Uninformed Search
000000000

BFS
00000

DFS
0000000000

# CS540 Introduction to Artificial Intelligence
## Lecture 17

Young Wu

Based on lecture slides by Jerry Zhu, Yingyu Liang, and Charles Dyer

August 2, 2022

Uninformed Search
●○○○○○○○○

BFS
○○○○○

DFS
○○○○○○○○○○○

# Learning vs Search
## Motivation

- In reinforcement learning, the reward and state transition need to be learned by taking actions.
- In search problems, the reward and state transitions are given.
- The problem is to find a sequence of actions that lead to the goal with minimum cost.

Uninformed Search
○●○○○○○○○

BFS
○○○○○

DFS
○○○○○○○○○○○

# Search Problem

Motivation

- State space $S$ is the set of all valid configurations.
- Initial states $I$ and goal states $G$ are subsets of $S$.
- Successor function $s'(s)$ given the current state $s$ is the set of states reachable in one step from $s$.
- There is a cost (or negative reward) associated with moving from $s$ to $s'(s)$.
- The search problem is the problem of finding a solution path from a state in $I$ to a state in $G$, usually with minimum total cost.

Uninformed Search
○○●○○○○○○

BFS
○○○○○

DFS
○○○○○○○○○○○

# State Space
## Motivation

- The states need to represent all necessary information about the game.
- The actions are discrete and deterministic and are determined by the successor function.
- Each possible action at state $s$ is associated with a state in the set $s'(s)$.

Uninformed Search
○○○●○○○○○

BFS
○○○○○

DFS
○○○○○○○○○○○

# Sizes of State Space

## Motivation

- Tic Tac Toe: $10^3$
- Checkers: $10^{20}$
- Chess: $10^{50}$
- Go: $10^{170}$

Uninformed Search
0000●0000

BFS
00000

DFS
0000000000

# State Space Graph
Definition

- A state space can be represented by a weighted directed graph $(V, E, c)$.
- $V$ is the set of vertices (also called nodes).
- $E$ is the set of edges (also called arcs). Each edge is directed from one vertex to another vertex and represents an action.
- $c$ is the cost (also called weights) associated with each edge. The costs are positive.

Uninformed Search
oooooo●ooo

BFS
ooooo

DFS
oooooooooo

# Search Problem on Graph

Definition

- Search starts at an initial state and finishes if one of the goal states is reached.

- The solution is a path in the graph from an initial state to a goal state.

- The cost of a solution is the sum of edge costs on the solution path.

- The optimal solution is the solution with the lowest cost.

# Expansion
### Definition

- Vertices that are explored so far are stored in a tree called the state space search tree.
- Expanding a vertex means to generate all successor vertices and add them (and the associated edges) to the state space search tree.
- The leaves of the search tree are unexpanded and are called the frontier (sometimes called the fringe).
- The search strategies differ in the order in which the vertices are expanded.

# Performance
Definition

- A search strategy is complete if it finds at least one solution.
- A search strategy is optimal if it finds the optimal solution.
- For uninformed search, the costs are assumed to be 1 for all edges $c = 1$.

Uninformed Search
○○○○○○○○○●

BFS
○○○○○

DFS
○○○○○○○○○○

# Complexity
### Definition

- The time complexity of a search strategy is the worst case maximum number of vertices expanded.

- The space complexity of a search strategy is the worst case maximum number of states stored in the frontier at a single time.

- Notation: the goals are $d$ edges away from the initial state. This means assuming a constant cost of 1, the optimal solution has cost $d$. The maximum depth of the graph is $D$.

- Notation: the branching factor is $b$, the maximum number of actions associated with a state.

$$b = \max_{s \in V} |s'(s)|$$

Uninformed Search
000000000

BFS
●0000

DFS
0000000000

# Breadth First Search

Description

- Use Queue (FIFO) for the frontier.
- Remove from the front, add to the back.

Uninformed Search
○○○○○○○○○

BFS
○●○○○○

DFS
○○○○○○○○○○

# Breadth First Search

### Algorithm

- Input: a weighted digraph $(V, E, c)$, initial states $I$ and goal states $G$.

- Output: a path from $I$ to $G$.

- EnQueue initial states.

$$Q = I$$

- While $Q$ is not empty and goal is not deQueued, deQueue $Q$ and enQueue its successors.

$$s = Q_0$$
$$Q = Q + s'(s)$$

Uninformed Search
000000000

BFS
00●00

DFS
0000000000

# Breadth First Search Performance

Discussion

- BFS is complete.
- BFS is optimal with $c = 1$.

Uninformed Search
000000000

BFS
000●0

DFS
0000000000

# Breadth First Search Complexity

Discussion

- Time complexity: the worst case occurs when the goal is the last vertex at depth $d$.

$$T = b + b^2 + ... + b^d$$

- Space complexity: the worst case is storing all vertices at depth $d$ is in the frontier.

$$S = b^d$$

Uninformed Search
000000000

BFS
0000●

DFS
0000000000

# BiDirectional Search

Discussion

- BFS from the initial states and goal states at the same time.

- The search stops when the two frontiers meet (have non-empty intersection) in the middle.

- The time and space complexity is the same as BFS with depth $\dfrac{d}{2}$ .

# Depth First Search
### Description

- Use Stack (LIFO) for the frontier.
- Remove from the front, add to the front.

Uninformed Search
000000000

BFS
00000

DFS
0●00000000

# Depth First Search

### Algorithm

- Input: a weighted digraph $(V, E, c)$, initial states $I$ and goal states $G$.
- Output: a path from $I$ to $G$.
- Push initial states.

$$S = I$$

- While $S$ is not empty and goal is not popped, pop $S$ and push its successors.

$$s = S_0$$
$$S = s'(s) + S$$

Uninformed Search
○○○○○○○○○

BFS
○○○○○

DFS
○○●○○○○○○○○

# Depth First Search Performance

Discussion

- DFS is incomplete if $D = \infty$.
- DFS is not optimal.

Uninformed Search
000000000

BFS
00000

DFS
0000000000

# Depth First Search Complexity
Discussion

- Time complexity: the worst case occurs when the goal is the root of the last subtree expanded in the whole graph.

$$T = b^{D-d+1}... + b^{D-1} + b^D$$

- Space complexity: the worst case is storing all vertices sharing the parents with vertices in the current path.

$$S = (b - 1) D + 1$$

# Iterative Deepening Search

Description

- DFS but stop if path length $> 1$
- repeat DFS but stop if path length $> 2$
- ...
- repeat DFS but stop if path length $> d$

Uninformed Search
○○○○○○○○○

BFS
○○○○○

DFS
○○○○○●○○○○

# Iterative Deepening Search
### Algorithm

- Input: a weighted digraph $(V, E, c)$, initial states $I$ and goal states $G$.
- Output: a path from $I$ to $G$.
- Perform DFS on the digraph restricted to vertices with depth $\leqslant 1$ from the initial state.
- Perform DFS on the digraph restricted to vertices with depth $\leqslant 2$ from the initial state.
- Repeat until the goal is deQueued.

Uninformed Search
○○○○○○○○○

BFS
○○○○○

DFS
○○○○○○○●○○○

# Iterative Deepening Search Performance
Discussion

- IDS is complete.
- IDS is optimal with $c = 1$.

Uninformed Search
○○○○○○○○○

BFS
○○○○○

DFS
○○○○○○○○●○○

# Iterative Deepening Search Complexity
### Discussion

- Time complexity: the worst case occurs when the goal is the last vertex at depth $d$.

$$T = db + (d - 1) b^2 + ... + 3b^{d-2} + 2b^{d-1} + 1b^d$$

- Space complexity: it has the same space complexity as DFS.

$$S = (b - 1) d$$

Uninformed Search
○○○○○○○○○

BFS
○○○○○

DFS
○○○○○○○○○○●○

# Non-Tree Search
Discussion

- If the state space is not a tree, search strategies need to remember the states that are already expanded.
- A vertex should be removed from the frontier if it is already expanded.

Uninformed Search
000000000

BFS
00000

DFS
0000000000●

# Uniformed vs Informed Search

Discussion

- Uninformed search means only the goal $G$ and the successor function $s'$ are given.
- Informed search means which non-goal states are better is also known.
- Usually, iterative deepening is used for uninformed search.