Depth-first Search

Depth-first search treats the frontier as a stack: it always selects the last element added to the frontier.

$$select(Node, [Node|Frontier], Frontier).$$

$$add\_to\_frontier(Neighbors, Frontier_1, Frontier_2) \leftarrow$$
$$append(Neighbors, Frontier_1, Frontier_2).$$

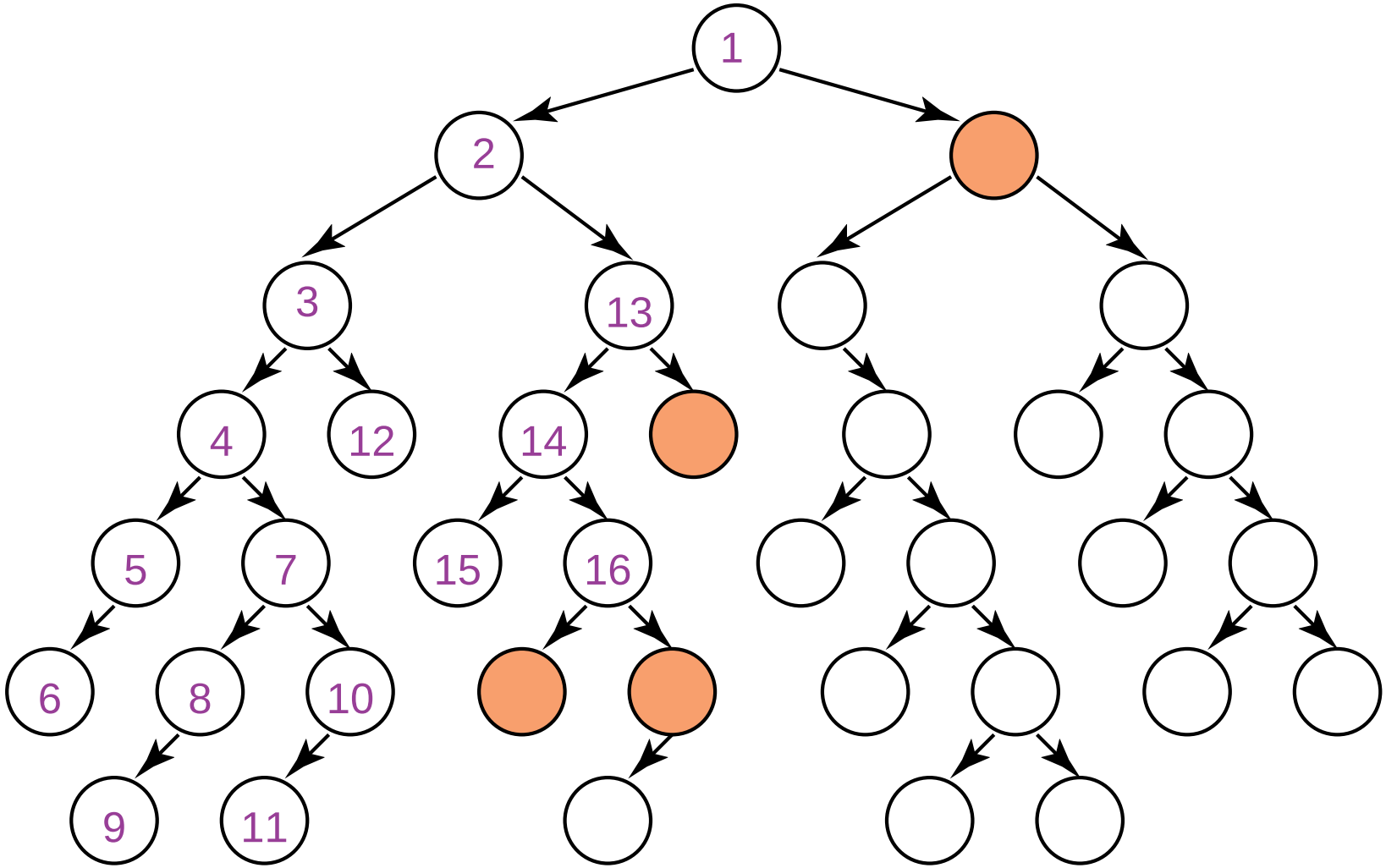
Frontier: $[e_1, e_2, \ldots]$

e_1 is selected. Its neighbors are added to the front of the stack.

e_2 is only selected when all paths from e_1 have been explored.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

Illustrative Graph — Depth-first Search



© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

Complexity of Depth-first Search

- Depth-first search isn't guaranteed to halt on infinite graphs or graphs with cycles.
- The space complexity is linear in the size of the path being explored.
- Search is unconstrained by the goal until it happens to stumble on the goal.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

Breadth-first Search

Breadth-first search treats the frontier as a queue: it always selects the earliest element added to the frontier.

$$select(Node, [Node|Frontier], Frontier).$$

$$add\_to\_frontier(Neighbors, Frontier_1, Frontier_2) \leftarrow$$
$$append(Frontier_1, Neighbors, Frontier_2).$$

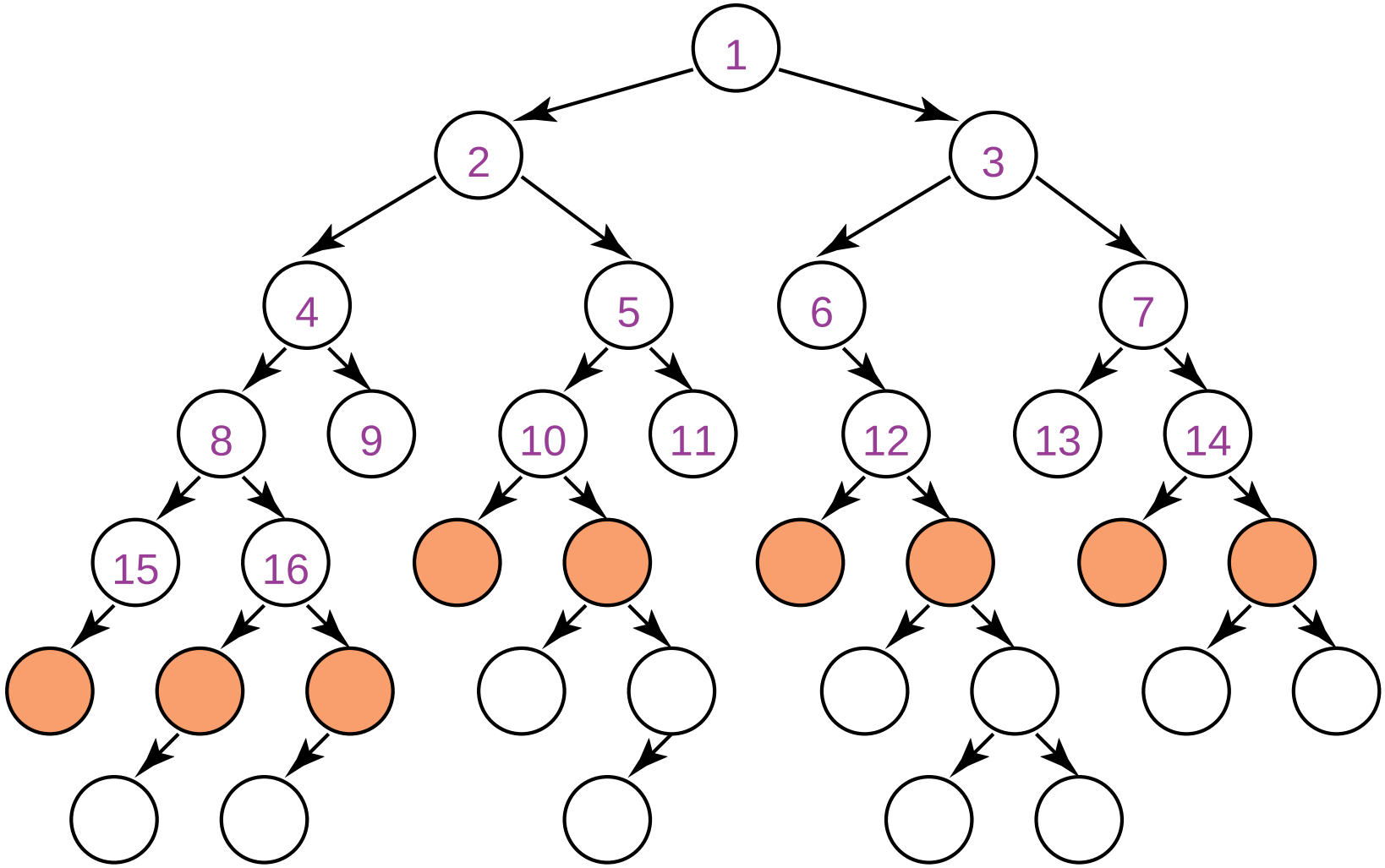
Frontier: $[e_1, e_2, \ldots]$

e_1 is selected. Its neighbors are added to the end of the queue.

e_2 is selected next.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

Illustrative Graph — Breadth-first Search



© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

Complexity of Breadth-first Search

- The branching factor of a node is the number of its neighbors.
- If the branching factor for all nodes is finite, breadth-first search is guaranteed to find a solution if one exists.
  It is guaranteed to find the path with fewest arcs.
- Time complexity is exponential in the path length:
  b^n, where b is branching factor, n is path length.
- The space complexity is exponential in path length: b^n.
- Search is unconstrained by the goal.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

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.
- Lowest-cost-first search finds the shortest path to a goal node.
- At each stage, it selects the shortest path on the frontier.
- The frontier is implemented as a priority queue ordered by path length.
- When arc costs are equal $\Longrightarrow$ breadth-first search.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998