binary trees

traversals, searches, game trees

slides
bit.ly/abhi-disc

attendance
bit.ly/abhi-attendance

announcements

1. HW 5 due ~~3/8 (tomorrow)~~ 3/29 tuesday after spring break
2. Lab 8 due 3/11 (friday)
3. Weekly survey due tomorrow!

trees 

trees

- structures that follow three basic rules

trees

- structures that follow three basic rules

1) if there are N nodes, there are N-1 edges

trees

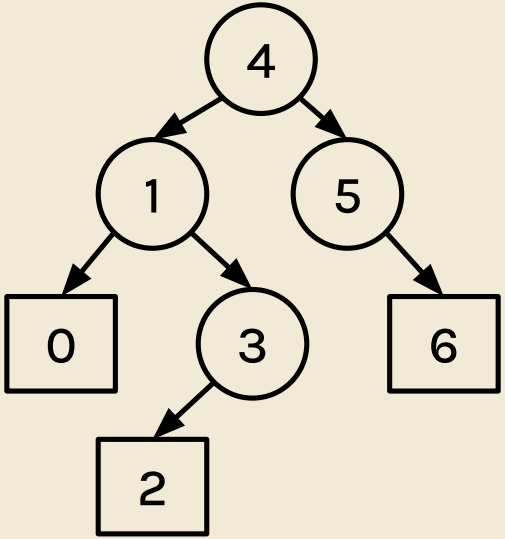
- structures that follow three basic rules

1) if there are N nodes, there are N-1 edges
2) there’s exactly one path from every node to every other node

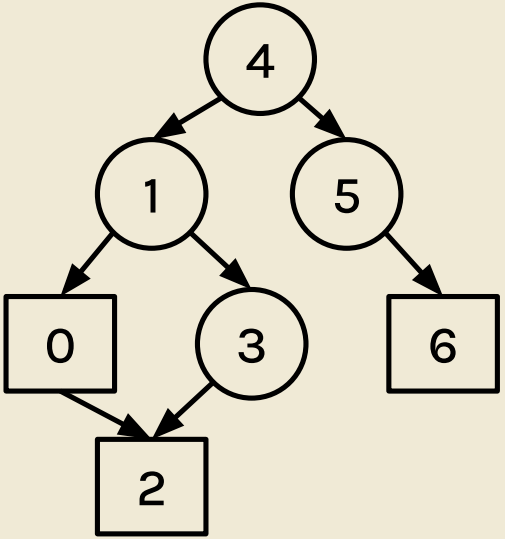
trees

- structures that follow three basic rules

1) if there are N nodes, there are N-1 edges
2) there's exactly one path from every node to every other node
3) there are no cycles (follows from 1 and 2)



valid tree



invalid tree

tree traversals

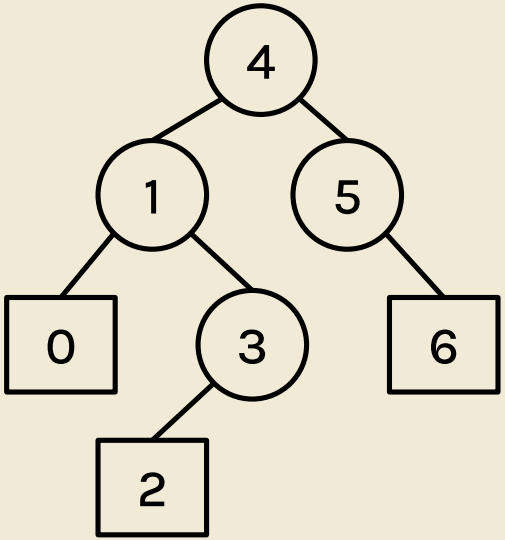
- **preorder:** visit node, then traverse children
- **inorder:** traverse left child, then node, then traverse right child
- **postorder:** traverse children, then visit node

tree traversals (informal)

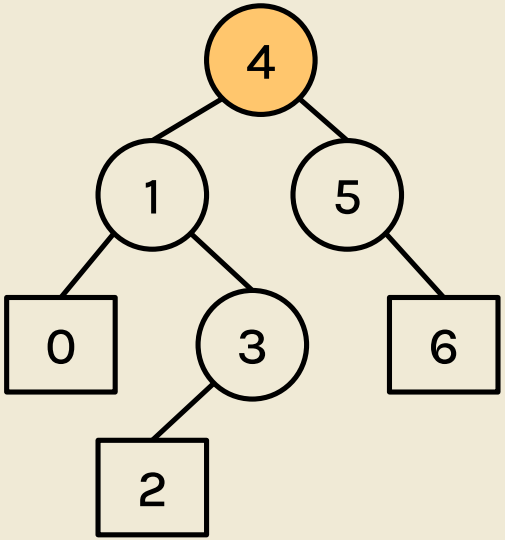
start by tracing the tree counterclockwise

- **preorder:** visit “left sides” of nodes
- **inorder:** visit “bottom sides” of nodes
- **postorder:** visit “right sides” of nodes

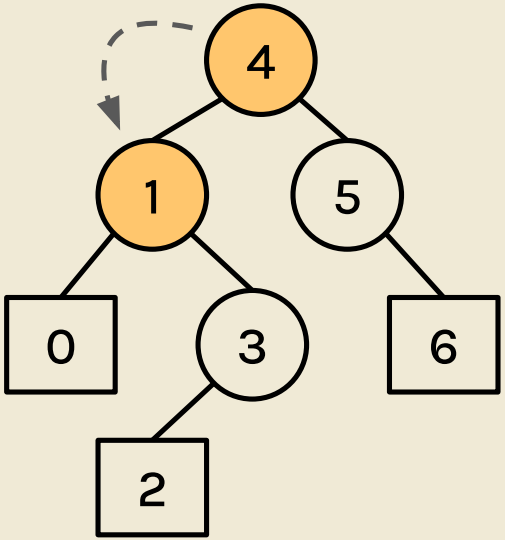
tree traversals (preorder)



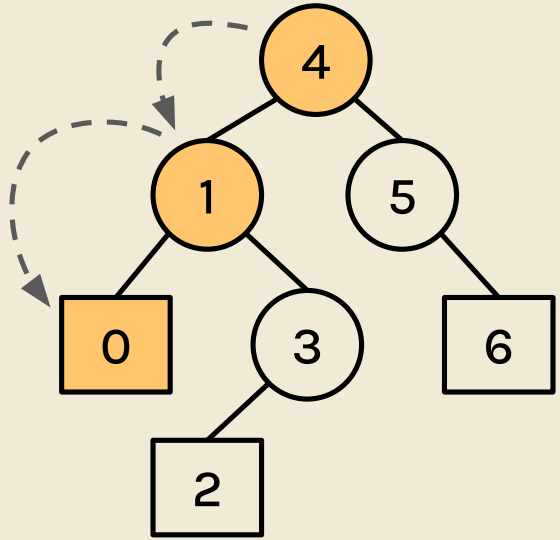
tree traversals (preorder)



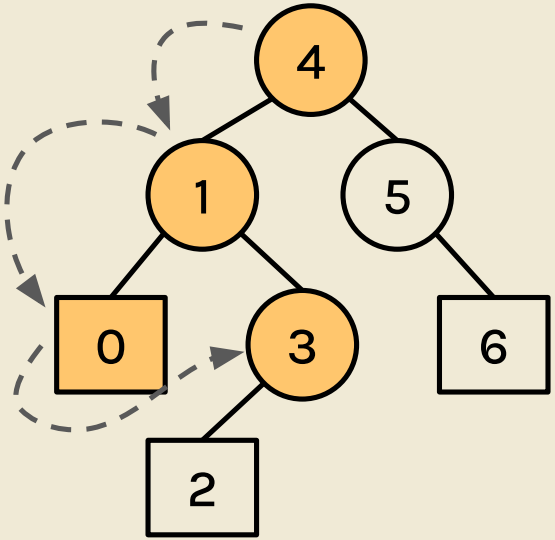
tree traversals (preorder)



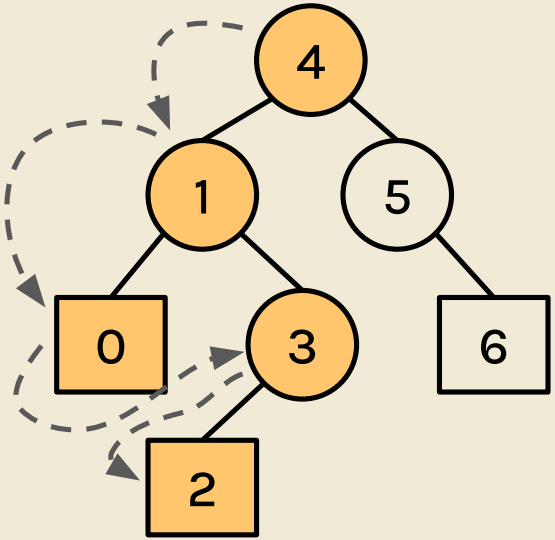
tree traversals (preorder)



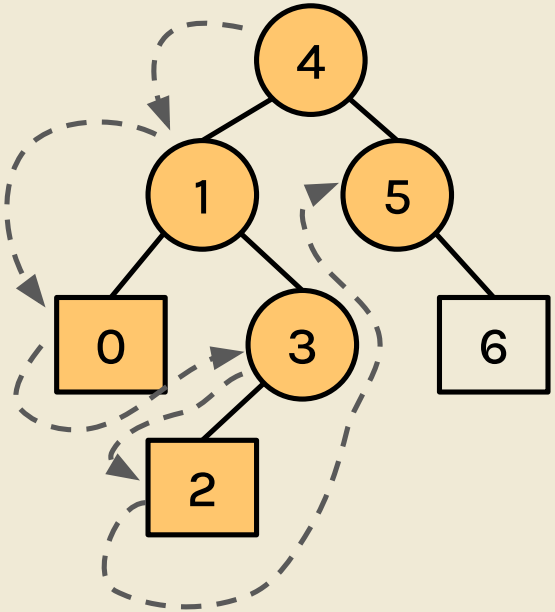
tree traversals (preorder)



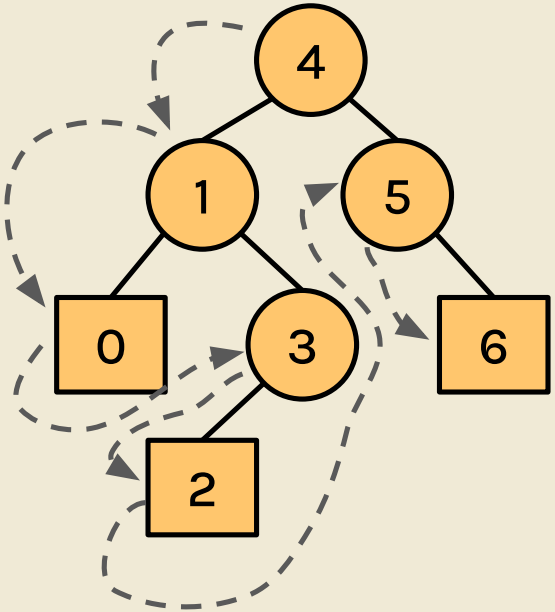
tree traversals (preorder)



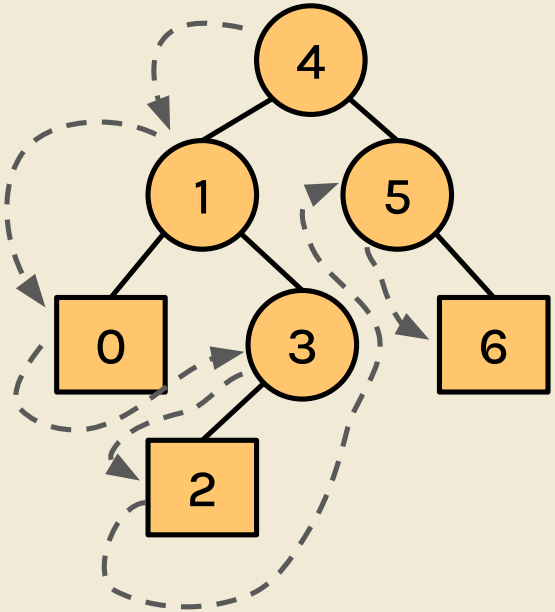
tree traversals (preorder)



tree traversals (preorder)

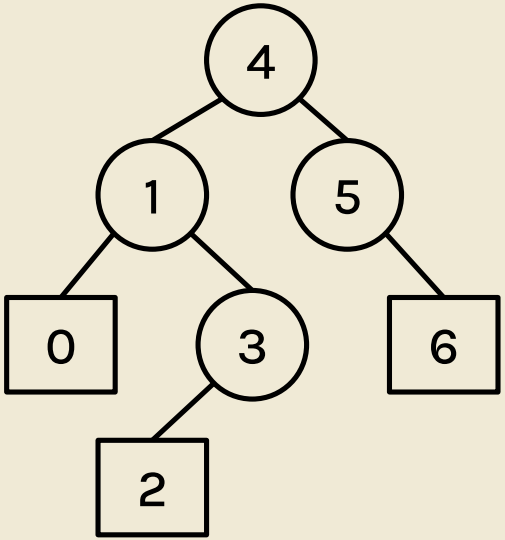


tree traversals (preorder)

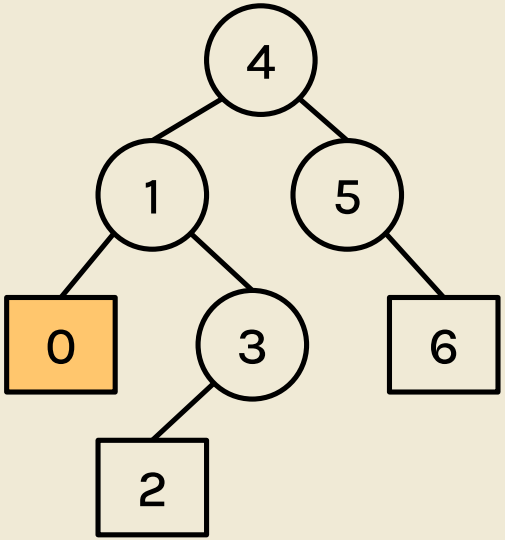


4, 1, 0, 3, 2, 5, 6

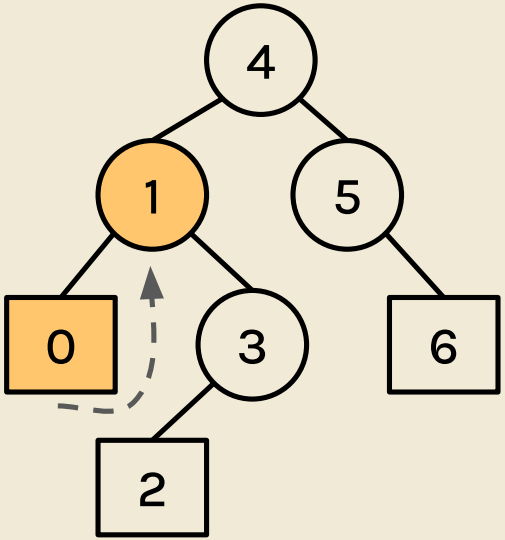
tree traversals (inorder)



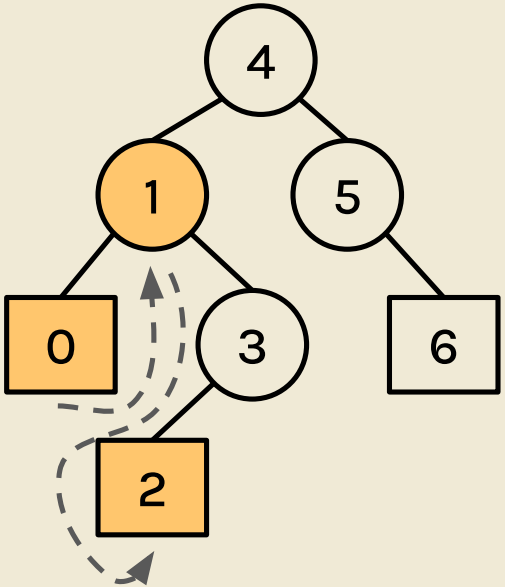
tree traversals (inorder)



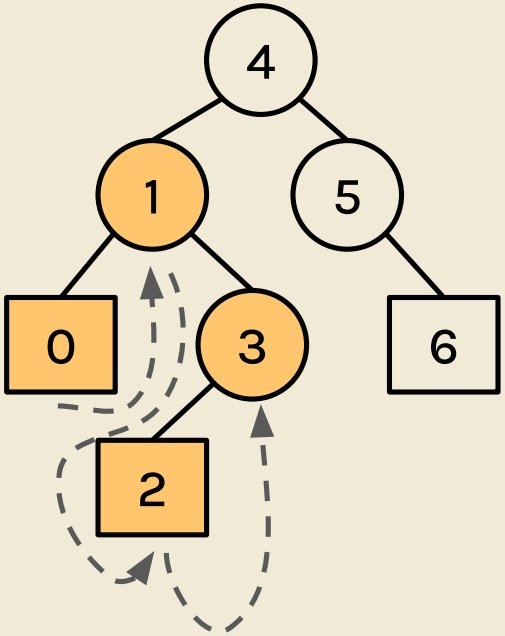
tree traversals (inorder)



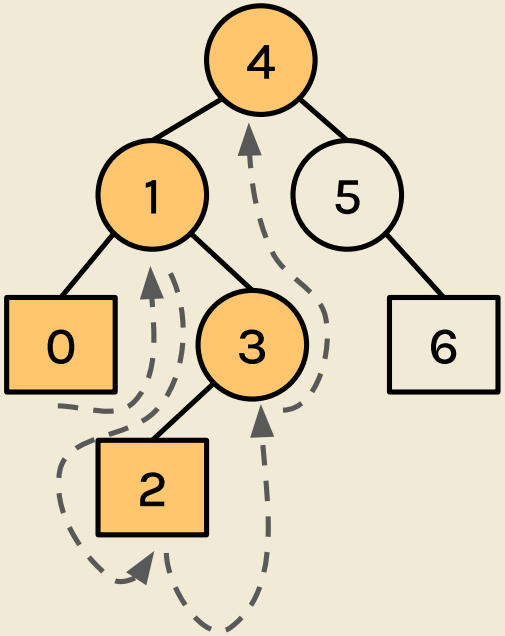
tree traversals (inorder)



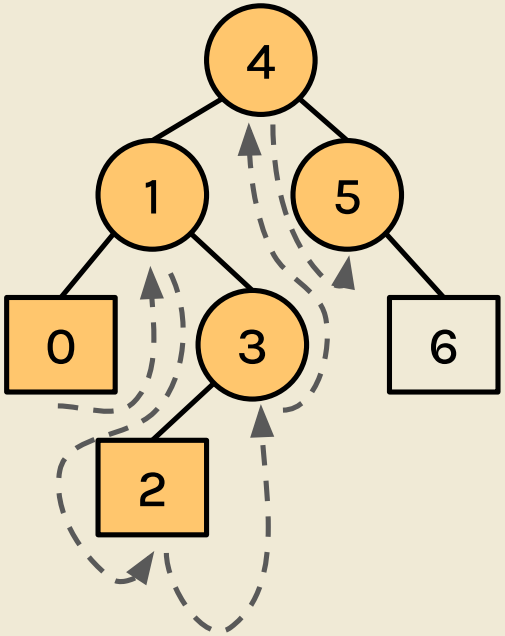
tree traversals (inorder)



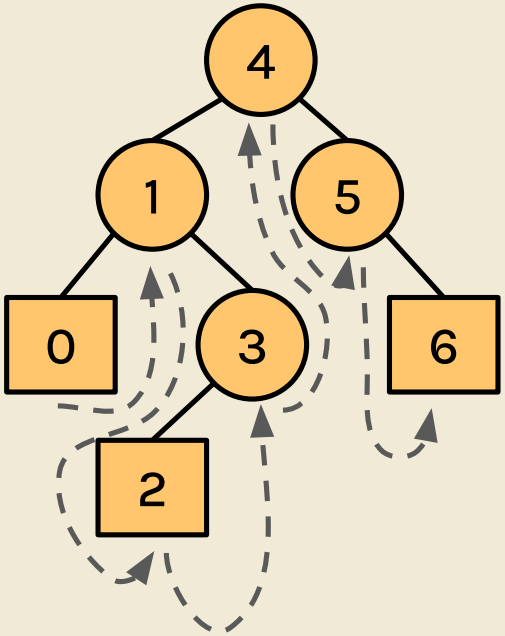
tree traversals (inorder)



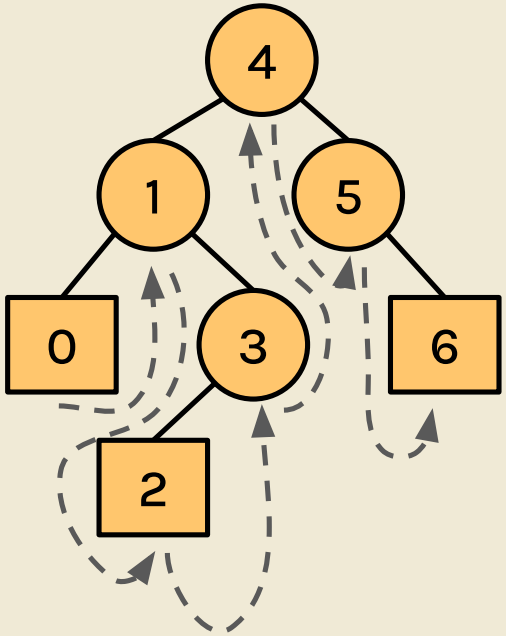
tree traversals (inorder)



tree traversals (inorder)

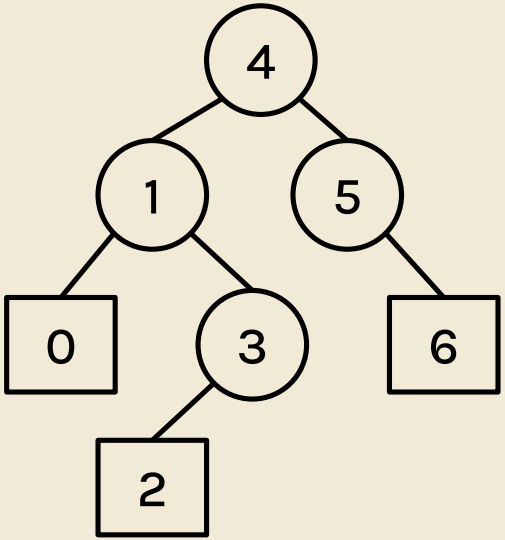


tree traversals (inorder)

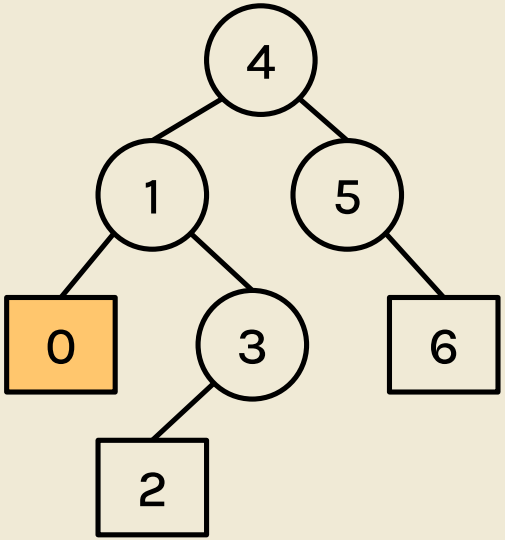


0, 1, 2, 3, 4, 5, 6

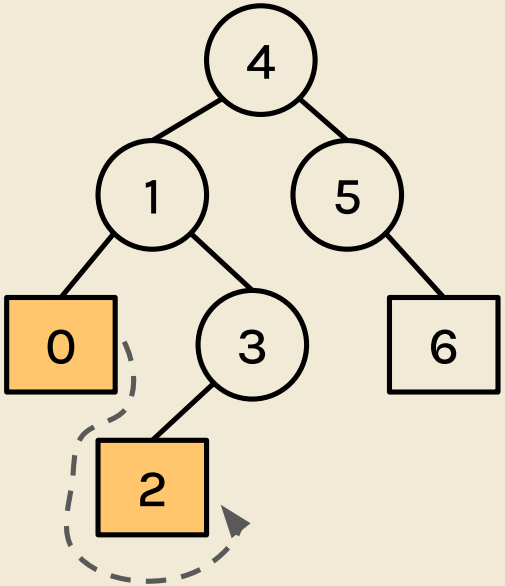
tree traversals (postorder)



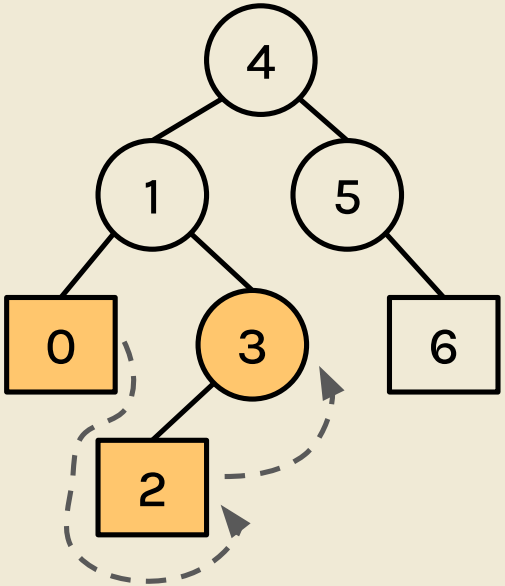
tree traversals (postorder)



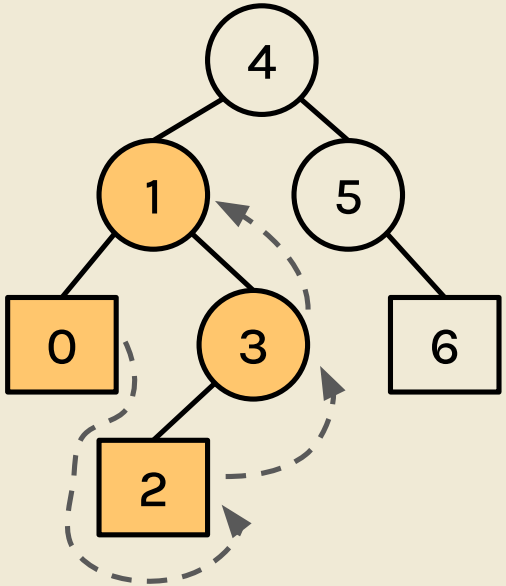
tree traversals (postorder)



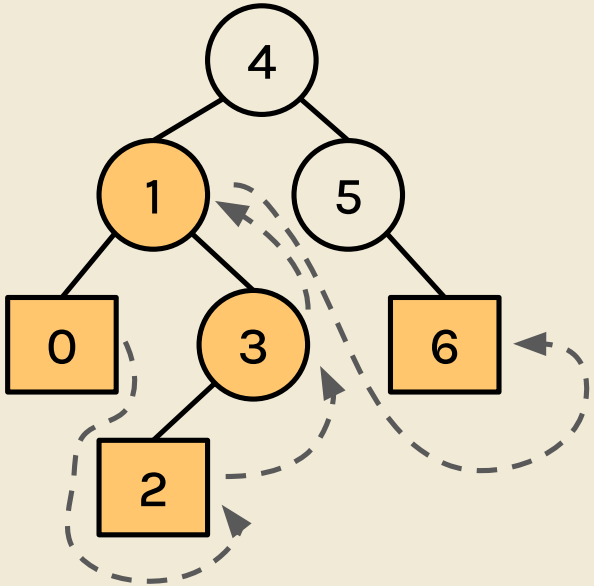
tree traversals (postorder)



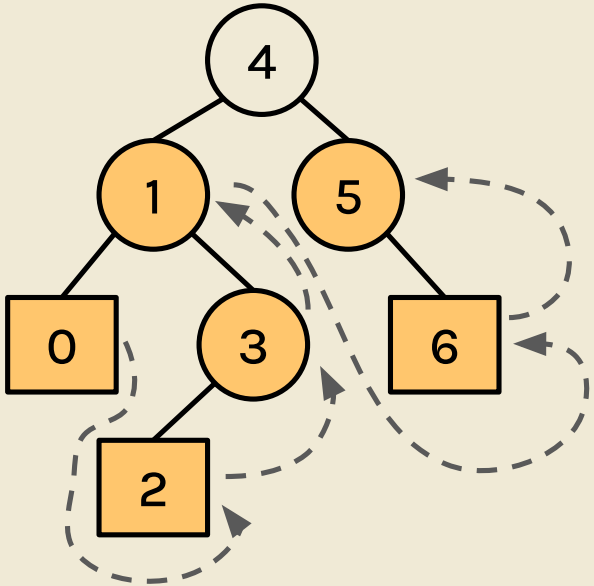
tree traversals (postorder)



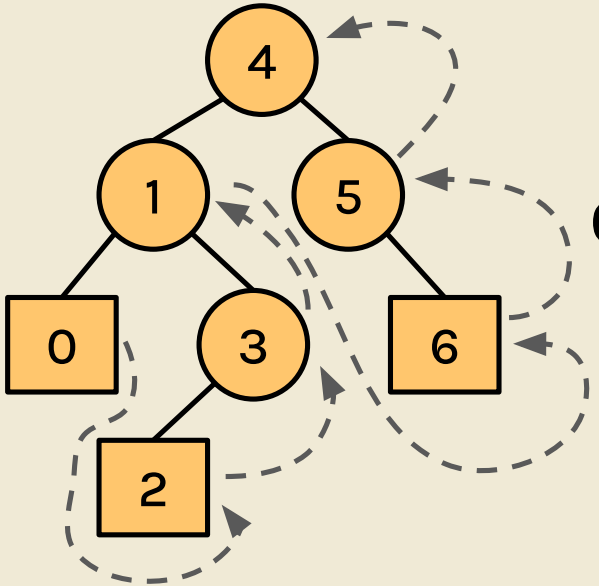
tree traversals (postorder)



tree traversals (postorder)



tree traversals (postorder)

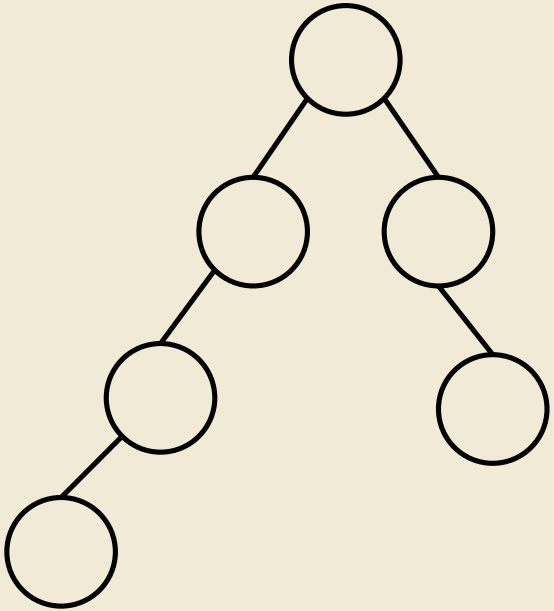


0, 2, 3, 1, 6, 5, 4

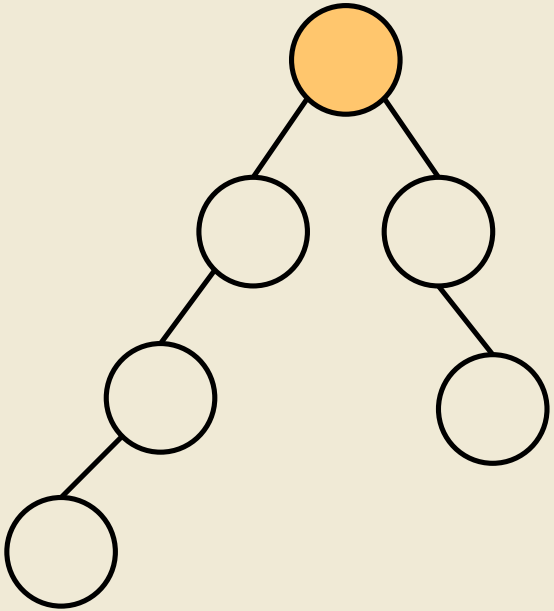
breadth first search

- visit nodes in order of distance to the source
- implemented through a queue

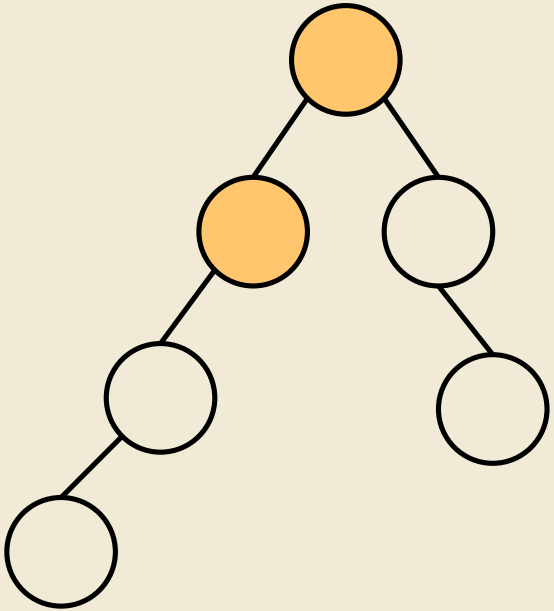
breadth first search



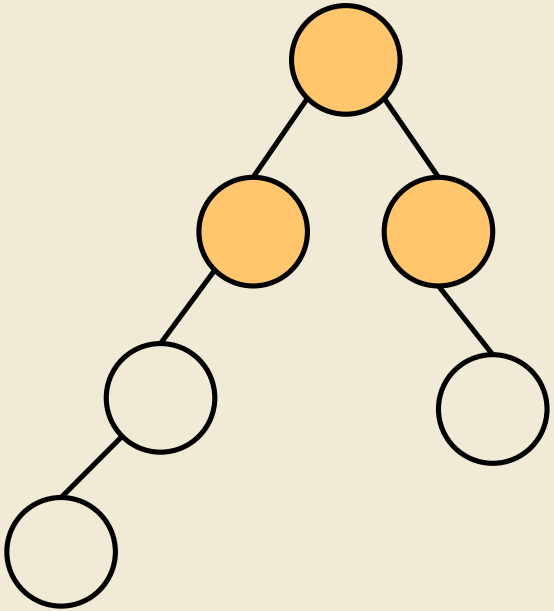
breadth first search



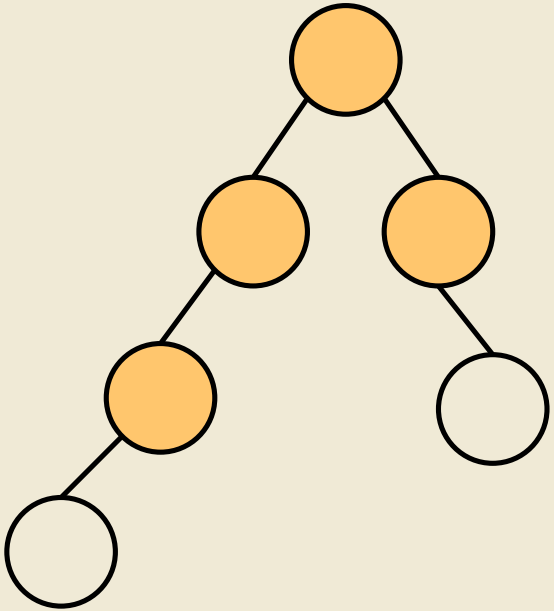
breadth first search



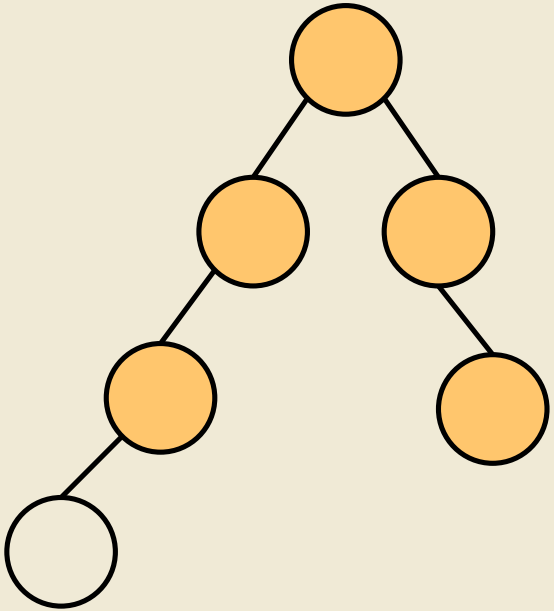
breadth first search



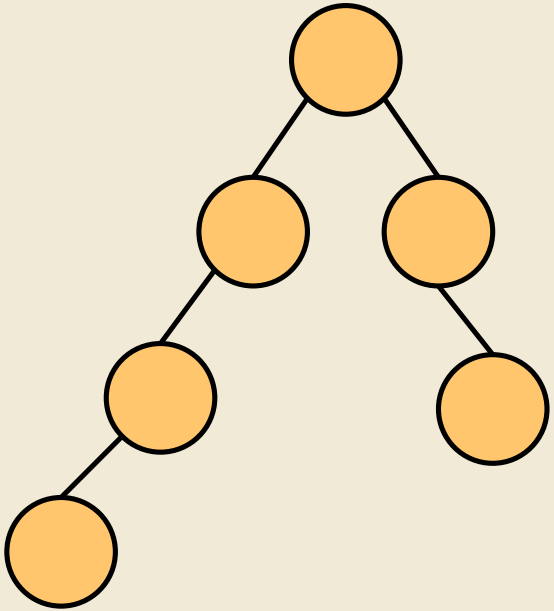
breadth first search



breadth first search



breadth first search



breadth first search (algorithm)

breadth first search (algorithm)

- put the starting node in the queue

breadth first search (algorithm)

- put the starting node in the queue
- pop the first item from the queue (the starting node) and visit it
  - when you visit a node, add all its children/”neighbors” to the queue

breadth first search (algorithm)

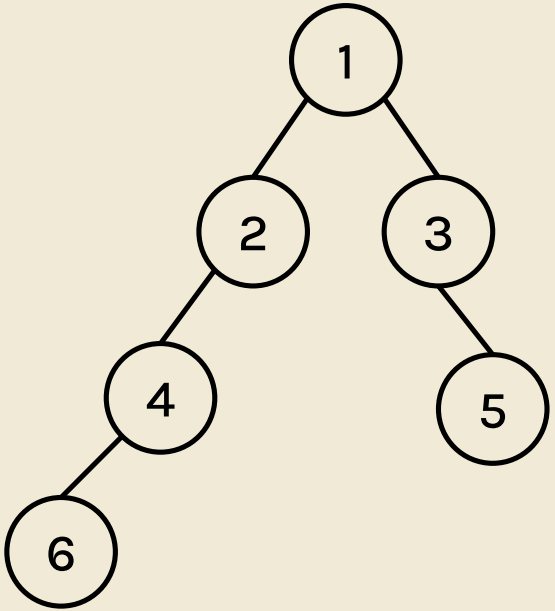
- put the starting node in the queue
- pop the first item from the queue (the starting node) and visit it
  - when you visit a node, add all its children/”neighbors” to the queue
- pop and visit the next item in the queue

breadth first search (algorithm)

- put the starting node in the queue
- pop the first item from the queue (the starting node) and visit it
  - when you visit a node, add all its children/"neighbors" to the queue
- pop and visit the next item in the queue

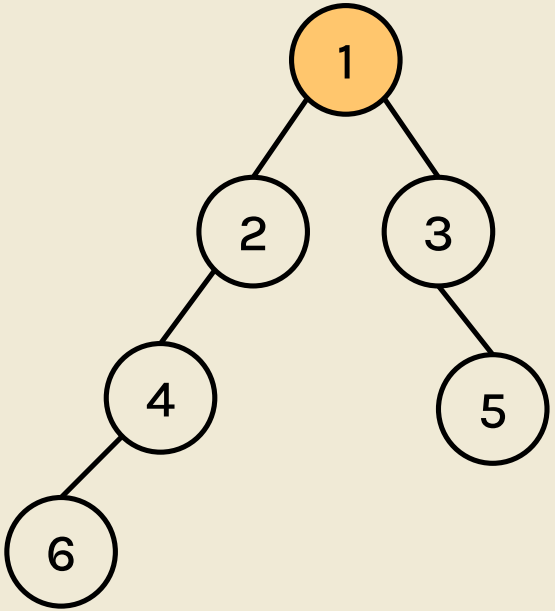
continue until there are no more items in the queue!

breadth first search



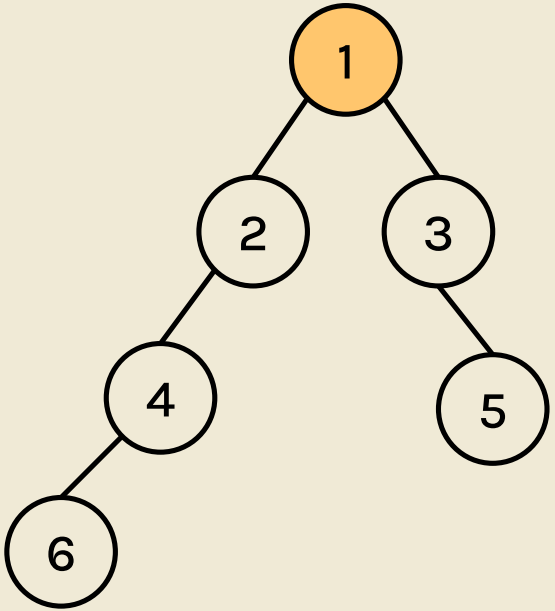
queue: {1}

breadth first search



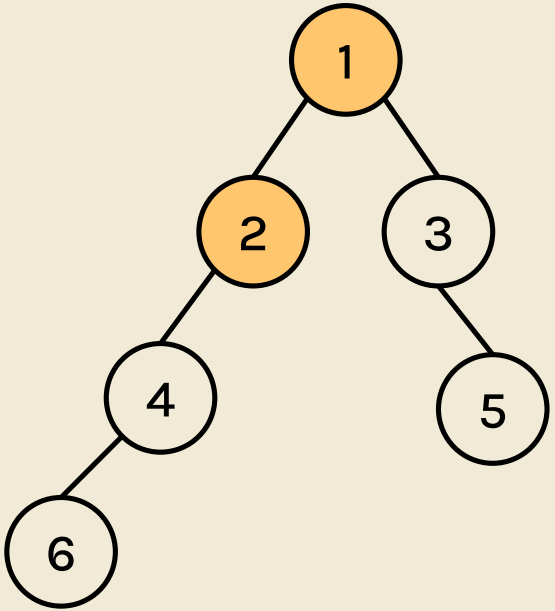
queue: {~~1~~, 2, 3}

breadth first search



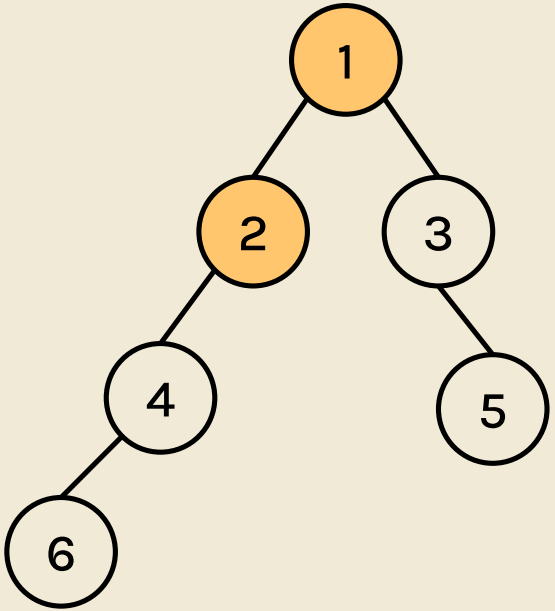
queue: {2, 3}

breadth first search



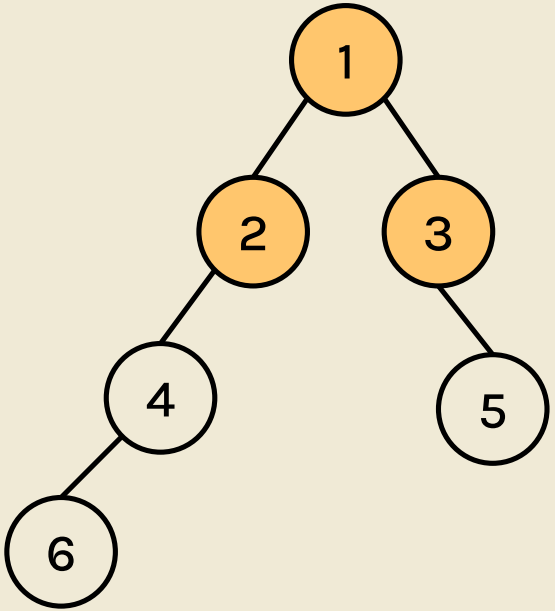
queue: {~~2~~, 3, 4}

breadth first search



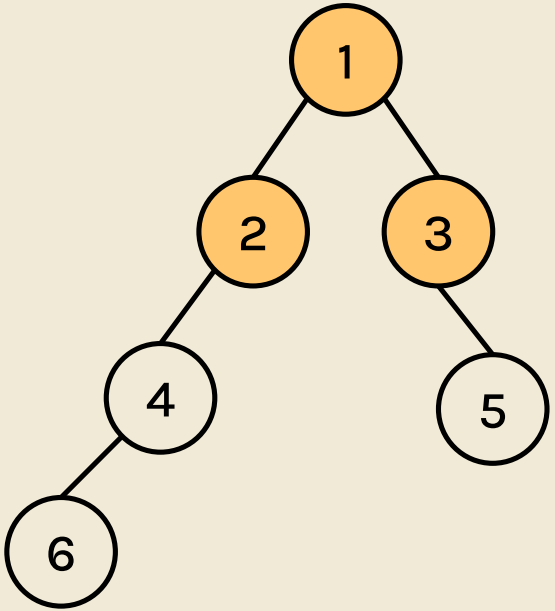
queue: {3, 4}

breadth first search



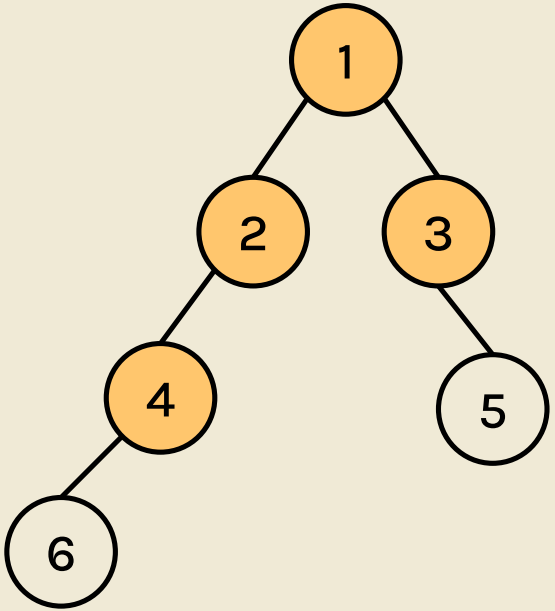
queue: {~~3~~, 4, 5}

breadth first search



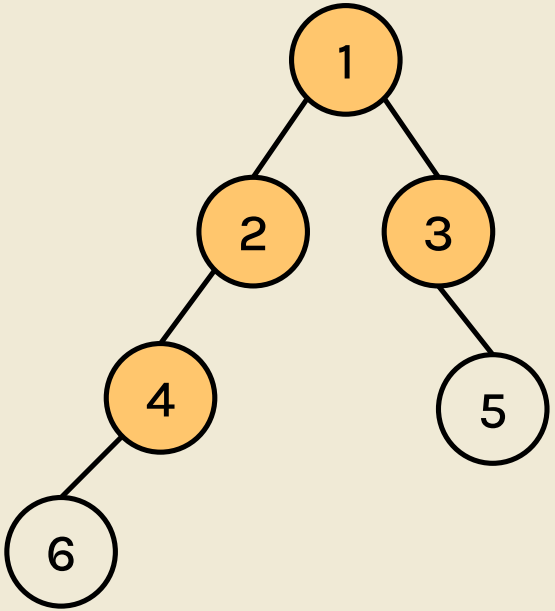
queue: {4, 5}

breadth first search



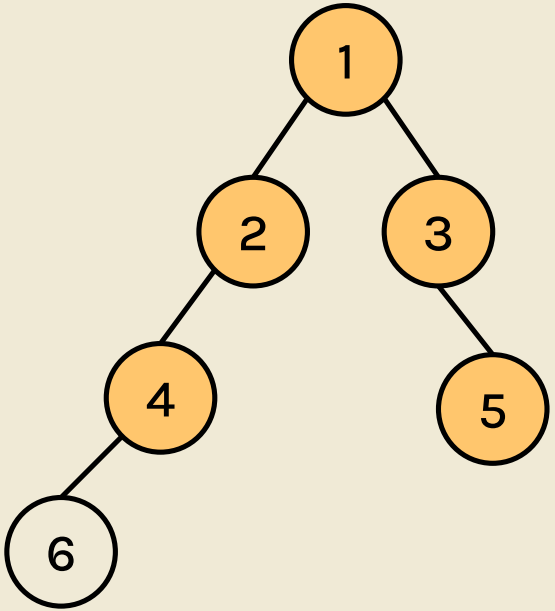
queue: {~~4~~, 5, 6}

breadth first search



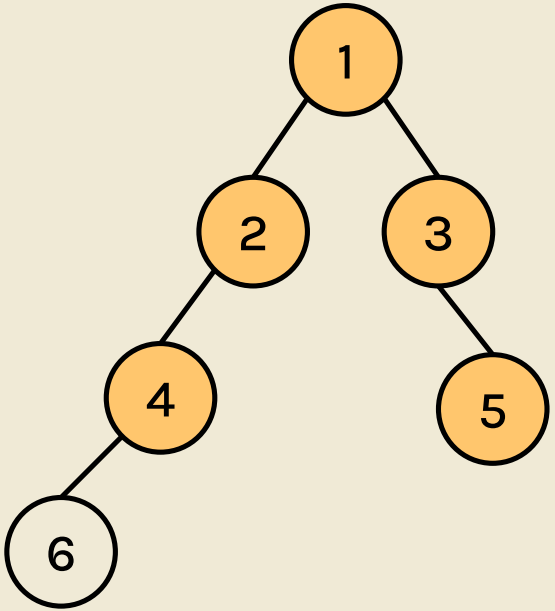
queue: {5, 6}

breadth first search



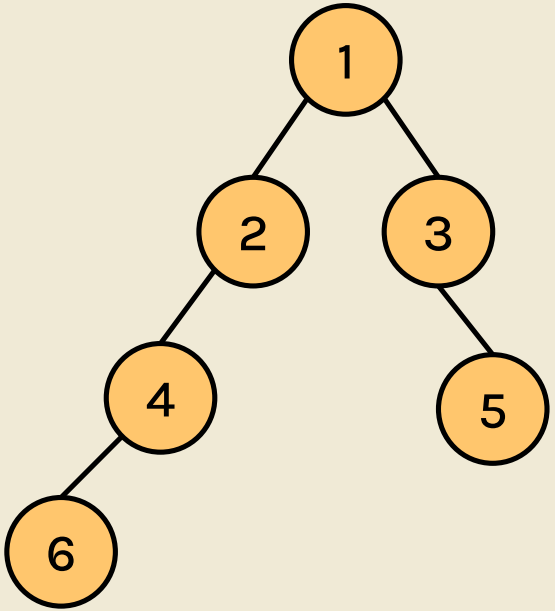
queue: {~~5~~, 6}

breadth first search



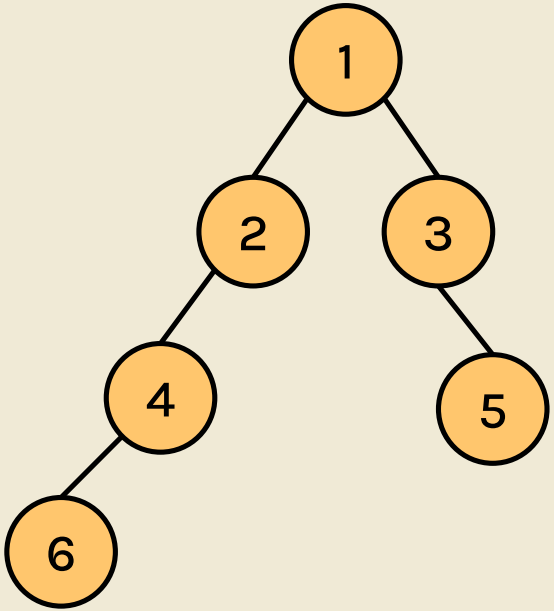
queue: {6}

breadth first search



queue: {~~6~~}

breadth first search



queue: {}

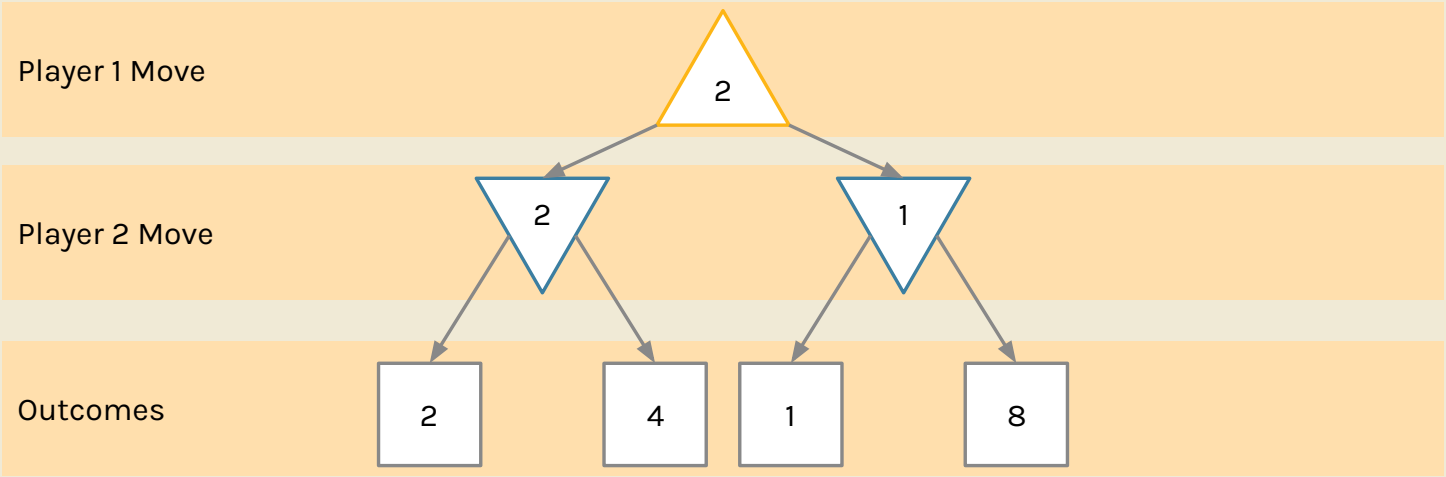
end of iteration!

game trees/min-max trees

- showcases the outcomes of a two player game
  - one player tries to maximize the total score, the other player tries to minimize it

game trees/min-max trees

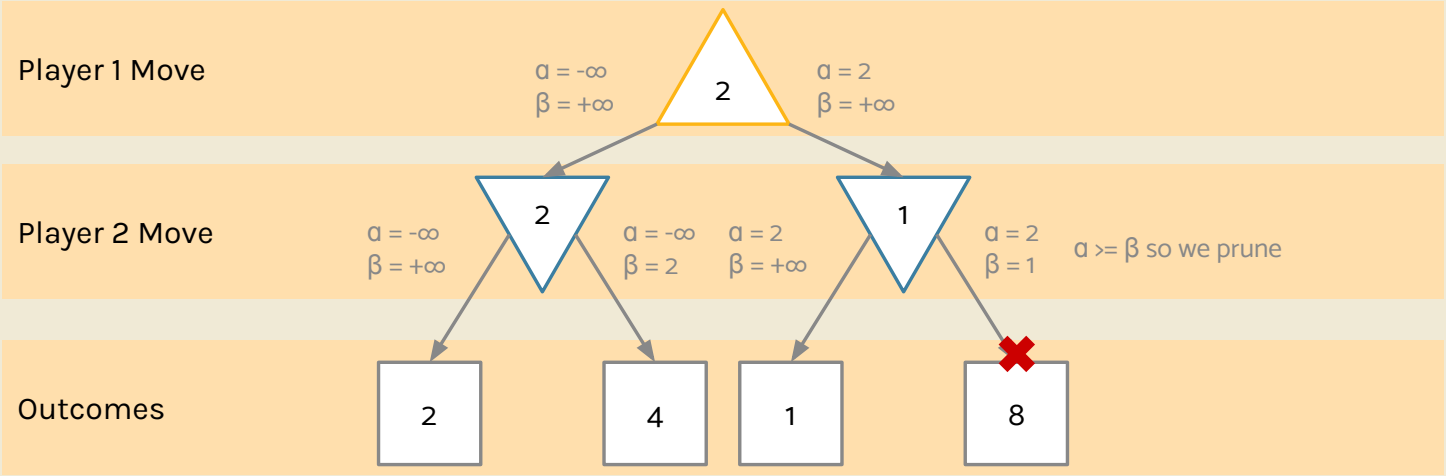
- showcases the outcomes of a two player game



alpha-beta pruning

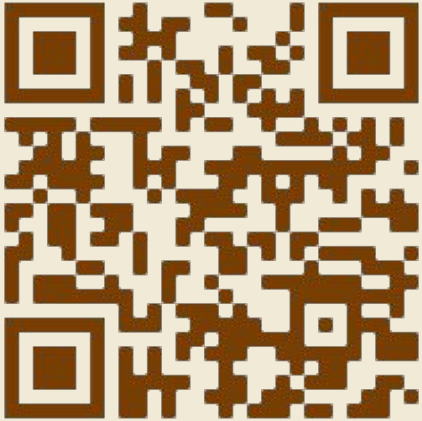
- reduces the number of nodes needed to visit to determine the best possible move for player one
- trees get combinatorially large!

alpha-beta pruning



worksheet
(on 61B website)

attendance
bit.ly/abhi-attendance



feedback
bit.ly/abhi-feedback

slides: **bit.ly/abhi-disc**