



Chapter 1
Search

Paragraph 1
Blind Search

Problem Solving in Fully Observable Static Environments

- Example:
 - You have a row boat and need to cross a river with your dog, goat, and salad. Unfortunately, apart from yourself only one more item/animal fits into the boat. For obvious reasons, you cannot leave the dog and the goat unattended - or the goat and the salad, for that matter. What do you do?

My Dog, My Goat, and My Salad

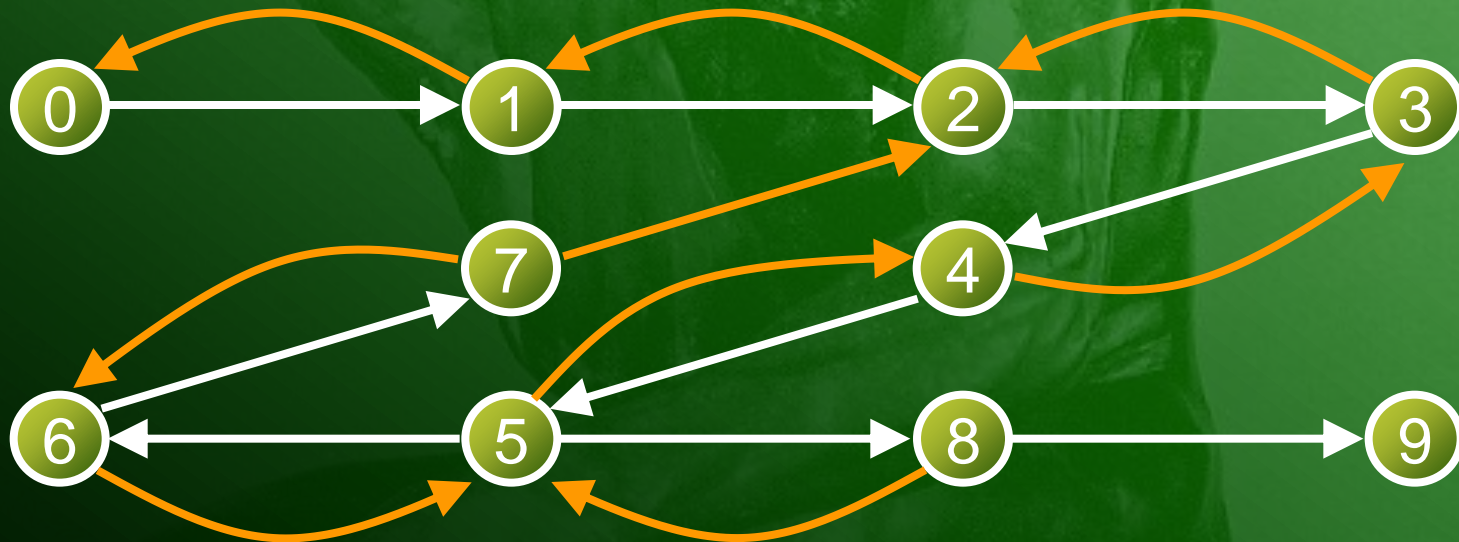


Search Problems

- Given are:
 - a set of (legal) states
 - an initial state (or start state)
 - a successor function
 - a set of goal states (or a goal test)
 - a step cost function (or path cost function)
- The task:
 - Find a path from the start state to one of the goal states (at minimal cost).

Search Strategies

- Backtracking (BT)
 - If the current state is a goal state, terminate. Otherwise choose the next successor of the current state that has not an ancestor. If no such state exists, backtrack and try from the next earlier state. If all reachable states are exhausted, terminate.



Search Strategies

- Questions that arise:
 - **Completeness:** Will BT always terminate and find a path to a goal state if one exists?
 - **Optimality:** Will BT find a minimum cost path?
 - **Time Complexity:** How much time will BT take?
 - **Space Complexity:** How much memory do we need to perform BT?

Backtracking

- If the state space is finite, BT is bound to find a solution. However, if the first successor was chosen poorly, BT may get stuck in a very long, or even infinite path. Therefore, BT is complete only when step sequences are bounded in length, and it is not optimal.
- While the space complexity is linear in the length of the path, the time complexity may grow exponentially.

Backtracking

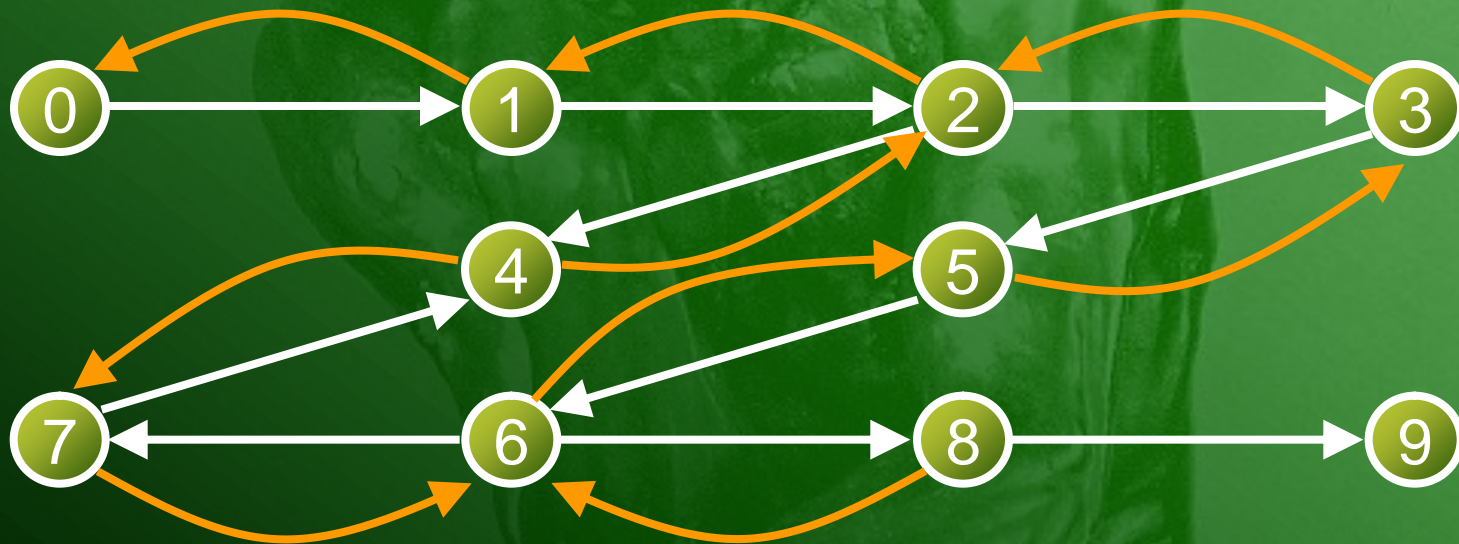
- Completeness: When finite depth
- Optimality: No ☹️
- Time Complexity: $O(b^m)$ ☹️
- Space Complexity: $O(m)$ 😊

b - is the branching factor

m - the maximum depth of the tree

Depth First Search (DFS)

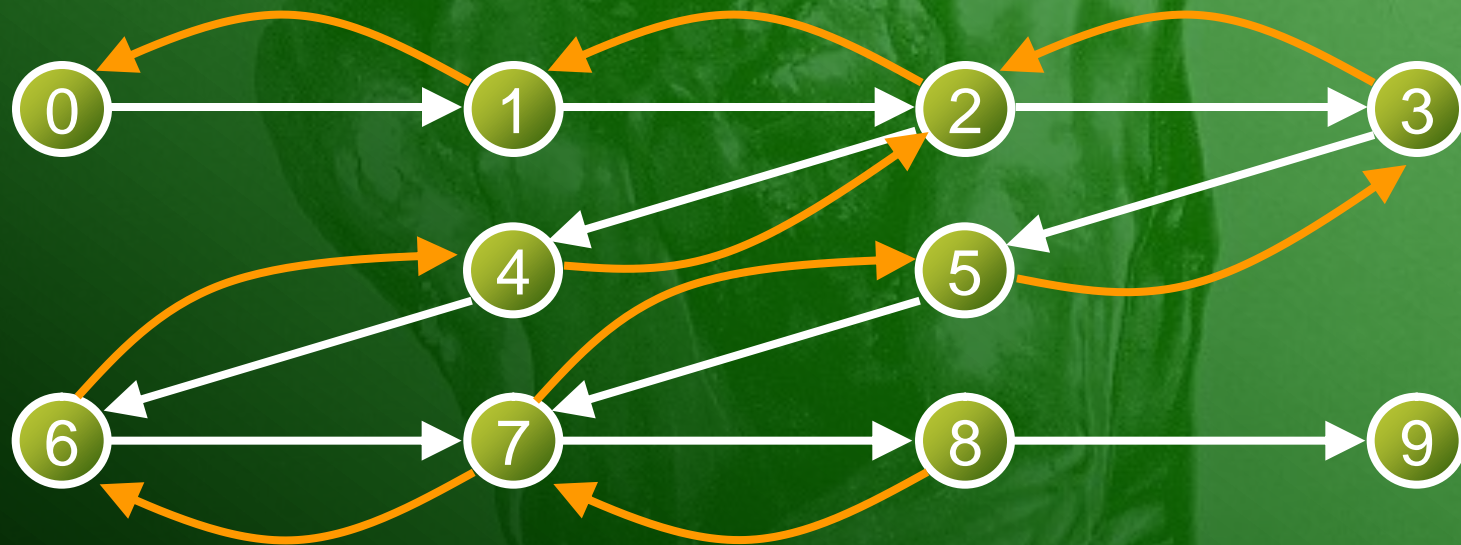
- Works like BT, but DFS adds all successors to a stack before moving on. Therefore, it inherits all properties of BT, albeit at a larger space complexity of $O(bm)$.



[The node numbers given denote when a node first enters the fringe. If we had denoted the time stamp when a node is being expanded, we would have seen the same sequence of **DFS numbers** as in BT.]

Breadth First Search (BFS)

- Breadth First Search expands the first node, then all its successors, then all their successors, and so on.



Breadth First Search

- BFS is complete when the number of alternative steps that can be taken (the branching factor) is finite. In case of uniform costs, BFS is even optimal.
- The time and space complexity can be huge, though:
 - Total number of nodes generated can be as large as
 - $b + b^2 + b^3 + \dots + b^d + b^{d+1} - b = \sum_{i=1..d+1} b^i - b$
 $= (b^{d+2} - b)/(b-1) - b$
 $= O(b^{d+1})$

Breadth First Search

- Completeness: When finite breadth
- Optimality: When uniform costs
- Time Complexity: $O(b^{d+1})$ ☹️
- Space Complexity: $O(b^d)$ ☹️

b - is the branching factor

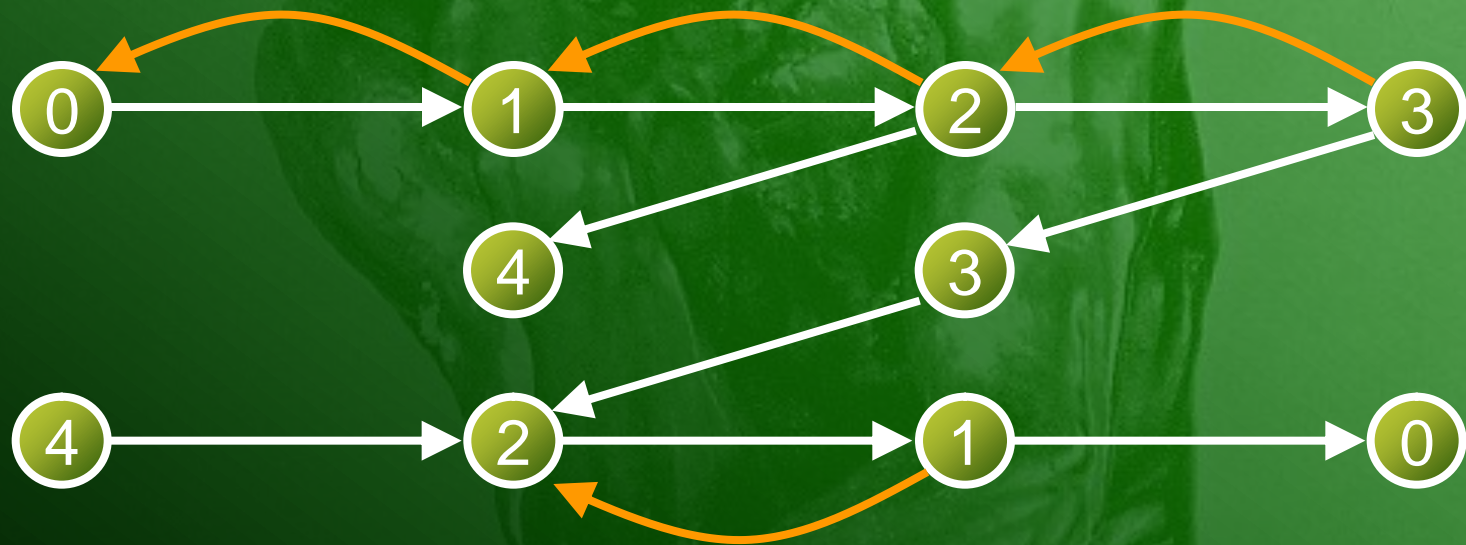
d - the maximum depth of a goal node

Bidirectional Search

- When
 1. the goal states can be found quickly, and
 2. a predecessors function is available

we can speed up BFS by starting two searches: one starts at the initial state, the other with a fringe consisting of all goal states.

Bidirectional Search



Bidirectional Search

- Completeness: When finite breadth
- Optimality: When uniform costs
- Time Complexity: $O(b^{d/2})$ ☺
- Space Complexity: $O(b^{d/2})$ ☹

- b - is the branching factor
- d - the maximum depth of a goal node

Iterative Deepening

- While BFS is in most cases complete and sometimes even optimal, its space requirements are very often prohibitively large.
- DFS on the other hand suffers from incompleteness in scenarios where sequences of steps can be arbitrarily long.
- We can rectify this shortcoming by introducing an increasing parameter that limits the maximum depth allowed for DFS.

Iterative Deepening

Depth Limit 0

0

Iterative Deepening

Depth Limit 1



Iterative Deepening

Depth Limit 2



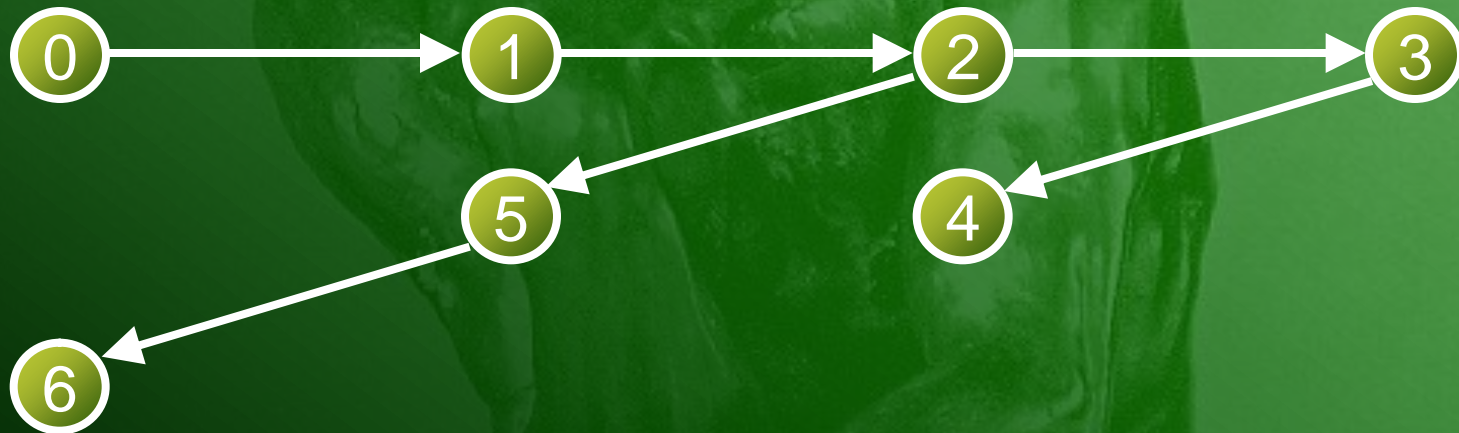
Iterative Deepening

Depth Limit 3



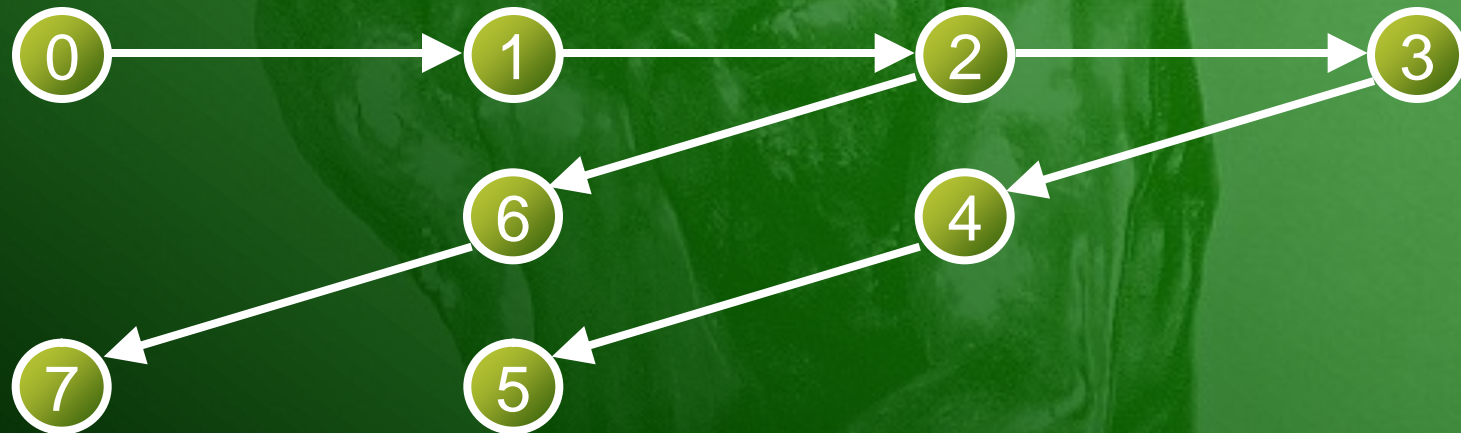
Iterative Deepening

Depth Limit 4



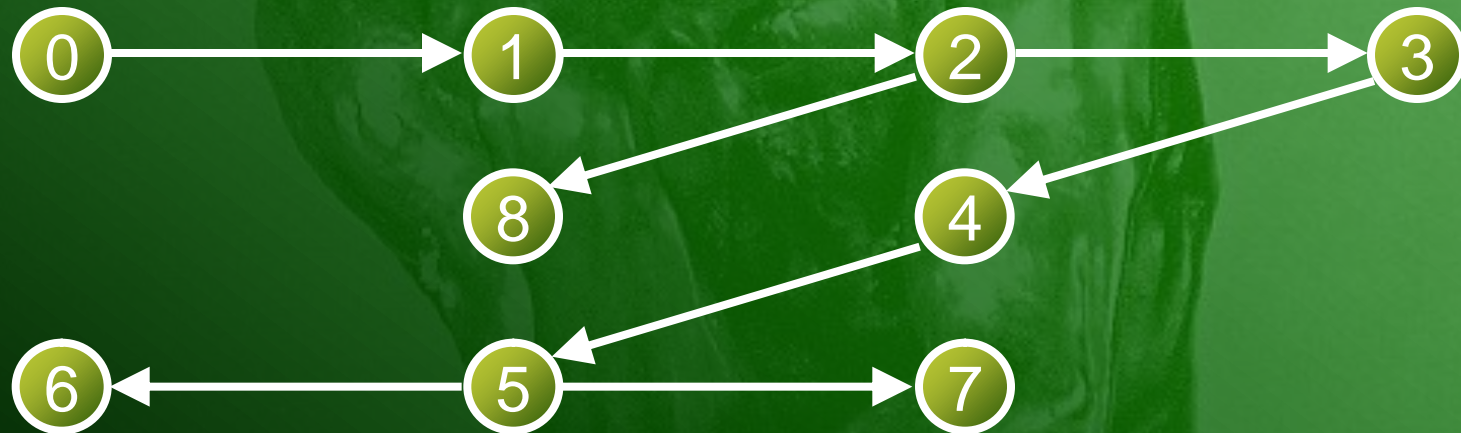
Iterative Deepening

Depth Limit 5



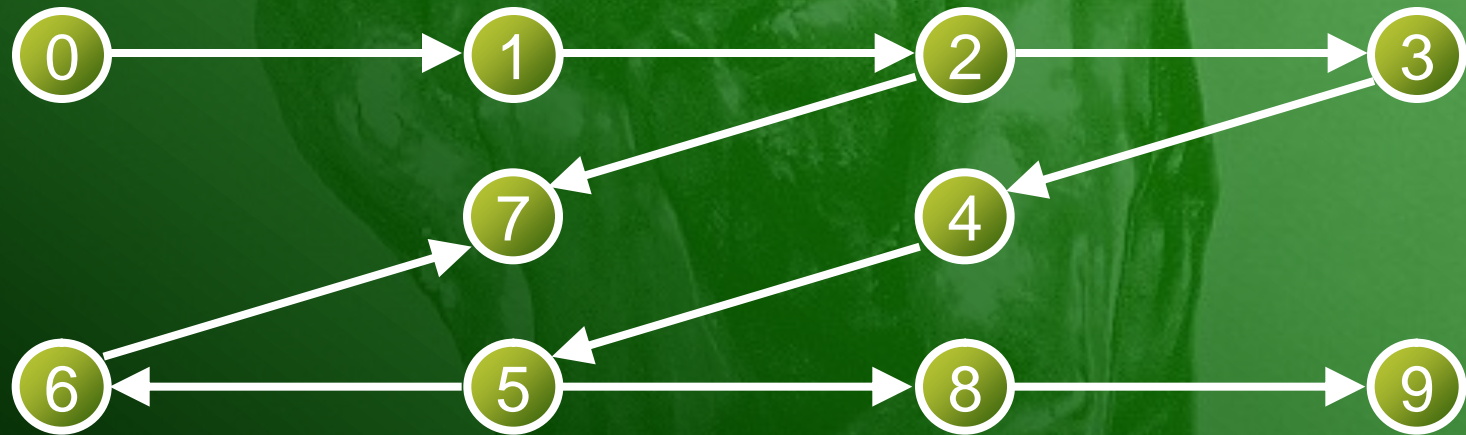
Iterative Deepening

Depth Limit 6



Iterative Deepening

Depth Limit 7



Iterative Deepening

- Completeness: When finite breadth
- Optimality: When uniform costs
- Time Complexity: $O(b^d)$
- Space Complexity: $O(bd)$ ☺

b - is the branching factor

d - the maximum depth of a goal node

Summary

Criterion	BFS	Bidir. Search	DFS	Iterative Deepening
Complete	when $b < \infty$	when $b < \infty$	No	when $b < \infty$
Optimal	when costs uniform	when costs uniform	No	when costs uniform
Time	$O(b^{d+1})$	$O(b^{d/2})$	$O(b^m)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{d/2})$	$O(bm)$	$O(bd)$



The End