INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

Dharwad

ज्ञानेन विकासः

Breadth-First Search and Depth-First Search

Dr. Animesh Chaturvedi

Assistant Professor: IIIT Dharwad

Young Researcher: Heidelberg Laureate Forum

Postdoc: King's College London & The Alan Turing Institute

PhD: IIT Indore MTech: IIITDM Jabalpur

HEIDELBERG LAUREATE FORUM

KING'S College LONDON

Indian Institute of Technology Indore

भारतीय प्रौद्योगिकी संस्थान इंदौर

The Alan Turing Institute

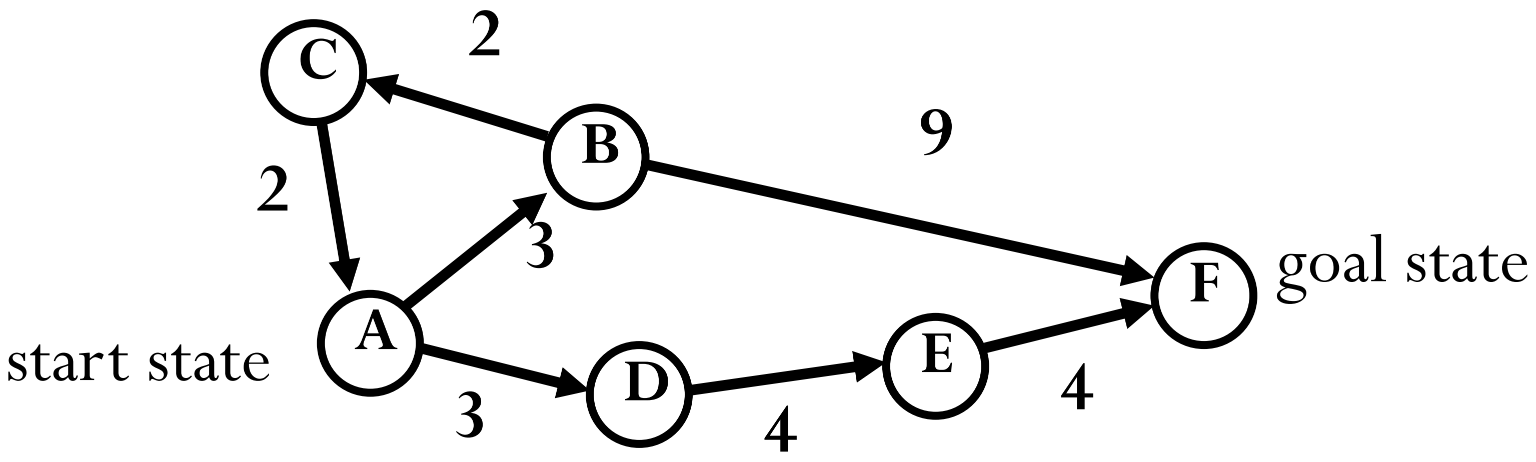
PDPM

Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

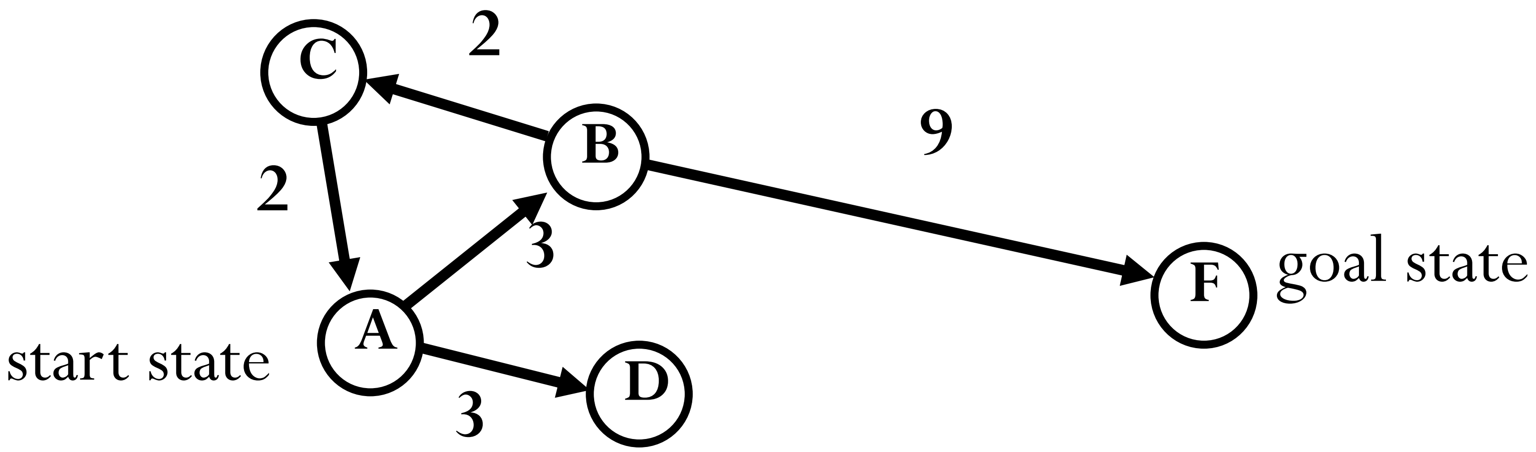
Search

- We have some actions that can change the state of the world
  - Change induced by an action perfectly predictable
- Try to come up with a sequence of actions that will lead us to a goal state
  - May want to minimize number of actions
  - More generally, may want to minimize total cost of actions
- Do not need to execute actions in real life while searching for solution!
  - Everything perfectly predictable anyway

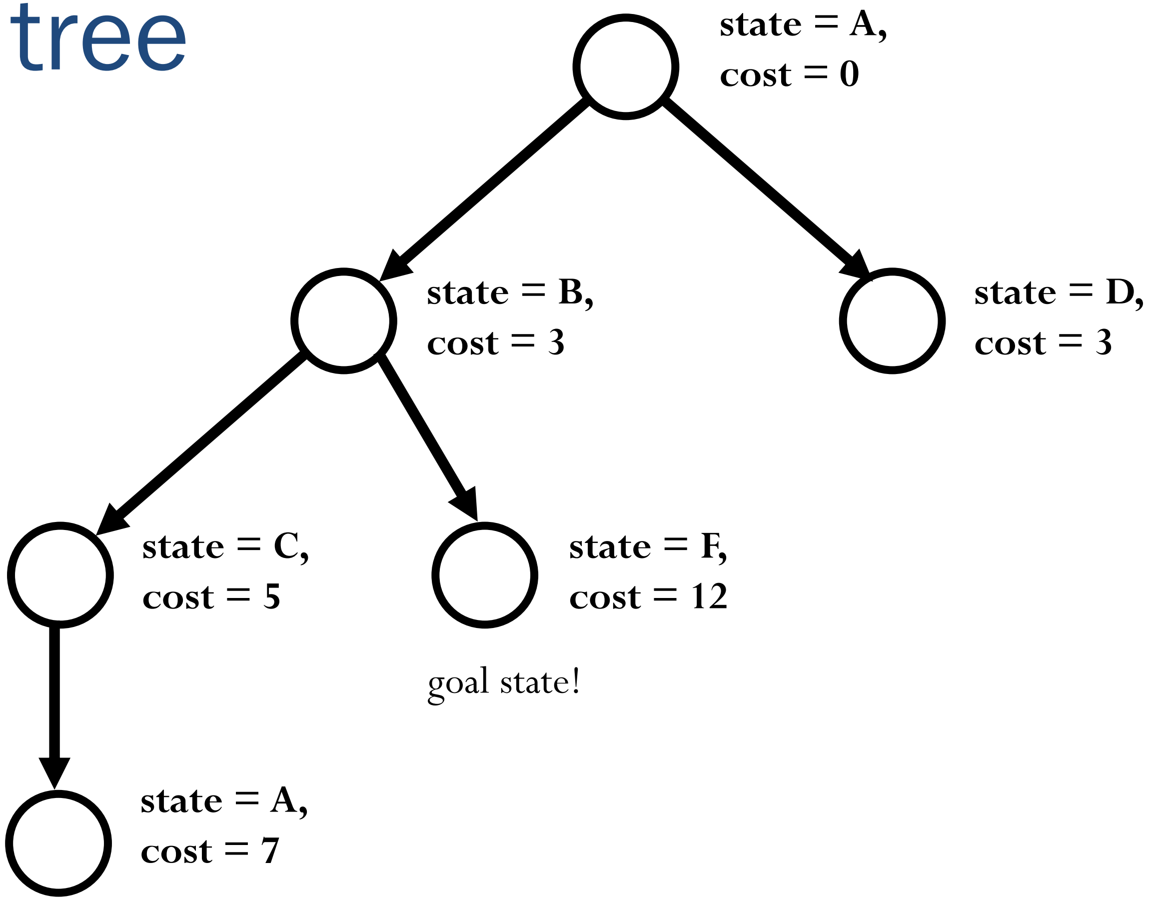
A simple example: traveling on a graph



Searching for a solution

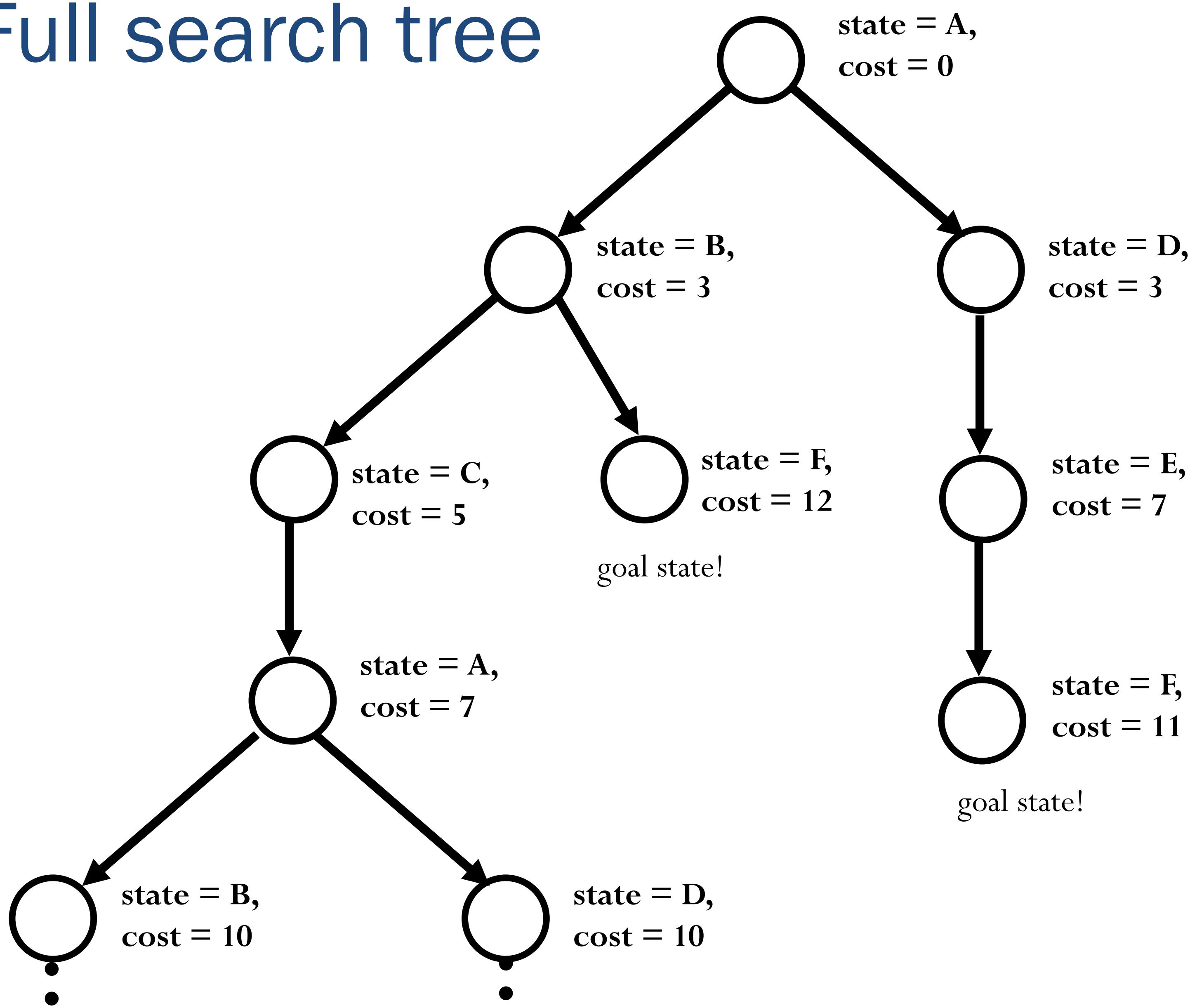


Search tree

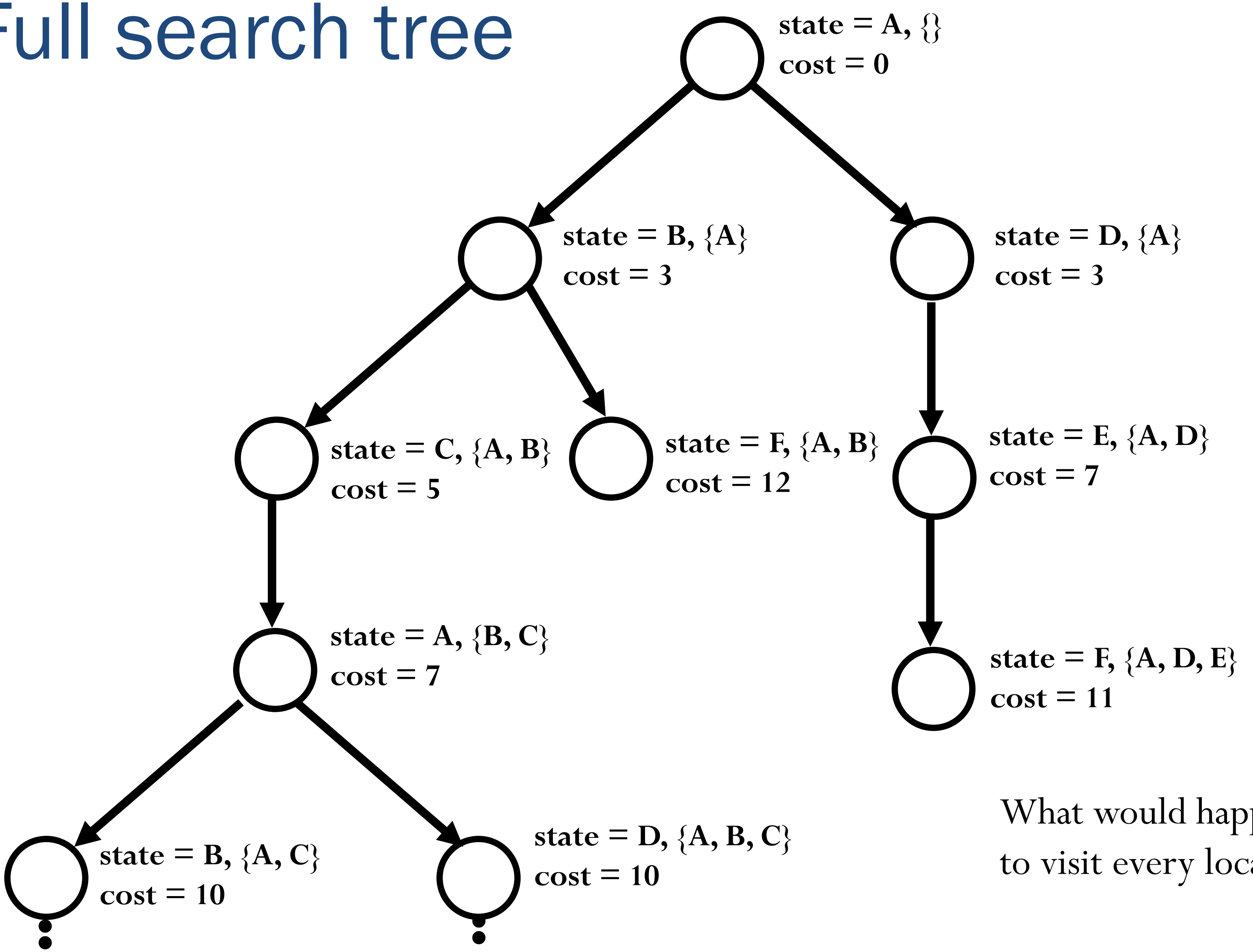


search tree nodes and states are not the same thing!

Full search tree



Full search tree



What would happen if the goal were to visit every location twice?

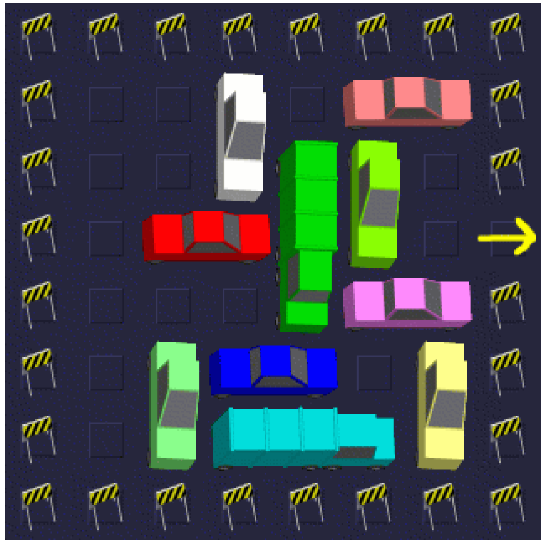
Key concepts in search

- Set of states that we can be in
  - Including an initial state…
  - … and goal states (equivalently, a goal test)
- For every state, a set of actions that we can take
  - Each action results in a new state
  - Typically defined by successor function
    - Given a state, produces all states that can be reached from it
- Cost function that determines the cost of each action (or path = sequence of actions)
- Solution: path from initial state to a goal state
  - Optimal solution: solution with minimal cost

Uninformed search

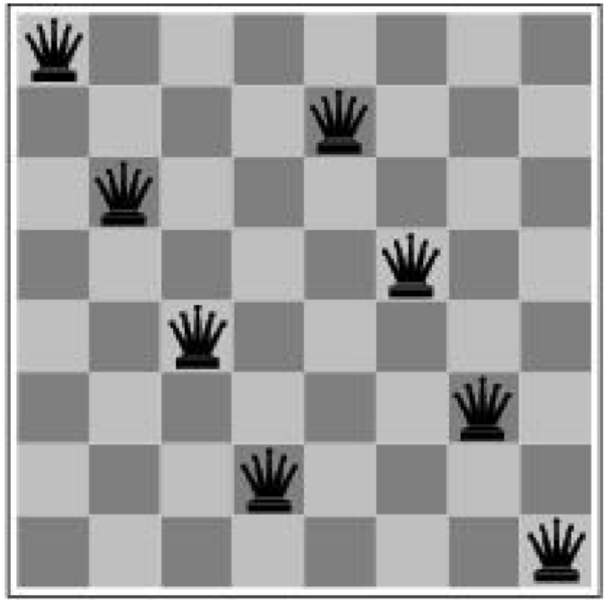
- Given a state, we only know whether it is a goal state or not
- Cannot say one nongoal state looks better than another nongoal state
- Can only traverse state space blindly in hope of somehow hitting a goal state at some point
  - Also called blind search
  - Blind does **not** imply unsystematic!

Searching Examples

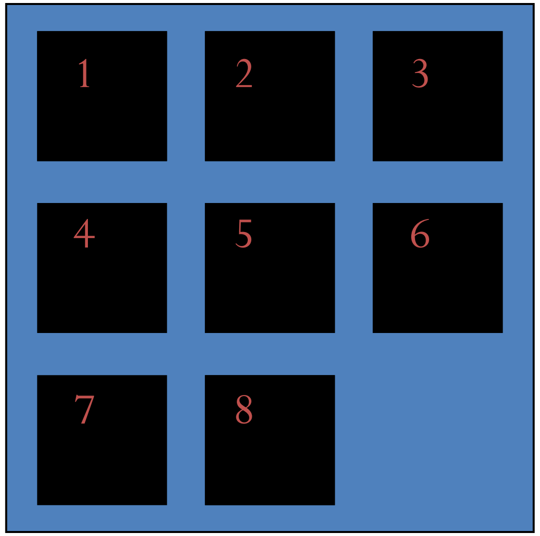


Rush Hour: Move cars forward and backward to “escape”

8-queens problem



8-puzzle



Logistics

Generic search algorithm

- Fringe = set of nodes generated but not expanded
- fringe := {initial state}
- loop:
  - if fringe empty, declare failure
  - choose and remove a node v from fringe
  - check if v's state s is a goal state; if so, declare success
  - if not, expand v, insert resulting nodes into fringe
- Key question in search: Which of the generated nodes do we expand next?

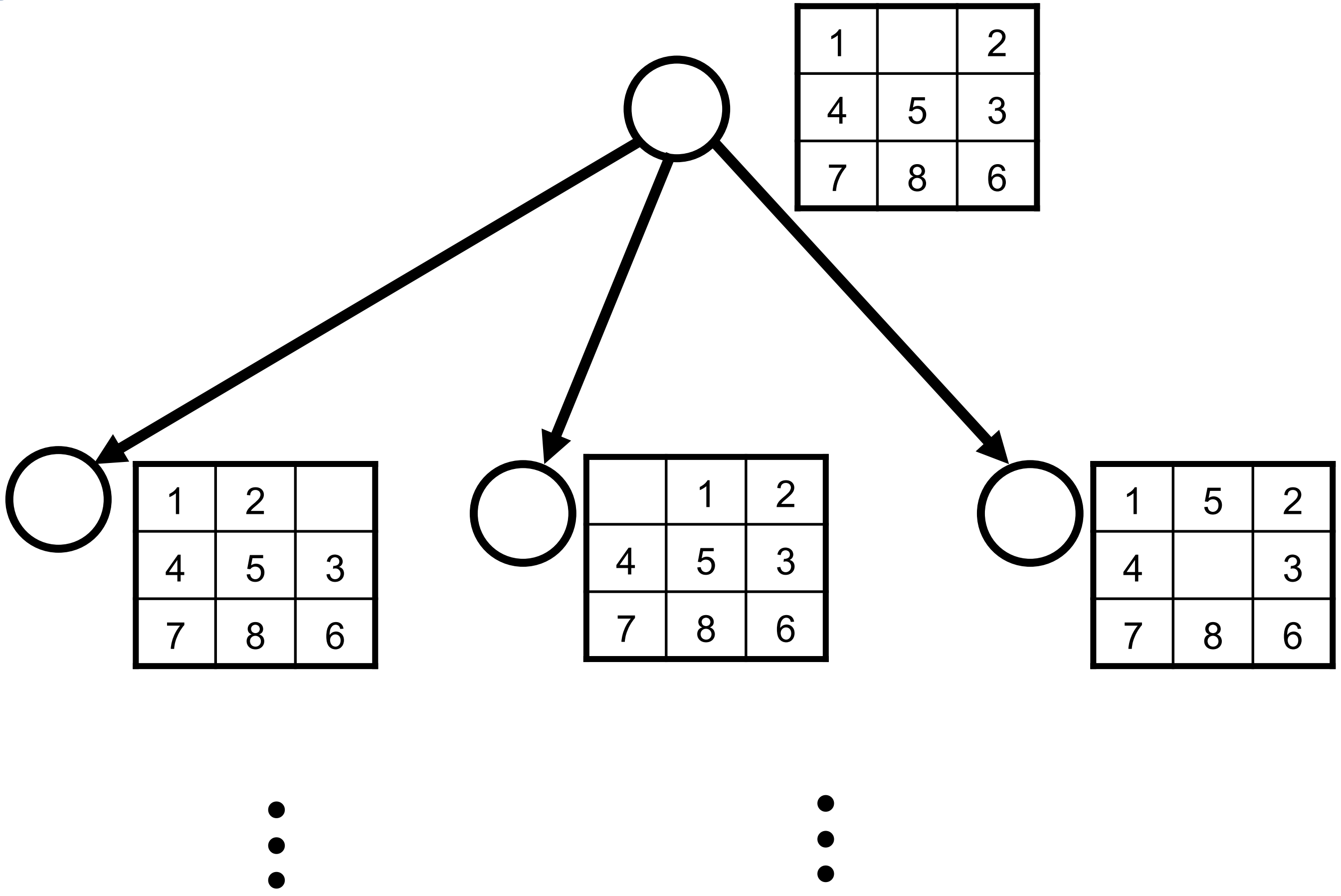
8-puzzle

1		2
4	5	3
7	8	6

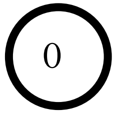
1	2	3
4	5	6
7	8	

goal state

8-puzzle



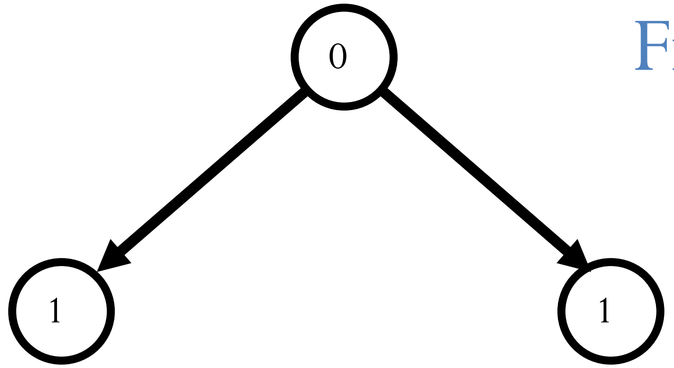
Breadth-First Search



First-In-First-Out (FIFO)
Queue

Animated PPT

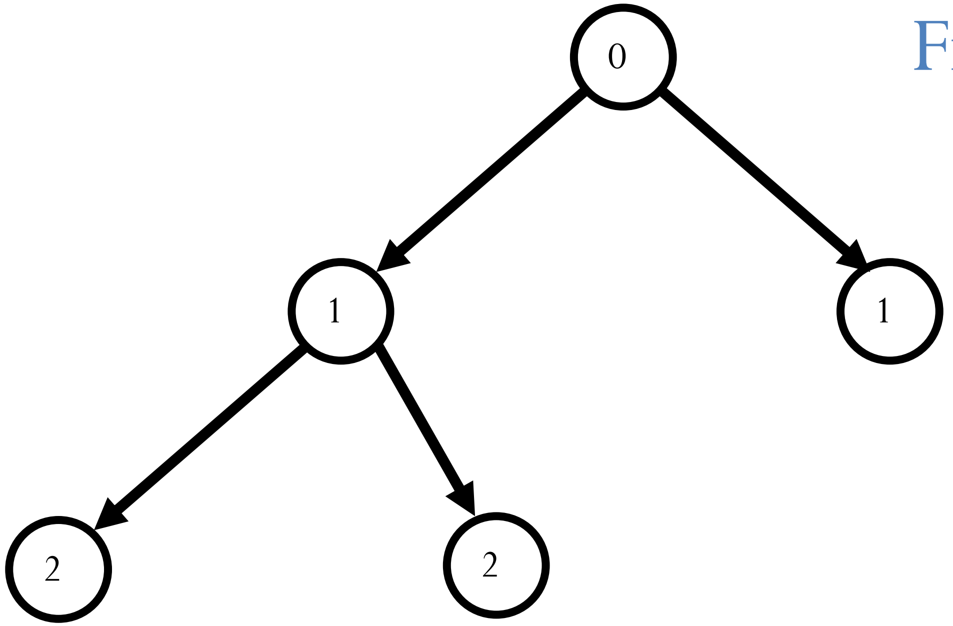
Breadth-First Search



First-In-First-Out (FIFO) Queue

Animated PPT

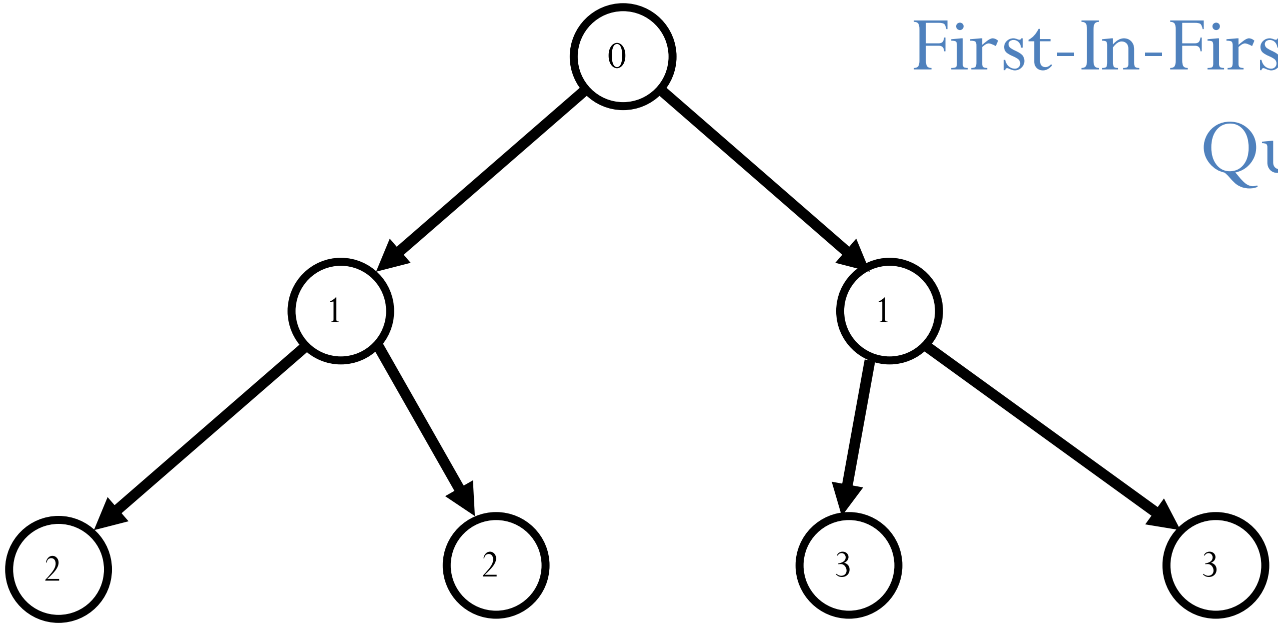
Breadth-First Search



First-In-First-Out (FIFO)
Queue

Animated PPT

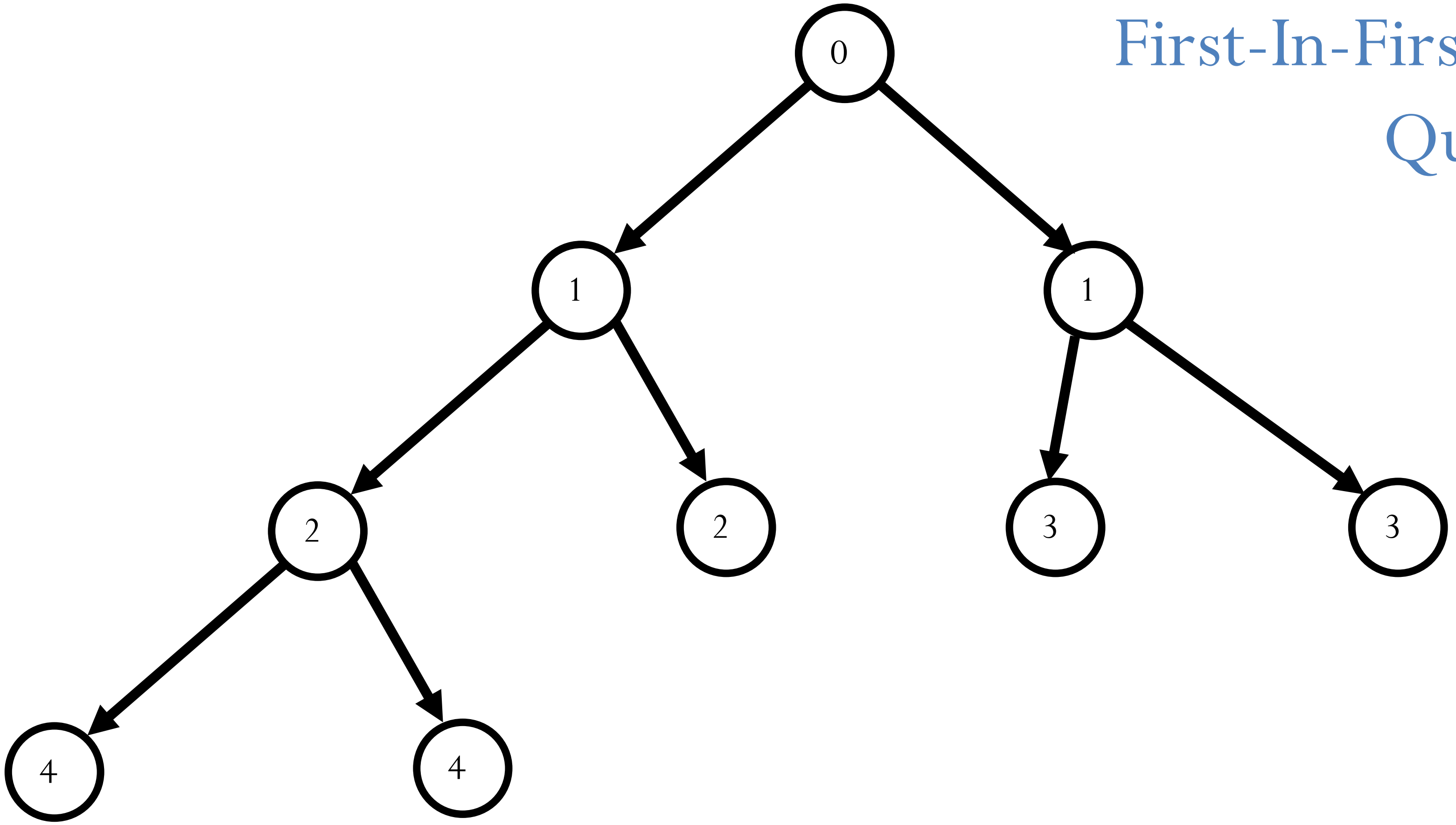
Breadth-First Search



First-In-First-Out (FIFO)
Queue

Animated PPT

Breadth-First Search

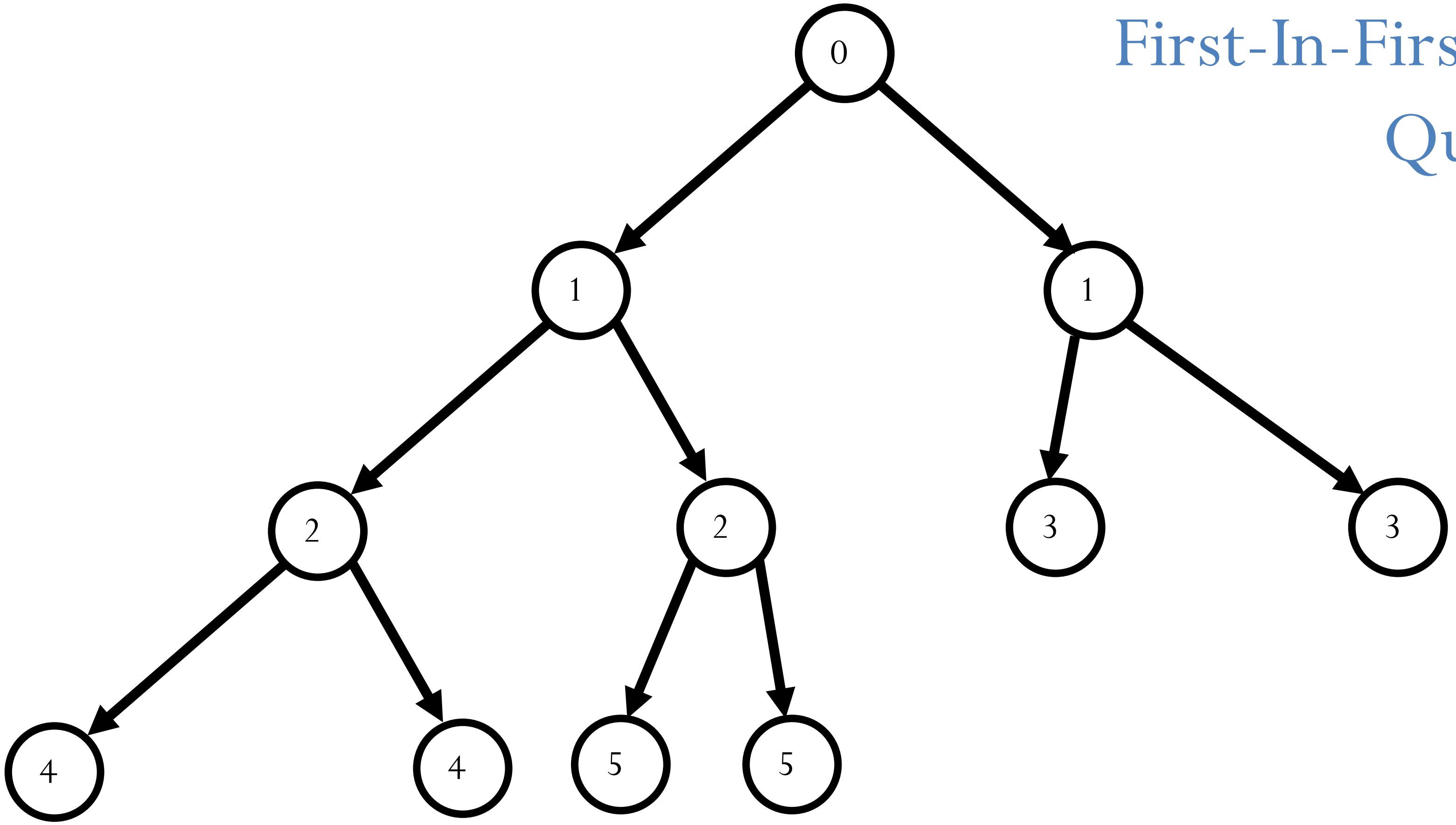


First-In-First-Out (FIFO)
Queue

Animated PPT

Breadth-First Search

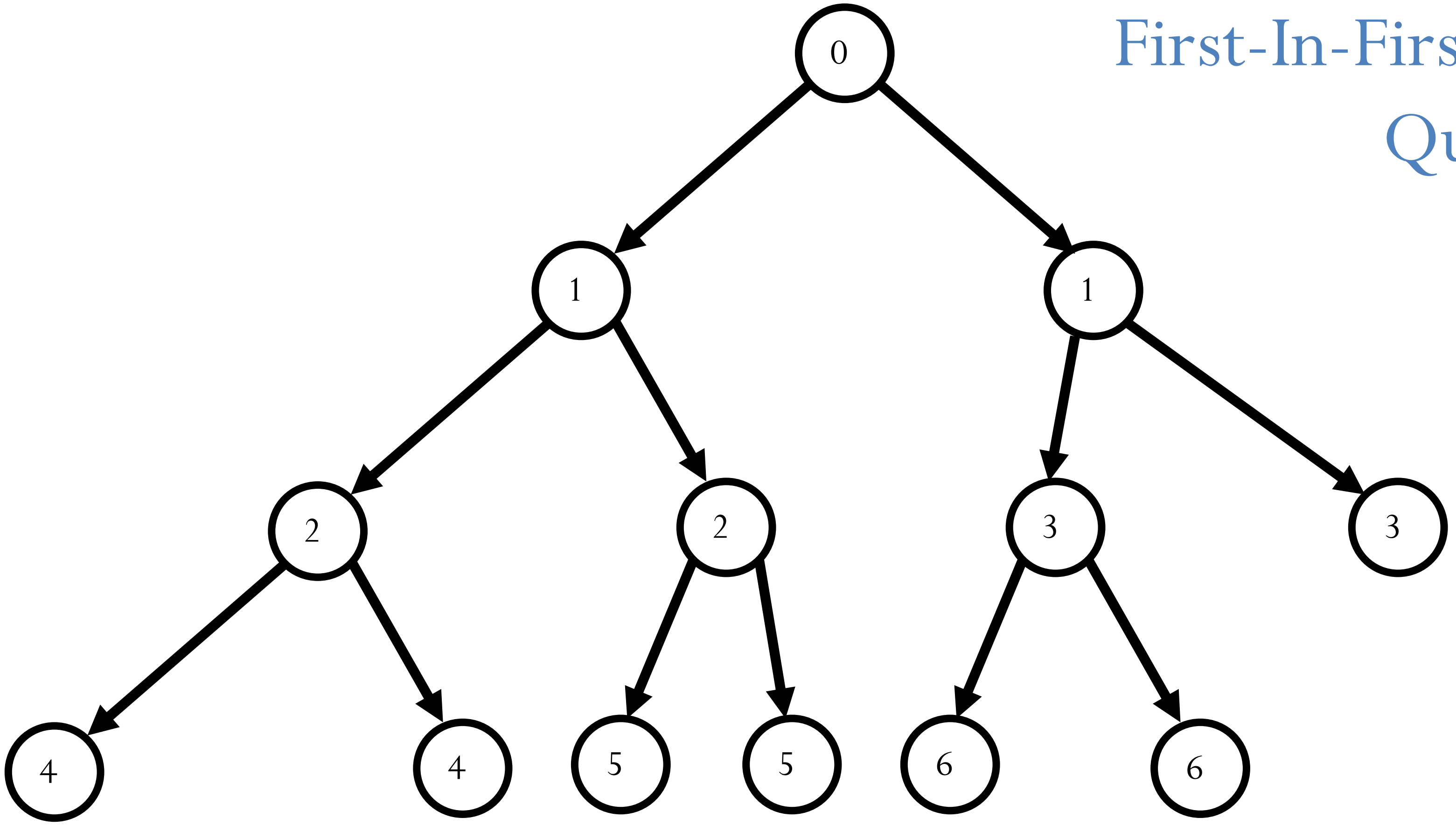
First-In-First-Out (FIFO)
Queue



Animated PPT

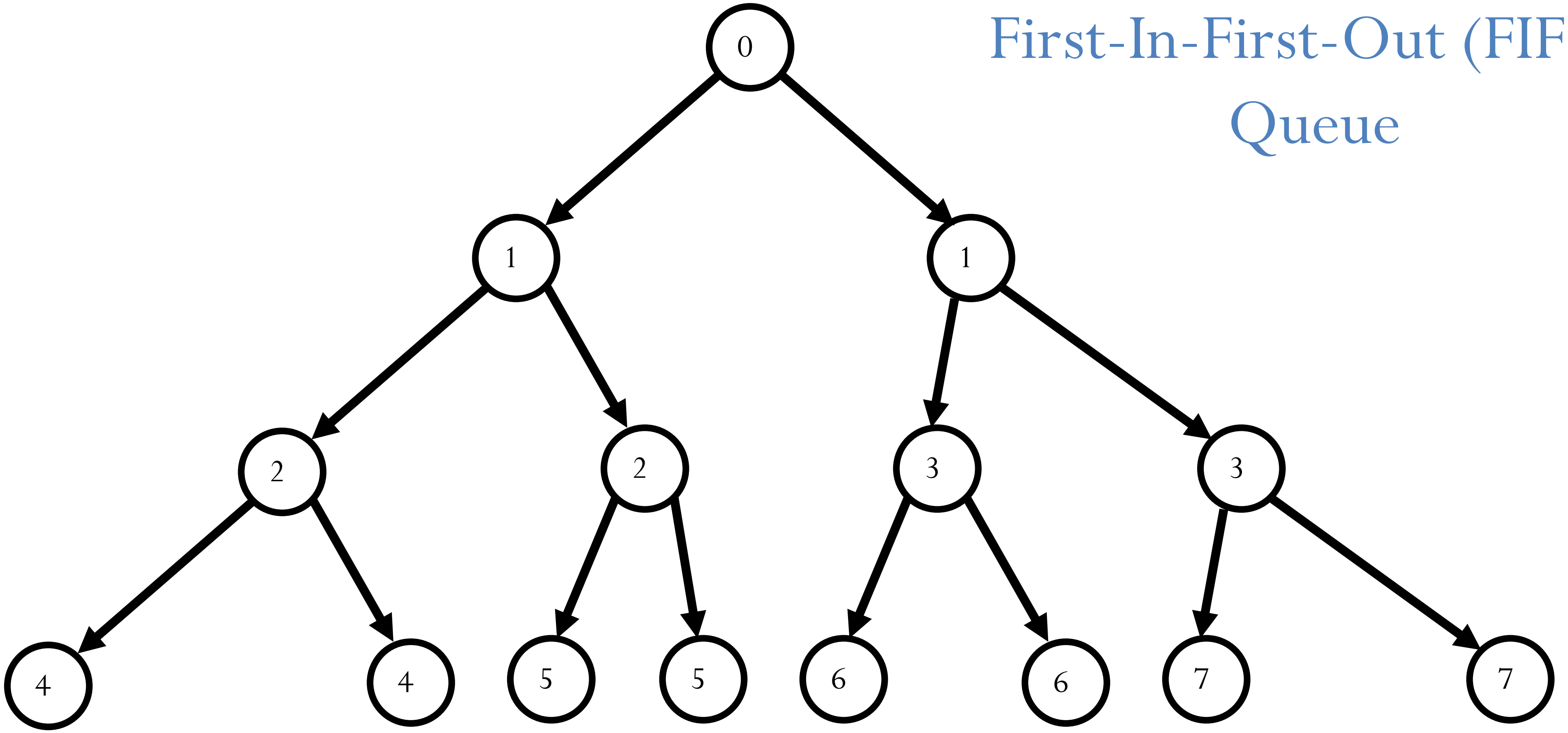
Breadth-First Search

First-In-First-Out (FIFO) Queue



Animated PPT

Breadth-First Search

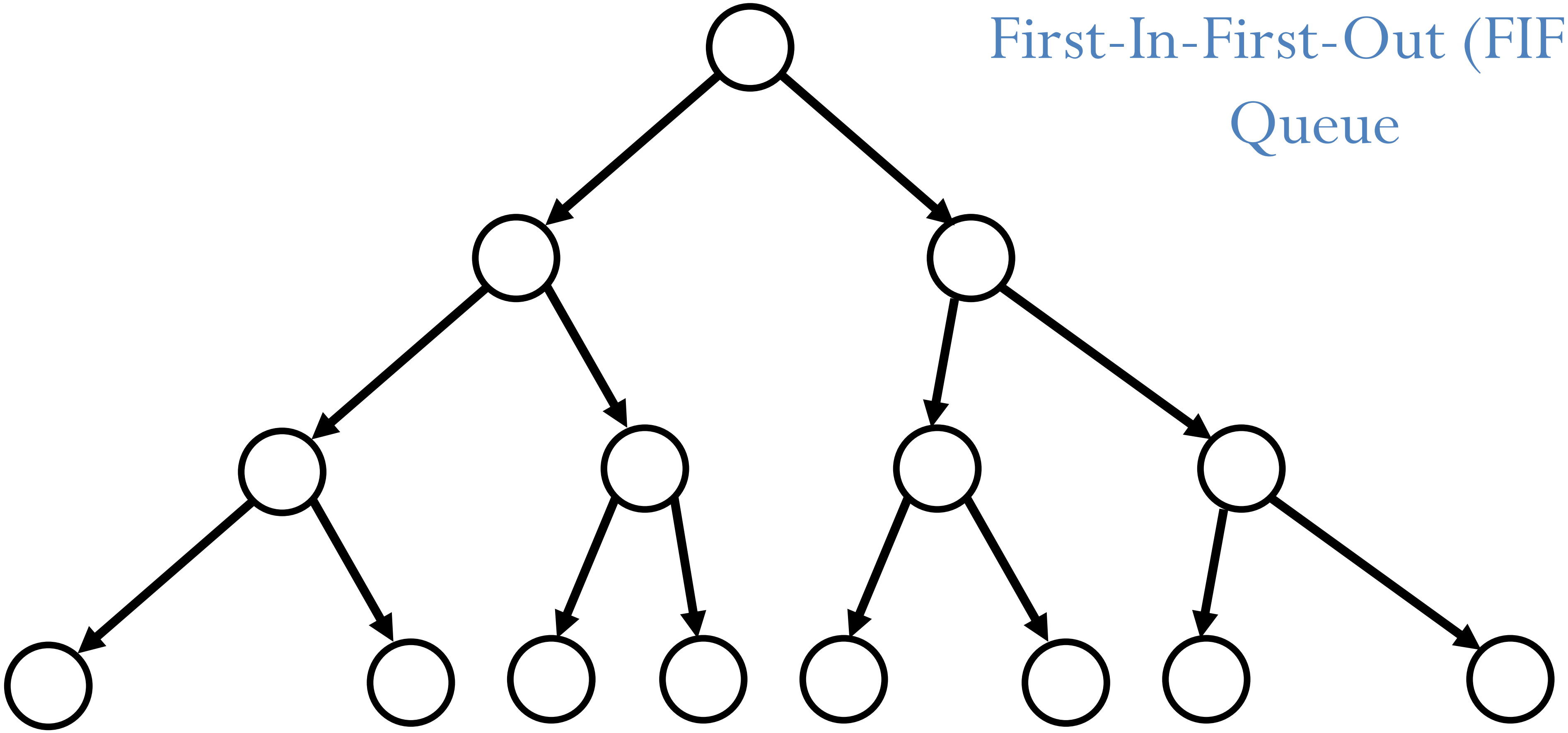


First-In-First-Out (FIFO)
Queue

Animated PPT

Breadth-First Search

First-In-First-Out (FIFO)
Queue

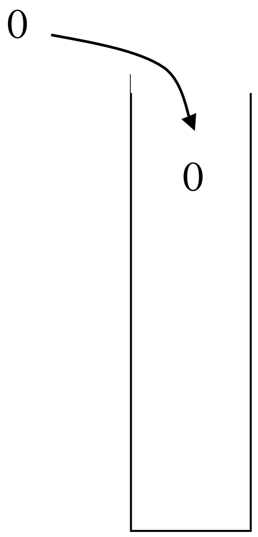


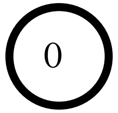
Animated PPT

Properties of Breadth-First Search

- Nodes are expanded in the same order in which they are generated
  - Fringe can be maintained as a First-In-First-Out (FIFO) queue
- BFS is complete: if a solution exists, one will be found
- BFS finds a shallowest solution
  - Not necessarily an optimal solution
- If every node has b successors (the branching factor), first solution is at depth d, then fringe size will be at least b^d at some point
  - This much space (and time) required ☹

Depth-First Search



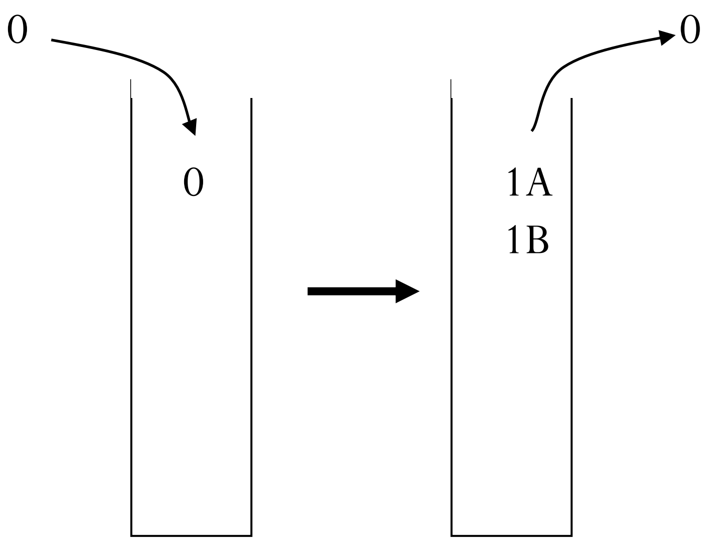


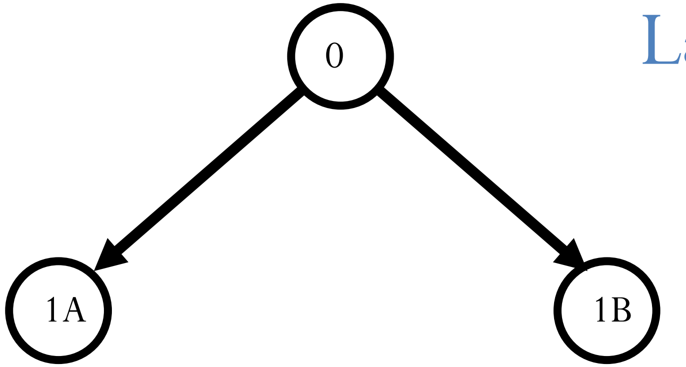
Last-In-First-Out (LIFO)

Stack

Animated PPT

Depth-First Search



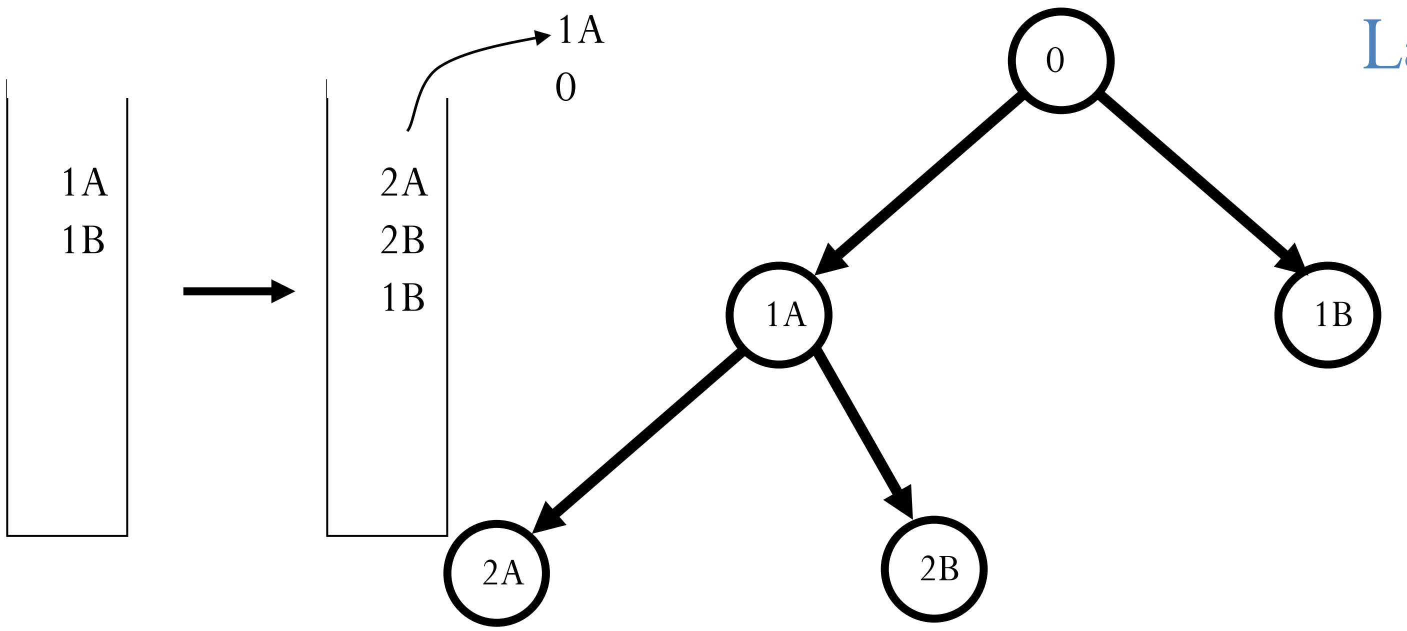


Last-In-First-Out (LIFO)

Stack

Animated PPT

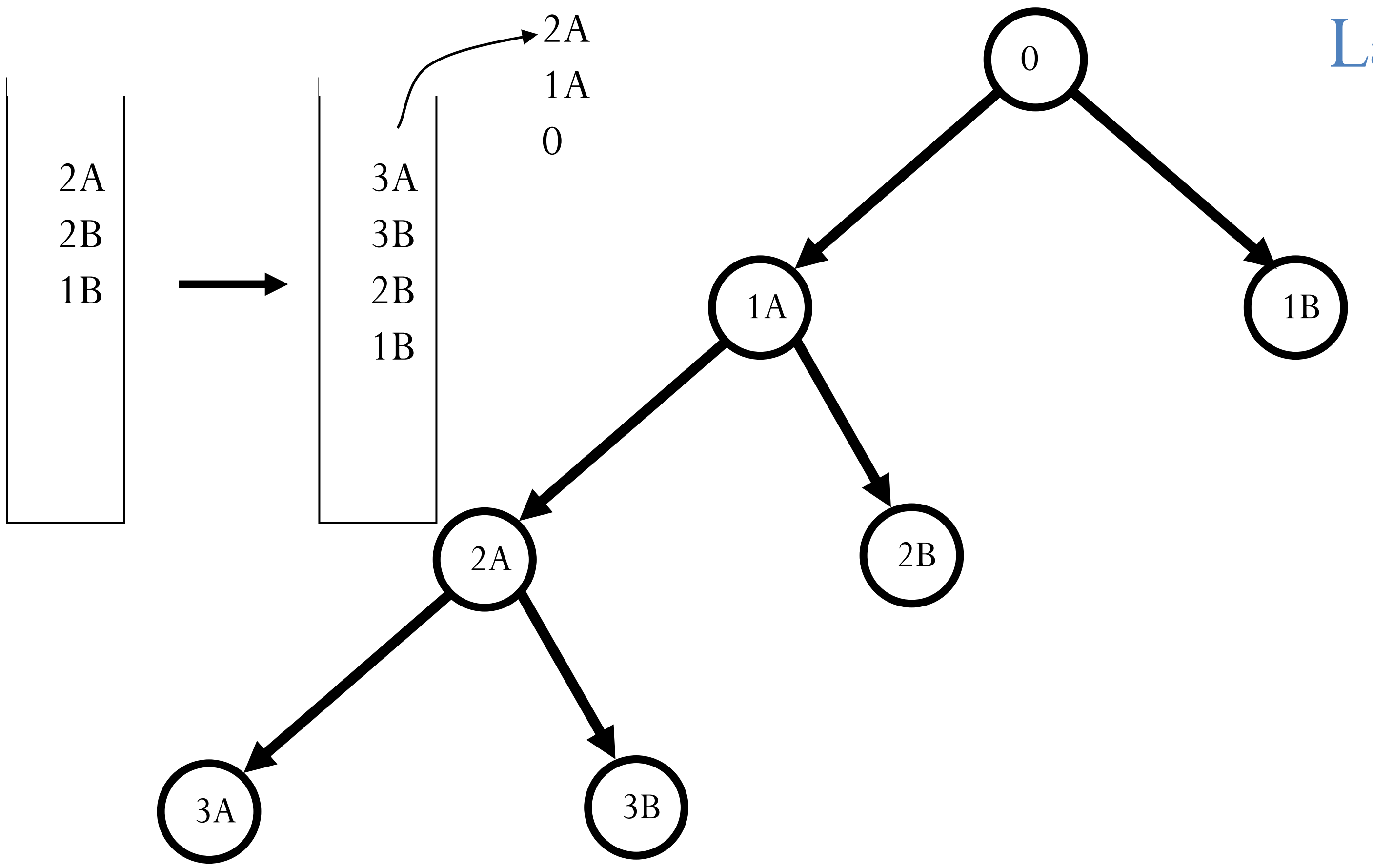
Depth-First Search



Last-In-First-Out (LIFO)
Stack

Animated PPT

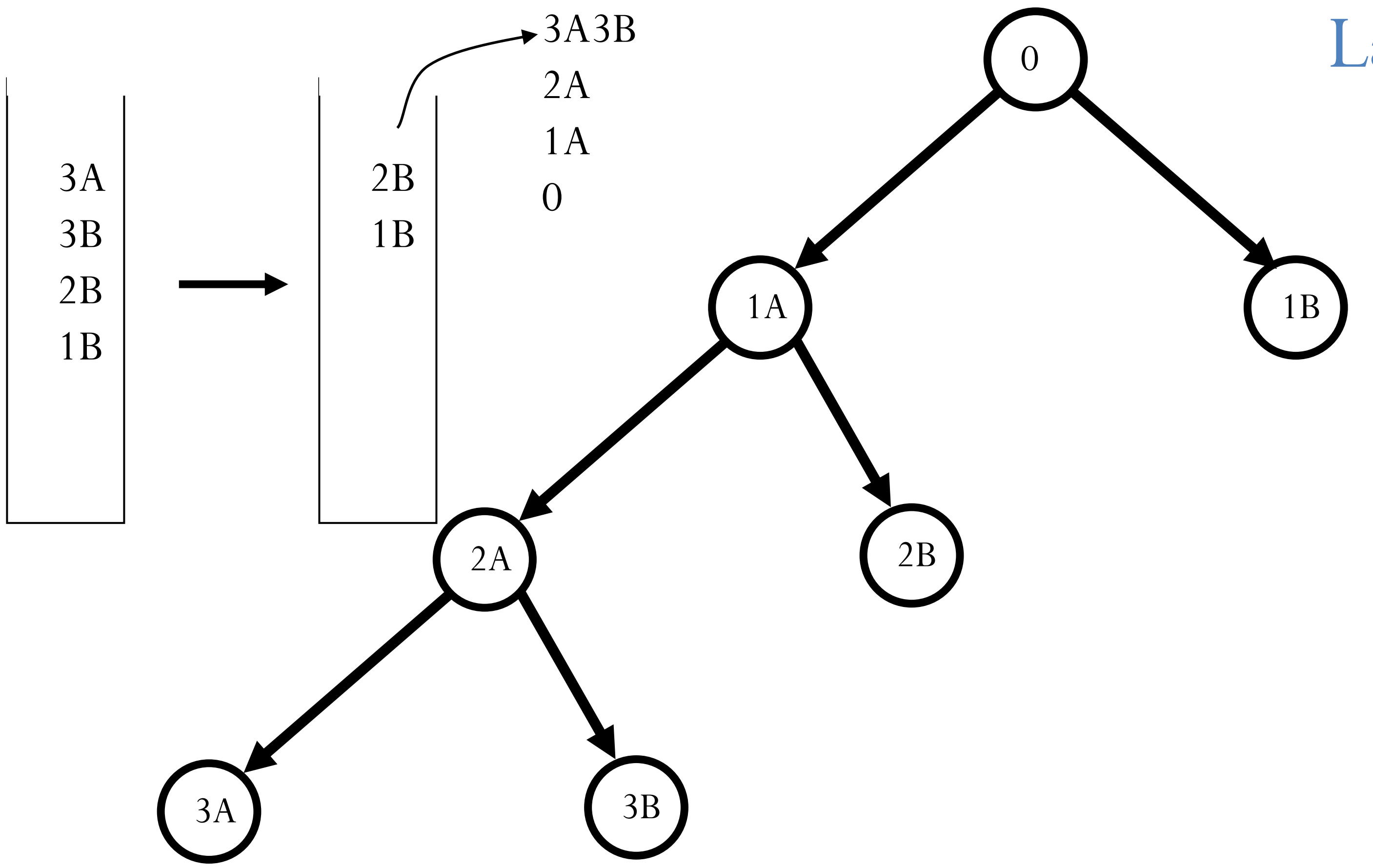
Depth-First Search



Last-In-First-Out (LIFO)
Stack

Animated PPT

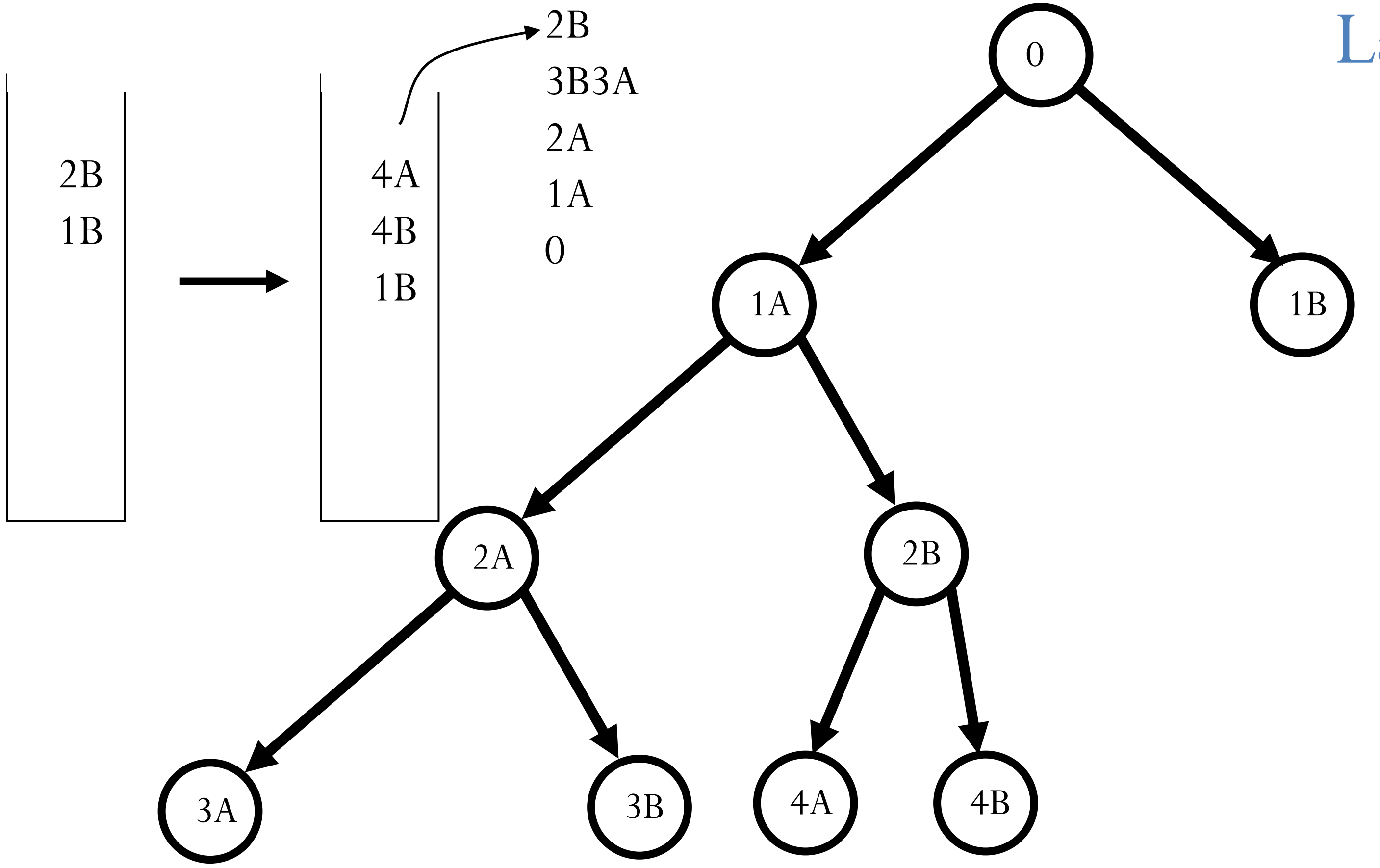
Depth-First Search



Last-In-First-Out (LIFO)
Stack

Animated PPT

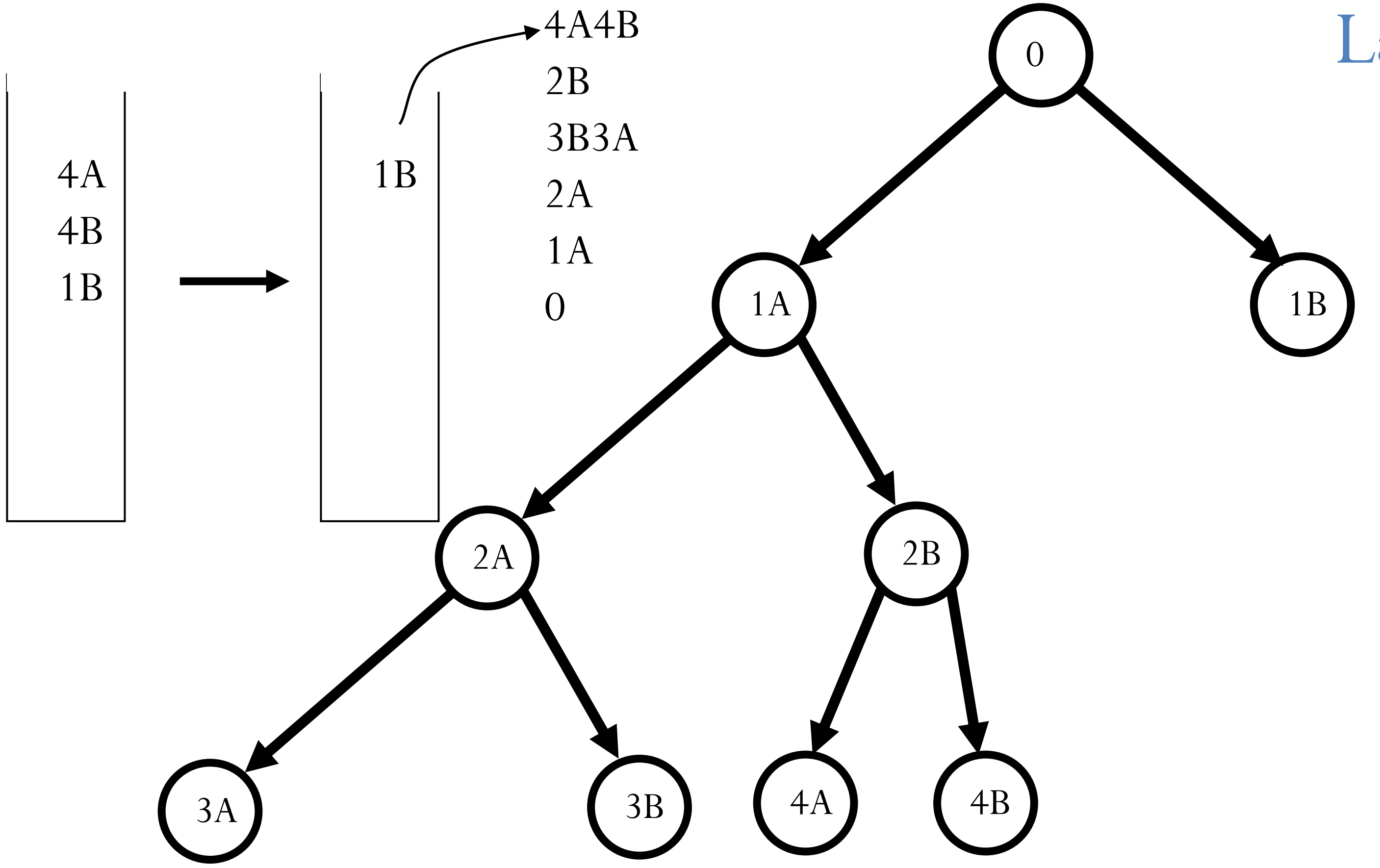
Depth-First Search



Last-In-First-Out (LIFO)
Stack

Animated PPT

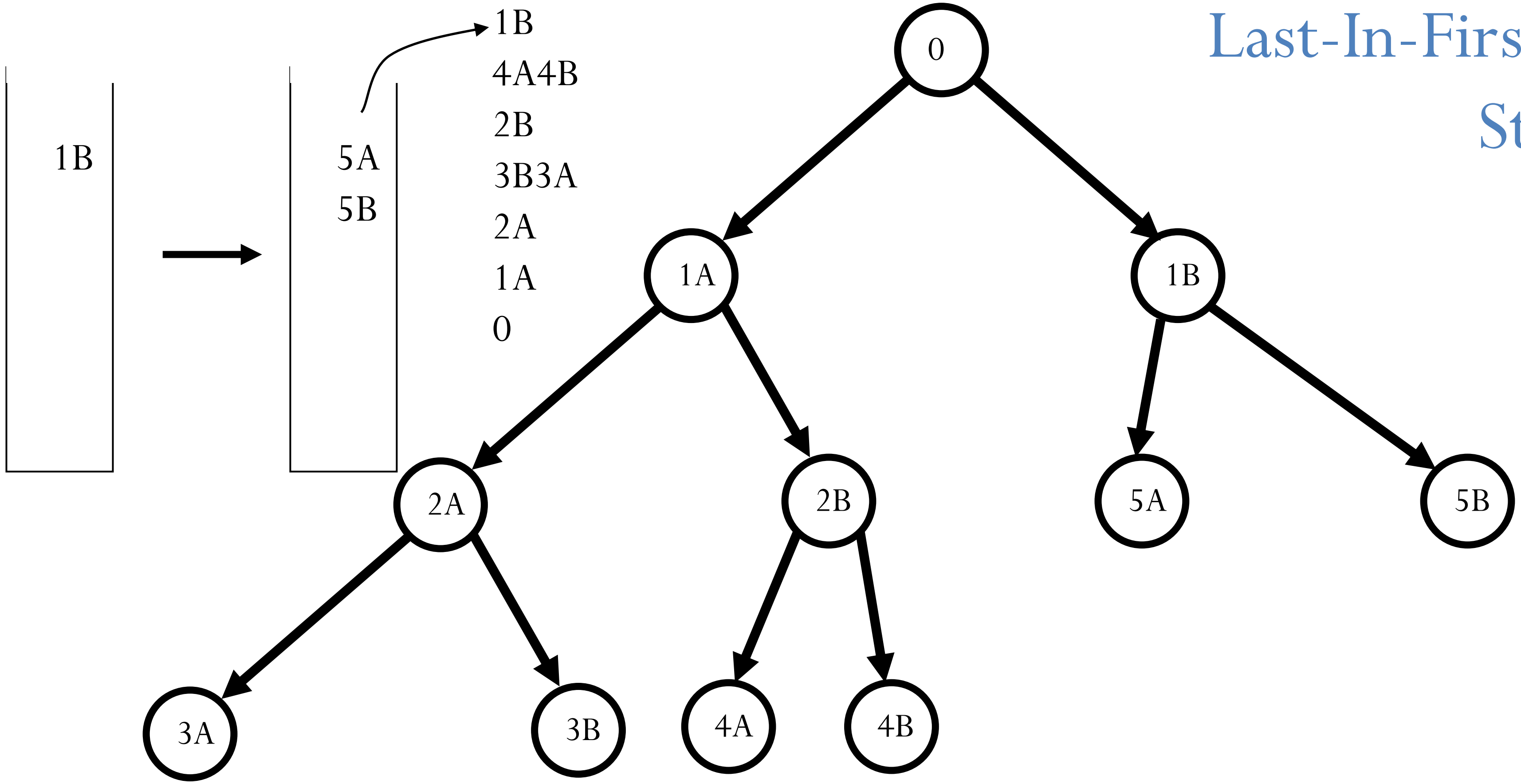
Depth-First Search



Last-In-First-Out (LIFO)
Stack

Animated PPT

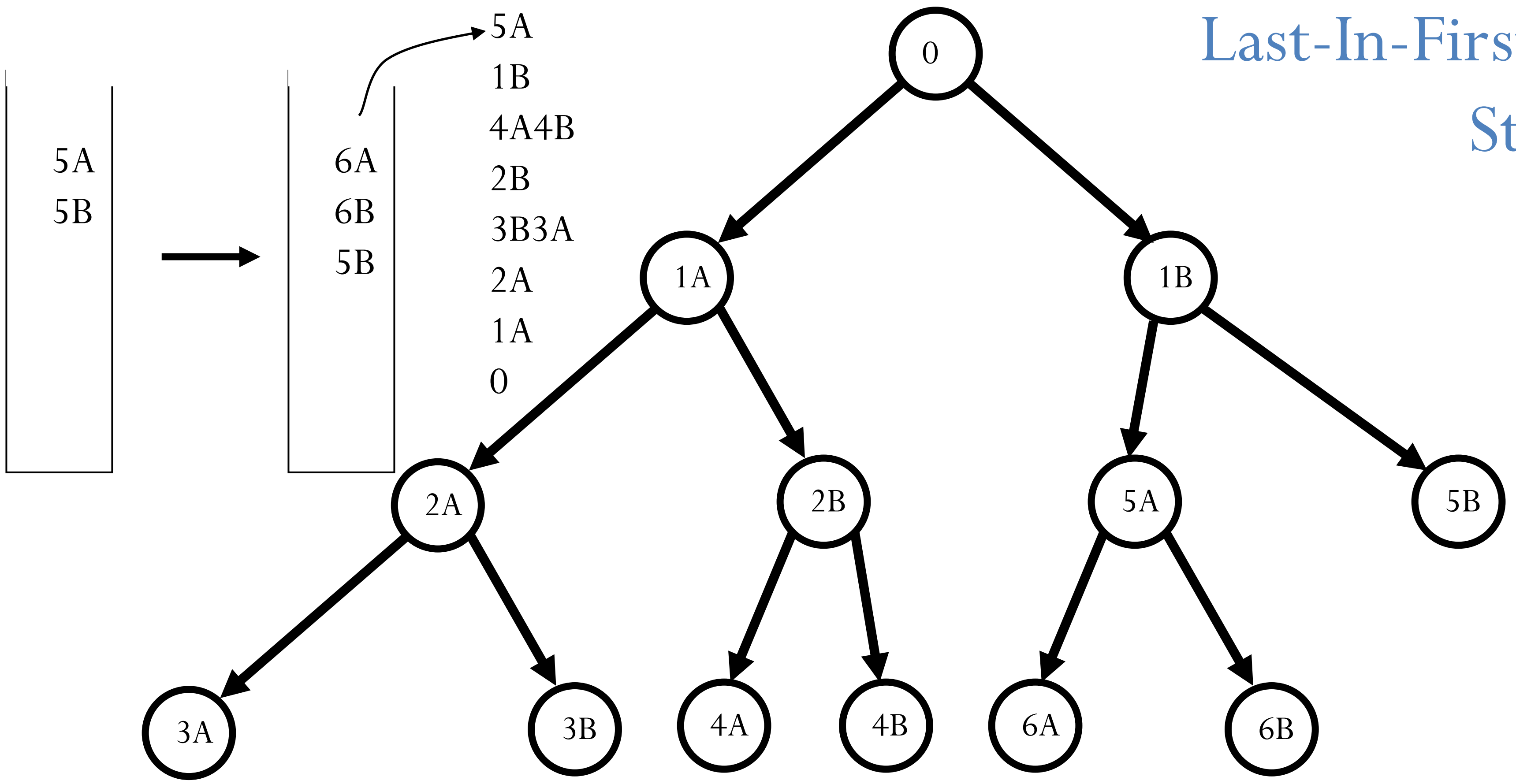
Depth-First Search



Last-In-First-Out (LIFO)
Stack

Animated PPT

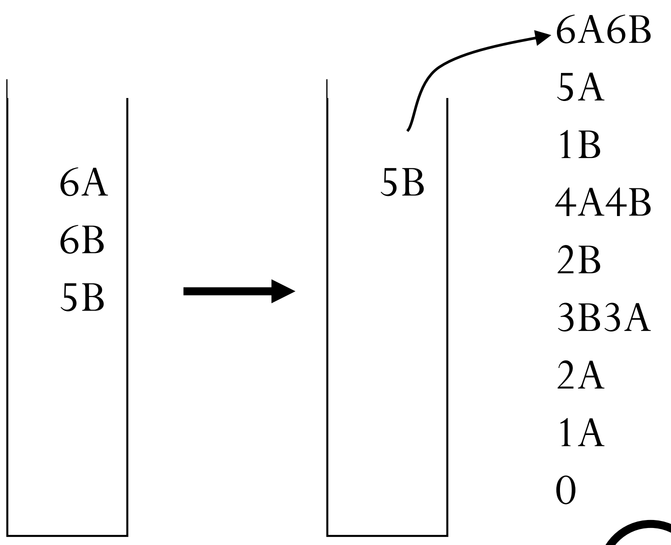
Depth-First Search

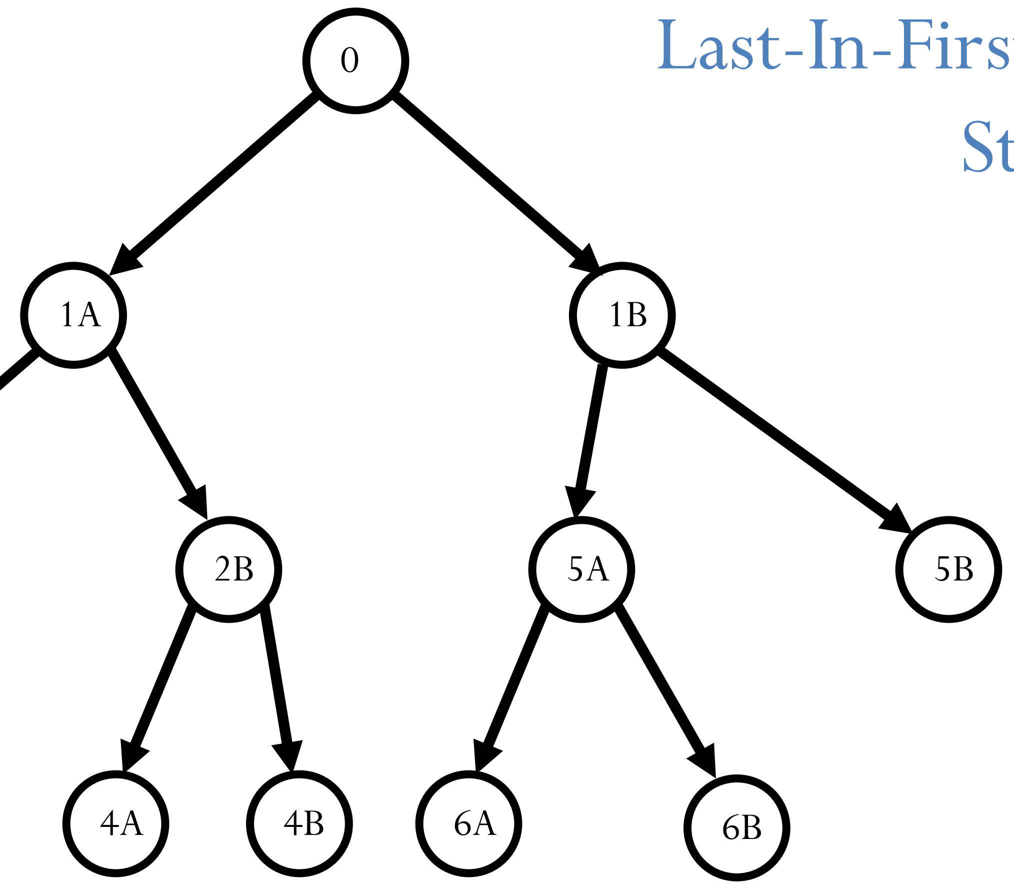


Last-In-First-Out (LIFO)
Stack

Animated PPT

Depth-First Search

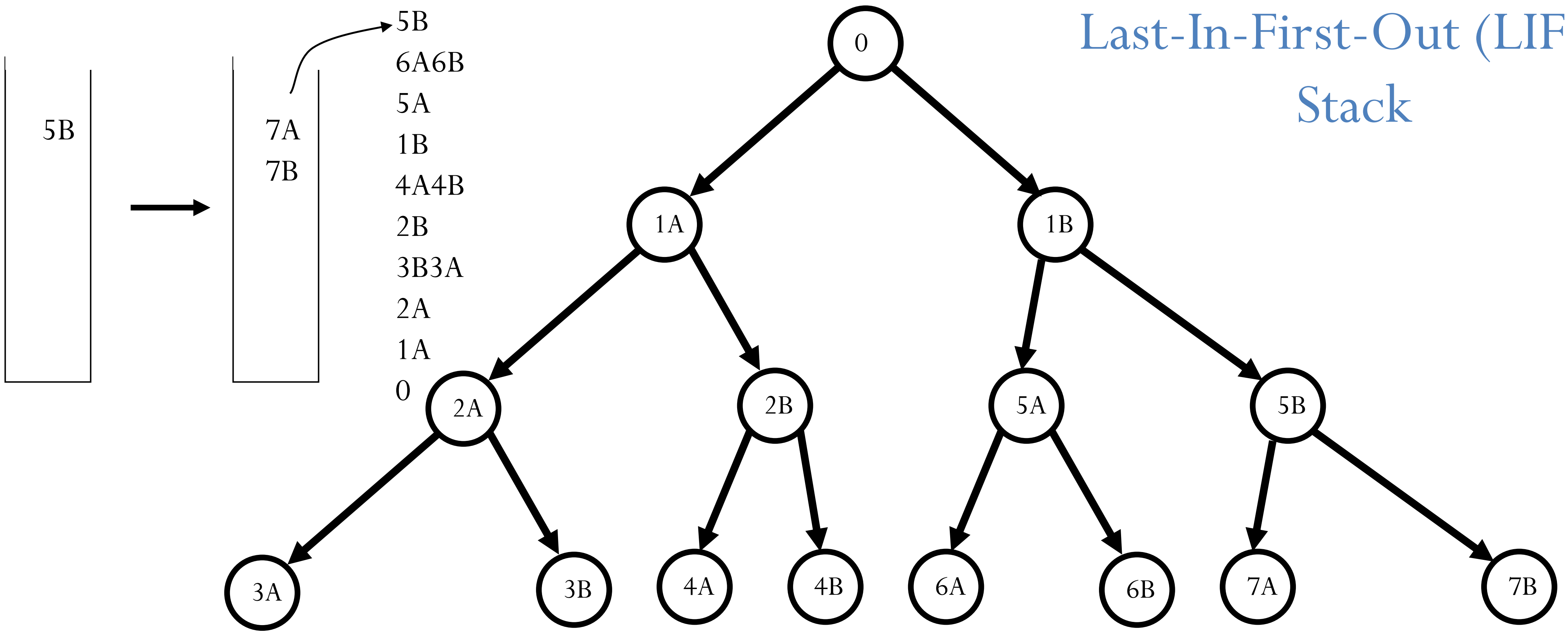




Last-In-First-Out (LIFO)
Stack

Animated PPT

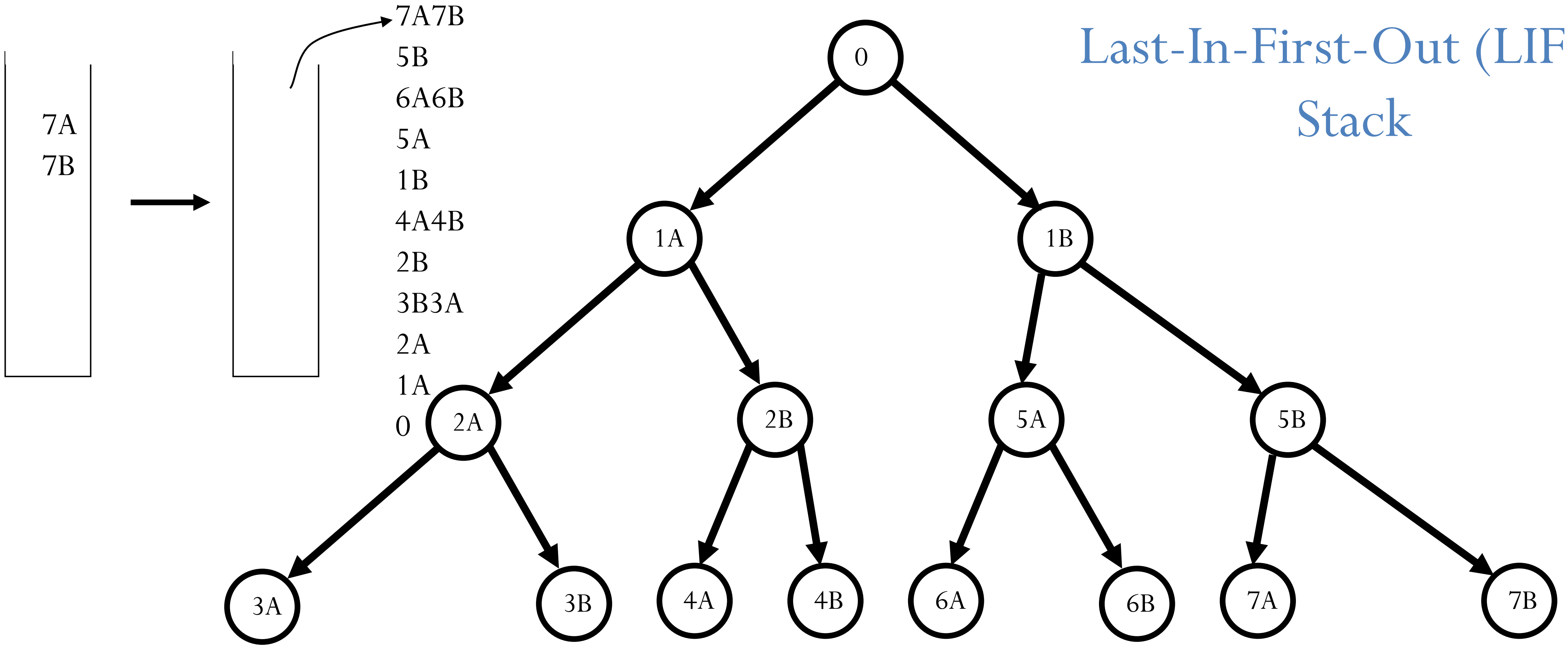
Depth-First Search



Last-In-First-Out (LIFO)
Stack

Animated PPT

Depth-First Search

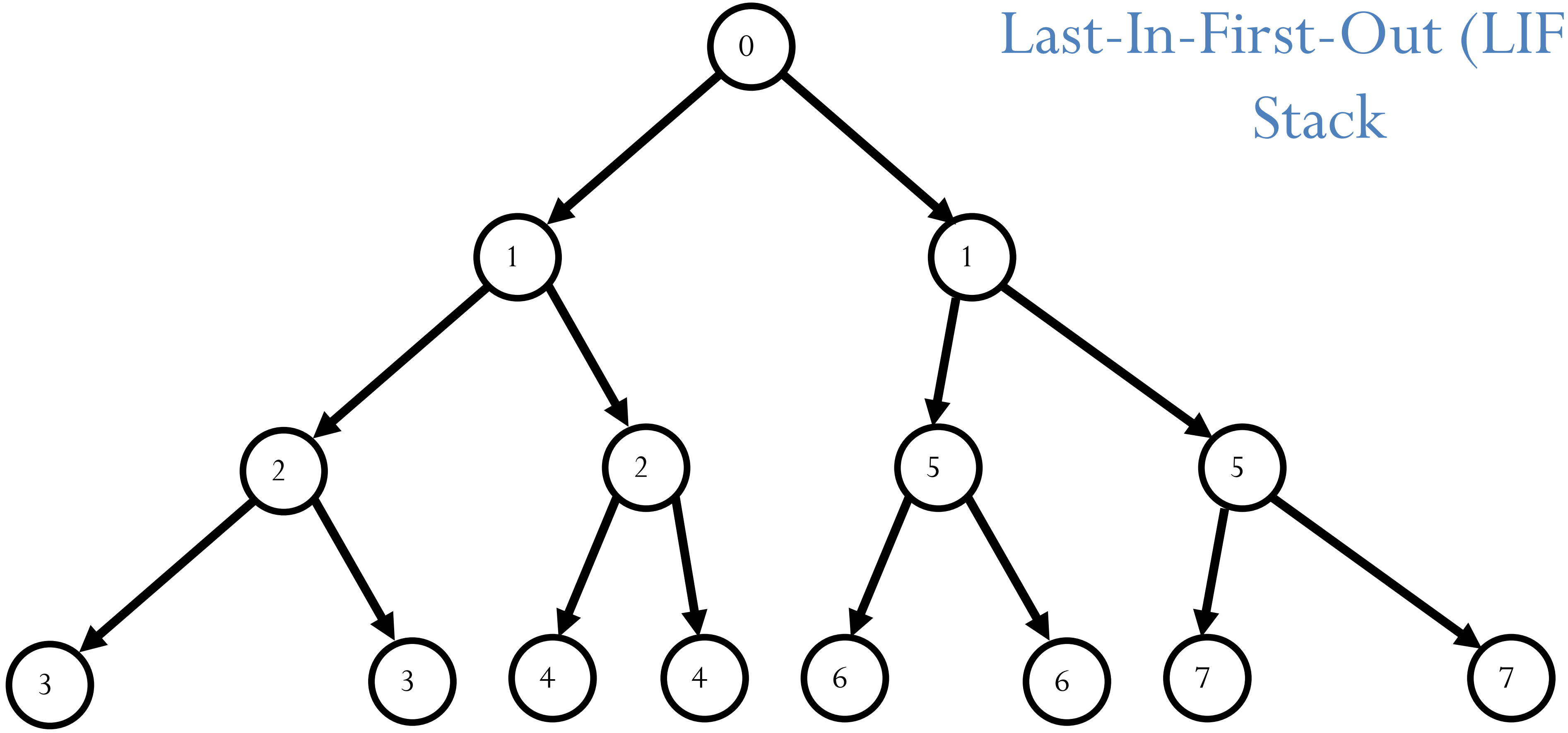


Last-In-First-Out (LIFO)
Stack

Animated PPT

Depth-First Search

Last-In-First-Out (LIFO)
Stack



Animated PPT

Implementing Depth-First Search

- Fringe can be maintained as a Last-In-First-Out (LIFO) queue (aka. a stack)
- Also easy to implement recursively:
- DFS(node)
  - If goal(node) return solution(node);
  - For each successor of node
    - Return DFS(successor) unless it is *failure*;
  - Return *failure;*

Properties of depth-first search

- Not complete (might cycle through nongoal states)
- If solution found, generally not optimal/shallowest
- If every node has b successors (the branching factor), and we search to at most depth m, fringe is at most bm
  - Much better space requirement ☺
  - Actually, generally don't even need to store all of fringe
- Time: still need to look at every node
  - $b^m + b^{m-1} + \ldots + 1$ (for $b>1$, $O(b^m)$)
  - Inevitable for uninformed search methods…

Combining good properties of BFS and DFS

- Limited depth DFS: just like DFS, except never go deeper than some depth d
- Iterative deepening DFS:
  - Call limited depth DFS with depth 0;
  - If unsuccessful, call with depth 1;
  - If unsuccessful, call with depth 2;
  - Etc.
- Complete, finds shallowest solution
- Space requirements of DFS
- May seem wasteful timewise because replicating effort
  - Really not that wasteful because **almost all effort at deepest level**
  - $db + (d-1)b^2 + (d-2)b^3 + \ldots + 1b^d$ is $O(b^d)$ for $b > 1$

Searching solution evaluation

- Comparing multiple searching algorithm based on
  - Completeness: does it always find a solution if one exist?
  - Time complexity: How long depends on number of nodes
  - Space complexity: Memory depends on number of nodes
  - Optimality: Find shortest path (or least cost solution)?
  - Systematicity: does it visit each state at most once?

ขอบคุณ

Thai

Grazie

Italian

תודה רבה

Hebrew

धन्यवादः

Sanskrit

ಧನ್ಯವಾದಗಳು

Kannada

Ευχαριστώ

Greek

Thank You

English

Gracias

Spanish

Спасибо

Russian

Obrigado

Portuguese

شكراً

Arabic

https://sites.google.com/site/animeshchaturvedi07

Merci

French

多謝

Traditional Chinese

धन्यवाद

Hindi

Danke

German

நன்றி

Tamil

Tamil

多谢

Simplified Chinese

ありがとうございました

Japanese

감사합니다

Korean