Artificial Intelligence

Lecture 4

Search Strategies

Reading: Russell's Chapter 3

Search strategies

- A search strategy is defined by picking the order of node expansion
- Strategies are evaluated along the following dimensions:
  - completeness: does it always find a solution if one exists?
  - Time complexity: number of nodes generated/expanded
  - space complexity: maximum number of nodes in memory
  - optimality: does it always find a least-cost/optimal solution?

- Time and space complexity are measured in terms of
  - *b:* maximum branching factor of the search tree
  - *d:* depth of the least-cost solution
  - *m*: maximum depth of the state space

Types of Search strategies

- Uninformed search strategies
- Informed search strategies

Uninformed search strategies

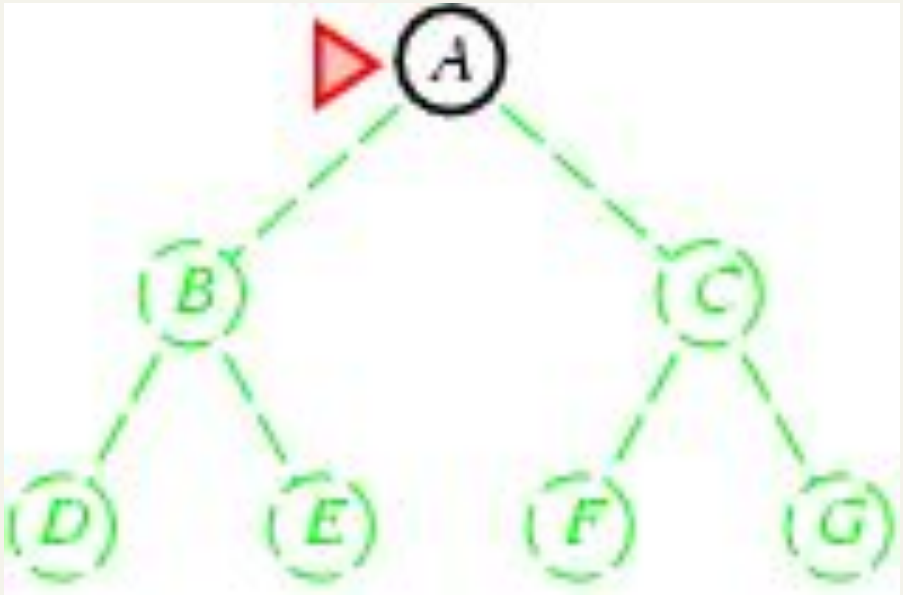
- Uninformed/blind search strategies use only the information available in the problem definition
- Generate successors and distinguish a goal state from a non goal state

Uninformed search strategies

- All search strategies are distinguished by the order in which nodes are expanded
  1. Breadth-first search
  2. Uniform-cost search
  3. Depth-first search
  4. Depth-limited search
     1. Iterative deepening search

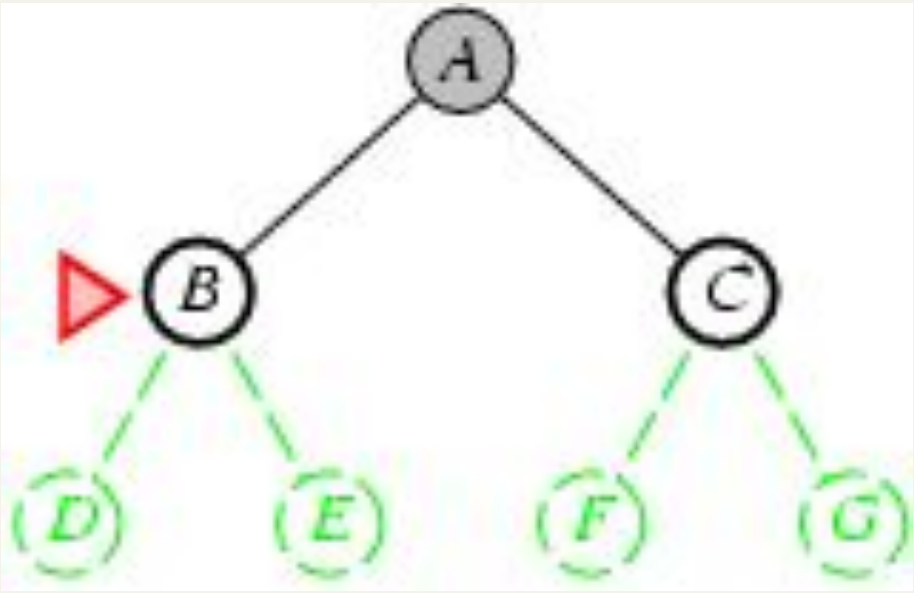
1. Breadth-first search

- Expand shallowest unexpanded node
- Implementation:
  - *fringe* is a FIFO queue, i.e., new successors go at end



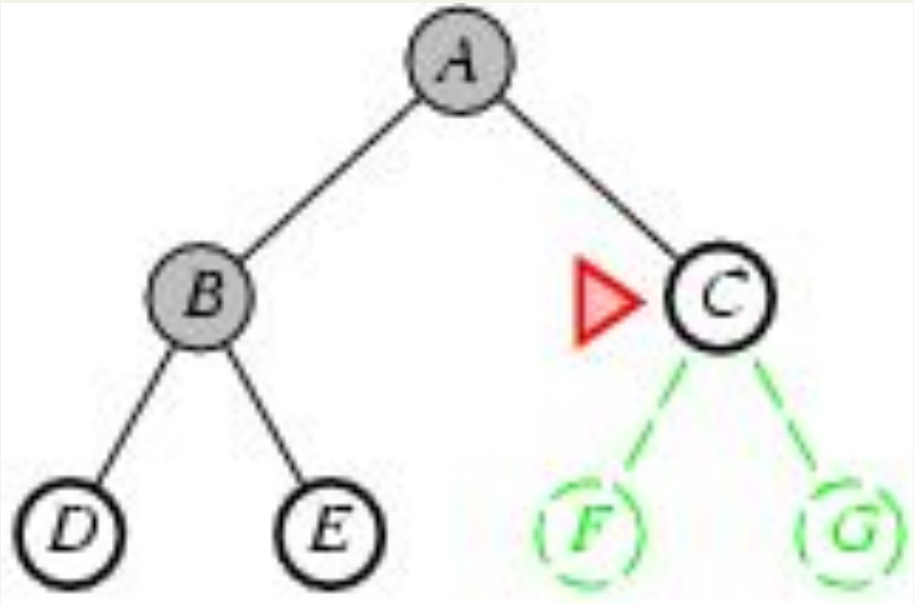
Breadth-first search

- Expand shallowest unexpanded node
- Implementation:
  - *fringe* is a FIFO queue, i.e., new successors go at end



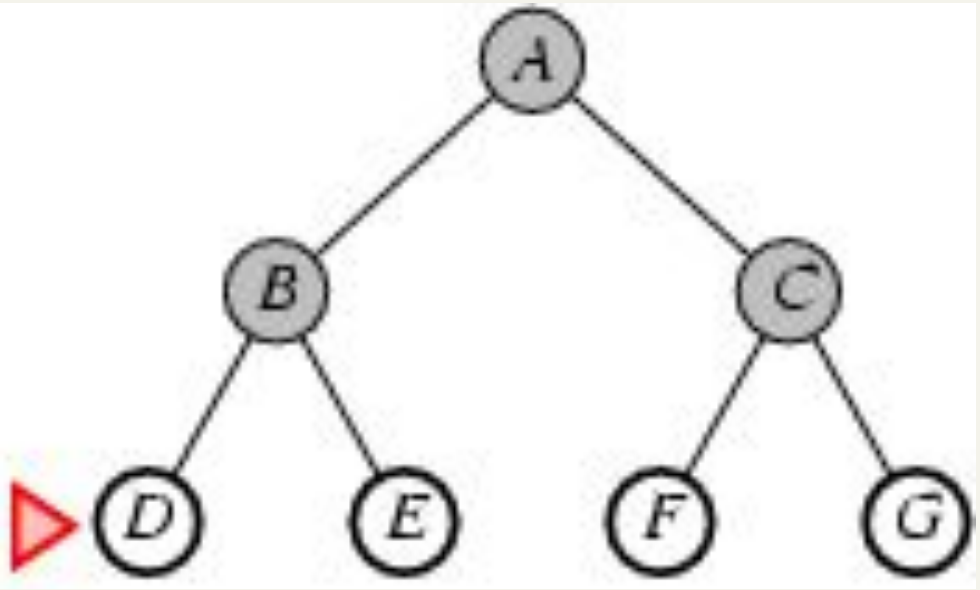
Breadth-first search

- Expand shallowest unexpanded node
- Implementation:
  - *fringe* is a FIFO queue, i.e., new successors go at end



Breadth-first search

- Expand shallowest unexpanded node
- Implementation:
  - *fringe* is a FIFO queue, i.e., new successors go at end



Properties of breadth-first search

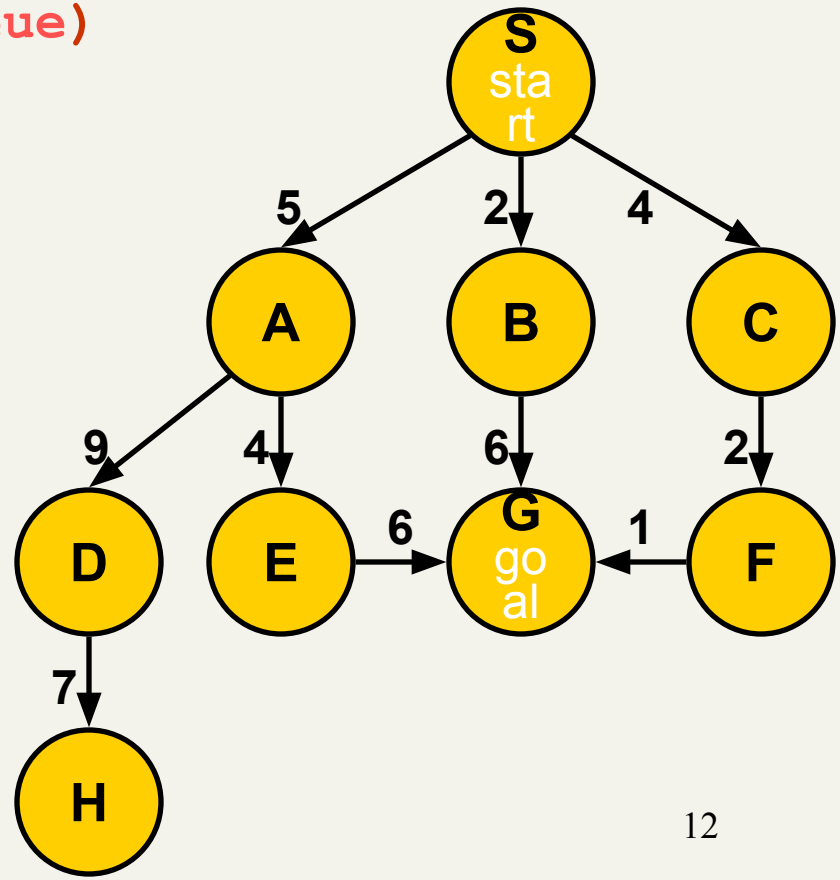
- Complete? Yes (if b is finite), BFS will eventually find it after expanding all shallower nodes
- Time? $1+b+b^2+b^3+\ldots+b^d+(b^{d+1}-b) = O(b^{d+1})$
- Space? $O(b^{d+1})$ (keeps every node in memory)
- Optimal? Yes (if cost = 1 per step)
- Space is the bigger problem

Breadth-First Search (BFS)

`generalSearch(problem, queue)`

# of nodes tested: 0, expanded: 0

expnd. node	Frontier list
	{S}

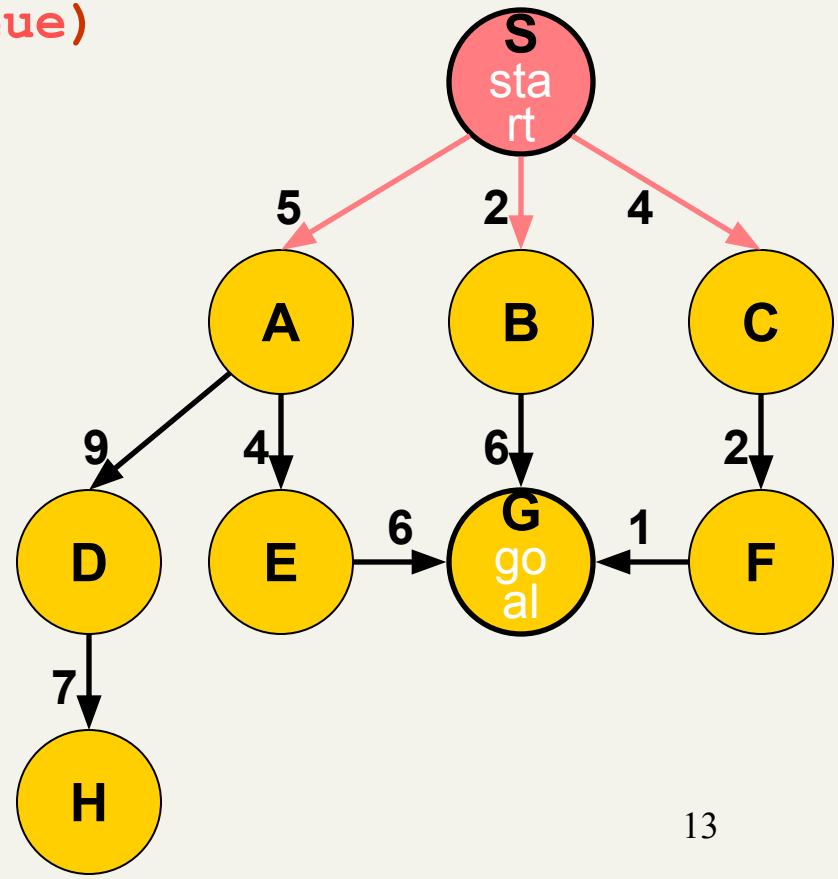


Breadth-First Search (BFS)

`generalSearch(problem, queue)`

# of nodes tested: 1, expanded: 1

expnd. node	Frontier list
	{S}
S not goal	{A,B,C}

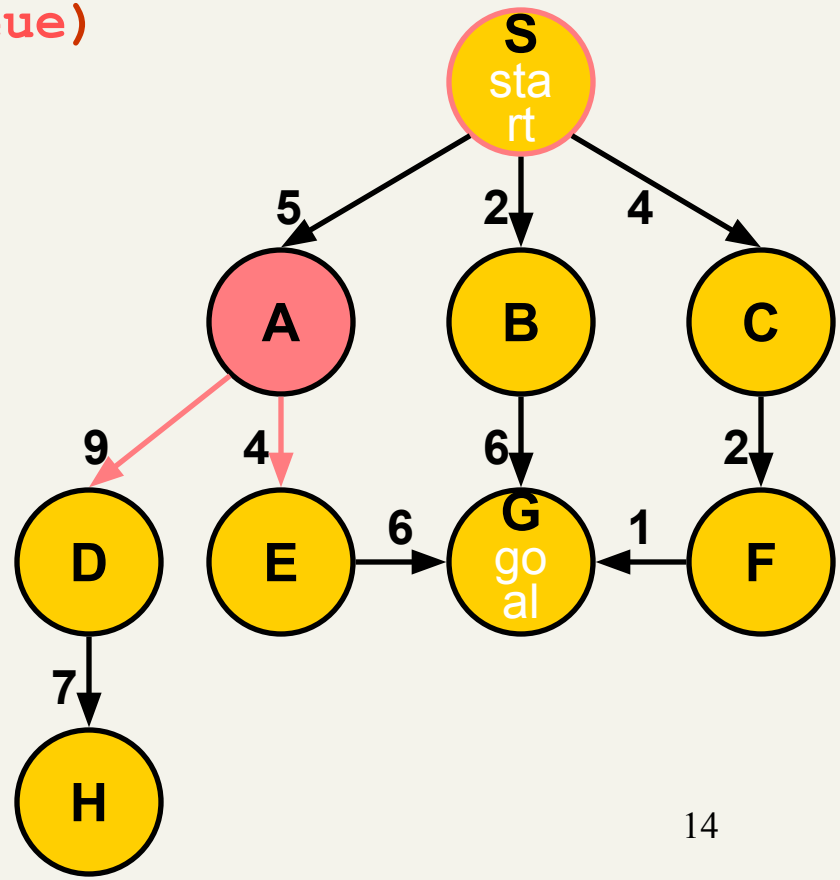


Breadth-First Search (BFS)

`generalSearch(problem, queue)`

# of nodes tested: 2, expanded: 2

expnd. node	Frontier list
	{S}
S	{A,B,C}
A not goal	{B,C,D,E}

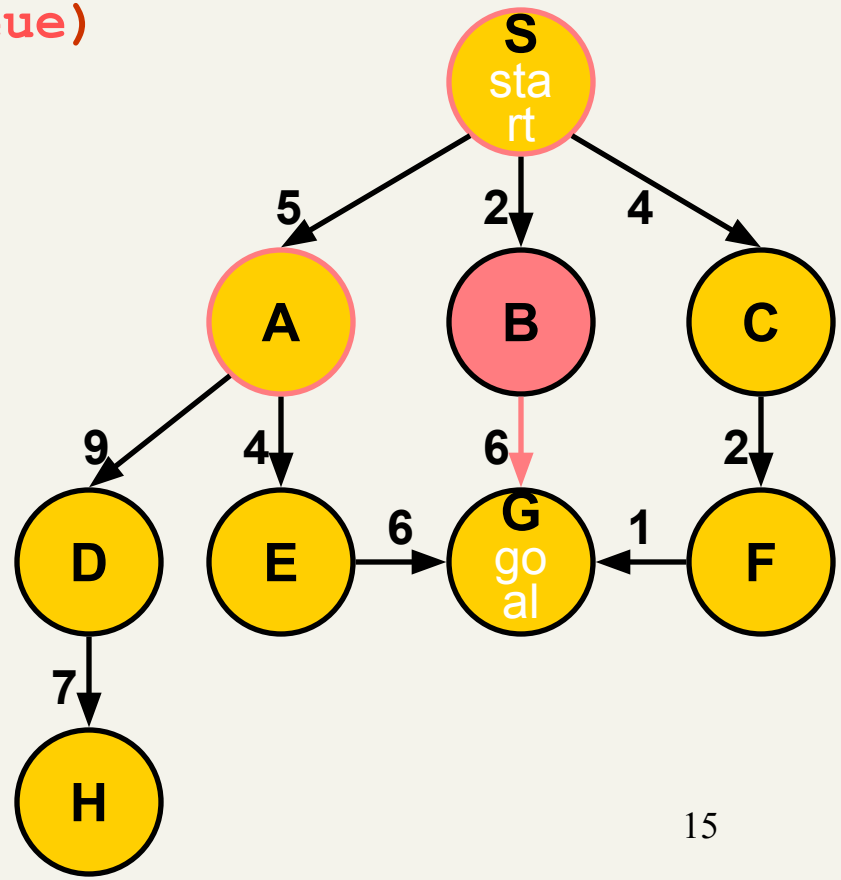


Breadth-First Search (BFS)

`generalSearch(problem, queue)`

# of nodes tested: 3, expanded: 3

expnd. node	Frontier list
	{S}
S	{A,B,C}
A	{B,C,D,E}
B not goal	{C,D,E,G}

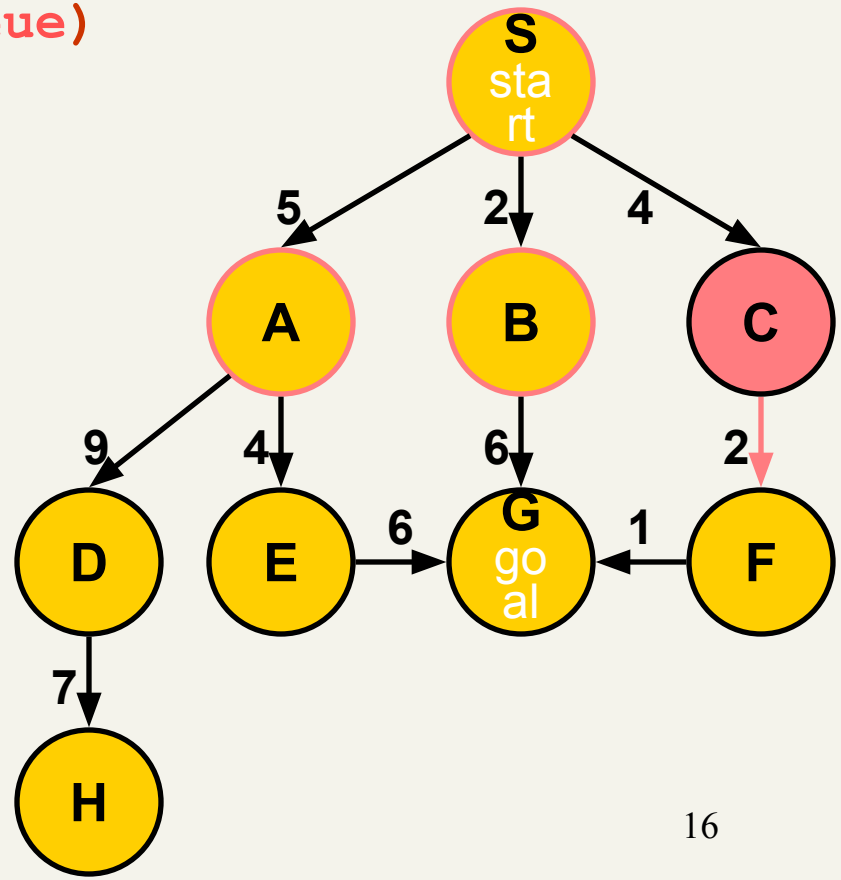


Breadth-First Search (BFS)

`generalSearch(problem, queue)`

# of nodes tested: 4, expanded: 4

expnd. node	Frontier list
	{S}
S	{A,B,C}
A	{B,C,D,E}
B	{C,D,E,G}
C not goal	{D,E,G,F}

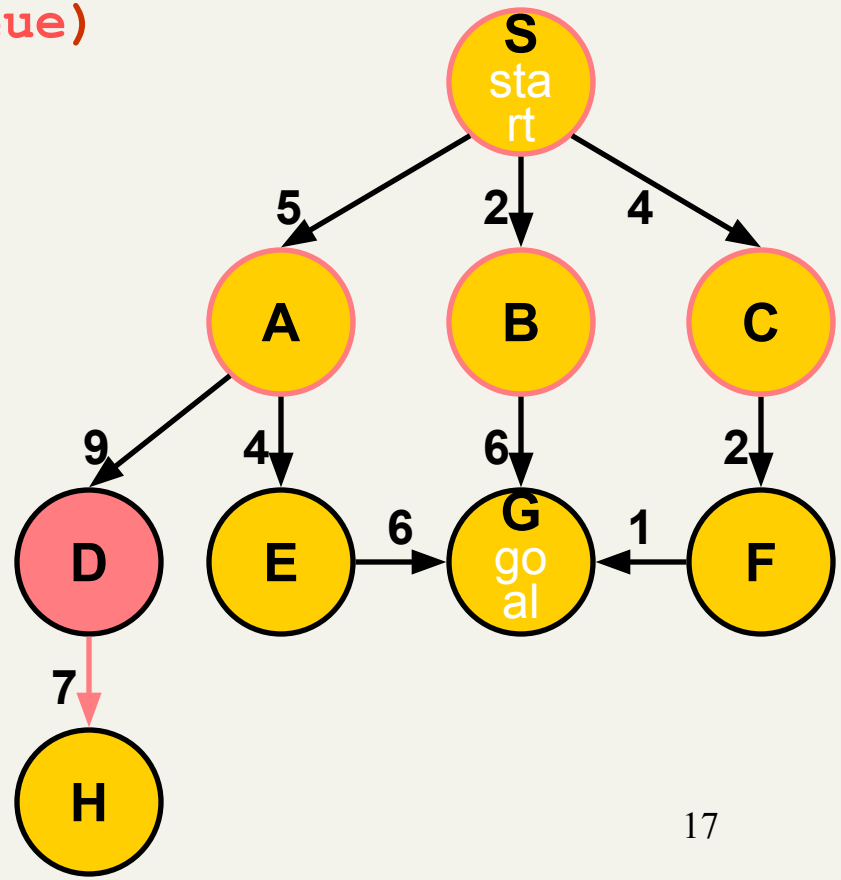


Breadth-First Search (BFS)

`generalSearch(problem, queue)`

# of nodes tested: 5, expanded: 5

expnd. node	Frontier list
	{S}
S	{A,B,C}
A	{B,C,D,E}
B	{C,D,E,G}
C	{D,E,G,F}
D not goal	{E,G,F,H}

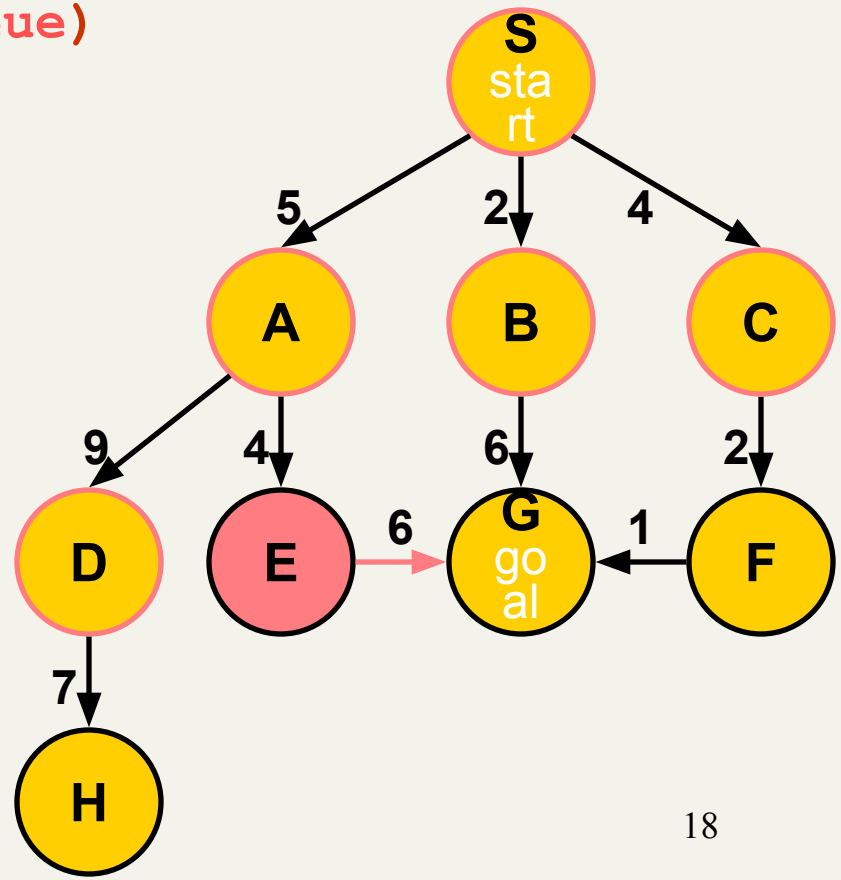


Breadth-First Search (BFS)

`generalSearch(problem, queue)`

# of nodes tested: 6, expanded: 6

expnd. node	Frontier list
	{S}
S	{A,B,C}
A	{B,C,D,E}
B	{C,D,E,G}
C	{D,E,G,F}
D	{E,G,F,H}
E not goal	{G,F,H,G}

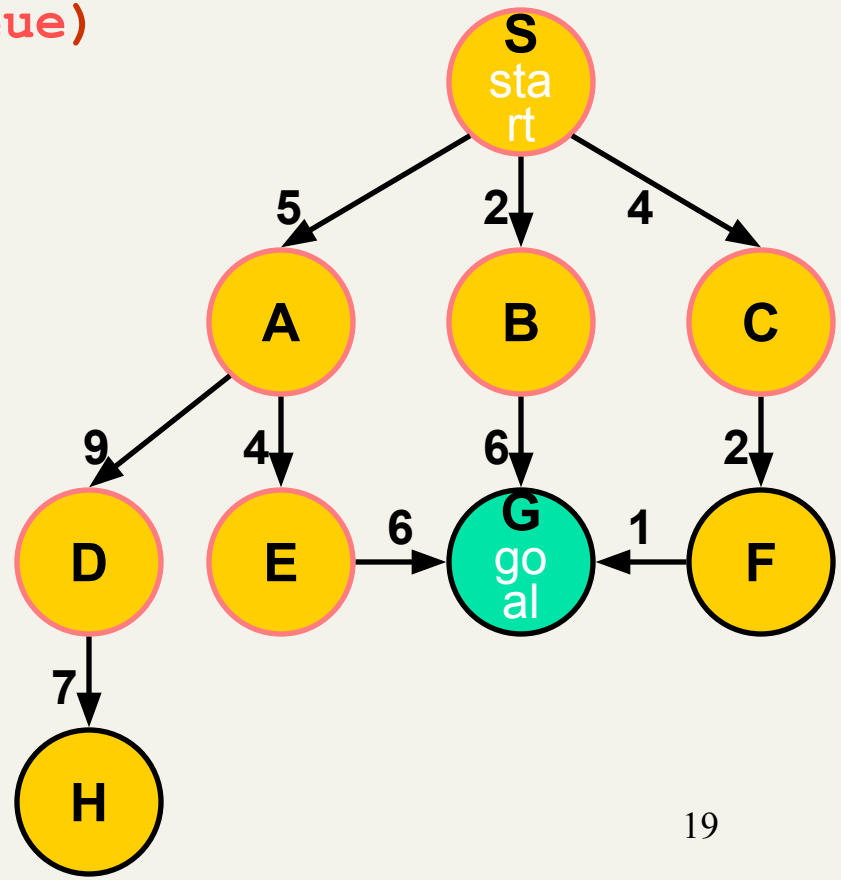


Breadth-First Search (BFS)

`generalSearch(problem, queue)`

# of nodes tested: 7, expanded: 6

expnd. node	Frontier list
	{S}
S	{A,B,C}
A	{B,C,D,E}
B	{C,D,E,G}
C	{D,E,G,F}
D	{E,G,F,H}
E	{G,F,H,G}
G goal	{F,H,G} no expand



Breadth-First Search (BFS)

`generalSearch(problem, queue)`

# of nodes tested: 7, expanded: 6

expnd. node	Frontier list
	{S}
S	{A,B,C}
A	{B,C,D,E}
B	{C,D,E,G}
C	{D,E,G,F}
D	{E,G,F,H}
E	{G,F,H,G}
G	{F,H,G}