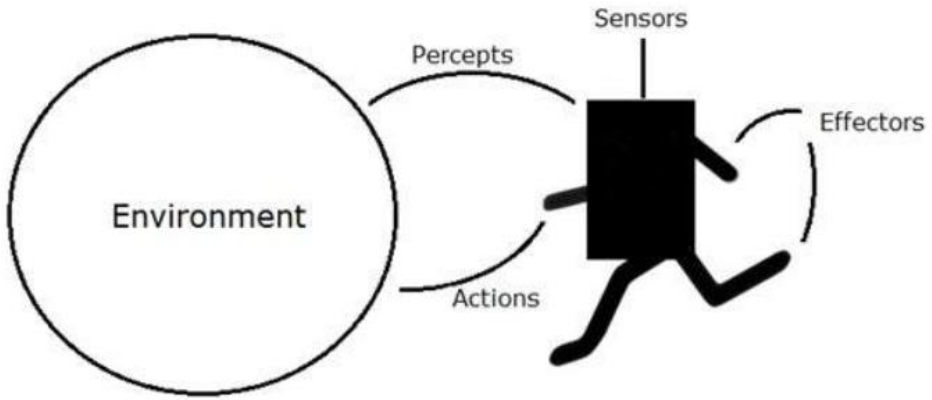
CS161
Discussion 2
Uninformed Search Algorithms

Agents

An agent ***perceives*** its ***environment*** through ***sensors*** and ***acts*** upon it through ***actuators***



Search Problem

A search problem consists of

- A state space $S = \{s_1, s_2, \ldots s_d\}$
- Initial State
- Actions: a set of possible actions
- Successor function (transition model): $F(s_t, a_t) = s_{t+1}$, sometimes with a path cost function
- Goal test : determine if solution is achieved.

A solution:

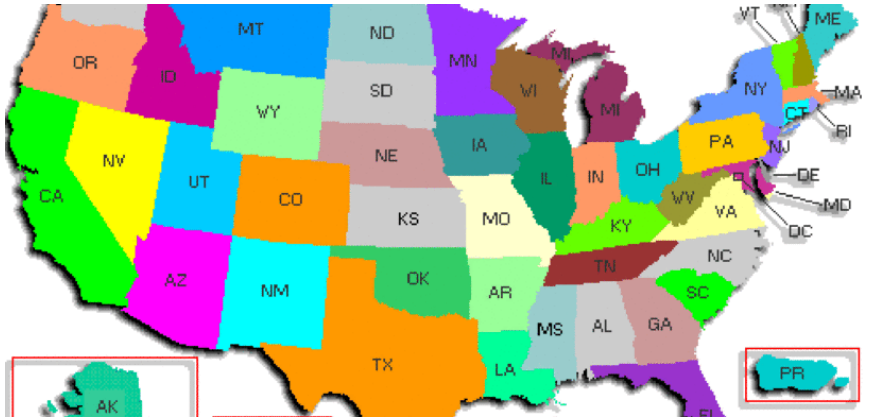
a sequence of actions that transform the initial state to a goal state

Problem formulation:

the process of deciding what actions and states to consider, given a goal.

Search Problem: Example - Travel from CA to MA

- **Initial State**: CA
- **State space**: 50 states in U.S.
- **Actions**: go to adjacent state. cost=distance
  - For example, Actions(CA)= {Go(OR), Go(AZ), Go(Nevada)}
- **Successor function** (transition model)
  - RESULT(In(CA), Go(AZ)) = In(AZ)
- **Goal test**:
  - state == MA?
- **Path cost function**
- Solution ?



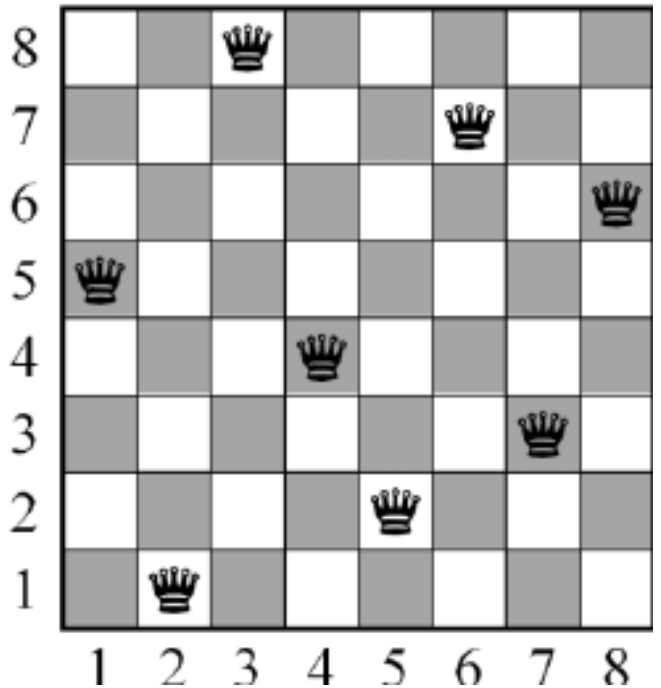
Search Problem: 8 queens

Objective:

Place eight queens on a chessboard such that no queen attacks any other.

(No two queens on the same row, column, diagonal)

(We don't care about how or how long you find the solution)



Search Problem: 8 queens

Formulate this problem.

- **States**: Any arrangement of 0 to 8 queens on the board is a state.
- **Initial state**: No queens on the board.
- **Actions**: Add a queen to any empty square.
- **Transition model**: Returns the board with a queen added to the specified square.
- **Goal test**: 8 queens are on the board, none attacked.

Search Problem: 8 queens

Formulate this problem.

- **States**: Any arrangement of o to 8 queens on the board is a state.
- **Initial state**: No queens on the board.
- **Actions**: Add a queen to any empty square.
- **Transition model**: Returns the board with a queen added to the specified square.
- **Goal test**: 8 queens are on the board, none attacked.

(We don't care about path cost here)

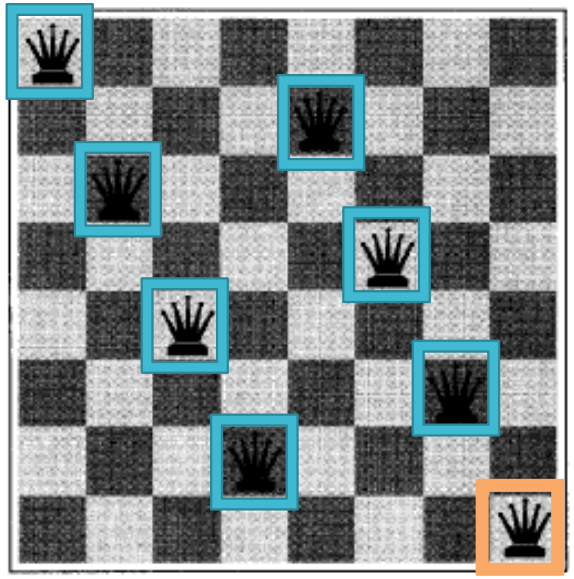
How many possible sequences?
$64 \times 63 \times \cdots \times 57 = 1.8 \times 10^{14}$

Search Problem: 8 queens

- **States**: All possible arrangements of n queens ($0 \leq n \leq 8$), one per column in the leftmost n columns, with no queen attacking another.
- **Actions**: Add a queen to any square in the leftmost empty column such that it is not attacked by any other queen.
- **Transition model**: Returns the board with a queen added to the specified square.
- **Goal test**: 8 queens are on the board, none attacked.

How large is the state space?

2057 (Why?)



Search Problem: 8 queens

- Incremental formulation
  - Start with an empty state
  - Each action adds a queen
- Complete-state formulation
  - Start with all 8 queens on the board
  - Moves queens around

In either case, path cost doesn't matter because only final state counts.

Search Problem Formulation – Exercise 1

- Two friends live in different cities on a map
- On every turn, we can simultaneously move each friend to a neighboring city
- Time(city $i \rightarrow$ neighbor j)=distance $d(i, j)$
- On each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin.
- We want the two friends to meet as quickly as possible.

❑ **Write a formulation for this search problem?**

Search Problem Formulation – Exercise 1

- Two friends live in different cities on a map
- On every turn, we can simultaneously move each friend to a neighboring city
- Time(city $i \rightarrow$ neighbor j)=distance $d(i, j)$
- On each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin.
- We want the two friends to meet as quickly as possible.

❑ **Write a formulation for this search problem?**

State:

Initial State:

Actions:

Path Cost:

Goal:

Search Problem Formulation – Exercise 2

- 3 missionaries and 3 cannibals are on one side of a river
- A boat can hold one or two people.
- **NOT** allowed (on either side): # of cannibals > # of missionaries
- Find a way to get everyone to the other side

Write a formulation for this search problem

Search Problem Formulation – Exercise 2

- 3 missionaries and 3 cannibals are on one side of a river
- A boat can hold one or two people.
- **NOT** allowed (on either side): # of cannibals > # of missionaries
- Find a way to get everyone to the other side

Write a formulation for this search problem

State:

Initial State:

Actions:

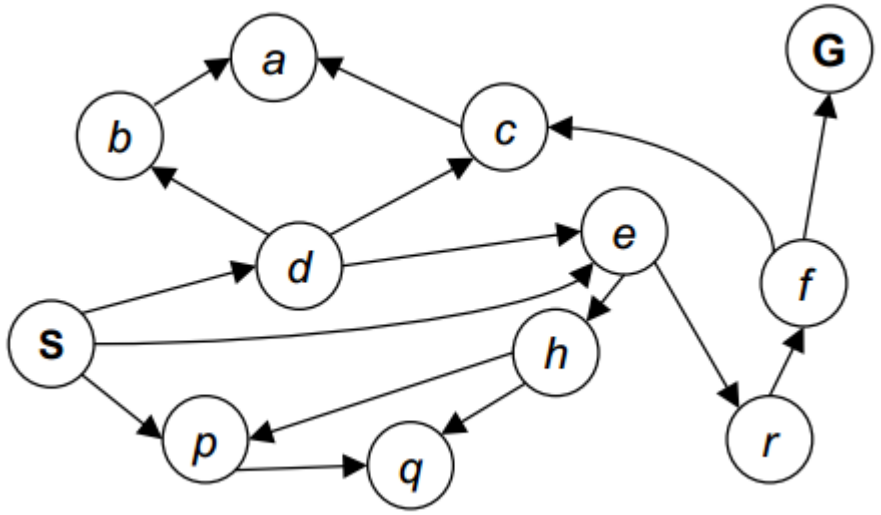
Path Cost:

Goal:

State space graph

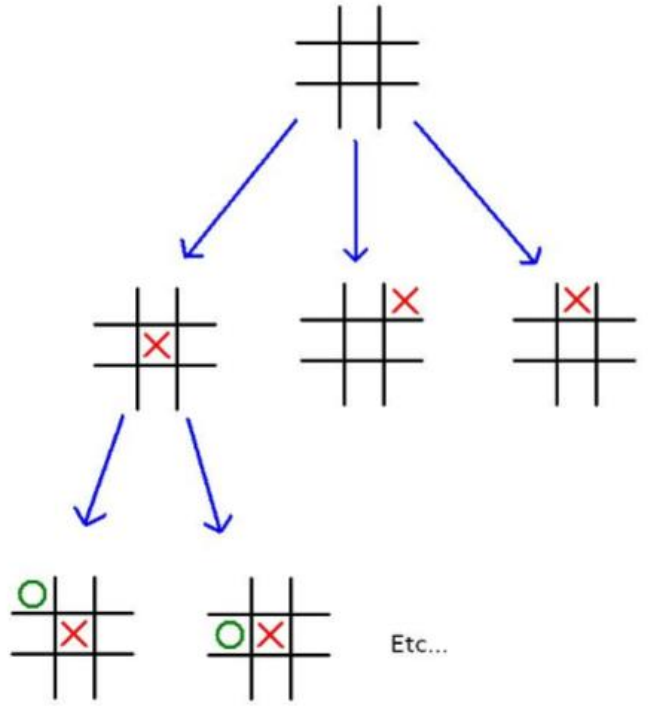
- Nodes are (abstracted) world configurations
- Arcs represent successors (action results)
- Goal test: one or a set of goal nodes

- Each state occurs only once!



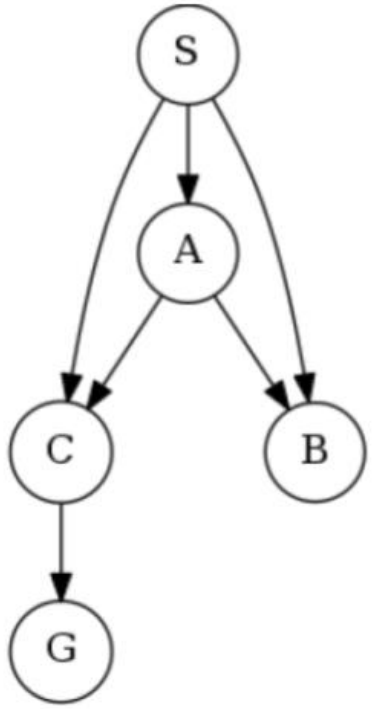
Search Tree

- A "what if" tree of plans and their outcomes
- Root: initial state
- Children: correspond to successors



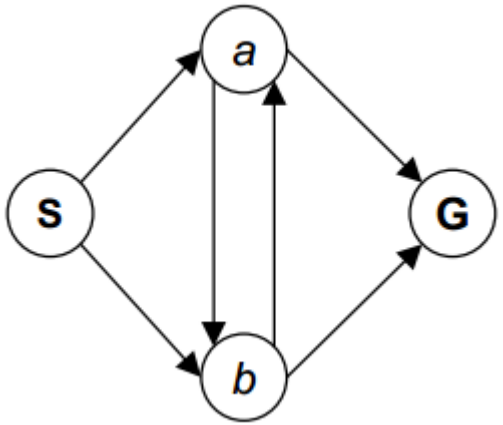
Exercise

- How many nodes are there in the search tree generated by the following state space graph?



State Space Graph vs Search Tree

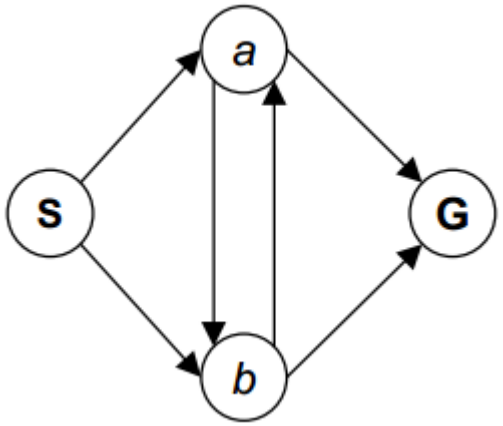
How big is the search tree?



4-state graph

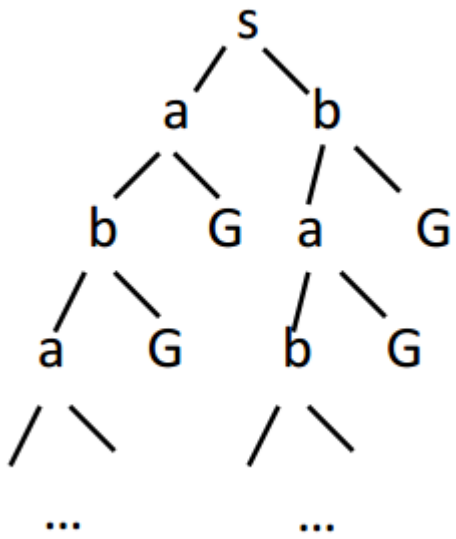
State Space Graph vs Search Tree

How big is the search tree?



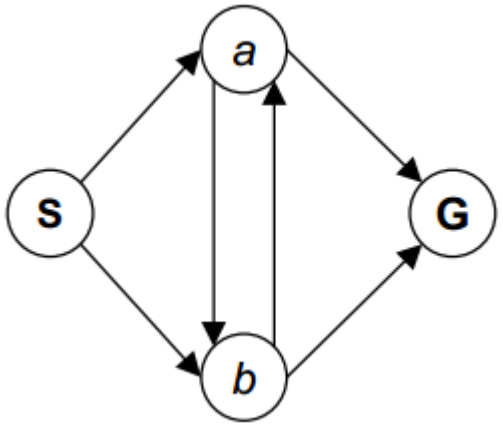
4-state graph

∞

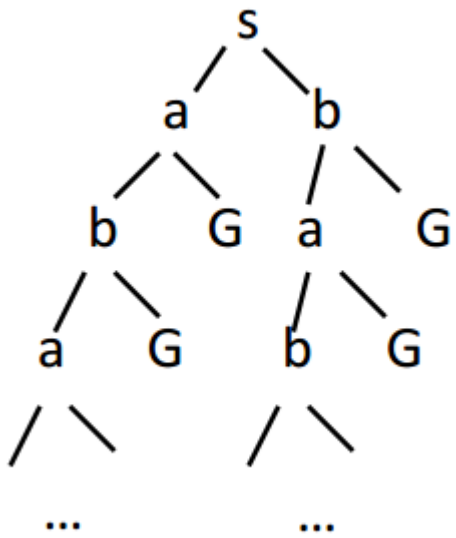


State Space Graph vs Search Tree

How big is the search tree? ∞



4-state graph



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

Search Problem

- Before an agent can start searching for solutions, a **goal** must be identified and a well-defined **problem** must be formulated

- Search problem formulation
  - **Initial State**
  - **A state space**
  - **Actions:** a set of possible actions
  - **Successor function** (transition model): $F(s_t, a_t) = s_{t+1}$ sometimes with a **path cost function**
  - **Goal test**: determine if solution is achieved.

  - A **solution**: a sequence of actions (a ***path***) that transform the initial state to a goal state

How to evaluate search algorithms

- completeness
- optimality
- time complexity
- space complexity

Uninformed Search

- BFS
- Uniform-cost search
- DFS, Depth-Limited Search
- Iterative Deepening
- Bidirectional search

BFS

- BFS
  - Expands shallowest nodes first
  - Complete
  - Optimal for unit step costs
  - Time complexity (*exponential*): # of generated nodes
    - $\mathrm{b} + \mathrm{b}^2 + \cdots + \mathrm{b}^{\mathrm{d}} = O(b^d)$ (*Goal test on generation* by default)
    - Goal test on expansion: $O(b^{d+1})$
  - Space complexity (*exponential*): $O(b^d)$
    - Fringe $O(b^d)$ Nodes in last generated layer

Uniform-Cost Search

- Expands the node with lowest path cost
- Uniform-cost search keeps going after a goal node has been generated.
  - Terminate after a goal node is expanded
- Optimal in general
  - Infinite loop if there is a path with an infinite sequence of zero-cost actions
- Complete
  - (If all step cost > a small positive constant ϵ)

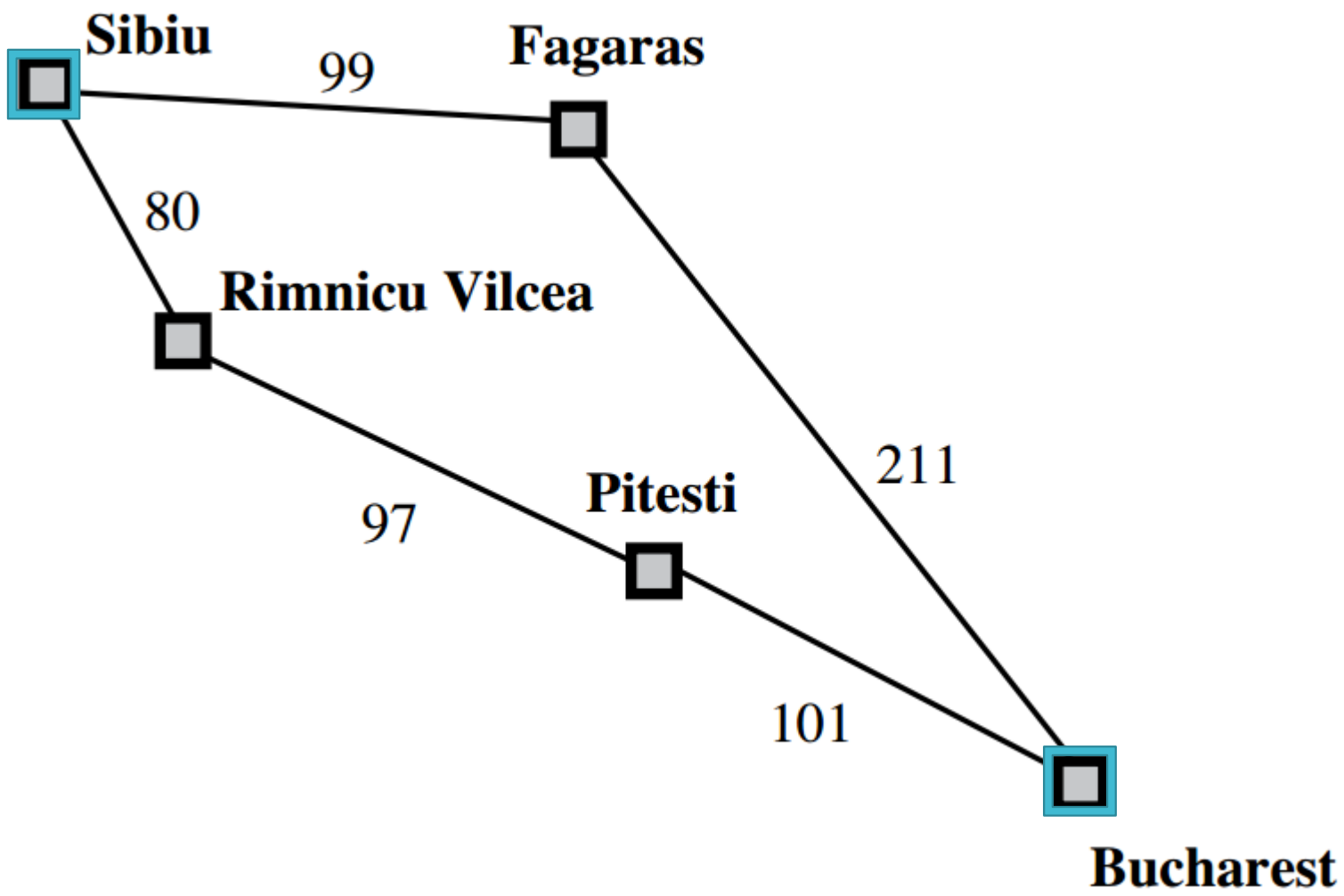
Uniform-Cost Search

- Time Complexity (Worst Case)
  - C: cost of optimal solution
  - Every action costs at least ϵ
  - Time complexity: $O(b^{1+\lfloor C/\epsilon \rfloor})$
    - When all step costs are equal: $O(b^{1+d})$ (test on expansion)
- Space Complexity (Worst Case)
  - Fringe: priority queue (priority: cumulative cost)
  - Worst case: roughly the last tier, $O(b^{1+\lfloor C/\epsilon \rfloor})$

Uniform-Cost Search

- Uniform-cost search and BFS
  - BFS stops after a goal node is generated (unless otherwise specified)
  - Uniform-cost search keeps going after a goal node has been generated

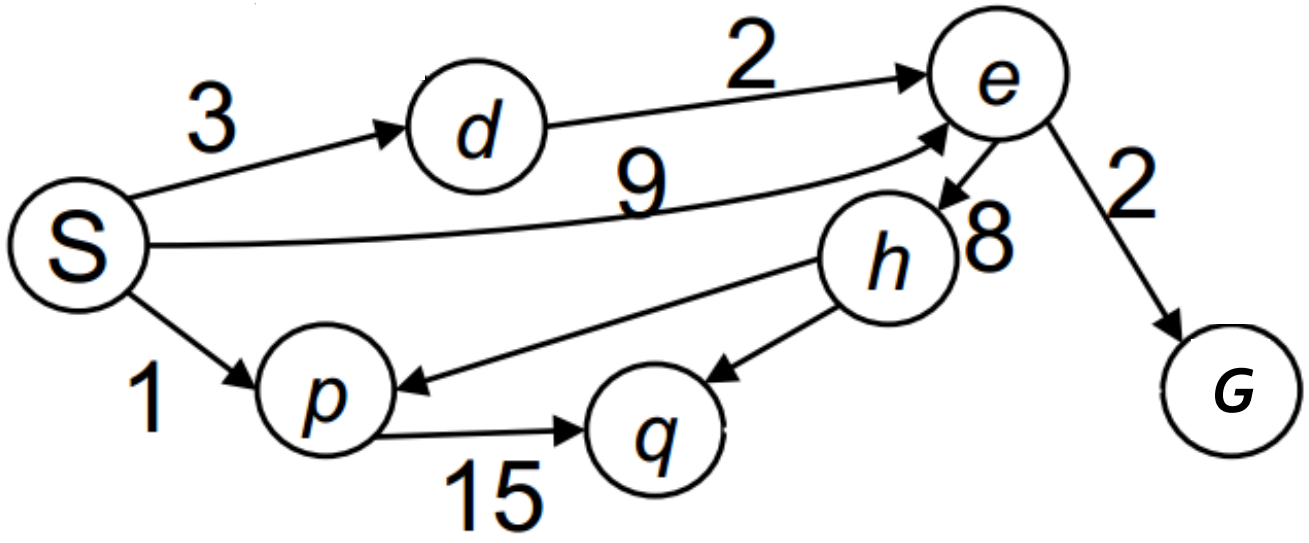
Uniform-Cost Search - Example



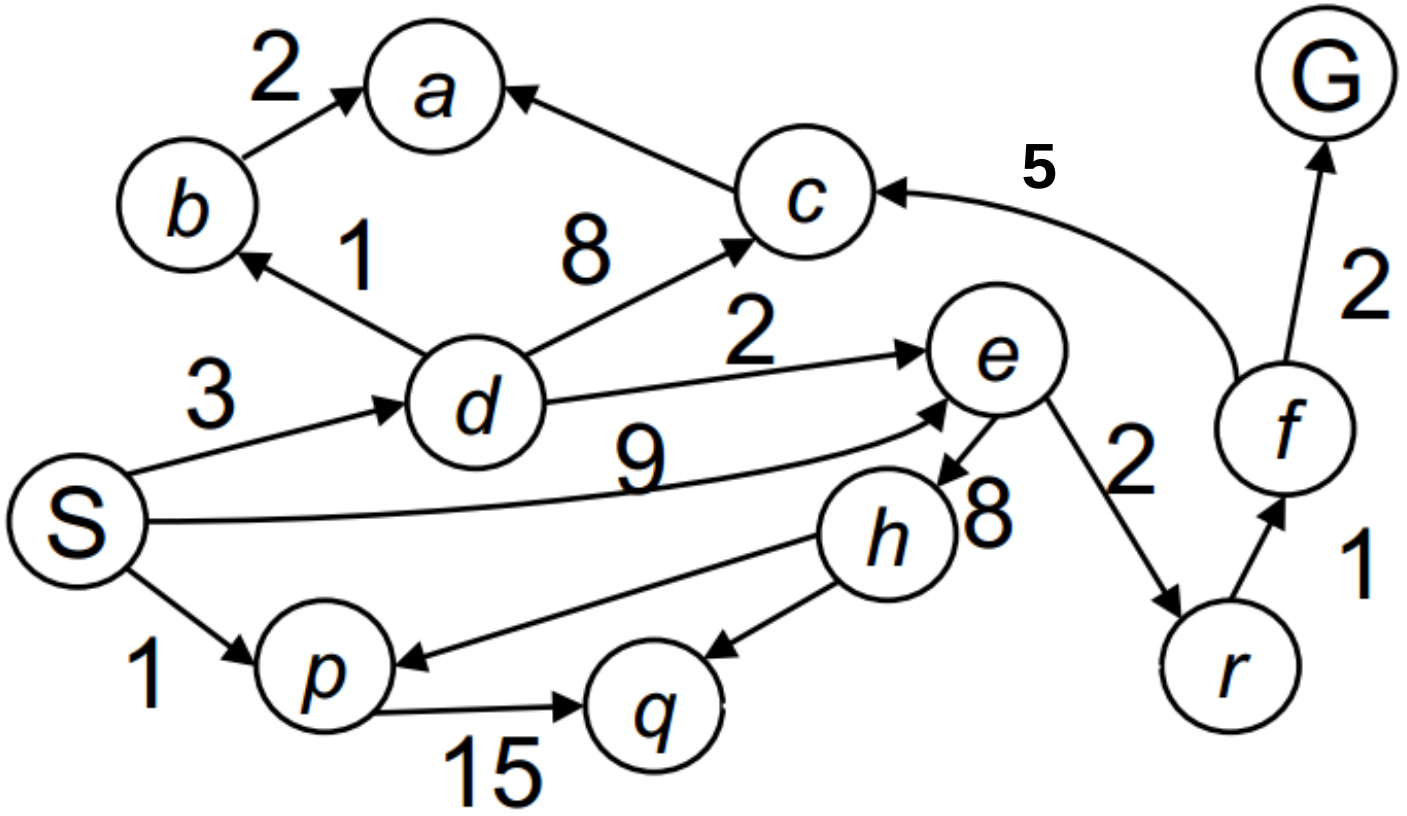
Uniform-Cost Search – Exercise 1

Use uniform-cost search

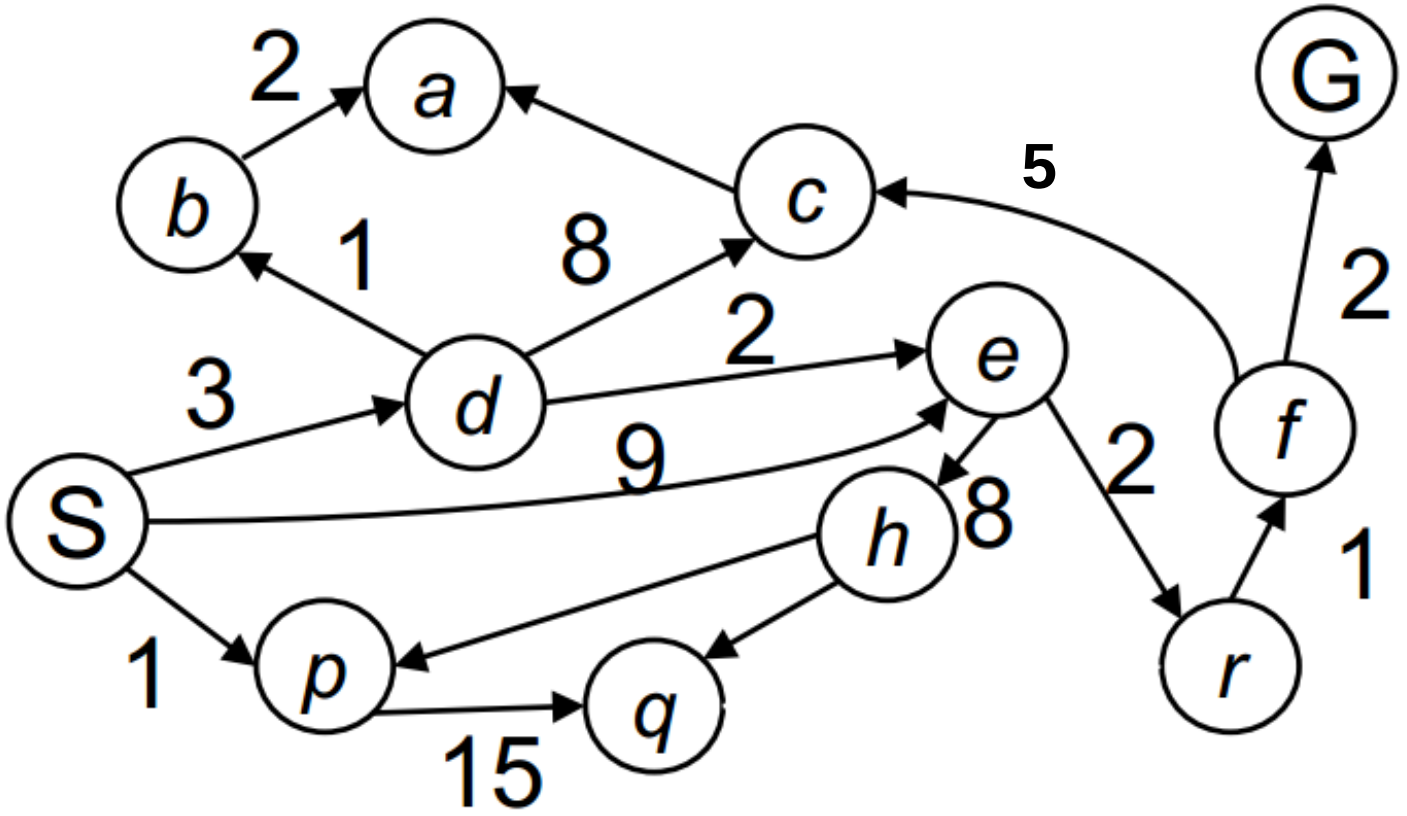
- Give the generated (partial) search tree
- Show in what order we expand nodes
- Return the optimal solution (path)



Uniform-Cost Search – Exercise 2



Uniform-Cost Search – Exercise 2



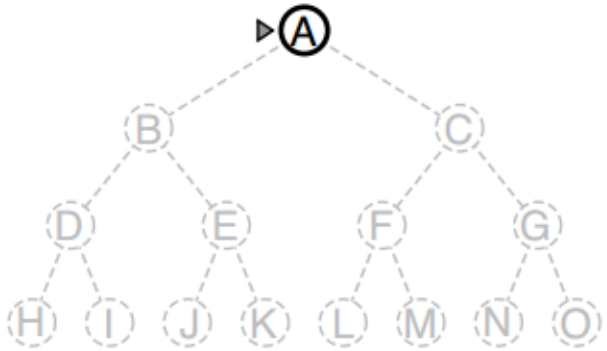
Think:
When does search terminate?

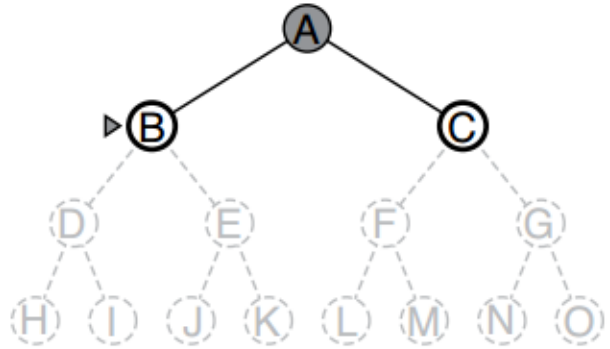
DFS and Depth-Limited Search

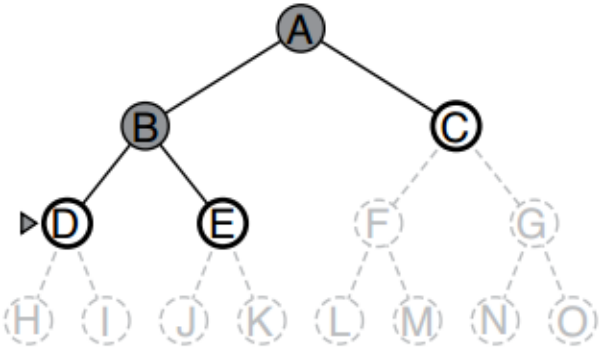
- DFS
  - Not optimal
  - Not complete
  - Time complexity $O(b^m)$ (m: maximum depth)
  - Space complexity $O(bm)$ Fringe: path and siblings along the path

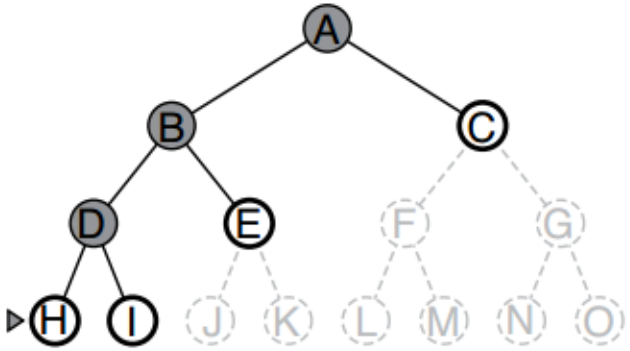
- Depth-Limited search: add a depth bound l
  - Complete
  - Not optimal
  - Time complexity $O(b^l)$
  - Space complexity $O(bl)$

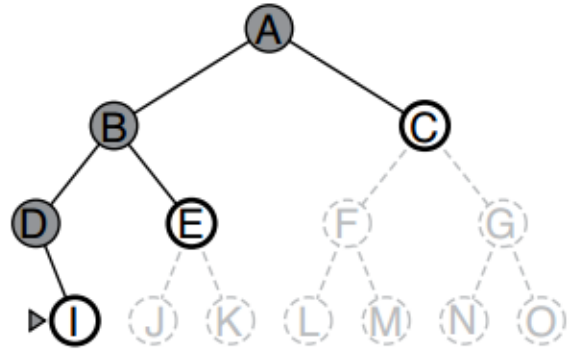
DFS

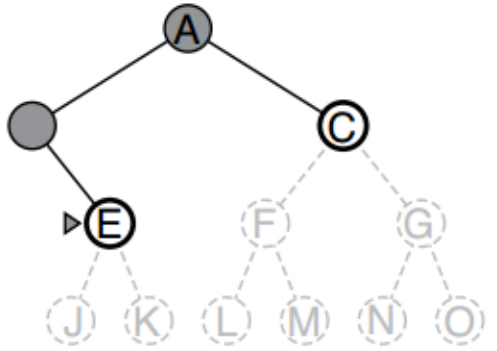

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O

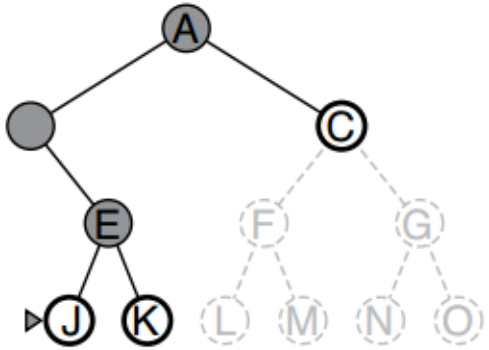

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O

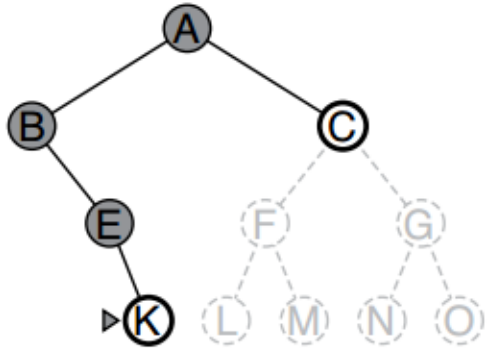

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O

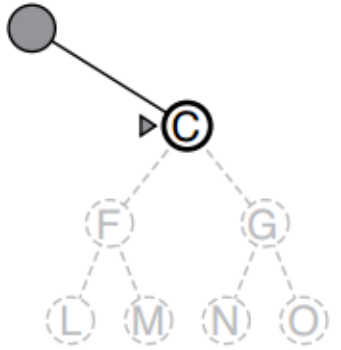

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O


A
B
C
D
E
F
G
I
J
K
L
M
N
O


A
C
E
F
G
J
K
L
M
N
O


A
C
E
F
G
J
K
L
M
N
O

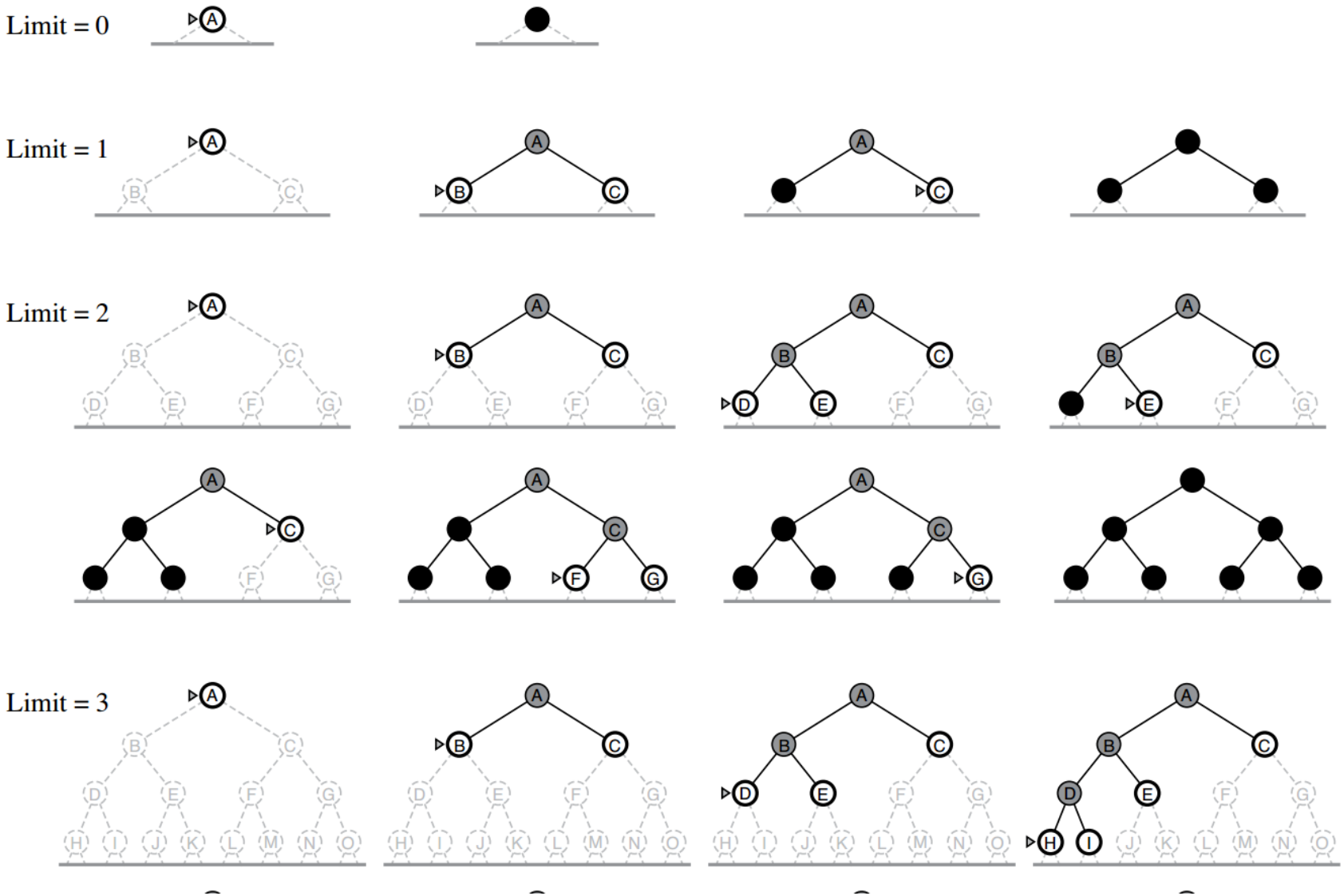

A
B
C
E
F
G
K
L
M
N
O


C
F
G
L
M
N
O

Iterative Deepening Search

- Depth-first search. Increase depth limits until a goal is found
- Complete; Optimal for unit step costs
- Space complexity of DFS: $O(bd)$ (d: depth of the shallowest solution)
- Time complexity comparable to BFS
  - (Analysis: see lecture slides)

Iterative Deepening



Bidirectional Search

- Run two simultaneous searches (BFS/Iterative Deepening)
  - One forward from the initial state
  - The other backward from the goal
- Replacing Goal test: Check whether the frontiers of the two searches intersect
  - The first found solution may not be optimal
    - Additional search is required to make sure there isn't short-cut across the gap! (Will show later)

Bidirectional Search (cont'd)

- What if we have multiple goal states?
  - For explicitly listed goal states: construct a new dummy goal state
    - Dummy goal state's immediate predecessors are all the actual goal states
  - For abstract description (e.g. "no queen attacks another queen")
    - Bidirectional search is difficult to use
- Time complexity & Space Complexity (Using two BFS)
  - $O(b^{d/2})$

Exercise - Word Ladder

- Input:
  - *beginWord*
  - *endWord* (endWord != beginWord)
  - A dictionary
- Transformation:
  - wordA -> wordB (e.g. "hit"-> "hot")
  - Only one letter can be changed at a time.
  - wordB must be in dictionary
- Question:
  - Transform *beginWord to endWord*
  - How many transformations we need at least?
    - Return -1 if no such sequence
- **How do you solve it?**

Exercise – Word Ladder

Example

- *beginWord* = "hit"
- *endWord* = "cog",
- *dictionary* = ["hot","dot","dog","lot","log","cog"]
- Output: 4
  - "hit" -> "hot" -> "dot" -> "dog" -> "cog"

https://leetcode.com/problems/word-ladder/

Bidirectional Search (cont'd)

We mentioned

- The first found solution may not be optimal
  - Additional search is required to make sure there isn't short-cut across the gap!

- When does this happen?
  - What if "hot" and "log" are connected in the word ladder example?

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.