Artificial Intelligence

Search… (continued)

Lecture 4

CS 444 – Spring 2019

Dr. Kevin Molloy

Department of Computer Science

James Madison University

Outline for Today

- Continuing discuss uninformed search methods and problem formulation.

- Short quiz

Problem 3.6b

Given a complete problem formulation for each of the following. Choose a formulation that is precise enough to be implemented.

A 3-foot tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. He would like to get the bananas. The room contains two stackable, movable, climbable 3 foot-high crates.

Initial state: As described (monkey, bananas suspended from ceiling, 2 crates on the floor in a room)

Goal state: Monkey has bananas.

Successor function: Hop on crate, hop off crate, move/push crate, place crate on top of a stack of crates, walk from a spot to another spot, grab bananas.

Cost function: Number of actions.

Problem 3.6d

Given a complete problem formulation for each of the following. Choose a formulation that is precise enough to be implemented.

You have three jugs, measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet. You can fill the jugs up, empty them out from one to another or onto the ground. You need to measure out exactly one gallon.

Initial state: Jugs empty [0, 0, 0]

Goal state: [x, y, 1] or [x, 1, z] or [1, y, z] (if too many states, we could state as one of the 3 jugs has exactly 1 gallon of water)

Successor function:

Fill([x, y, z],(1 || 2|| 3)) → [12, y, z] or [x, 8, z] or [x, y, 3]

Empty([x, y, z], (1 || 2 || 3)) → [0, y, z] or [x, 0, z] or [x, y, 0]

Transfer (x,y) transfer the contains of y into x until either y is empty OR x is at capacity.

Cost function: Number of actions.

Properties of Breadth-first Search (BFS)

Problems:

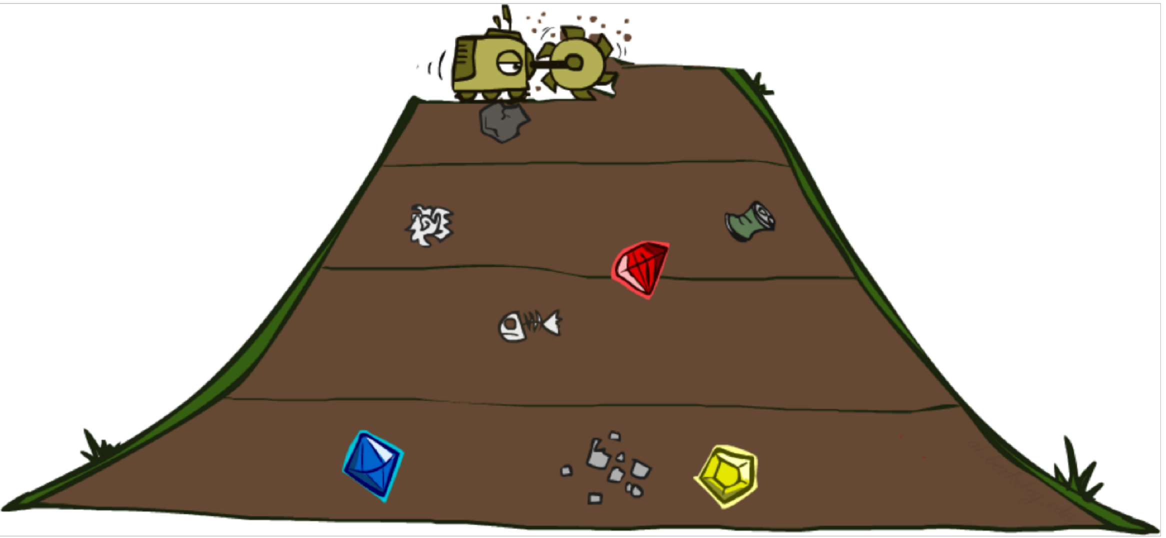
- If the path cost is a non-decreasing function of the depth of the goal node, BFS is optimal (uniform cost search a generalization).
- A graph with no weights can be considered to have edges of weight 1, in this case, BFS is optimal.
- BFS will find the shallowest goal after expanding all shallower nodes (if branching factor is finite). Hence, BFS is complete.
- BFS is not very popular because time and space complexity are exponential (with respect to d).
- Memory requirements of BFS are a bigger problem.

Time and Memory Requirements for BFS

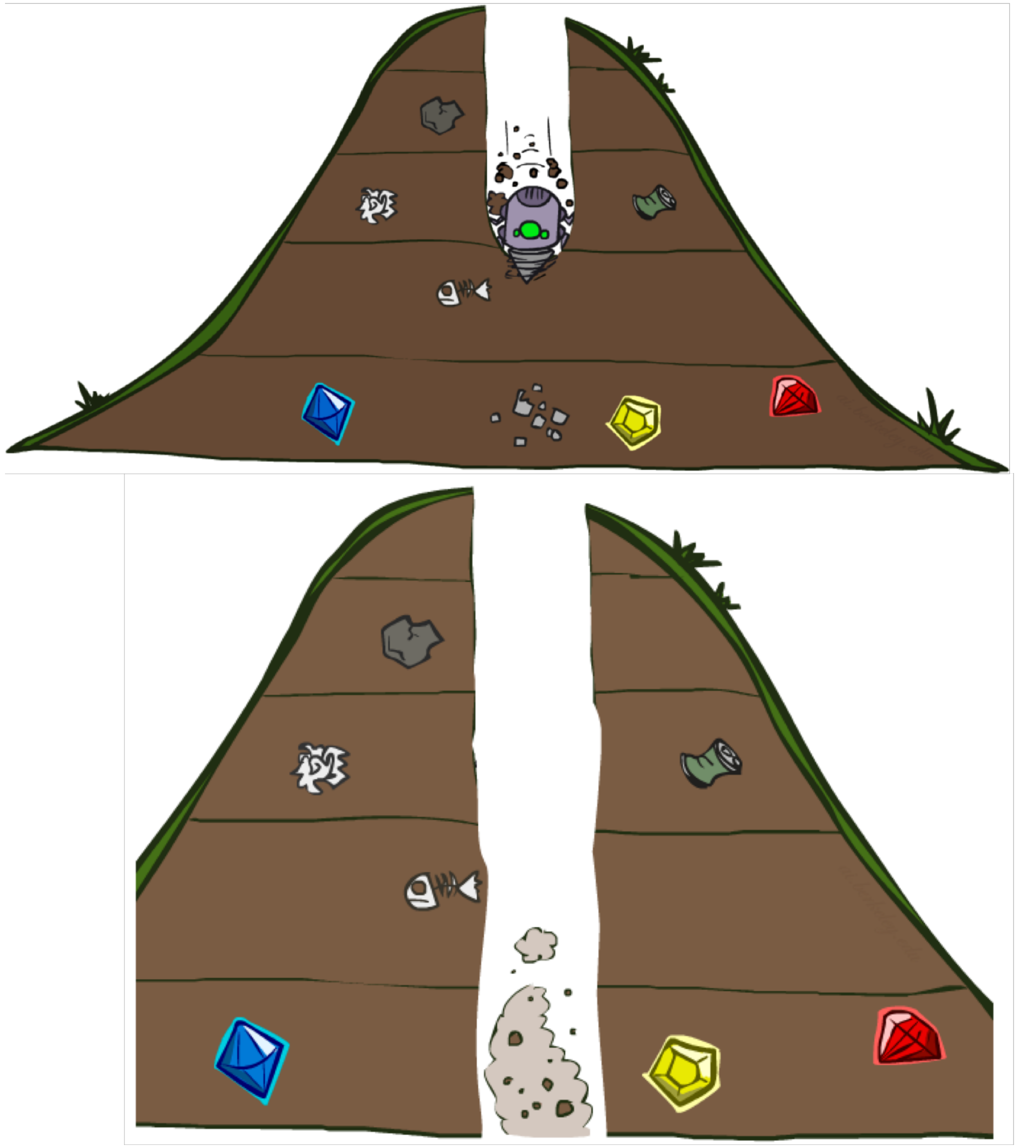
Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 KB
4	11,110	11 milliseconds	10.6 MB
6	10^6	1.1 seconds	1 GB
8	10^8	2 minutes	103 GB
10	10^{10}	3 hours	10 TB
12	10^{12}	13 days	1 PB
14	10^{14}	3.5 years	99 PB
16	10^{16}	350 years	10 EB

For a branching factor of b = 10; 1 million nodes/second and 1,000 byte nodes.

Properties of Search Strategies



Breadth First Search



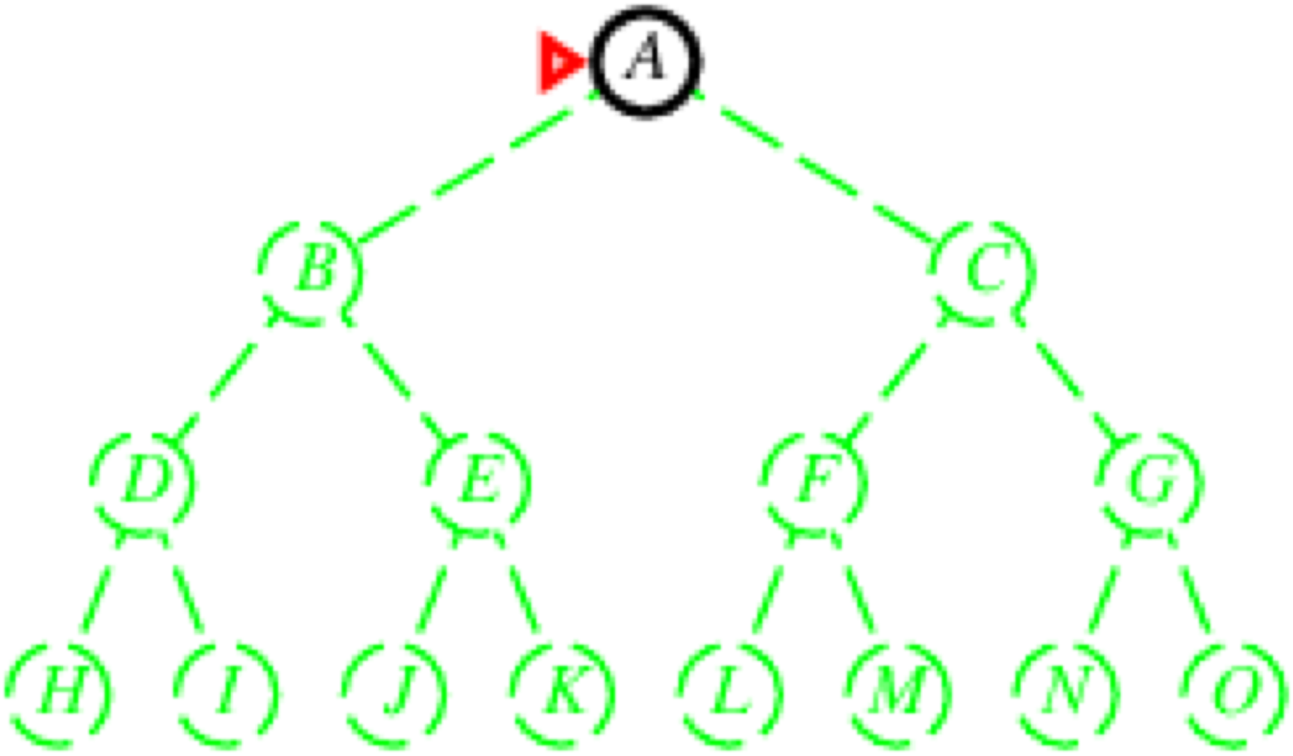
Depth First Search

Depth-first Search (DFS)

Strategy: Expand deepest unexpanded node

Implementation:

Fringe = last-in first-out (LIFO), i.e., unvisited successors at front (F is a stack)

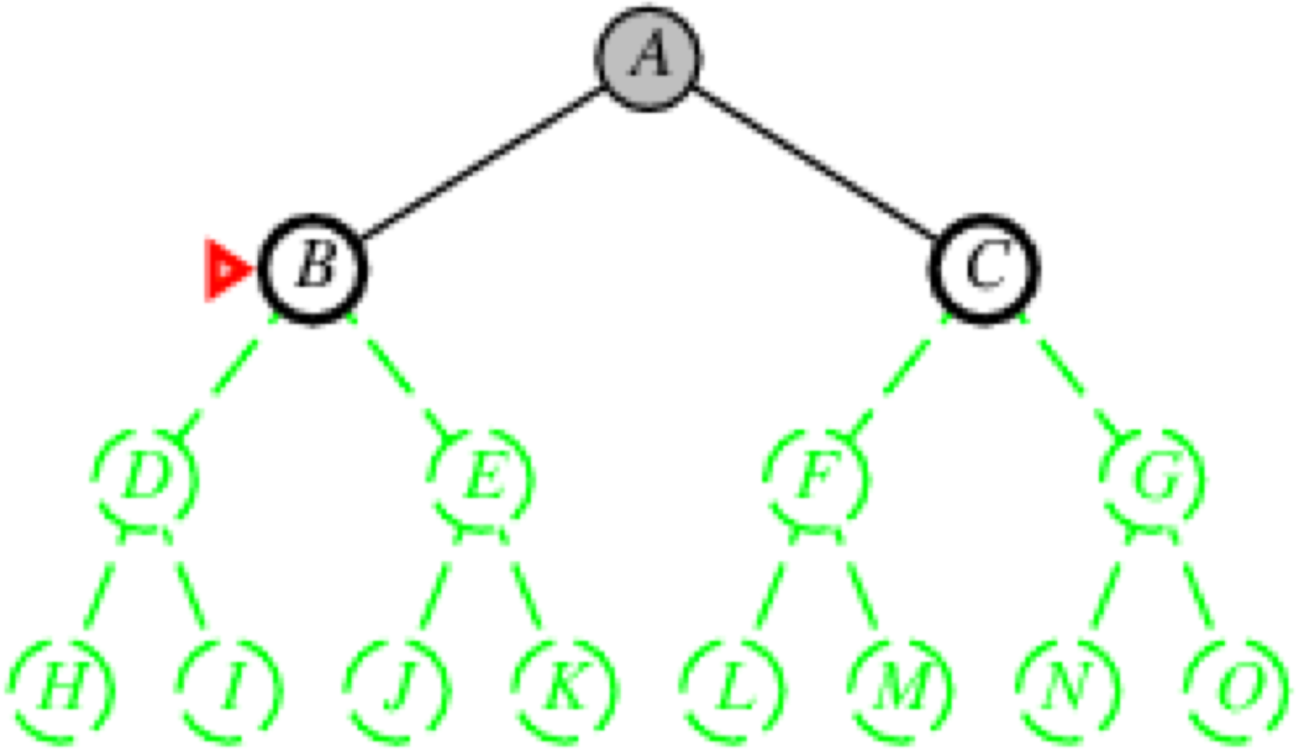


Depth-first Search (DFS)

Strategy: Expand deepest unexpanded node

Implementation:

Fringe = last-in first-out (LIFO), i.e., unvisited successors at front (F is a stack)

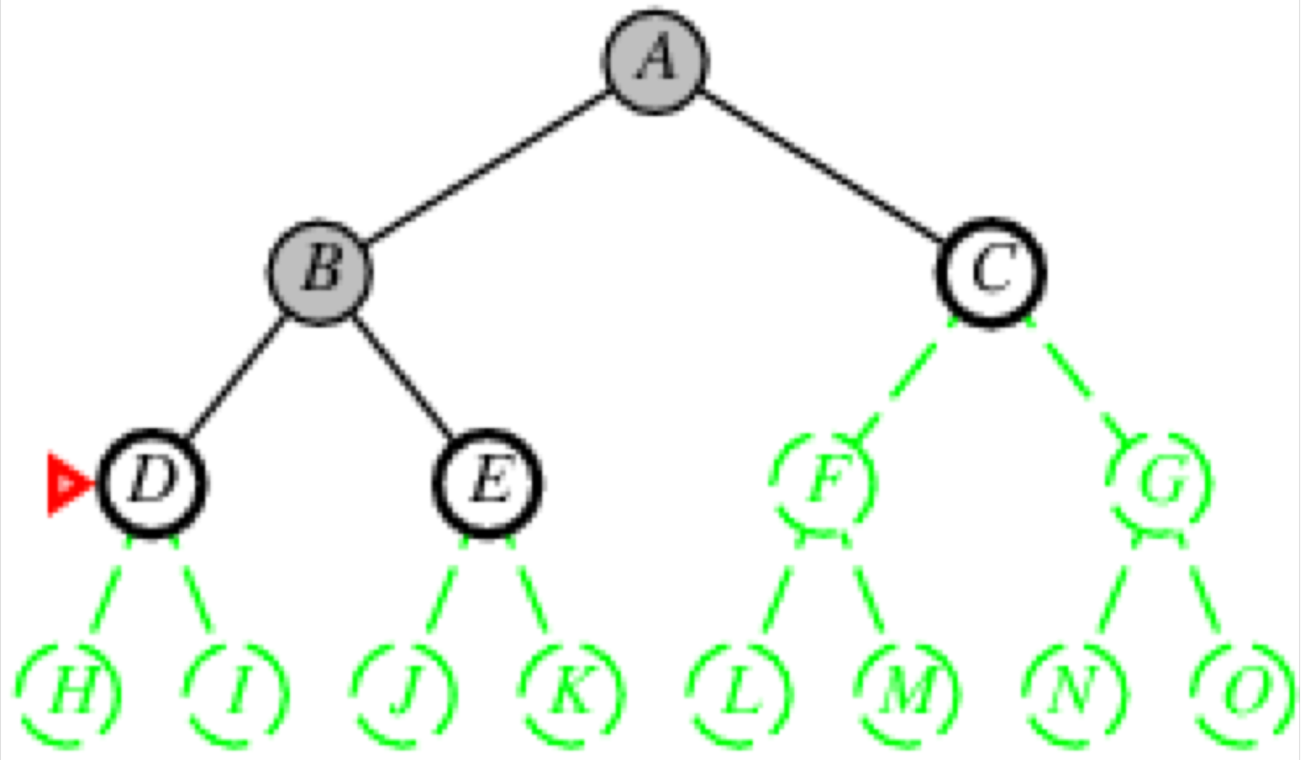


Depth-first Search (DFS)

Strategy: Expand deepest unexpanded node

Implementation:

Fringe = last-in first-out (LIFO), i.e., unvisited successors at front (F is a stack)

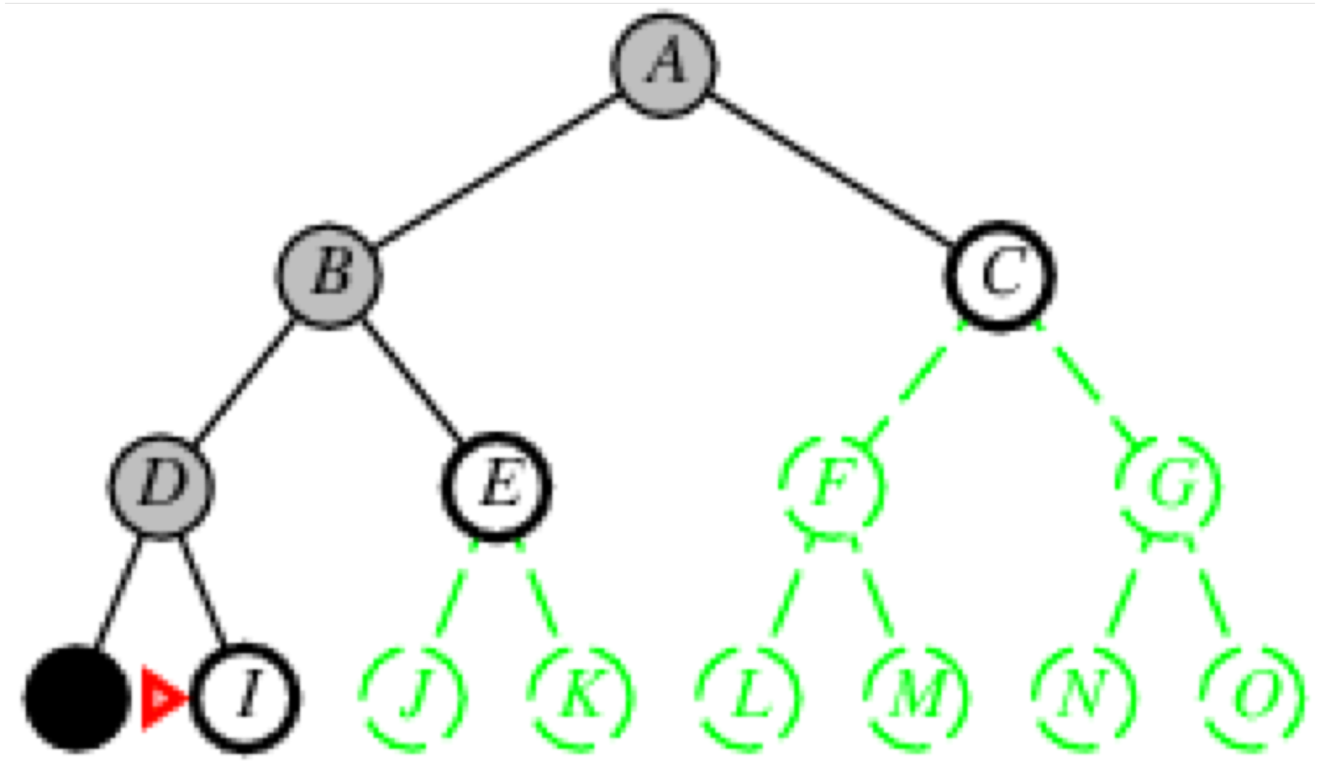


Depth-first Search (DFS)

Strategy: Expand deepest unexpanded node

Implementation:

Fringe = last-in first-out (LIFO), i.e., unvisited successors at front (F is a stack)

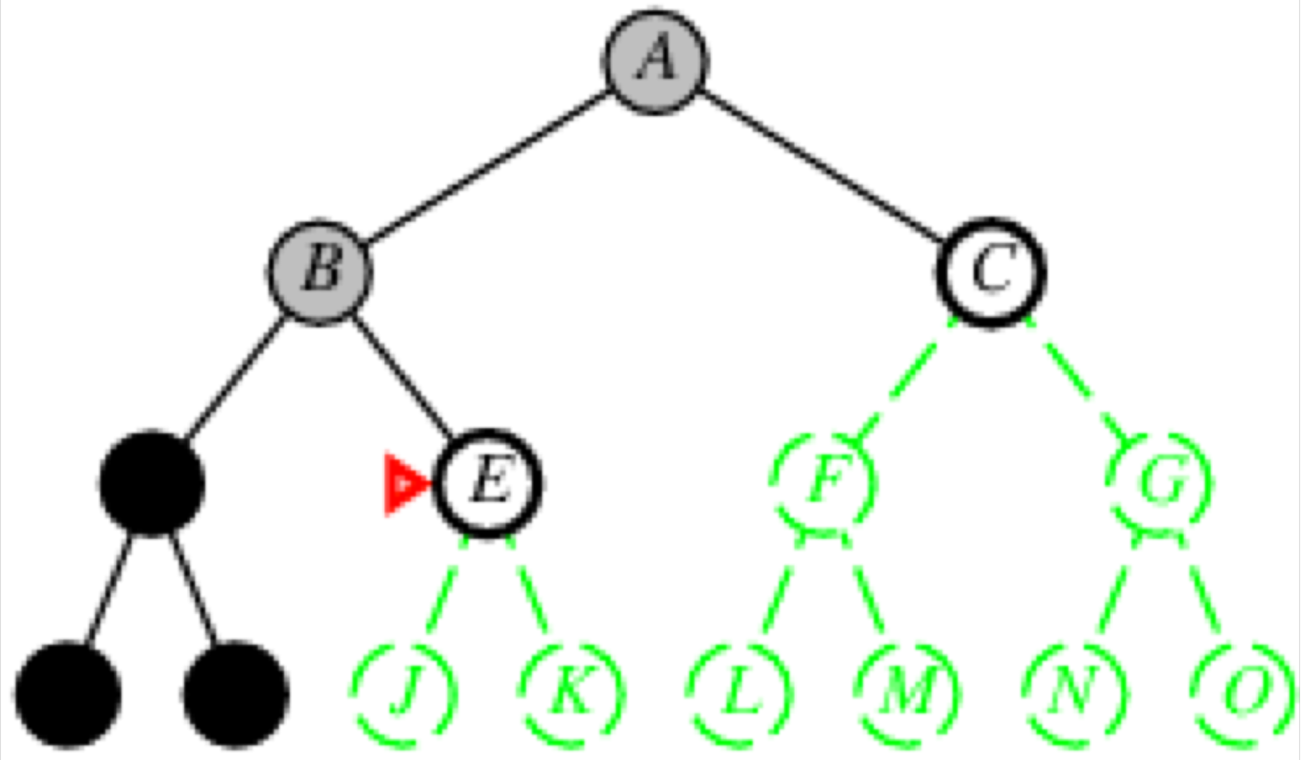


Depth-first Search (DFS)

Strategy: Expand deepest unexpanded node

Implementation:

Fringe = last-in first-out (LIFO), i.e., unvisited successors at front (F is a stack)

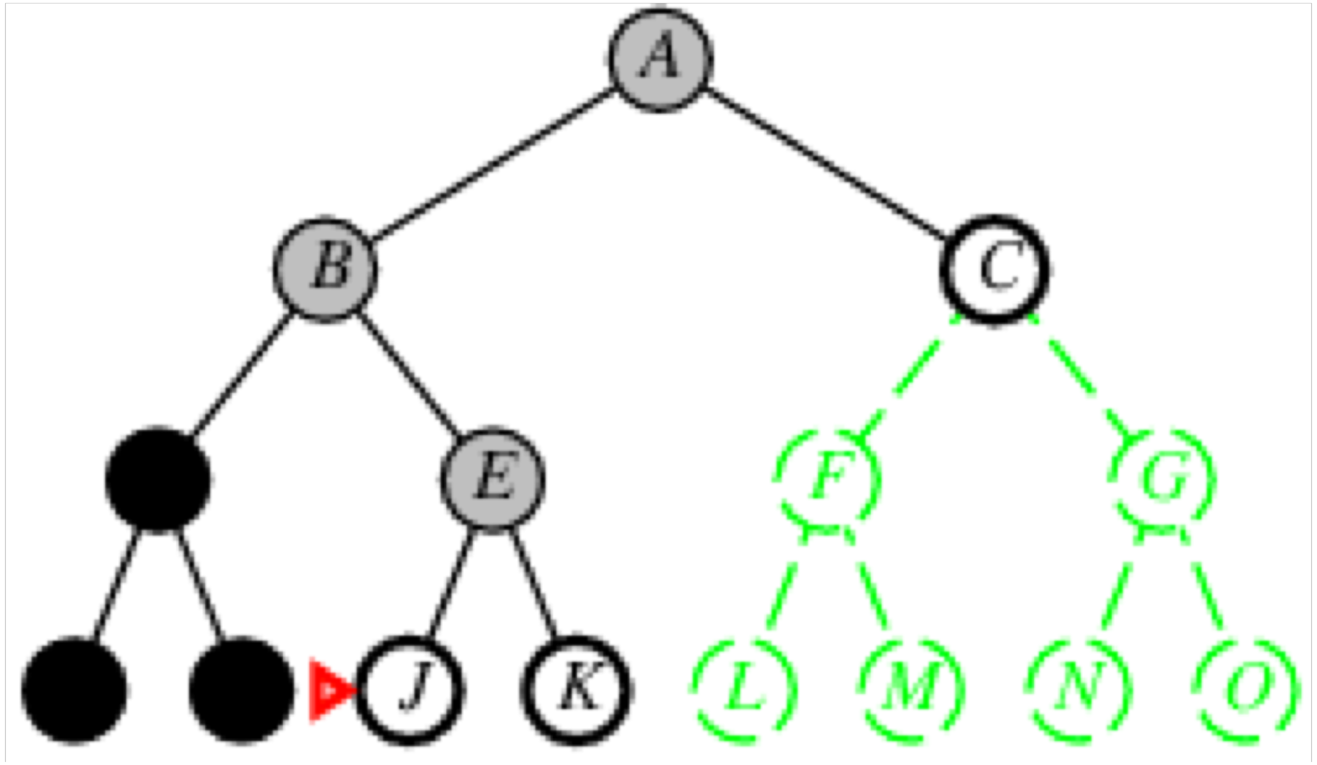


Depth-first Search (DFS)

Strategy: Expand deepest unexpanded node

Implementation:

Fringe = last-in first-out (LIFO), i.e., unvisited successors at front (F is a stack)

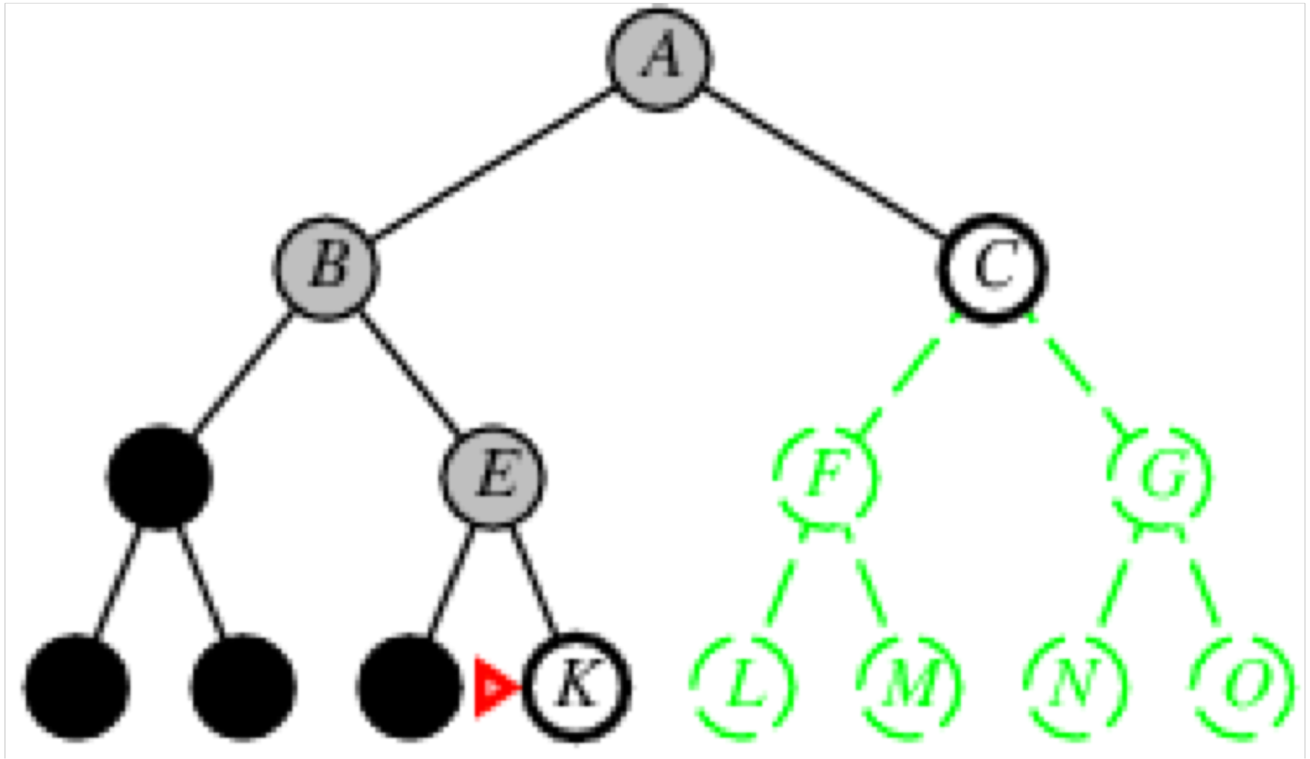


Depth-first Search (DFS)

Strategy: Expand deepest unexpanded node

Implementation:

Fringe = last-in first-out (LIFO), i.e., unvisited successors at front (F is a stack)



Properties of Depth-first Search (DFS)

Complete?	No. Fails in infinite-depth spaces (space with loops). Modify to avoid repeated states can make it finite.
Time?	$O(b^m)$: terrible if m is much larger than d.
Space?	O(bm) i.e., linear in space !!!
Optimal?	No

Time and space complexity are measured in terms of:

- *b* – maximum branching factor of the search tree
- *d* – depth of the least-cost solution
- *m* – maximum depth of the state space (may be ∞)

DFS Summary

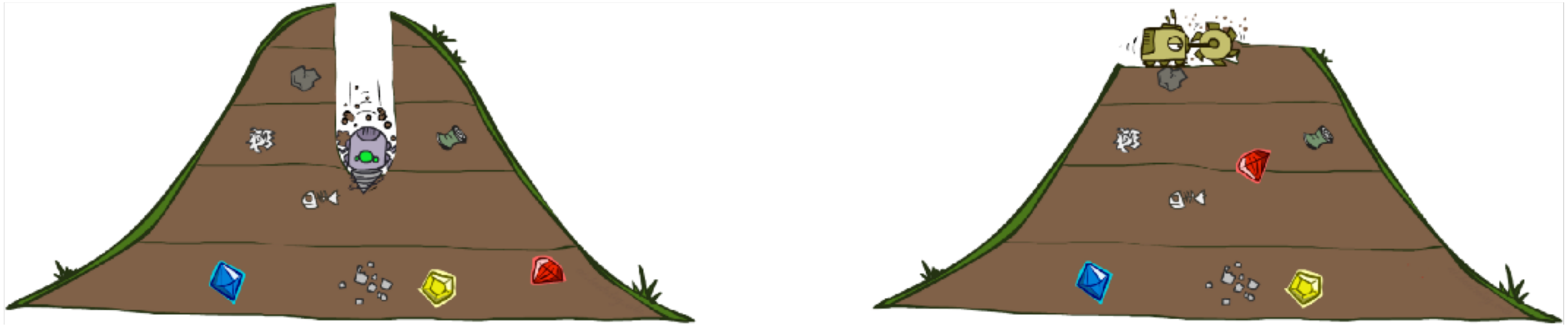
Behavior

- Expands the deepest node in the tree
- Backtracks when reaches a non-goal node with no descendants

Problems:

- Makes a wrong choice and can go down an infinite path even though the solution may be very close to initial vertex
- DFS is not optimal
- If subtree is of unbounded depth and contains no solutions, DFS will never terminate.
- Hence, DFS is not complete (in general)
- Let b be the maximum number of successors of any node, d be the depth of the shallowest goal, and m be the maximum length of any path in the search tree
- Time complexity is $O(b^m)$ and space complexity is $O(b \cdot m)$

Comparing BFS and DFS



When will BFS outperform DFS?

When will DFS outperform BFS?

Depth-limited Search (DLS)

- One problem with DFS is presence of infinite paths.

- DLS limits the depth of a path in the search tree of DFS.
  - Modifies DFS by using a predetermined depth limit of d_l.
  - DLS is incomplete if the shallowest goal is beyond the depth limit d_l.
  - DLS is not optimal if $d < d_l$
  - Time complexity is now $O(b^{dl})$ and space complexity is $O(b \cdot d_l)$

Iterative Deepening Search (IDS)

- Finds the best depth limit by incrementing dl until goal is found at $d_l = d$.
- Can be viewed as running DLS consecutive values of d_l
- IDS combines the benefits of both DFS and BFS
- Like DFS, its space complexity is $O\ (b \cdot d)$
- Like BFS, it is complete when the branching factor is finite, and is optimal if the path cost is a non-decreasing function of the depth of the goal node
- Its time complexity is $O(b^d)$
- IDS is the preferred uninformed search when the state space is large, and the depth of the solution is not known.

Summary of Uninformed Search Algorithms

Criterion	BFS	DFS	DLS	IDS
Complete?	Yes	No	Yes if dl ≥ d	Yes
Time	b^{d+1}	b^m	b^{dl}	b^d
Space	b^{d+1}	bm	bd_l	bd
Optimal?	Yes*	No	No	Yes*