Lecture 09
Solving Problems by Searching

Artificial Intelligence

Dr. Ahmed Matreen

Today’s Agenda

- Searching for Solutions
- Uninformed Search Strategies

Searching For Solutions

- Having formulated some problems…how do we solve them?
- Search through a state space
- Use a search tree that is generated with an initial state and successor functions that define the state space

Searching For Solutions

- A ***state*** is (a representation of) a physical configuration

- A ***node*** is a data structure constituting part of a search tree
  - Includes parent, children, depth, path cost

- States do not have children, depth, or path cost

- The EXPAND function creates new nodes, filling in the various fields and using the SUCCESSOR function of the problem to create the corresponding states

Uninformed Search Strategies

- ***<u>Uninformed</u>*** strategies use only the information available in the problem definition
  - Also known as blind searching

- Breadth-first search
- Depth-first search
- Uniform-cost search
- Depth-limited search
- Iterative deepening search

Comparing Uninformed Search Strategies

- Completeness
  - Will a solution always be found if one exists?
- Time
  - How long does it take to find the solution?
  - Often represented as the number of nodes searched
- Space
  - How much memory is needed to perform the search?
  - Often represented as the maximum number of nodes stored at once
- Optimal
  - Will the optimal (least cost) solution be found?

Comparing Uninformed Search Strategies

- Time and space complexity are measured in
  - b – maximum branching factor of the search tree
  - m – maximum depth of the state space (Max path length)

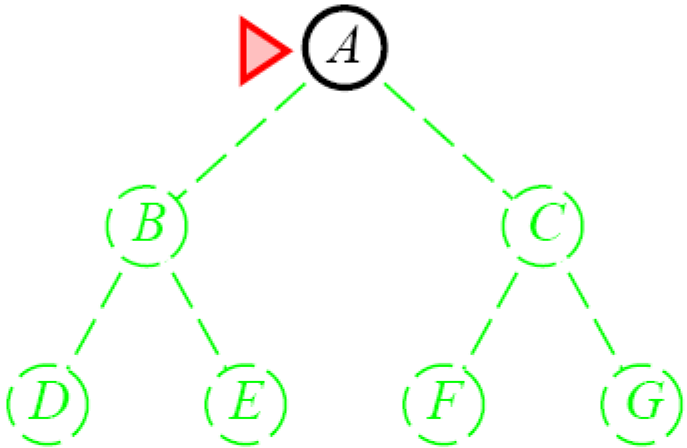
Breadth-First Search

- Recall from Data Structures the basic algorithm for a breadth-first search on a graph or tree

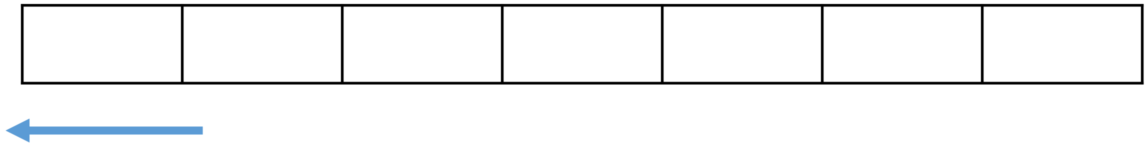
- Expand the ***shallowest*** unexpanded node

- Place all new successors at the end of a FIFO queue

Breadth-First Search

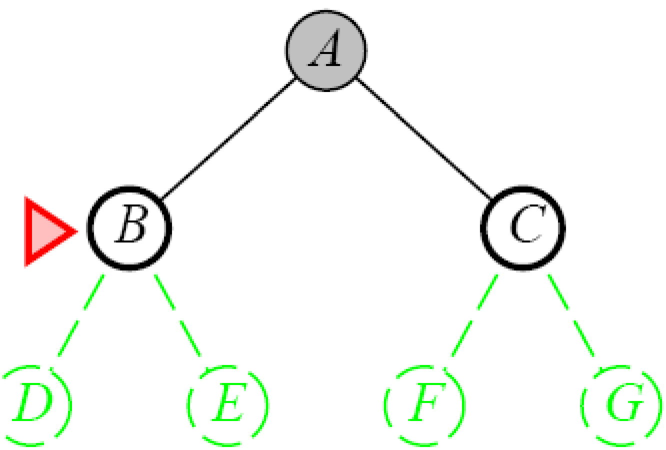


Queue-FIFO List



Front

Breadth-First Search

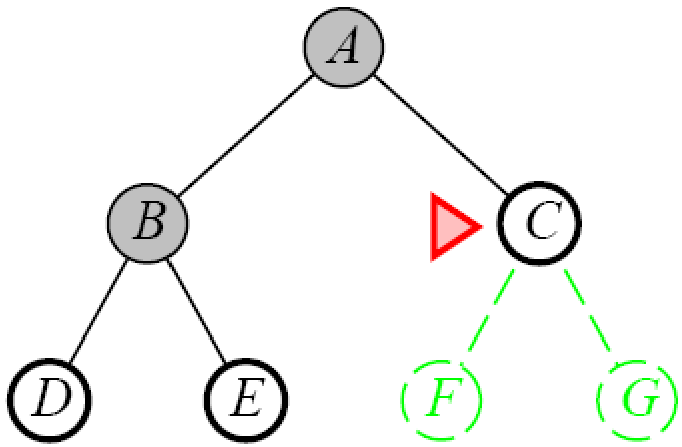


Queue-FIFO List

A						

Front

Breadth-First Search

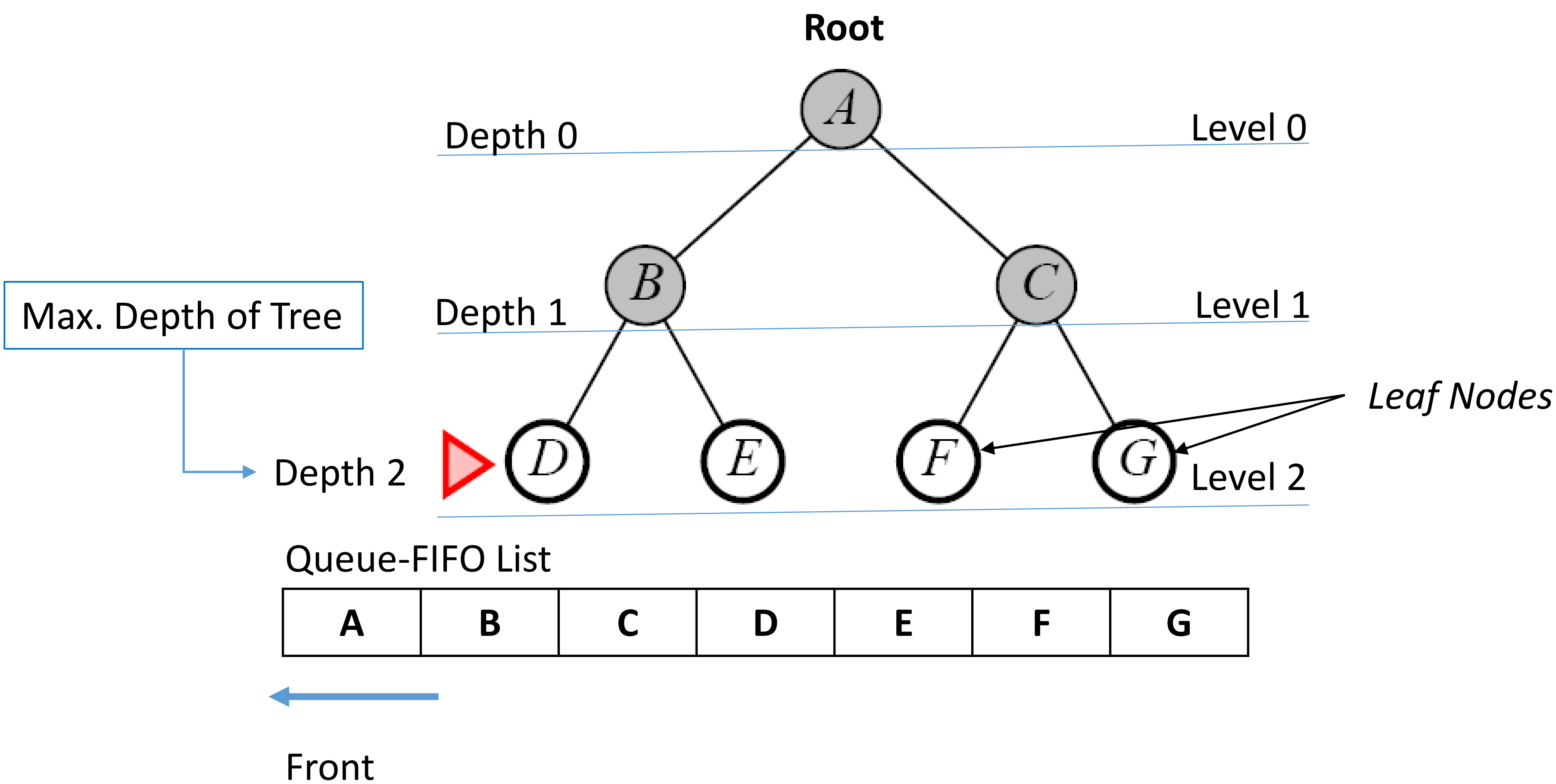


Queue-FIFO List

A	**B**					

Front

Breadth-First Search



Properties of Breadth-First Search

- Complete
  - Yes if b (max branching factor) is finite
  - But it wouldn't be if the branching factor for any node was infinite
- Time
  - $1 + b + b^2 + \ldots + b^d + b(b^m\text{-}1) = O(b^{m+1})$
  - $O(b^{m+1})$: Must examine every node in the tree
  - exponential in m

Properties of Breadth-First Search

- Space
  - $O(b^{m+1})$
  - Keeps every node in memory
  - This is the big problem; an agent that generates nodes at 10 MB/sec will produce 864000 MBs in 24 hours
- Optimal
  - Yes (if cost is 1 per step); not optimal in general

Using Breadth-First Search

- When is BFS appropriate?
  - space is not a problem
  - it's necessary to find the solution with the fewest arcs
  - although all solutions may not be shallow, at least some are
- When is BFS inappropriate?
  - space is limited
  - all solutions tend to be located deep in the tree
  - the branching factor is very large

Lessons From Breadth First Search

- The memory requirements are a bigger problem for breadth-first search than is execution time

- Exponential-complexity search problems cannot be solved by uniformed methods for any but the smallest instances