Learning Objectives

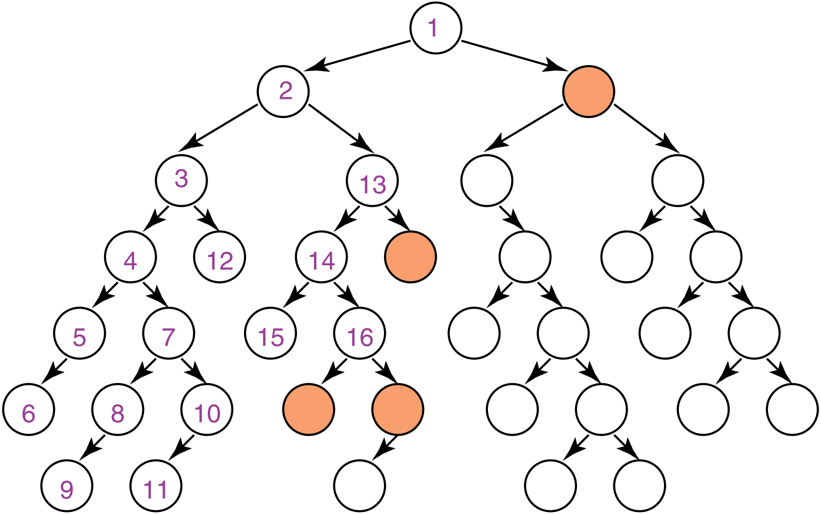
At the end of the class you should be able to:

- demonstrate how depth-first search will work on a graph
- demonstrate how breadth-first search will work on a graph
- predict the space and time requirements for depth-first and breadth-first searches

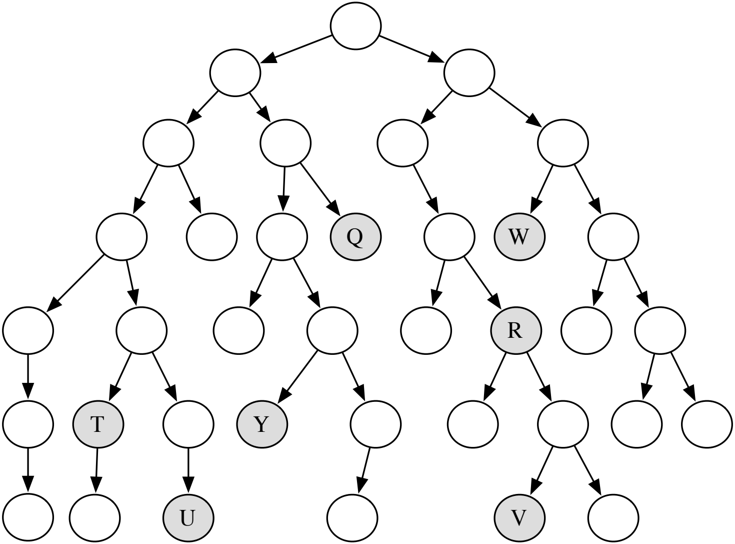
Depth-first Search

- Depth-first search treats the frontier as a stack
- It always selects one of the last elements added to the frontier.
- If the list of paths on the frontier is $[p_1, p_2, \ldots]$
    - p_1 is selected. Paths that extend p_1 are added to the front of the stack (in front of p_2).
    - p_2 is only selected when all paths from p_1 have been explored.

Illustrative Graph — Depth-first Search



Which shaded goal will a depth-first search find first?



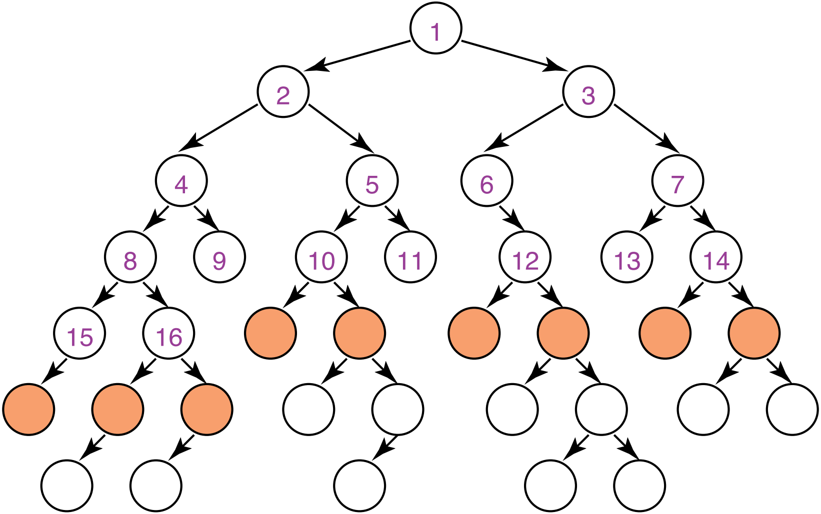
Complexity of Depth-first Search

- Does depth-first search guarantee to find the shortest path or the path with fewest arcs?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?
- What is the space complexity as a function of length of the path selected?
- How does the goal affect the search?

Breadth-first Search

- Breadth-first search treats the frontier as a queue.
- It always selects one of the earliest elements added to the frontier.
- If the list of paths on the frontier is $[p_1, p_2, \ldots, p_r]$:
  - p_1 is selected. Its neighbors are added to the end of the queue, after p_r.
  - p_2 is selected next.

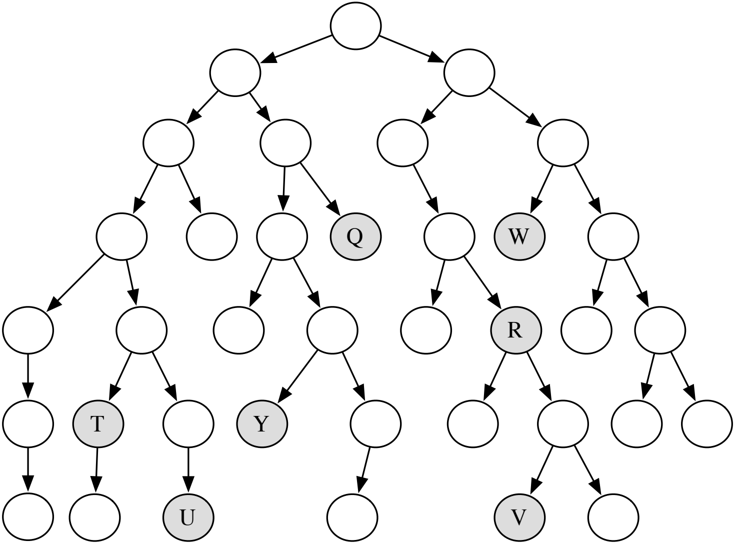
Illustrative Graph — Breadth-first Search



Complexity of Breadth-first Search

- Does breadth-first search guarantee to find the shortest path or the path with fewest arcs?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of the length of the path selected?
- What is the space complexity as a function of the length of the path selected?
- How does the goal affect the search?

Which shaded goal will a breadth-first search find first?



Lowest-cost-first Search

- Sometimes there are costs associated with arcs. The cost of a path is the sum of the costs of its arcs.

$$cost(\langle n_0, \ldots, n_k \rangle) = \sum_{i=1}^{k} |\langle n_{i-1}, n_i \rangle|$$

  An optimal solution is one with minimum cost.
- At each stage, lowest-cost-first search selects a path on the frontier with lowest cost.
- The frontier is a priority queue ordered by path cost.
- It finds a least-cost path to a goal node.
- When arc costs are equal $\Longrightarrow$ breadth-first search.

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added			
Breadth-first	First node added			
Lowest-cost-first	Minimal $cost(p)$			

Complete — if there a path to a goal, it can find one, even on infinite graphs.
Halts — on finite graph (perhaps with cycles).
Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	Exp

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path