CS540 Introduction to Artificial Intelligence
Lecture 15

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

July 4, 2020

Lecture Indices

Admin

- Lecture 13 will be recorded and posted after the final exam.
- Lecture 14 is a midterm review so it will be skipped.
- Lecture 15 is this lecture.
- Pre-recorded lectures are official lectures for Summer 2020 and covers all relevant materials for homework and exams.
- BBCU lectures are only summaries, reviews, additional examples, and discussions.
- Quizzes are alternatives for putting all the weights on the exams, they are not mandatory, just a way to encourage interactions and discussions.

40%

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.

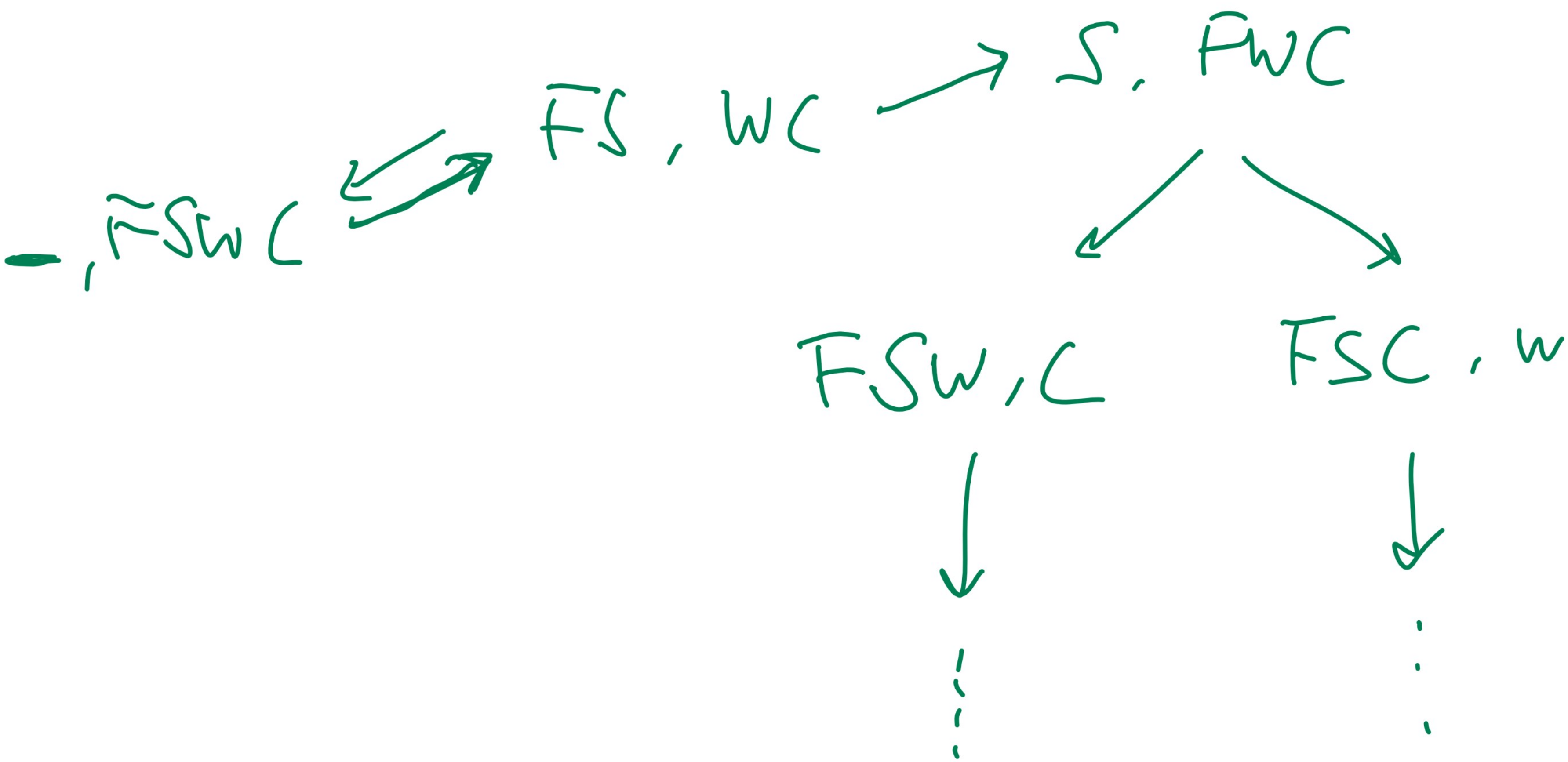
Search Problem Applications

Motivation

- Puzzles and games.
- Navigation: route finding. Google Map
- Motion planning.
- Scheduling.

Wolf, Sheep, Cabbage Example

Motivation



Search Problem

Motivation

_, FSWC

FS, WC

S, FWC

cost = 1

- State space S is the set of all valid configurations.
- Initial states I and goal states G are subsets of S. FSWC, _
- 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.

_, FSWC

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 $s'(s)$.

8 Puzzle Example

Motivation

Sizes of State Space

Motivation

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

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.

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.

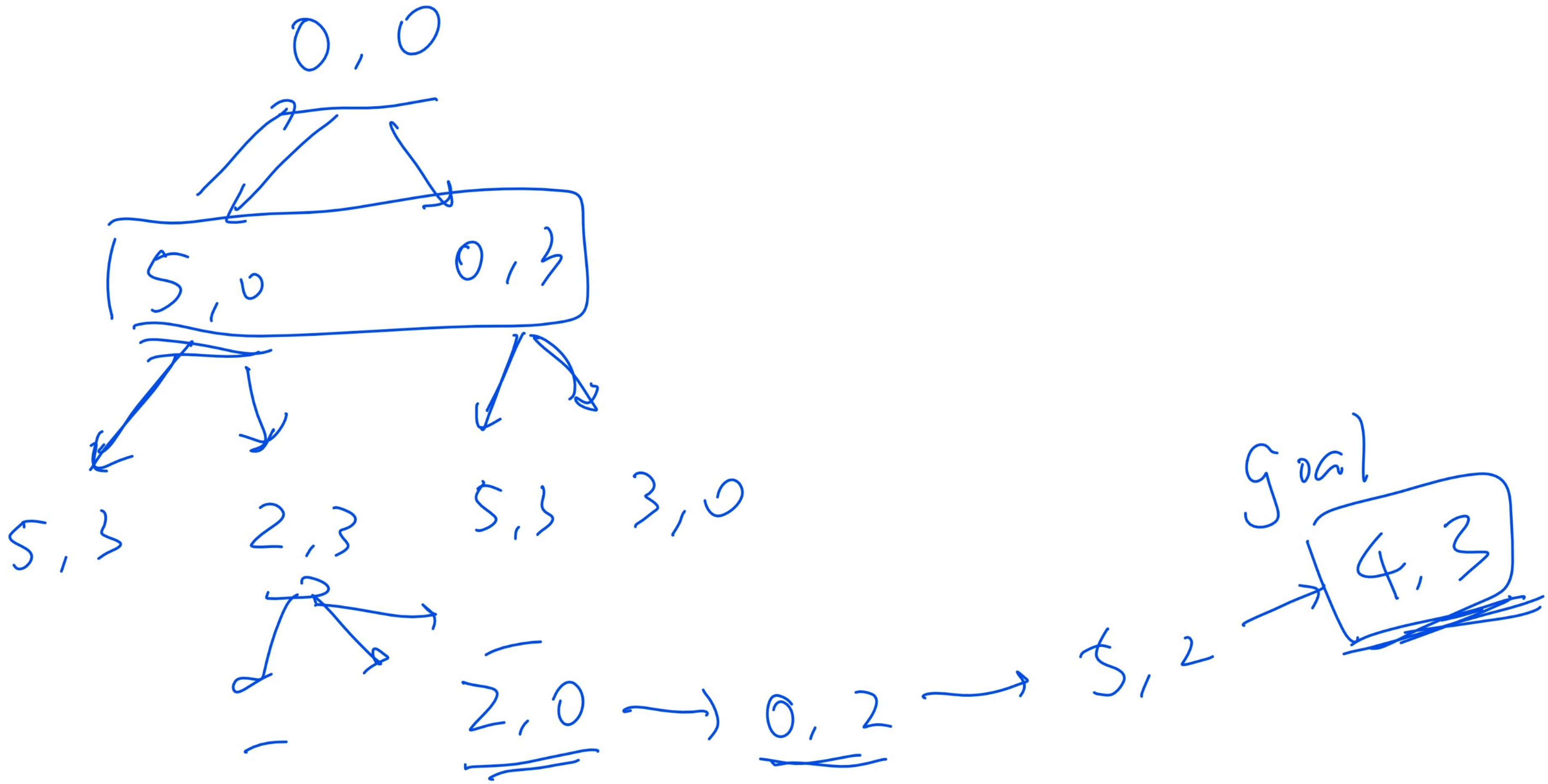
Expansion

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.

Water Jugs Example

Definition



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 $c = 1$.

Complexity

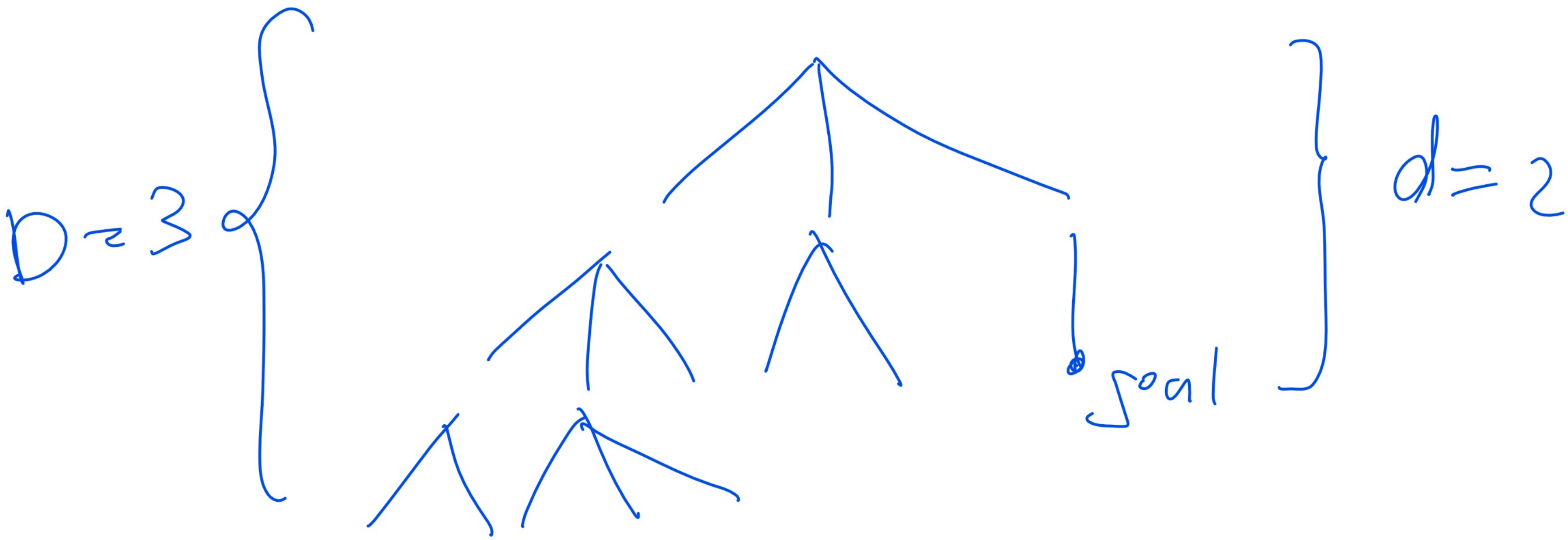
Definition

- 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.

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

Search Tree Diagram

Definition



$b = \max\{3, 2, 1\}$
$= 3$

Breadth First Search

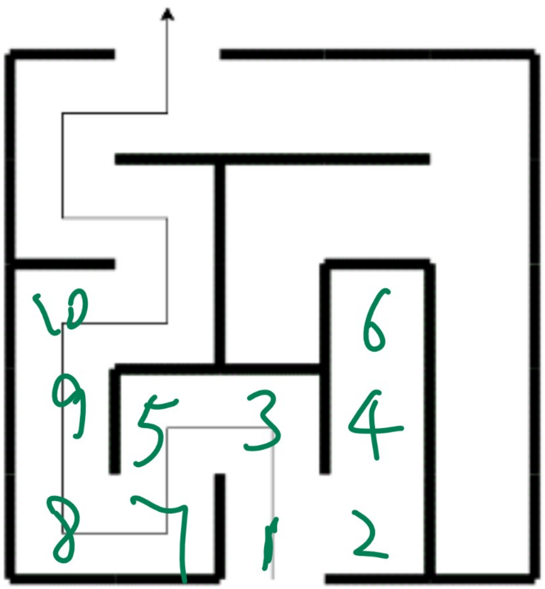
Description

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

Maze BFS Example

Definition

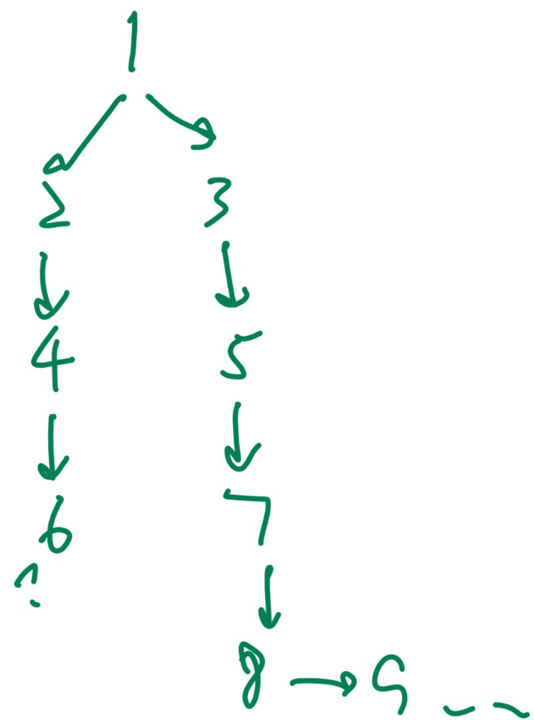
P5



Queue }

~~1~~ ~~2~~, ~~3~~, 4, ~~5~~, ~~6~~, ~~7~~

expand



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.

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

Breadth First Search Performance

Discussion

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

Breadth First Search Complexity

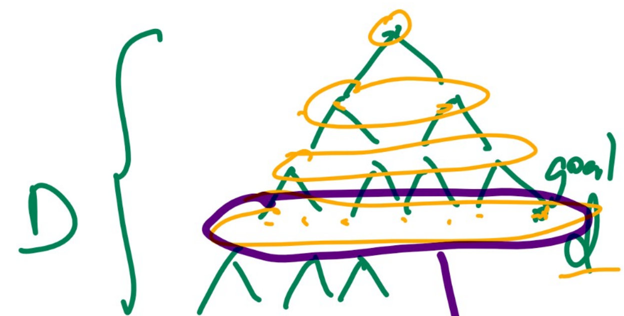
Discussion

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

1+

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

$O(b^d)$



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

b^d

$$S = b^d$$

$O(b^d)$

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 $\frac{d}{2}$.

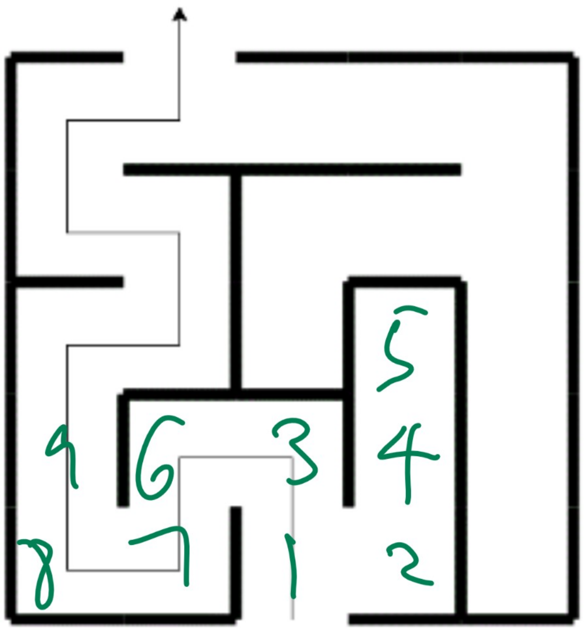
Depth First Search

Description

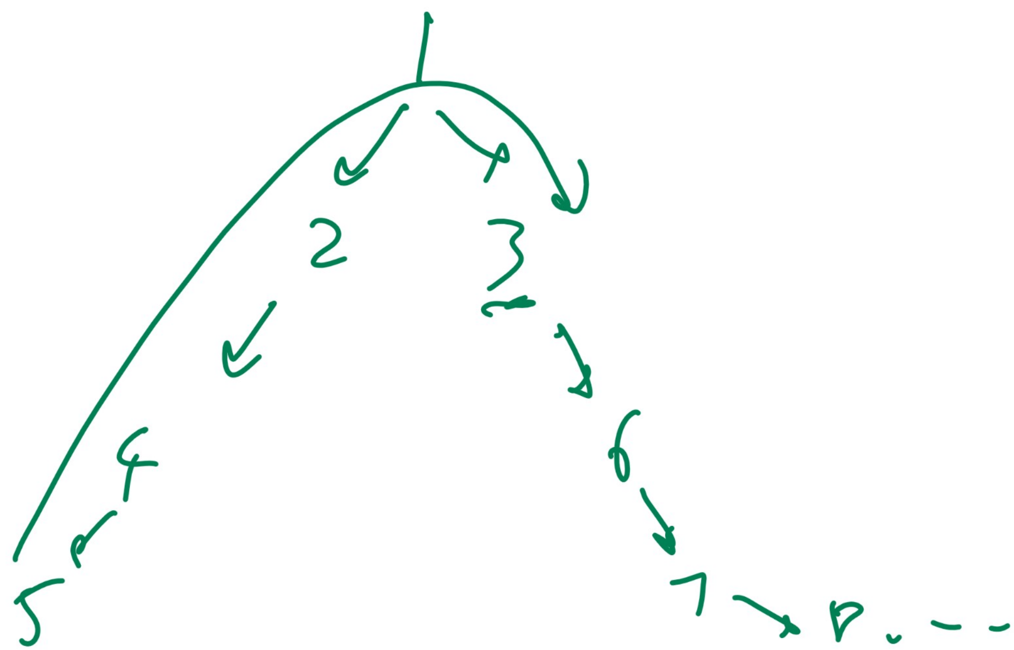
- Use Stack (LIFO) for the frontier.
- Remove from the front, add to the front.

Maze DFS Example

Definition



7 ~~6~~ ~~5~~ ~~4~~ ~~2~~, ~~3~~ ~~1~~



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.

P5

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

Depth First Search Performance

Discussion

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

Depth First Search Complexity

Discussion

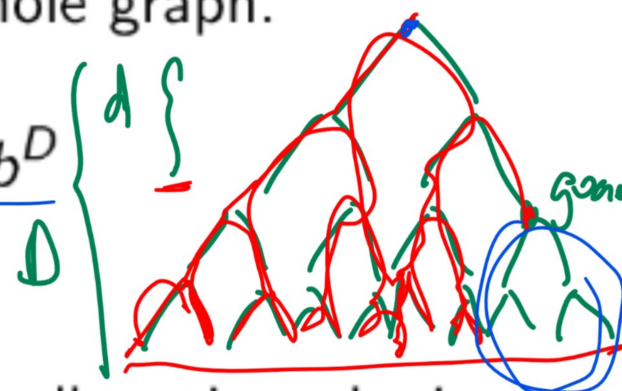
$1 + b^2 + b^3 + \dots b^D$

$- b^2 + b^3 + \dots b^{D-d}$

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

BFS

$$T = 1 + \underbrace{b^{D-d+1} \dots + b^{D-1} + b^D}_{O(b^D)}$$

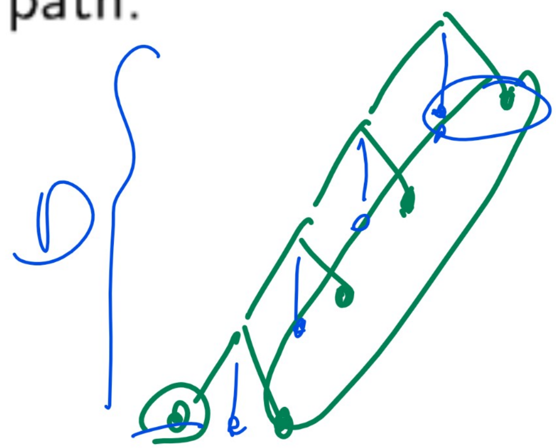


- Space complexity: the worst case is storing all vertices sharing the parents with vertices in the current path.

DFS

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

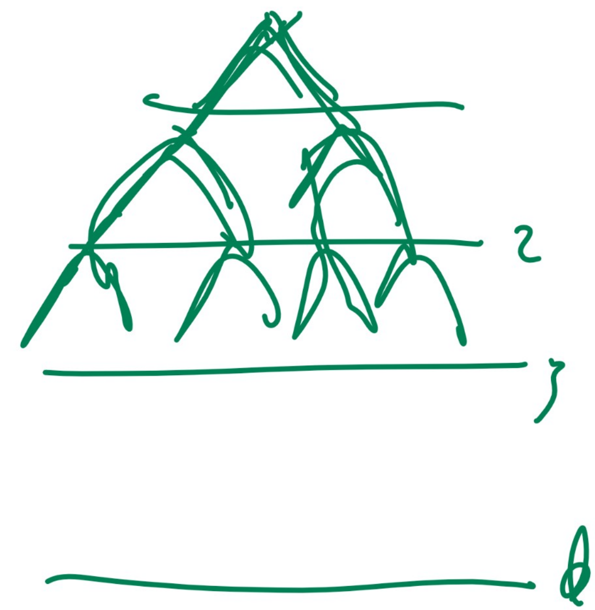
$O(bD)$



Iterative Deepening Search

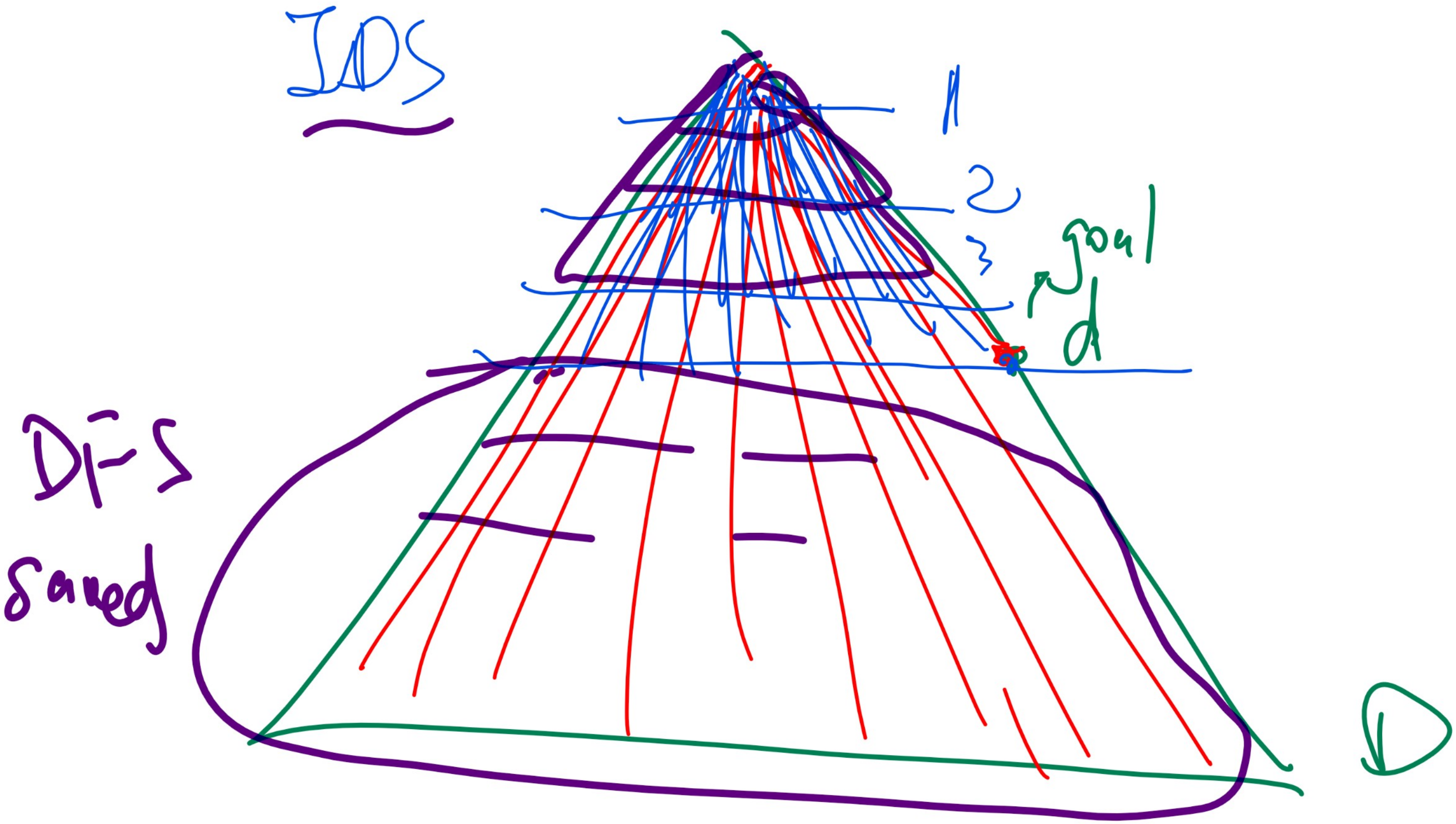
Description

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



Maze IDS Example

Definition



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.

Iterative Deepening Search Performance

Discussion

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

Iterative Deepening Search Complexity

Discussion

limit depth

1 b

2 $b + b^2$

3 $b + b^2 + b^3$

d $b + b^2 + b^3 + \ldots b^d$

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

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

$< b^d$ $\le b^d$

BFS $\rightarrow$ $O(b^d)$

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

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

$1\times$ d

DFS $\rightarrow$

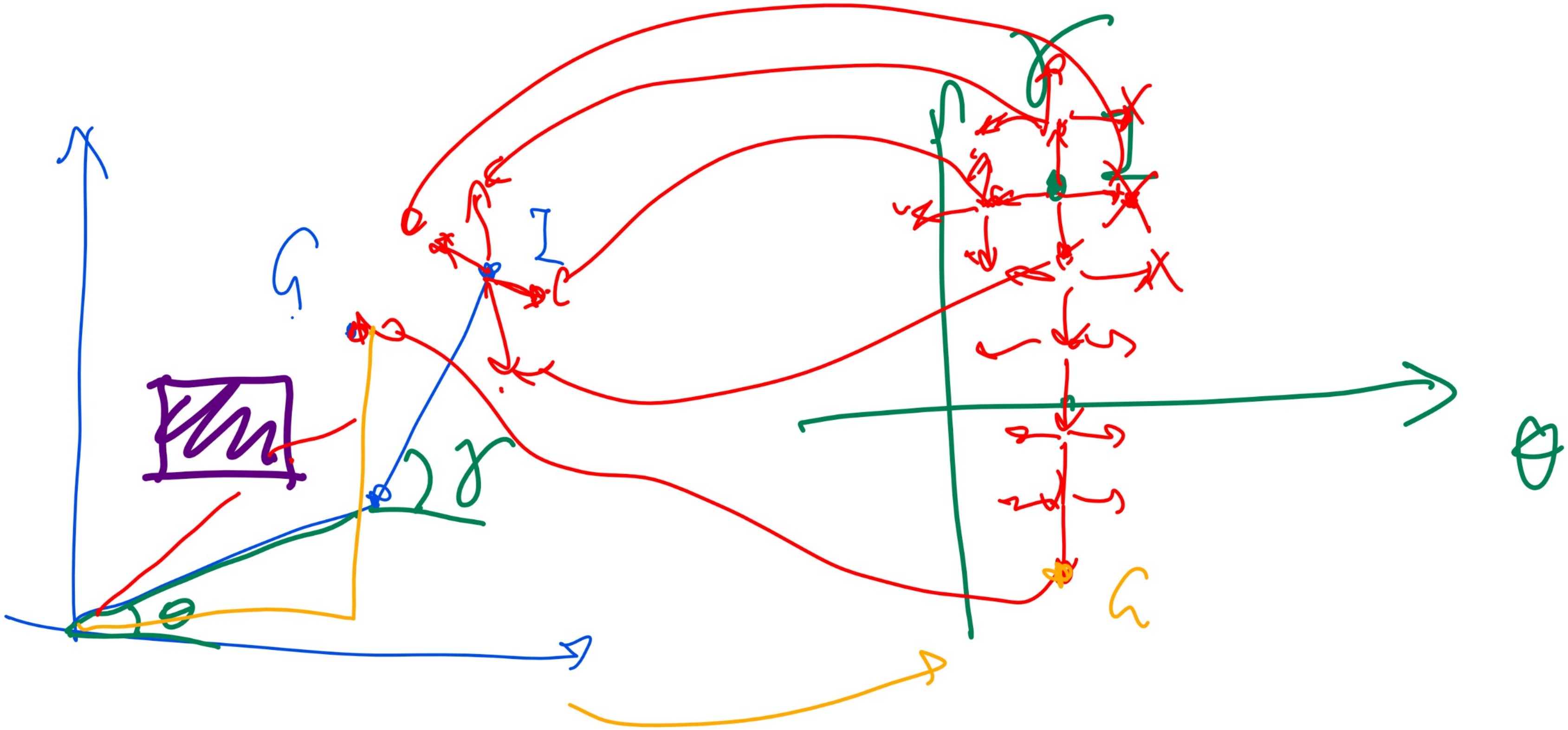
Non-Tree Search

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.

Configuration Space

Discussion



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.