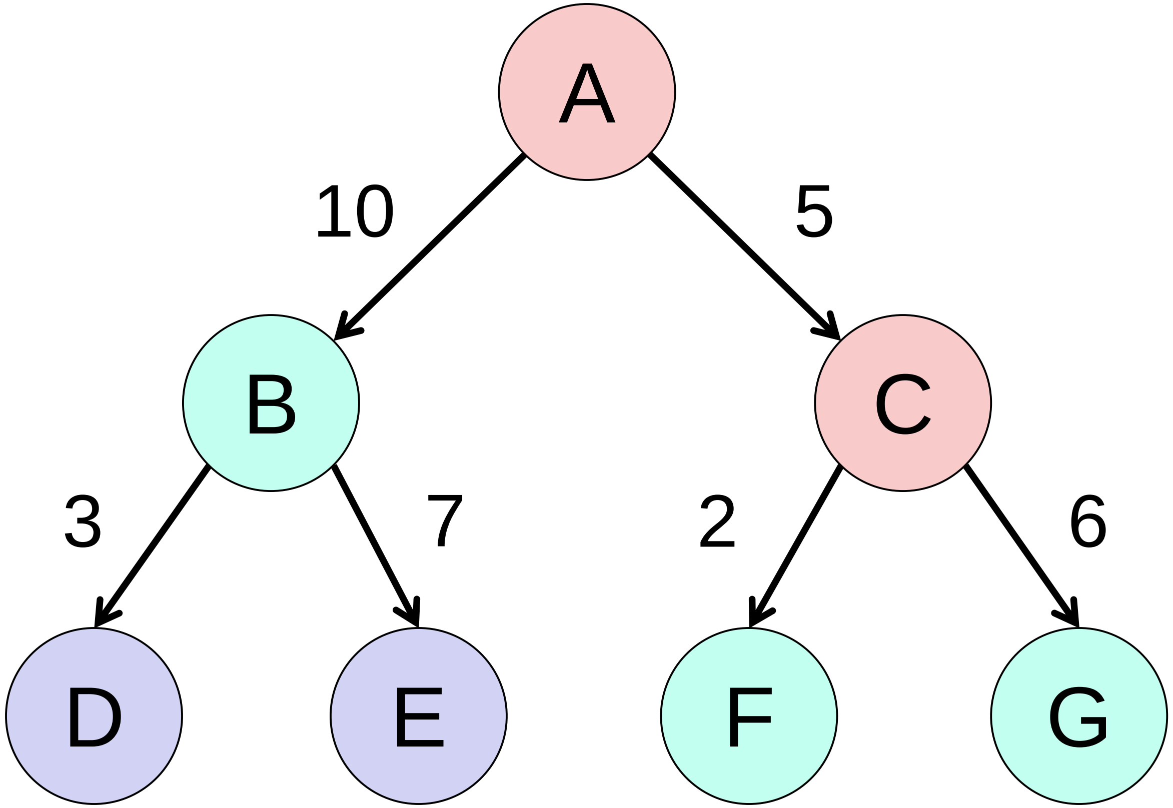


Frontier (Queue)

1) ~~A~~(0)

2) C(5) B(10)

Uniform-Cost Search



Frontier (Queue)

1) ~~A~~(0)

2) ~~C~~(5) B(10)

3) F(7) B(10) G(11)

Analysis of Uniform-Cost Search

Implementation: priority queue ordered by path cost

Complete? Yes if there are no zero cost actions

Optimal? Yes

Time and Space Complexity

b = branching factor (2 in our example)

e = minimum cost for any action

$O(b^{1+[C^*/e]})$ where C* is the path cost of the optimal solution

Uninformed Search

Breadth-first search

Uniform-cost search

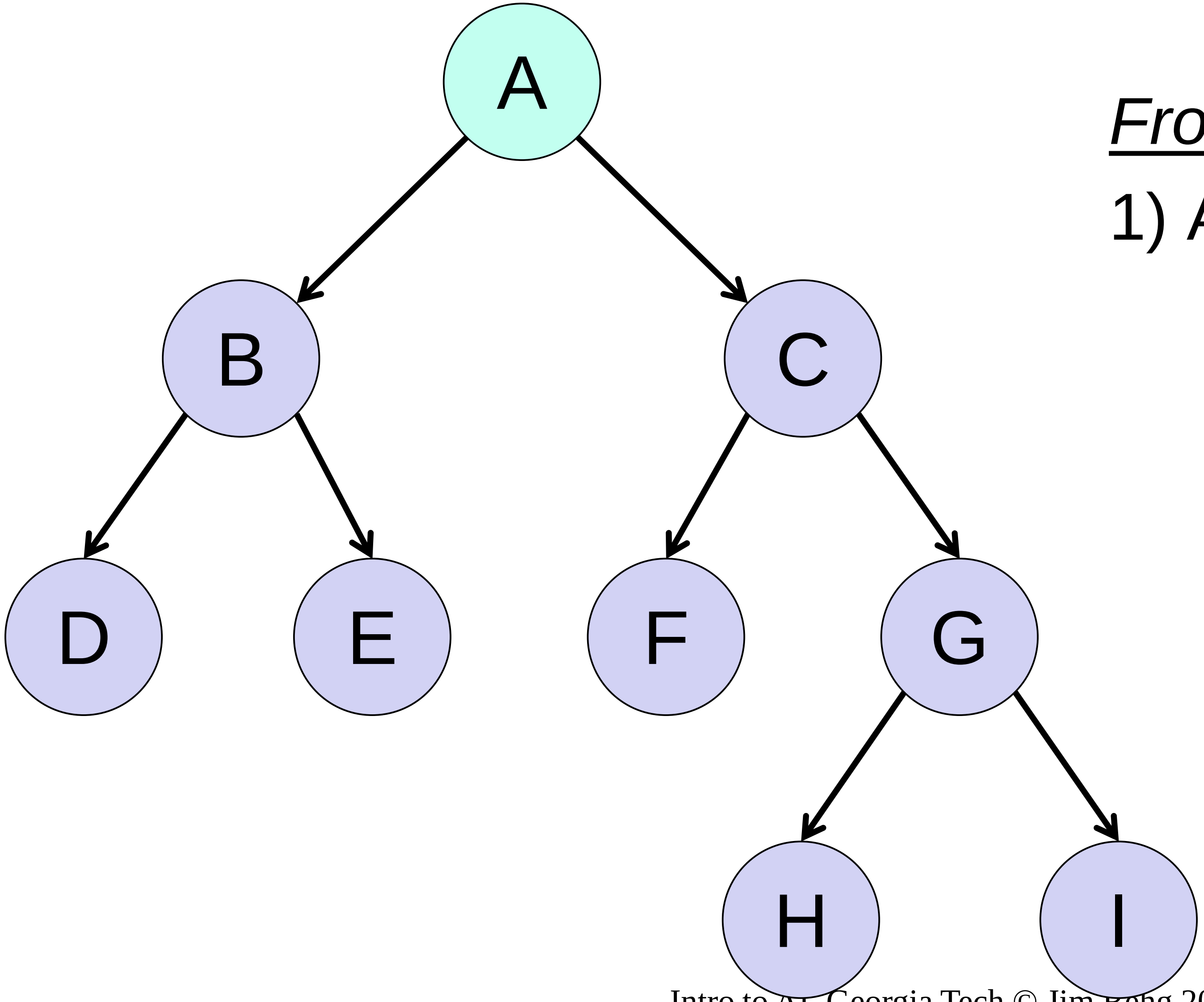
Depth-first search

Depth-limited search

Iterative-deepening search

Uninformed = Search method uses only information from problem

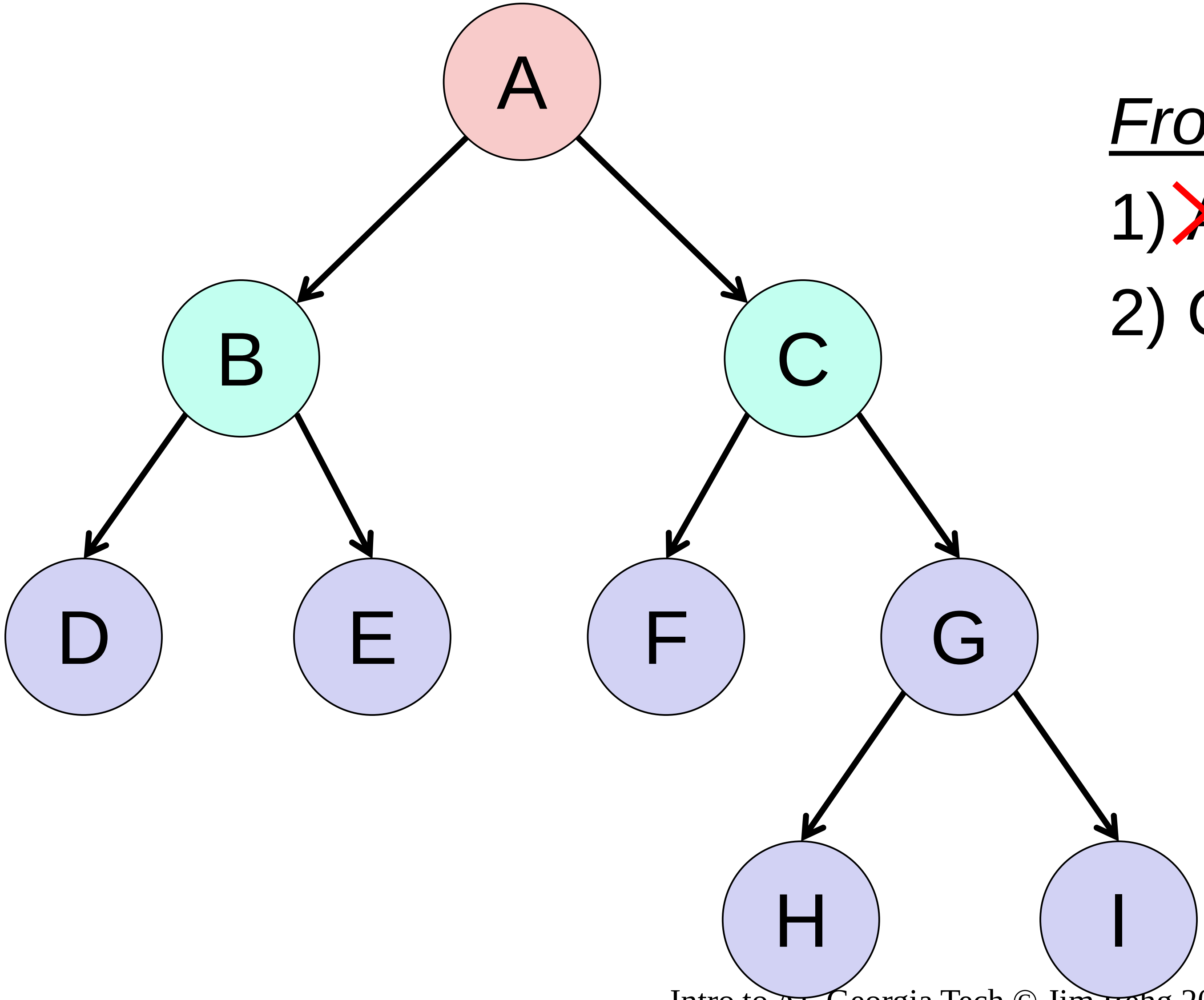
Depth-First Search



Frontier (Queue)

1) A

Depth-First Search

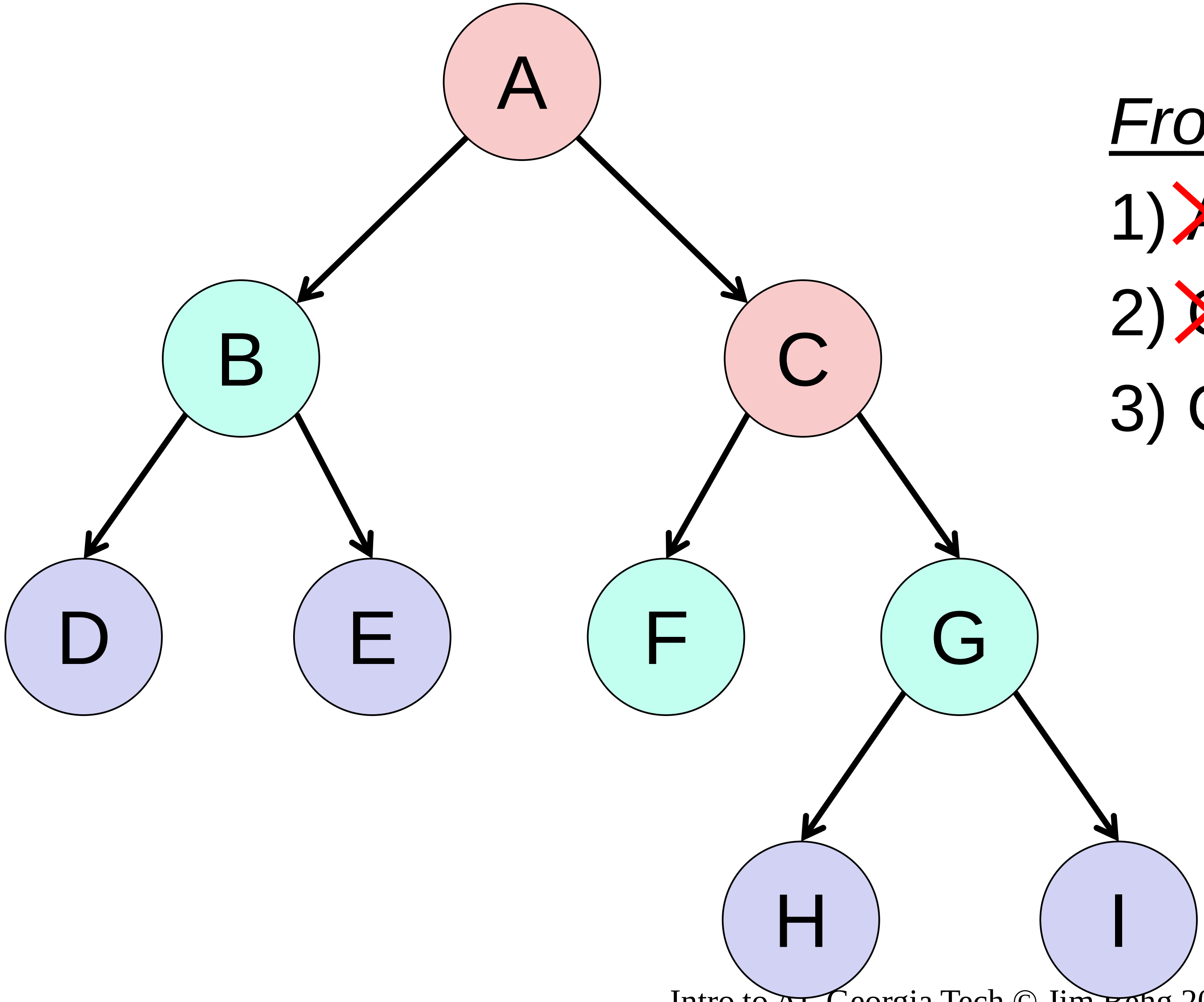


Frontier (Queue)

1) ~~A~~

2) C B

Depth-First Search



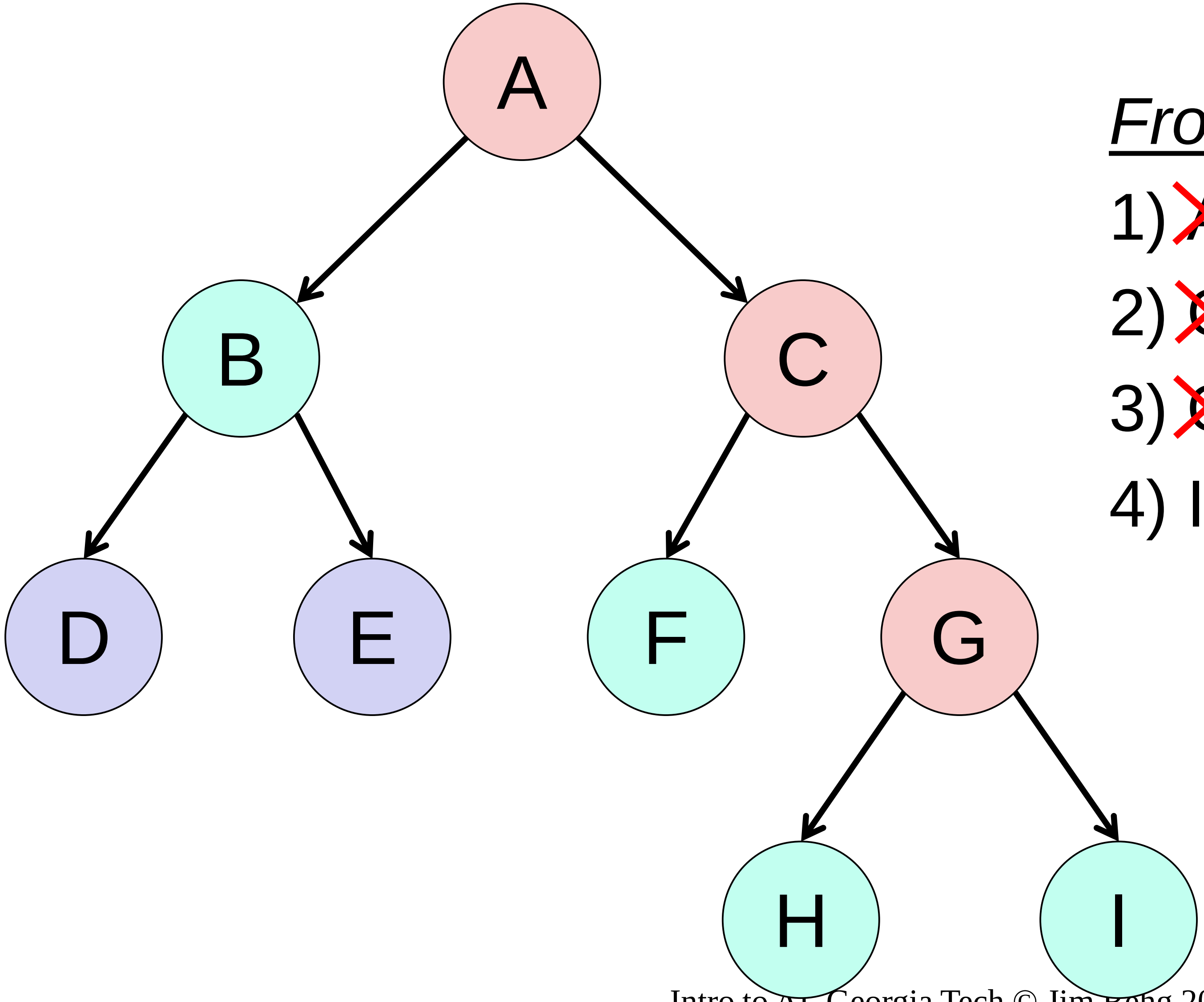
__Frontier (Queue)__

1) ~~A~~

2) ~~C~~ B

3) G F B

Depth-First Search



__Frontier (Queue)__

1) ~~A~~

2) ~~C~~ B

3) ~~G~~ F B

4) I H F B

Analysis of Depth-First Search

Implementation: LIFO queue

Complete? No, infinite loops are possible using tree-search

(graph-search version is complete for finite state spaces)

Optimal? No, can select a deep solution over a more shallow one

Time Complexity

m = maximum depth of any node in tree

$O(b^m)$ – can be larger than size of state space, larger than d

Space Complexity

O(bm) – linear in space, much better than breadth-first

Summary

Tree-based search methods maintain a frontier consisting of the currently-active nodes

Search methods differ in the order by which nodes are expanded, which is implemented through different types of queues

Breadth-First (BF) search is complete and can be made optimal for variable action costs via Minimum-Cost search, but time and space complexity is prohibitive

Depth-First (DF) search is neither complete nor optimal in its tree-based form, and time complexity can be prohibitive, but space complexity is attractive

Questions?