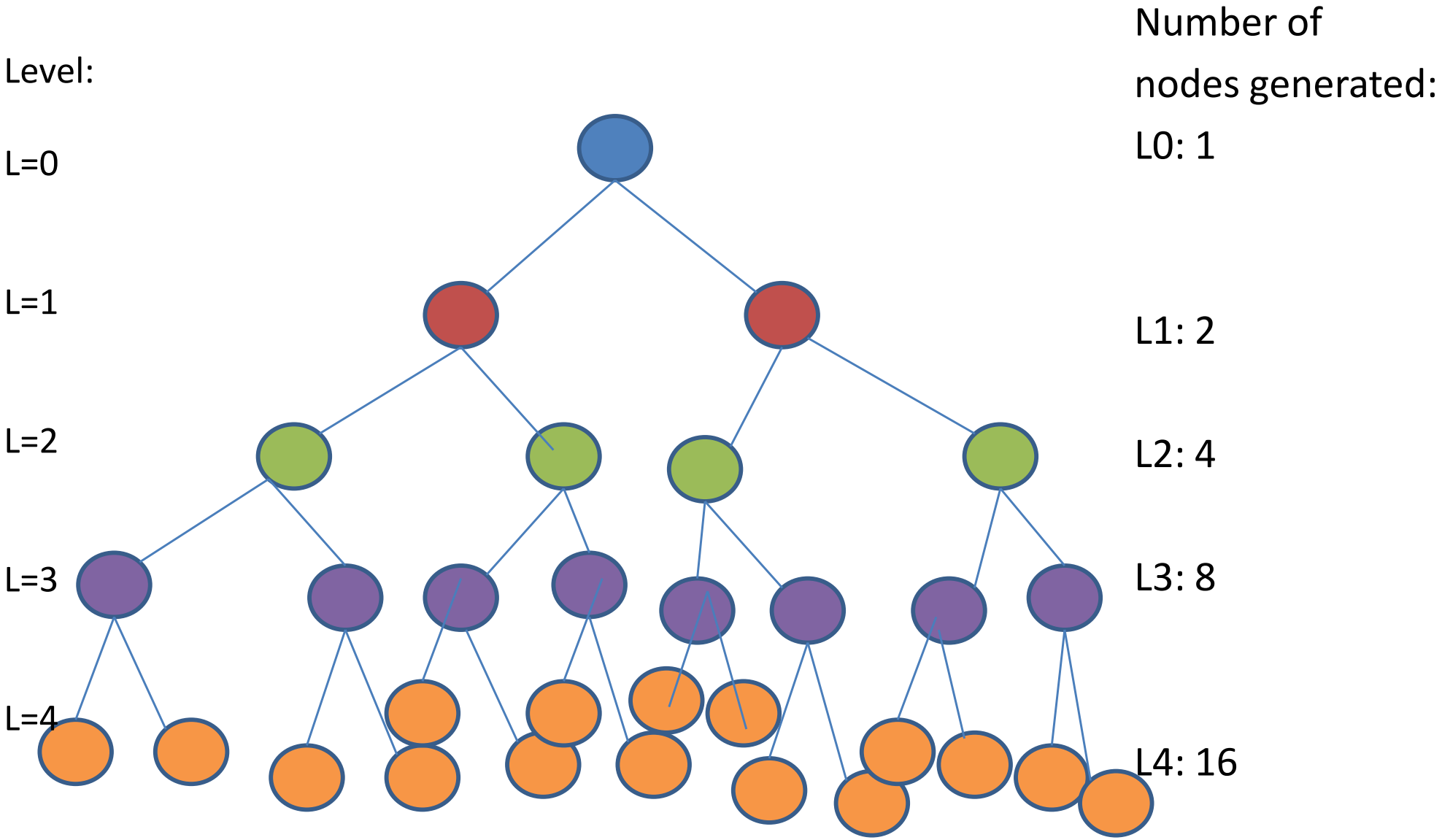
Breadth-first search



Breadth-first search

Level:

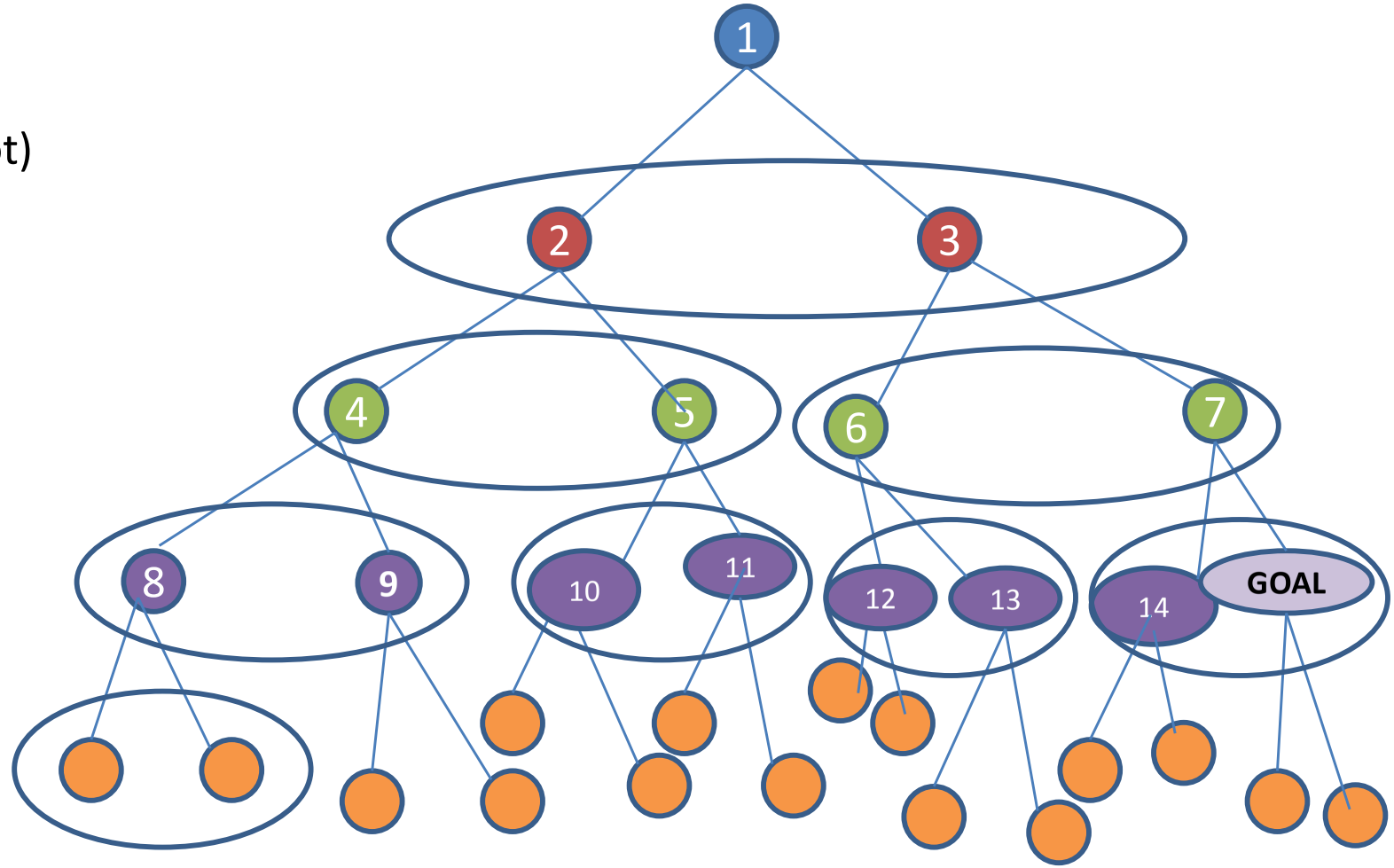
L=0
(root)

L=1

L=2

L=3

L=4



Number of nodes generated:
L0: 1

L1: 2

L2: 4

L3: 8

L4: 16

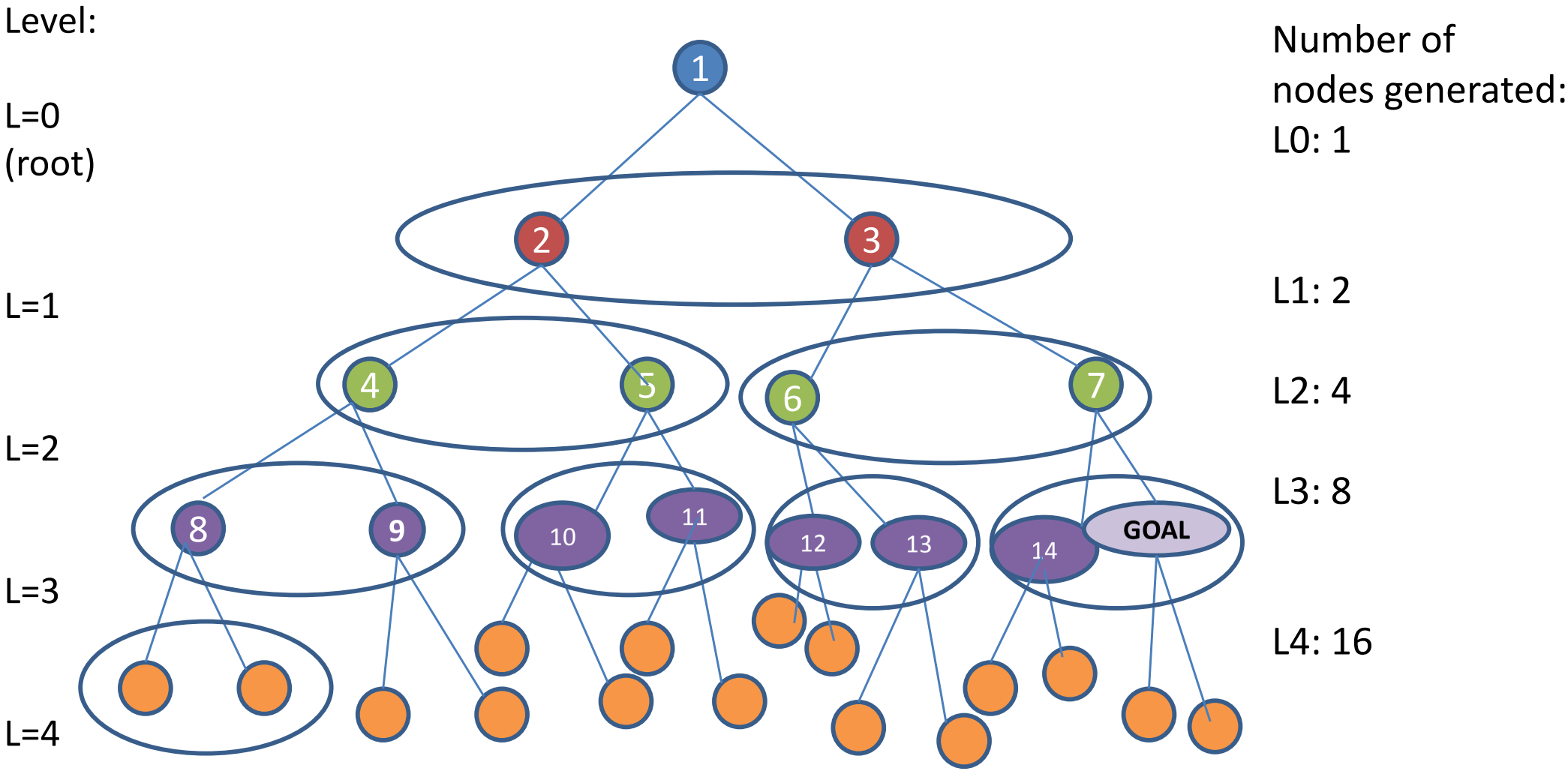
Every state has b=2 successors:

- At root level search tree generates 2 nodes (each of which generates 2 more nodes, so…)
- At 1st level search tree generates 4 nodes (…)
- At 2nd level search tree generates 8 nodes (…)
- At 3rd level search tree generates 16 - 2 nodes (GOAL node is at this level)

Time/space complexity:

$$b + b^2 + b^3 + (b^4 - b) = O(b^{L+1})$$

Breadth-first search



Explore node(s) first, than add successor nodes in the fringe:

Every state has b=2 successors:

After examining root level : generates 2 nodes

After examining 1st level: generates 4 nodes

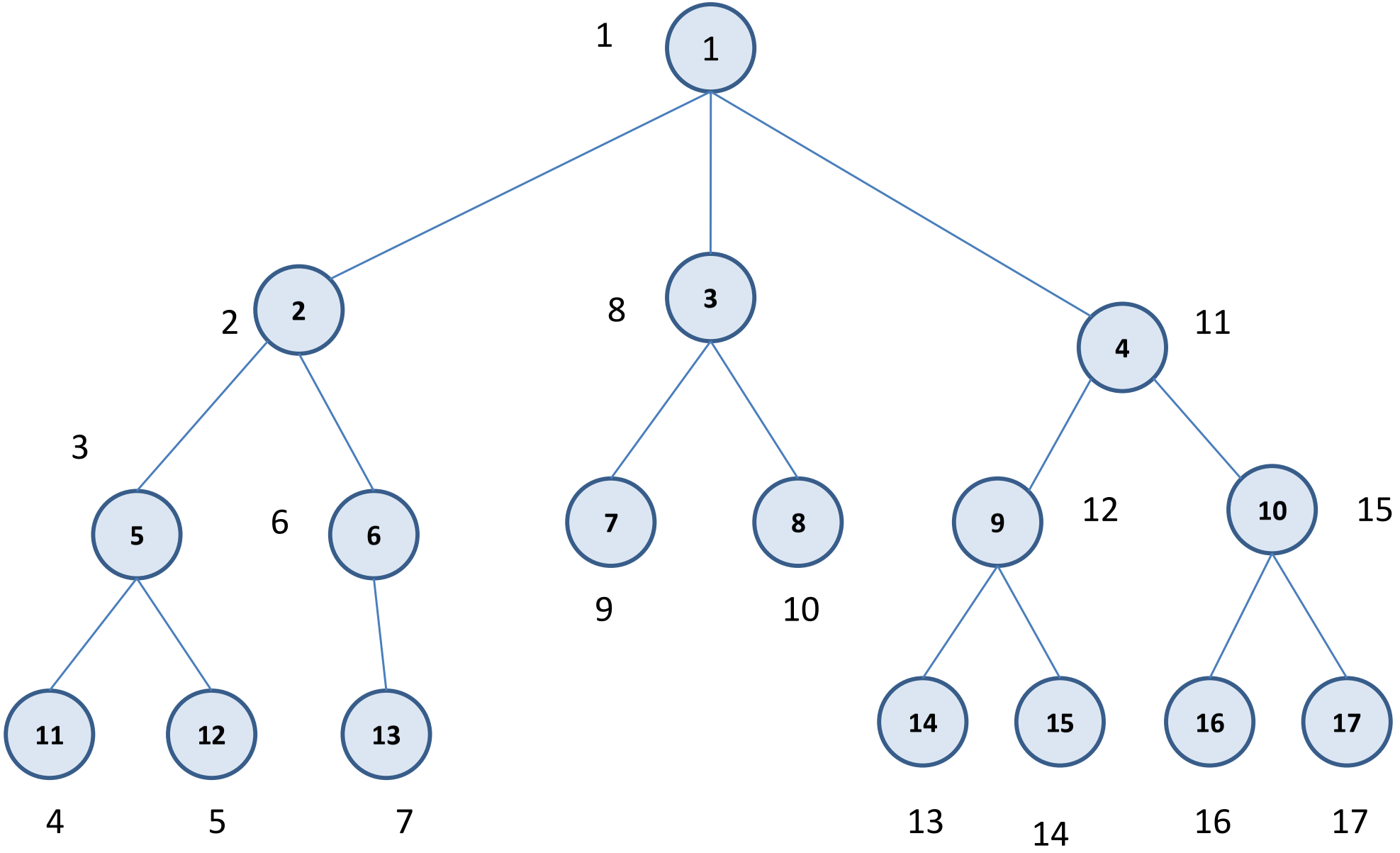
After examining 2nd level: generates 8 nodes

Examining 3rd level – GOAL node found!!!!!

Time/space complexity:

$$b + b^2 + b^3 = O(b^1)$$

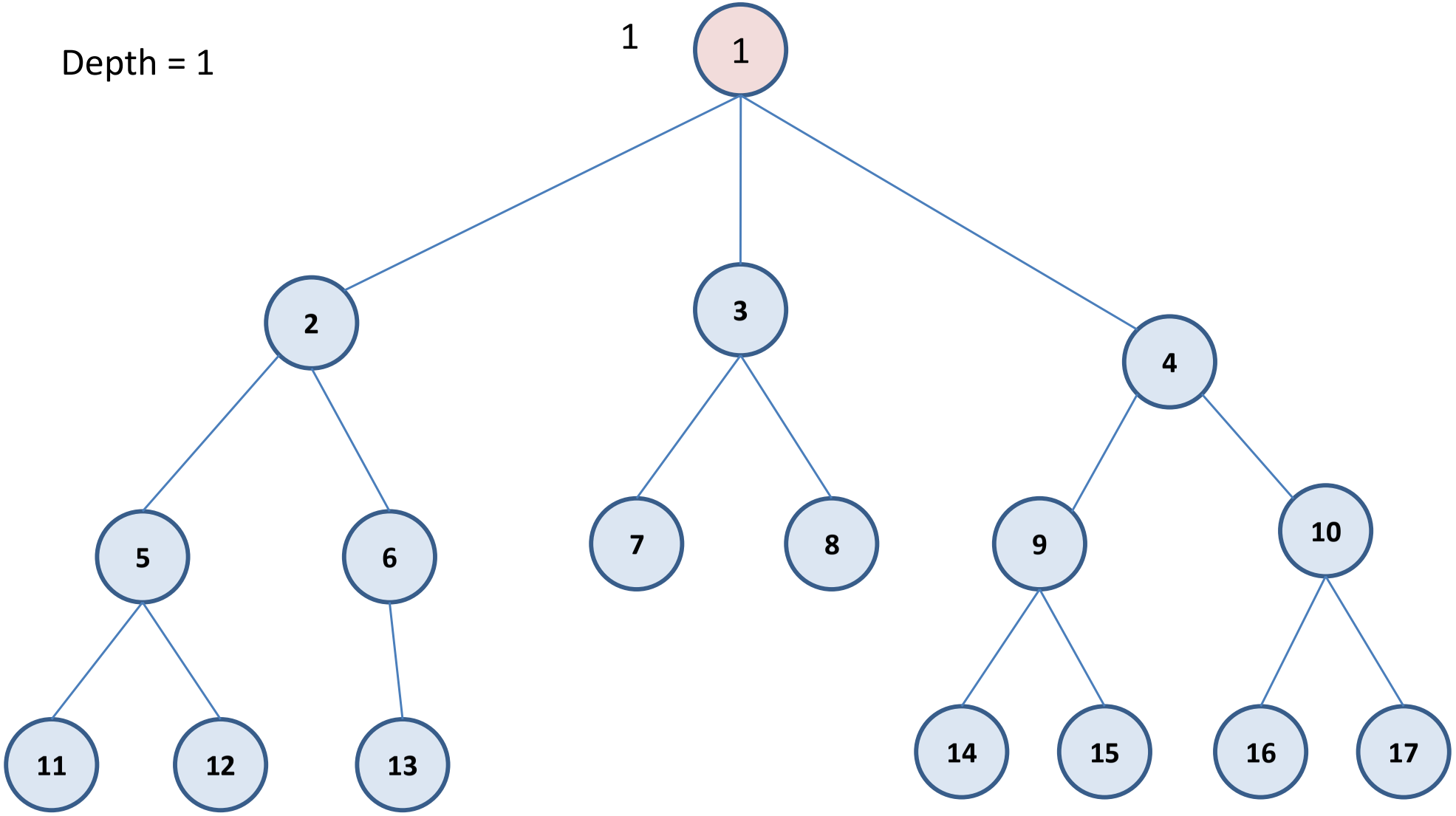
Depth-First Search



Order: 1, 2, 5, 11, 12, 6, 13, 3, 7, 8, 4, 9, 14, 15, 10, 16, 17

Depth-First Iterative-Deepening Search

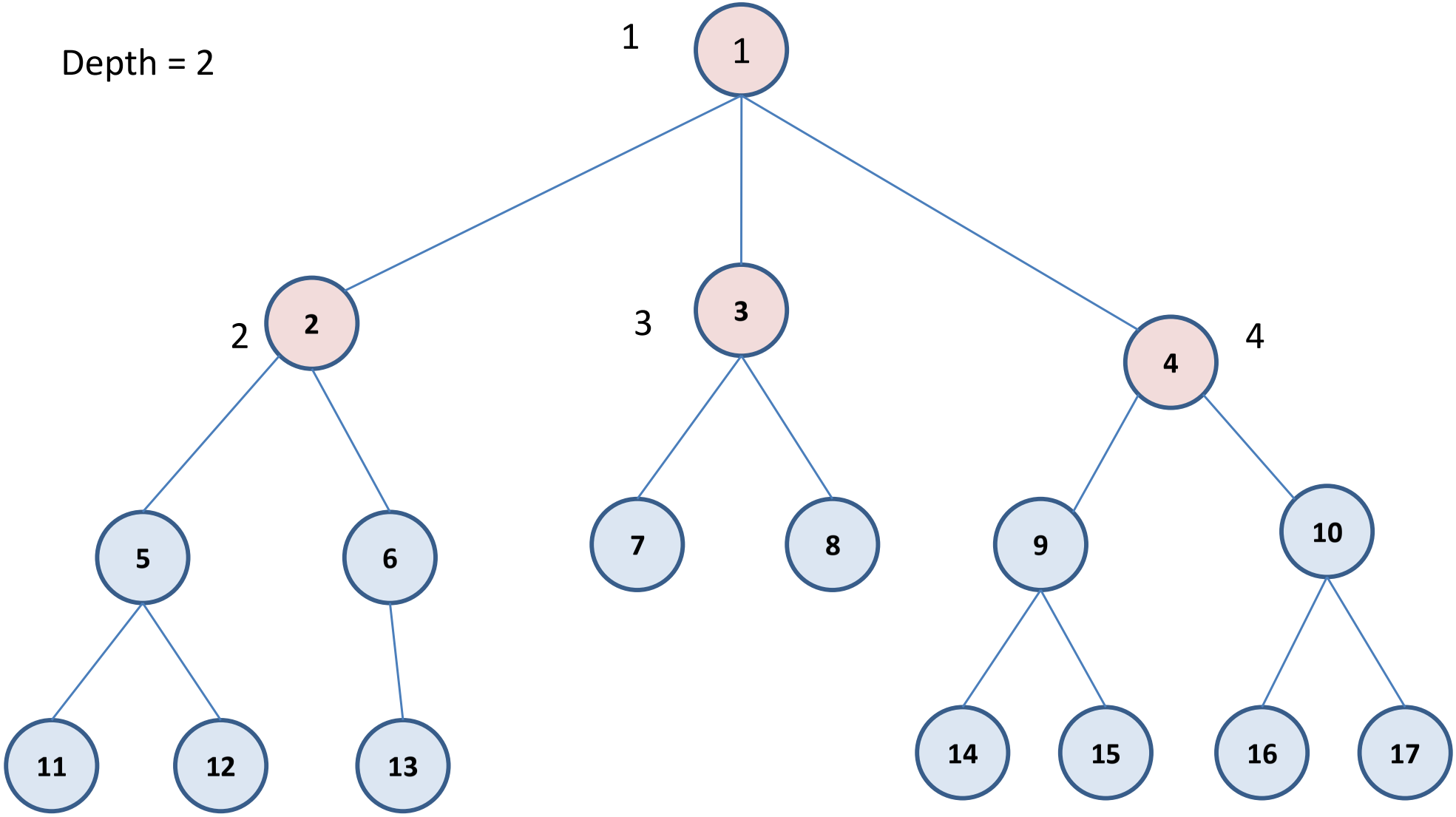
Depth = 1



Order: 1

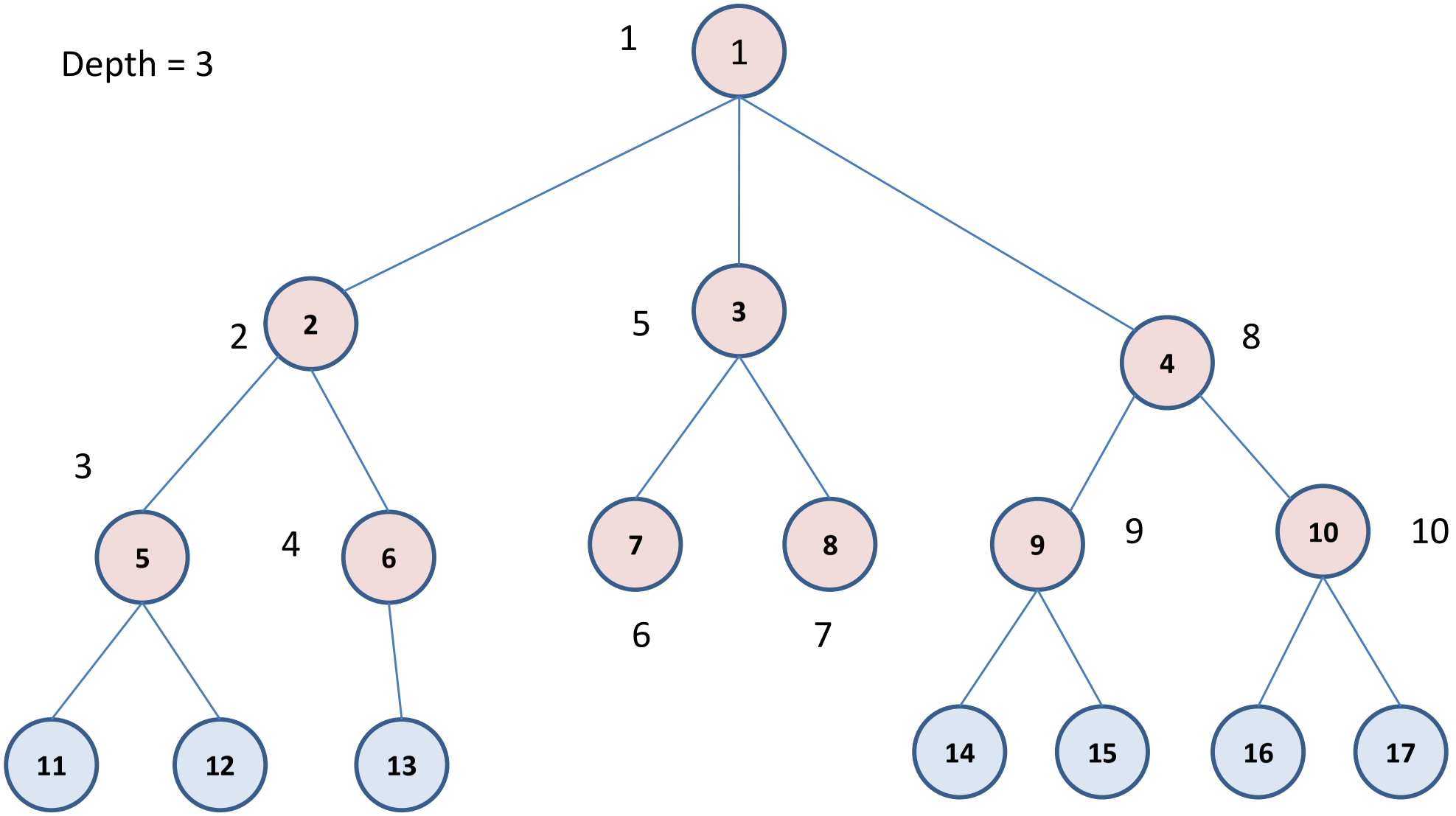
Depth-First Iterative-Deepening Search

Depth = 2



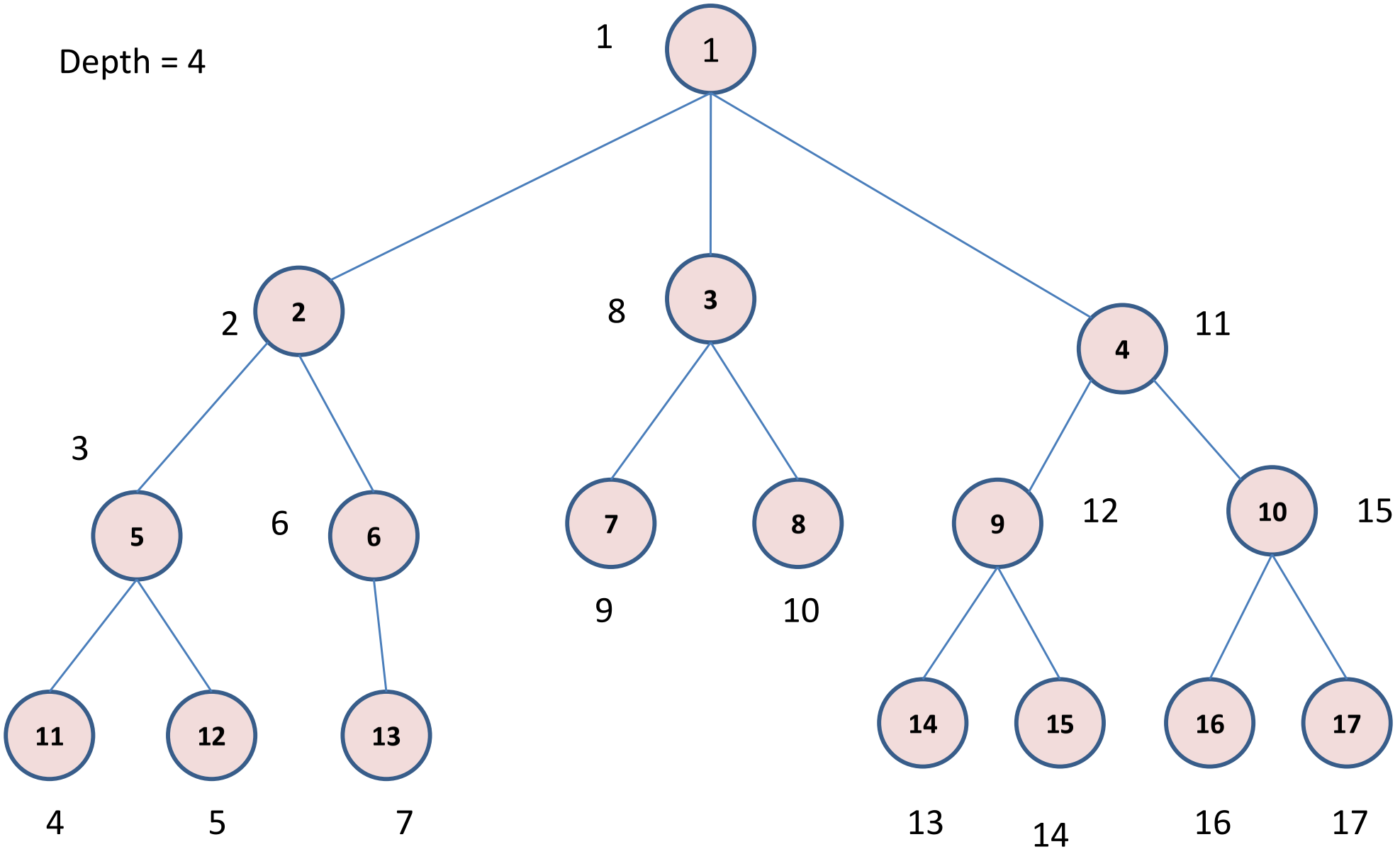
Order: 1, 2, 3, 4

Depth-First Iterative-Deepening Search



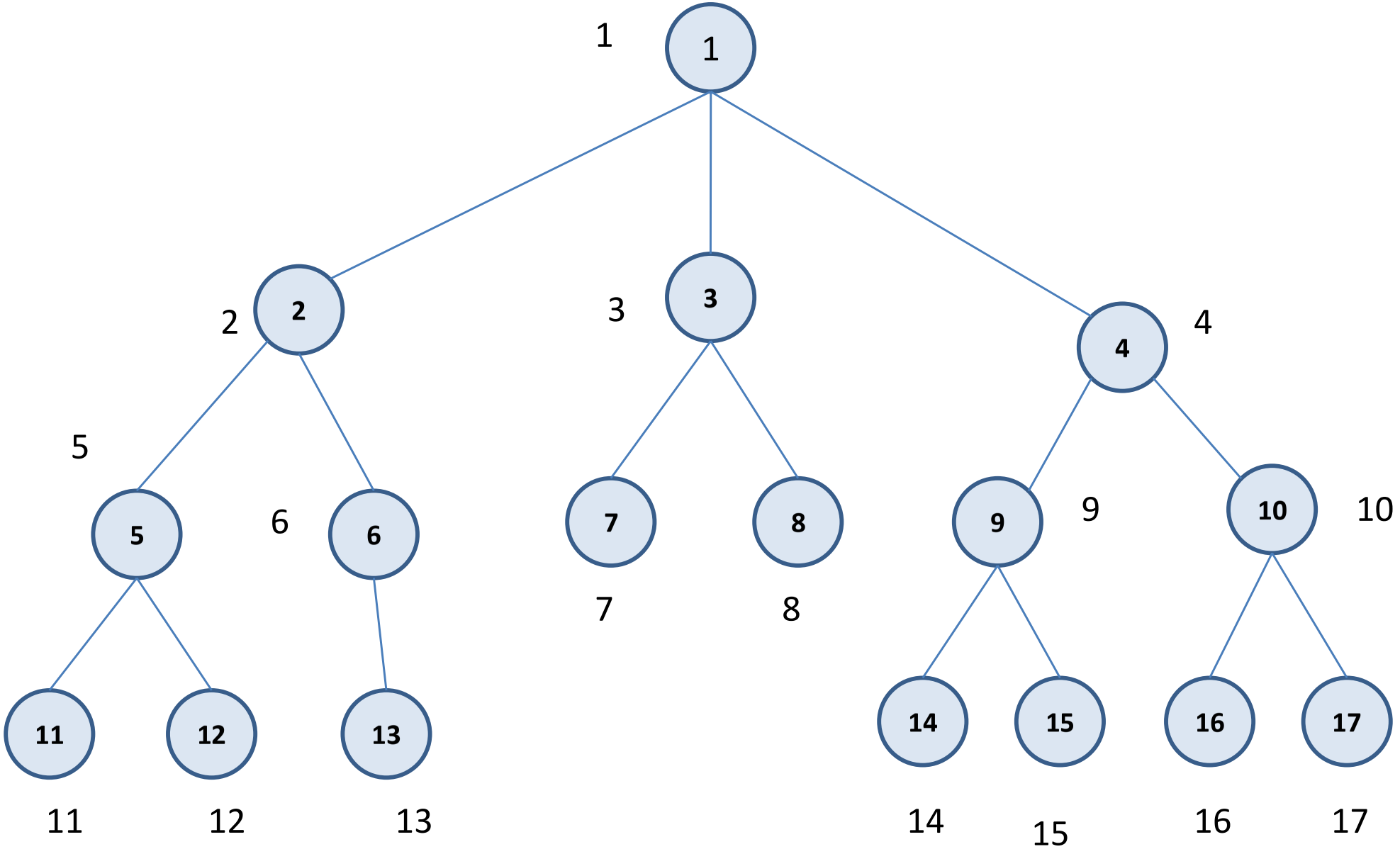
Order: 1, 2, 5, 6, 3, 7, 8, 4, 9, 10

Depth-First Iterative-Deepening Search



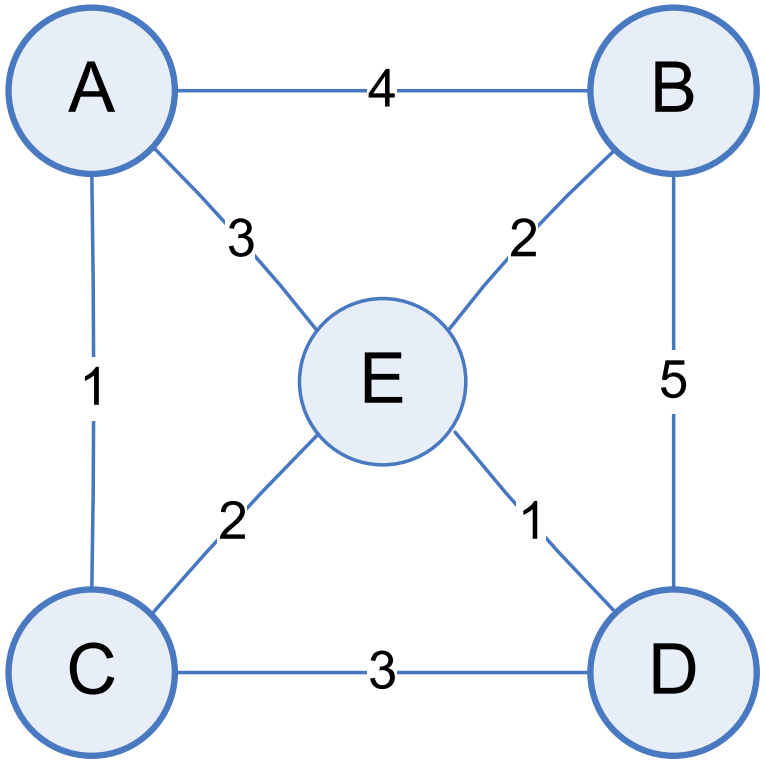
Order: 1, 2, 5, 11, 12, 6, 13, 3, 7, 8, 4, 9, 14, 15, 10, 16, 17

Breath-First Search



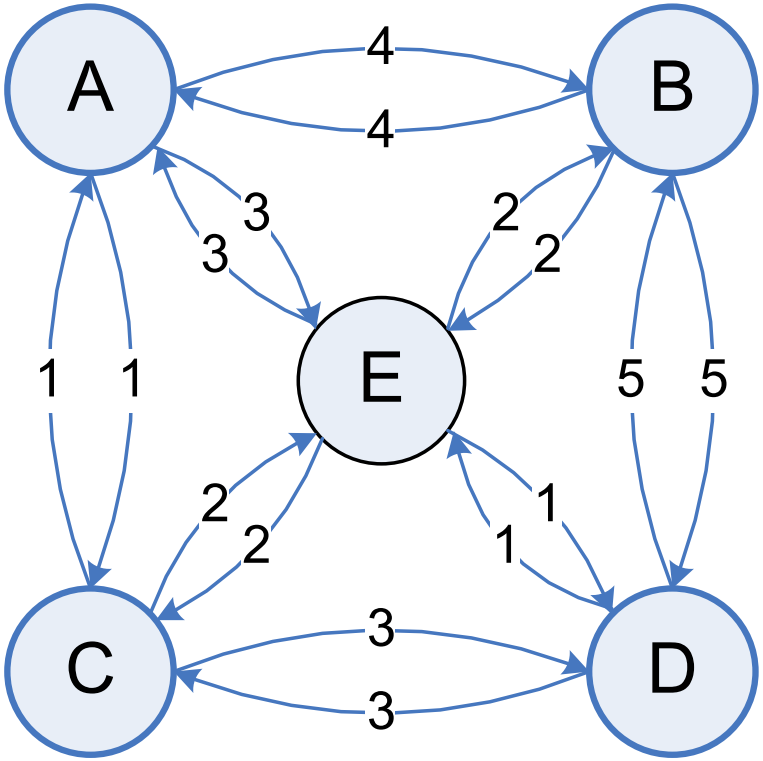
Order: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17

Traveling Salesman Problem



- Formulation:
  - States: cities
  - Initial state: A
  - Successor function: Travel from one city to another connected by a road
  - Goal test: the trip visits each city only once that starts and ends at A.
  - Path cost: traveling time

Traveling Salesman Problem

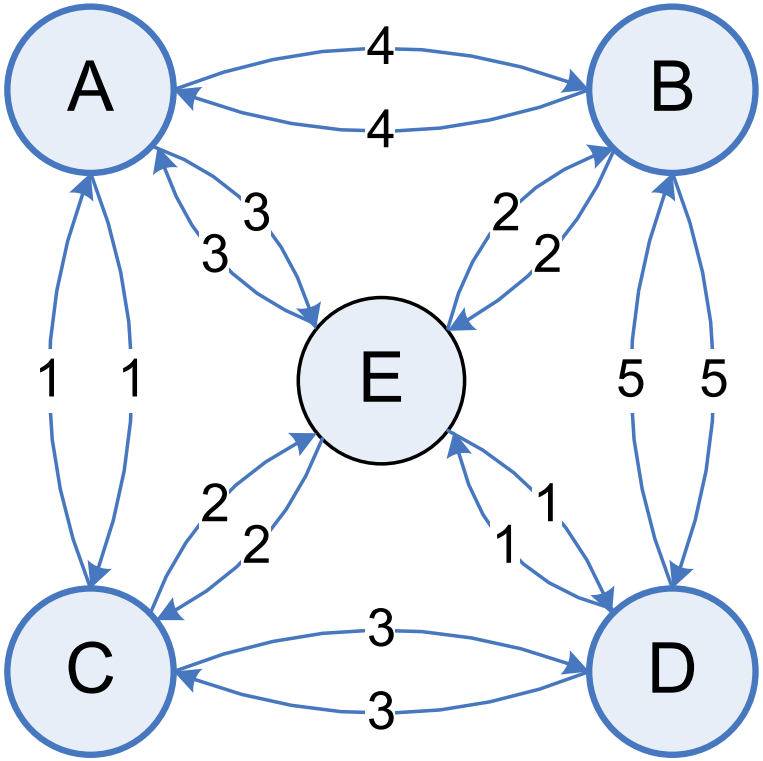


Can be represented as a graph

- Nodes – states
- Arcs – actions

States: A, B, C, D, E

Traveling Salesman Problem



Complete state space:

- Initial state (state A)
- All possible states and actions:

State A:
-go right to B, cost 4
-go down-right to E, cost 3
-go down to C, cost 1

State B:
-go left to A, cost 4
-go down to D, cost 5
-go down-left to E, cost 2

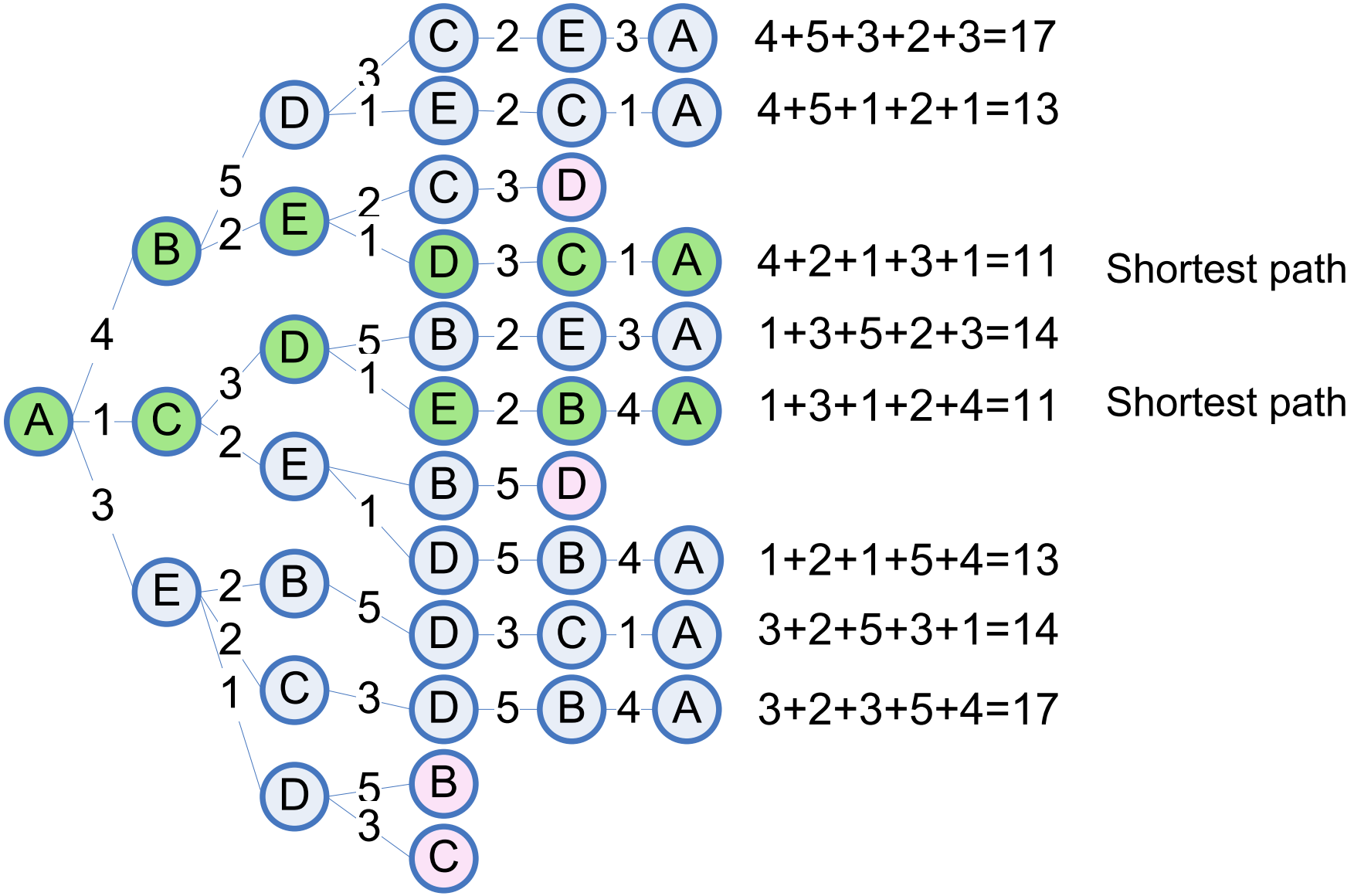
State C:
-go up to A, cost 1
-go right to D, cost 3
-go up-right to E, cost 2

State D:
-go up to B, cost 5
-go left to C, cost 3
-go up-left to E, cost 1

State E:
-go up-left to A, cost 3
-go up-right to B, cost 2
-go down left to C, cost 2
-go down-right to D, cost 1

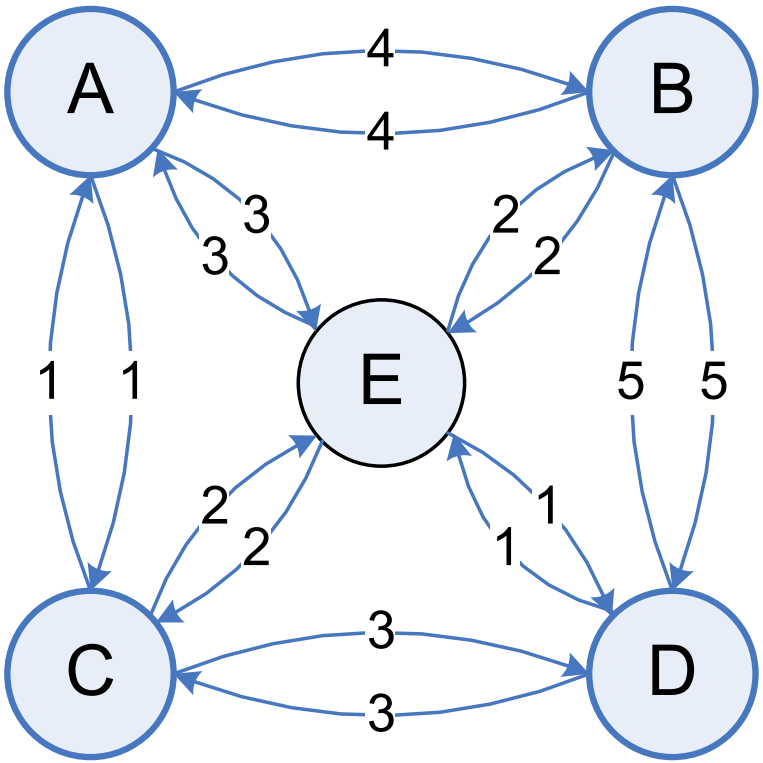
Traveling Salesman Problem

Breath-first search –the shortest trip from A, that visits all cities



Traveling Salesman Problem

Time and Space complexity: DFS vs. BFS

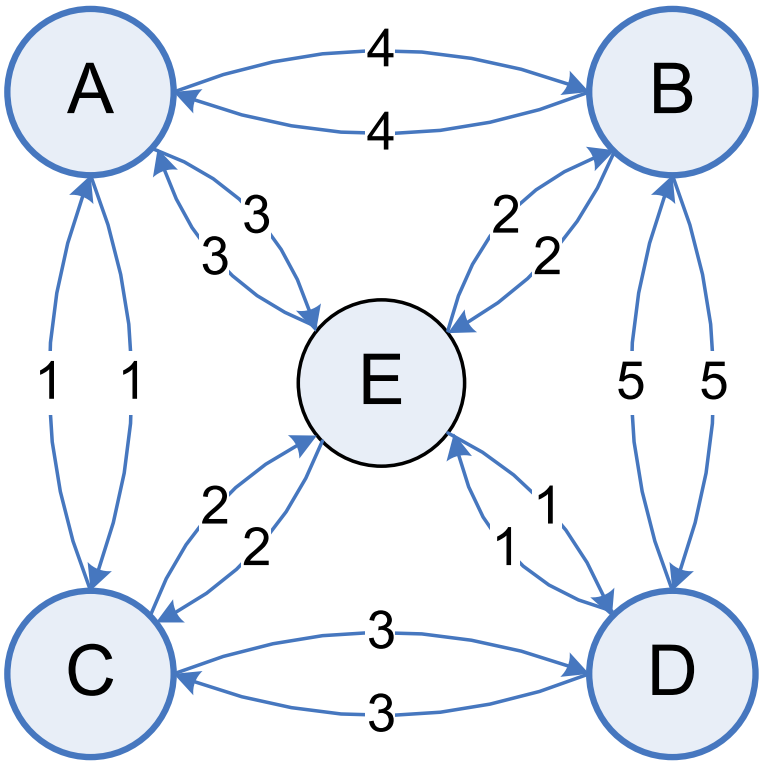


$depth_{goal} = depth_{max}$

	Time	Space
BFS:	$O(b^d)$	$O(b^d)$
DFS:	$O(b^d)$	$O(bd)$

Traveling Salesman Problem

Uniform-cost search?



Uniform-cost search algorithm is optimal with positive cost function. It will find the path with the lowest path cost.

Therefore, if cost = traveling time, uniform-cost search **will work well** with this problem