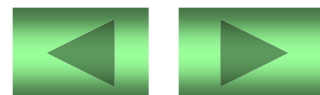


# Uninformed Search Algorithms

- Two categories of search algorithms are available:
  - Uninformed search algorithms
  - Informed search algorithms
- Uninformed Search Algorithms
  - Uninformed (blind) strategies use only the information available in the problem definition



# Uninformed Search Algorithms

- There are many uniformed search algorithms:
  - Breadth-first search
  - Uniform-cost search
  - Depth-first search
  - Depth-limited search
  - Iterative deepening

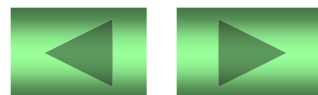
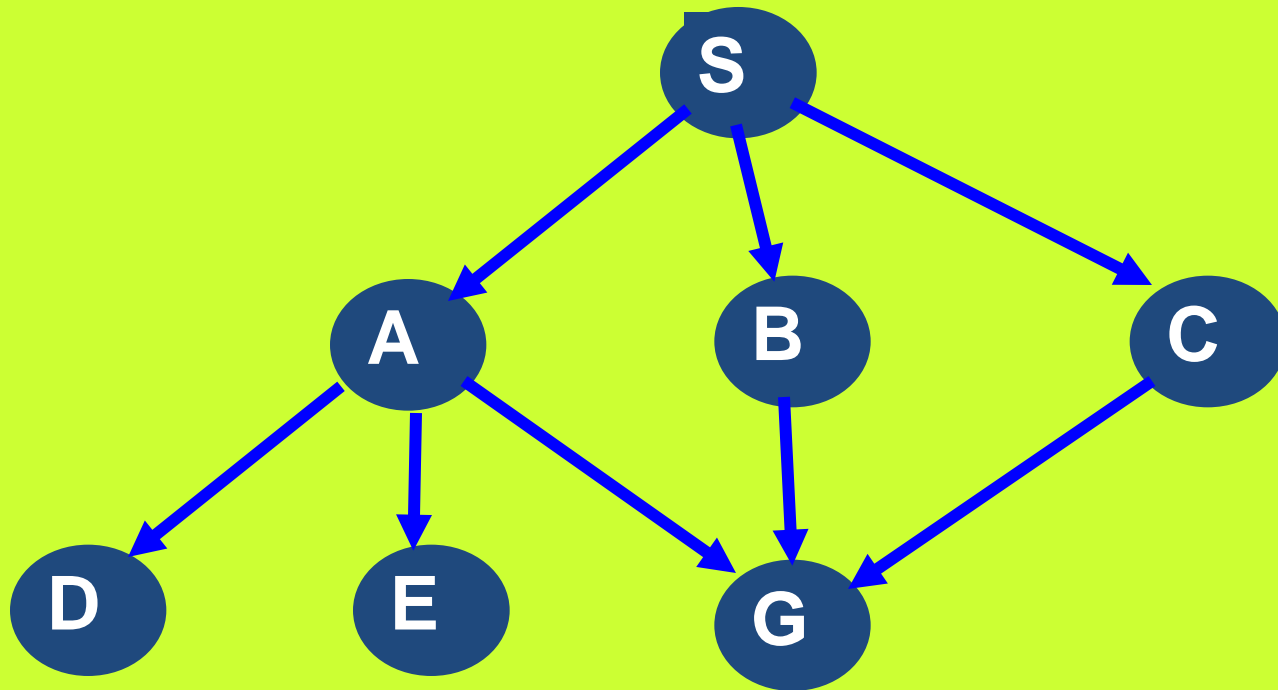


# Breadth-First Search (BFS)

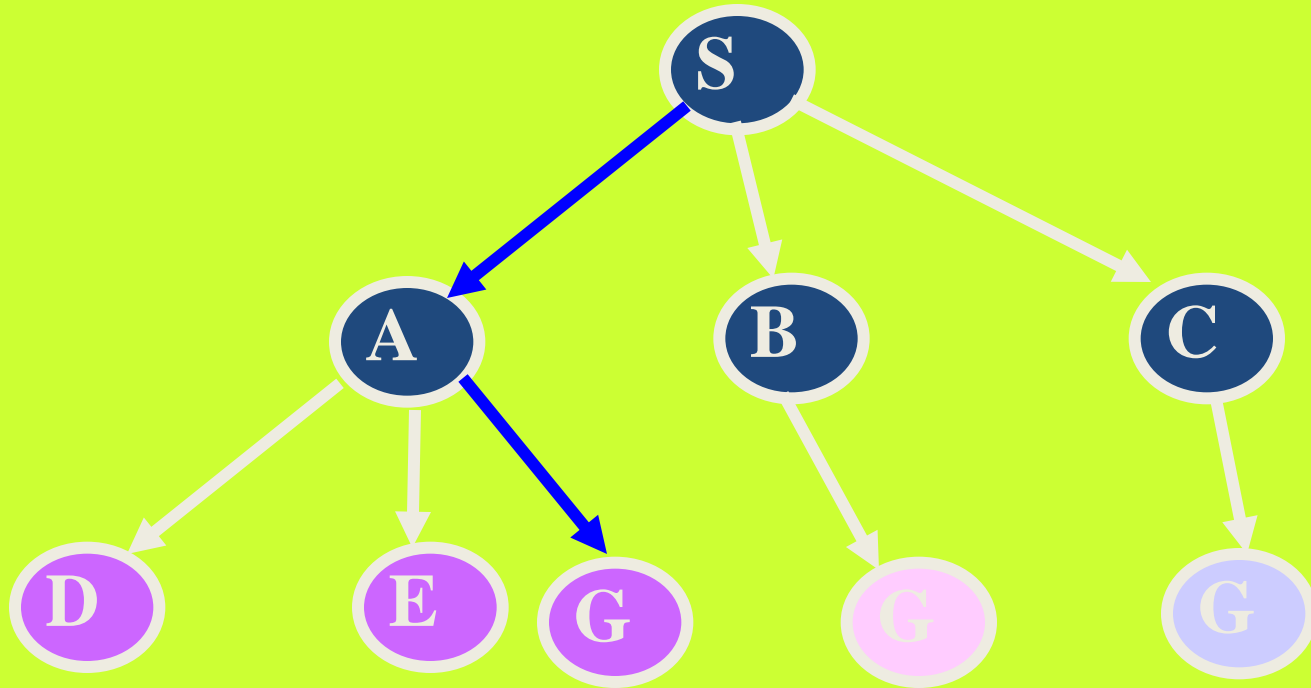
- Strategy: expands the **shallowest** node
- Implementation: put successors nodes at the end of the queue (**FIFO queue**)



# Breadth-First Search

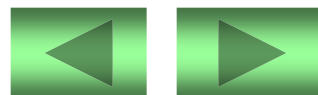


# Breadth-First Search



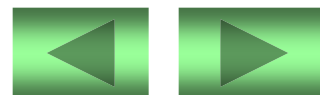
**Nodes are visited as follows:  
S A B C D E G**

**Path:  
S A G**



# Breadth-First Search: Properties

- Complete: Yes if  $b$  (maximum number of children of a node) is finite
- Time:  $1 + b + b^2 + \dots + b^d = (b^{d+1} - 1)/(b-1)$  nodes =  $O(b^d)$  (exponential in  $d$  (depth) of least-cost solution)
- Space:  $O(b^d)$ , keeps every node in memory (serious problem: use of lots of space)
- Optimal: Not optimal in general

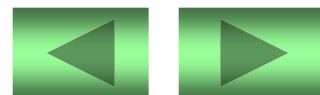
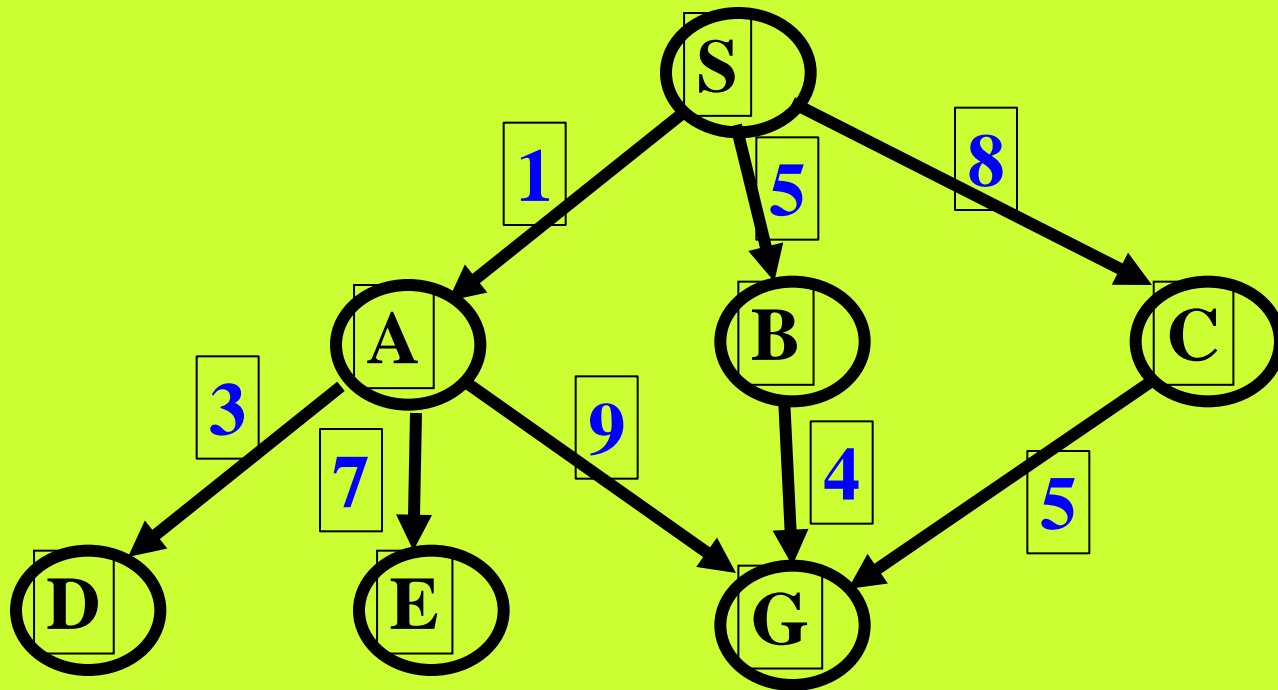


# Uniform-Cost Search

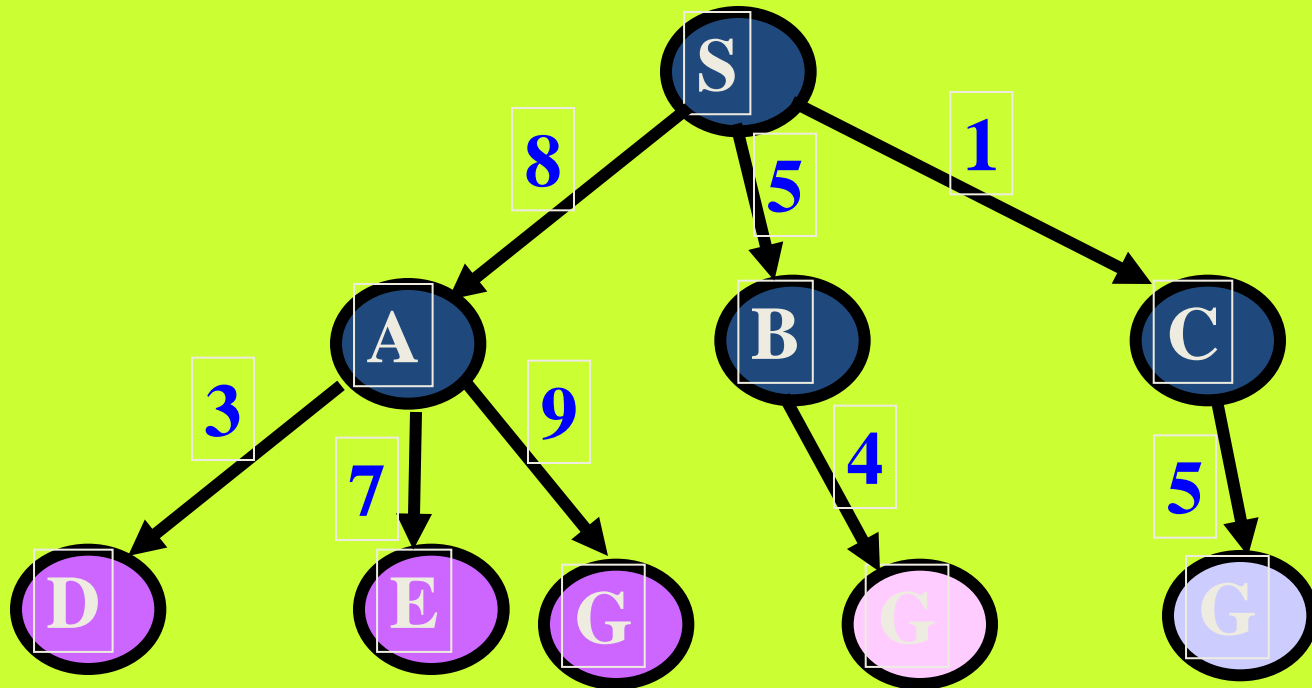
- Strategy: expands the **least-cost** unexpanded node
- Implementation: Queue ordered by path cost
- **Note:** Also Called “*Dijkstra's Algorithm*” in the algorithms literature and similar to “*Branch and Bound Algorithm*” in operations research literature



# Uniform-Cost Search

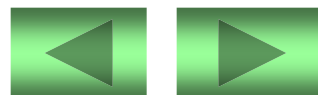


# Uniform-Cost Search



**Nodes are visited as follows:**  
**S C B A G D E**

**Path:**  
**S C G**



# Uniform-Cost Search: Properties

- Complete: Yes
- Time:  $O(b^d)$
- Space:  $O(b^d)$
- Optimal: Yes

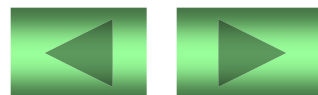
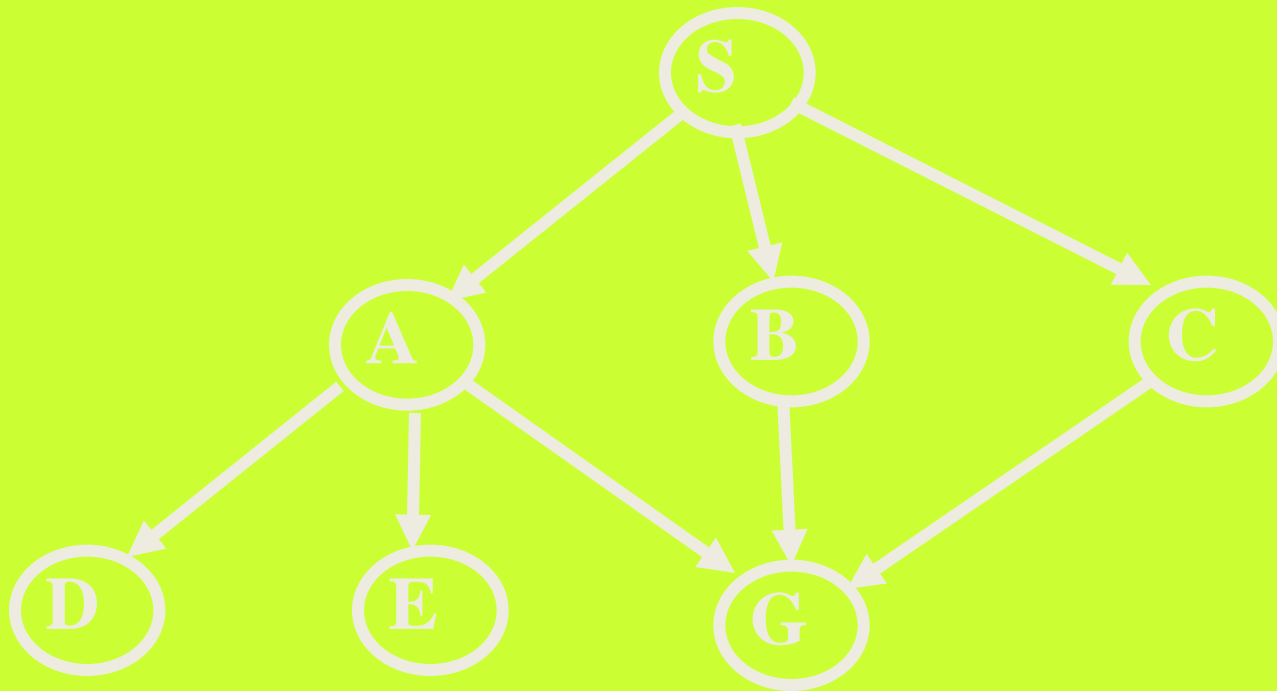


# Depth-First Search (DFS)

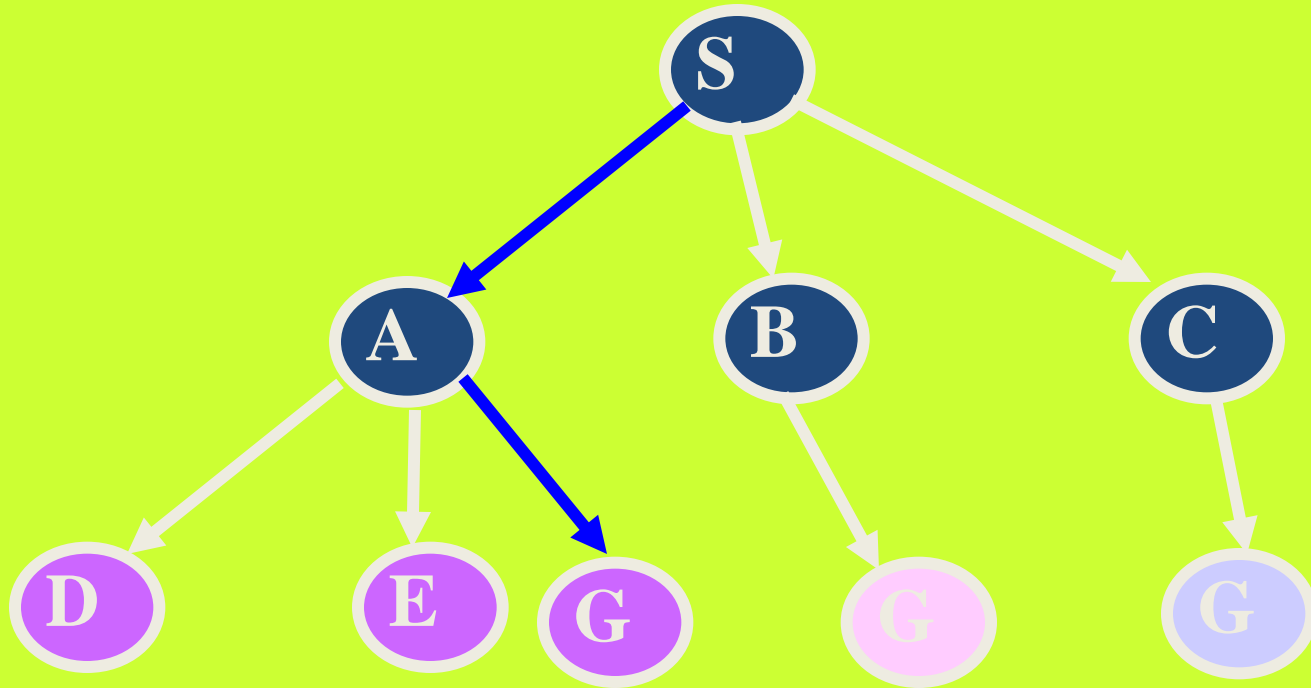
- Strategy: expands the **deepest** node
- Implementation: put successors nodes at the front of the queue (**LIFO: stack**)



# Depth-First Search



# Depth-First Search

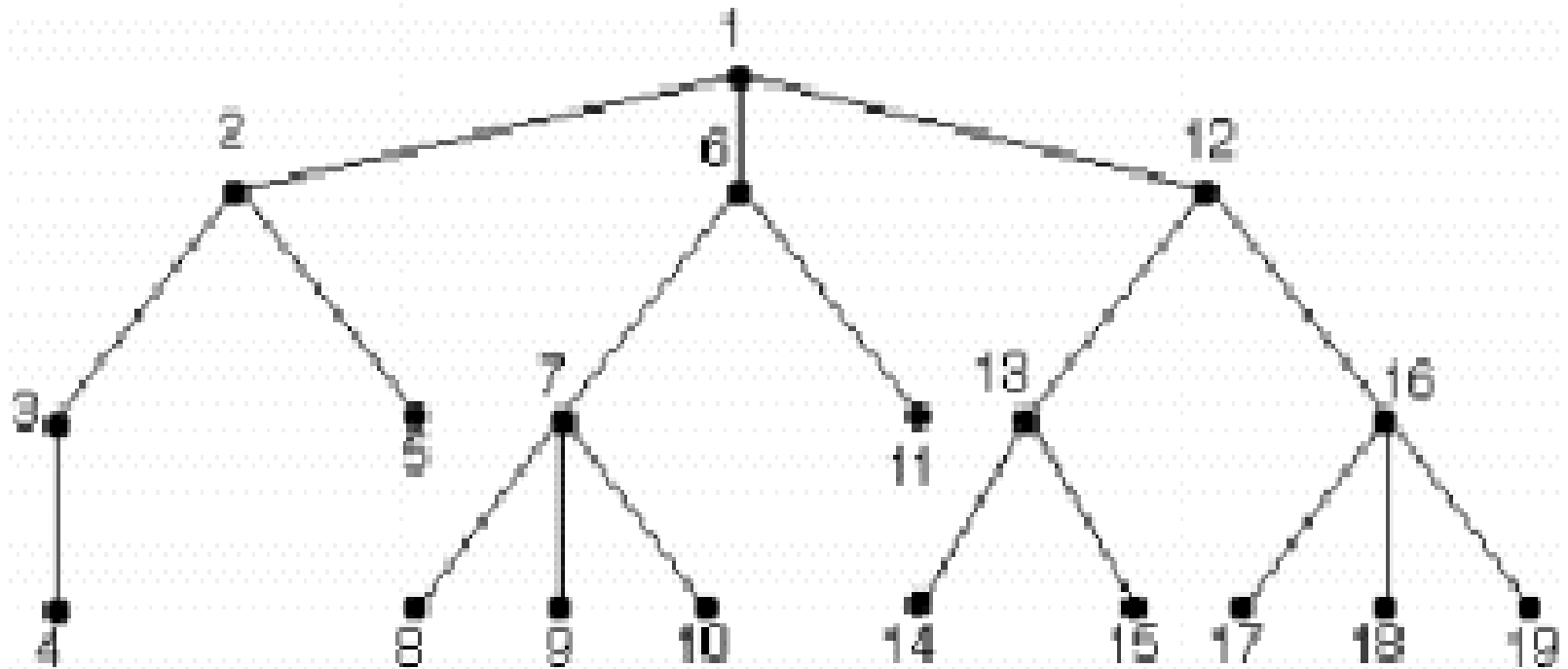


**Nodes are visited as follows:**  
**S A D E G B C**

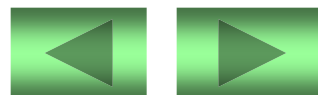
**Path:**  
**S A G**



# Depth-First Search

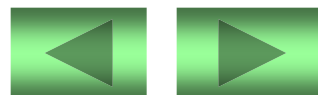


- Nodes are numbered in the order of their exploration



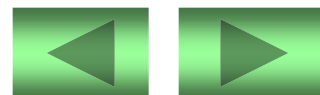
## Depth-First Search: Properties

- Complete:
  - No: fails in infinite-depth spaces, spaces with loops
  - Modify to avoid repeated states along path by marking visited nodes => complete in finite spaces
- Time:  $O(b^m)$ : exponential in  $m$  (maximum depth) of state space (horrible if  $m \gg d$  (depth) of least-cost solution)
- Space:  $O(b * m)$ , i.e.; linear space
- Optimal: No

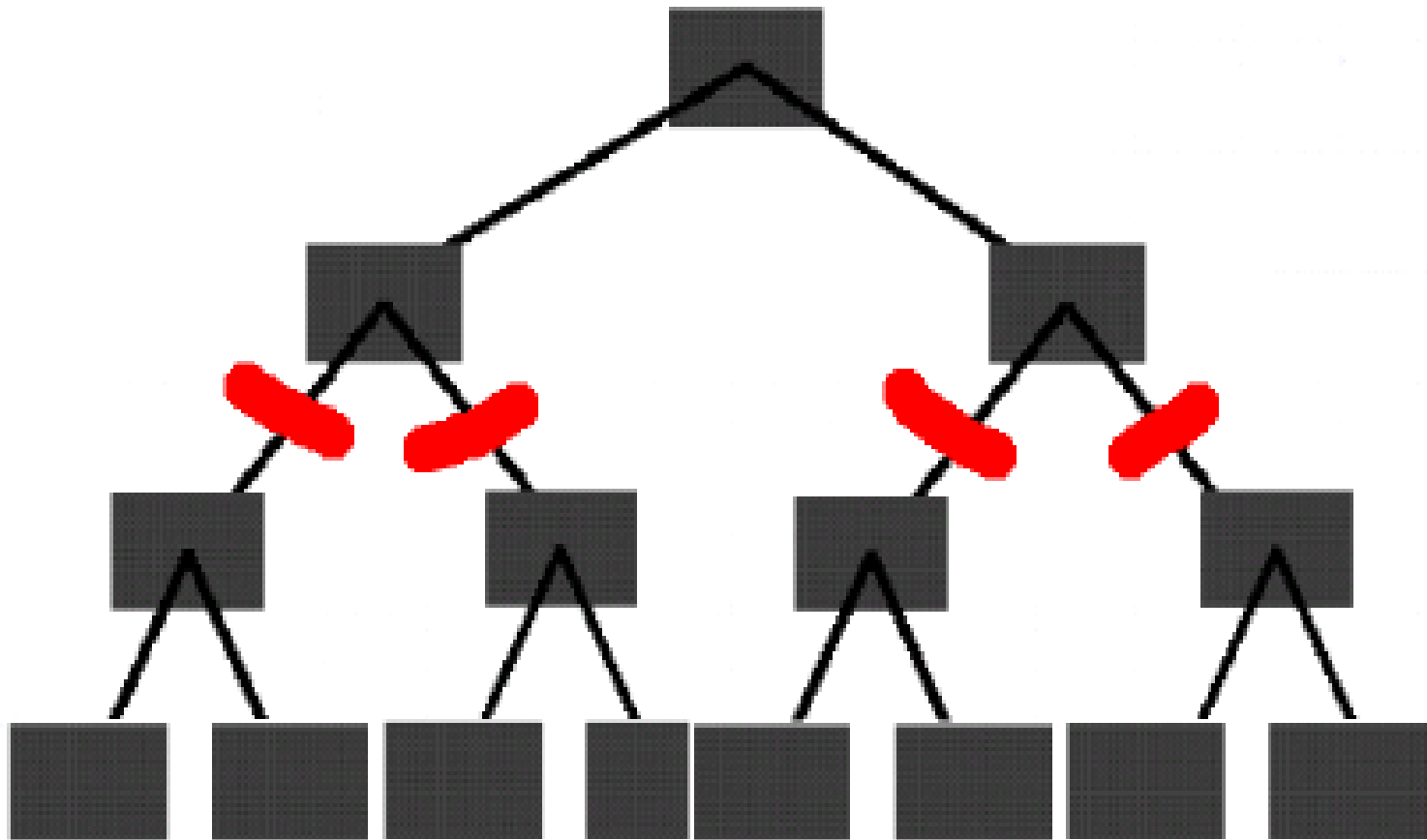


# Depth-Limited Search

- Strategy:
  - Depth-first search with depth limit  $L$
- Implementation:
  - Nodes at given depth  $L$  considered as having no successors



# Depth-Limited Search



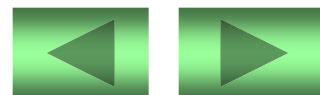
# Depth-Limited Search: Properties

- Completeness:
  - Yes, if  $L \geq d$
- Time complexity:
  - $O(b^L)$
- Space complexity:
  - $O(b * L)$
- Optimality:
  - No



# Iterative Deepening Search

- The problem with search in limited depth is to fix the good value of  $L$ , because the goal may be at deeper level
- Iterative deepening has to try all values possible of  $L$  starting at  $L = 0$



# Iterative Deepening Search

- Advantages of BFS and DFS are combined to get:
  - optimal and complete (as BFS)
  - saves memory space (as DFS)
- Iterative deepening search is recommended when the space of search is large and the solution depth is unknown



# Iterative Deepening Search

- Iterative deepening search can work as follows:
  - First apply DFS to depth 0 (i.e., start node has no successors), then, if no solution reached, apply DFS to depth 1, etc.
  - Repeat this process until solution is found



# Iterative Deepening Search: The Algorithm

function IDS (problem)

  for depth  $\leftarrow 0$  to  $\infty$  do

    result  $\leftarrow$  Depth-Limited-Search      (problem,  
    depth)

    if result  $\neq$  cutoff then

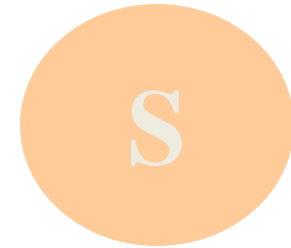
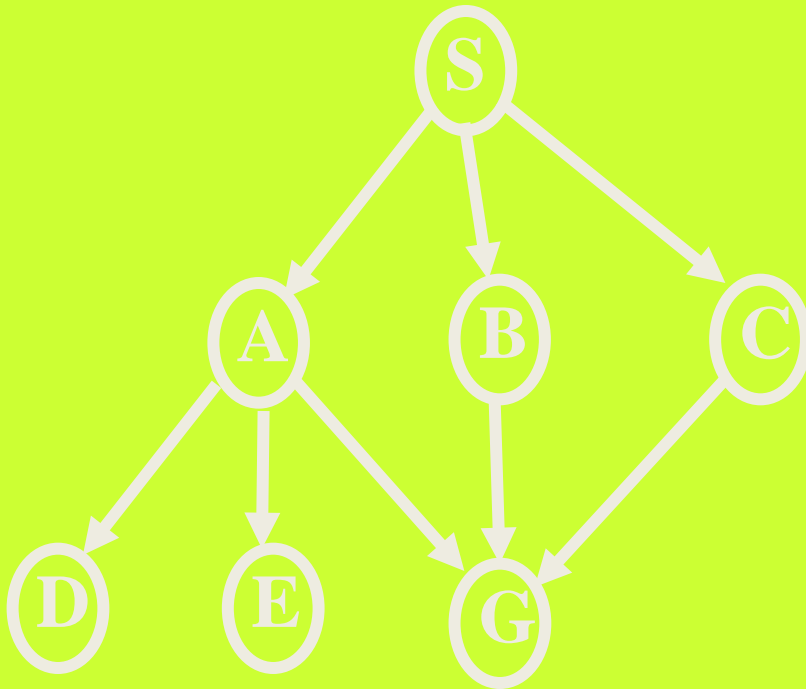
      return result

end



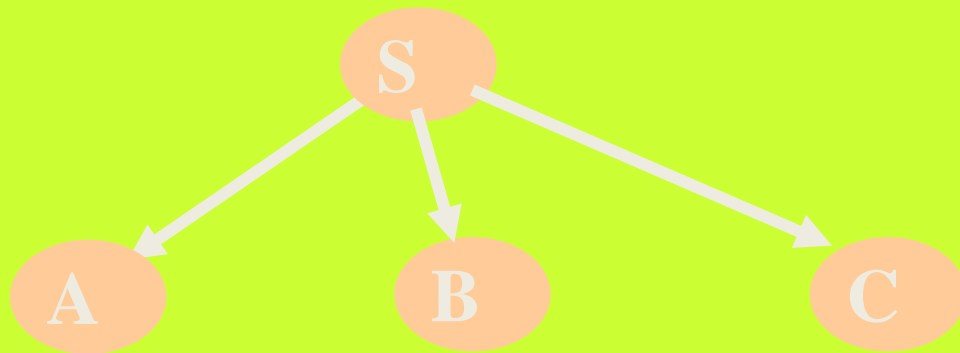
# Iterative Deepening Search

- $L = 0$



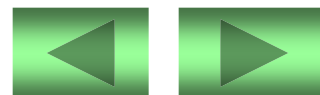
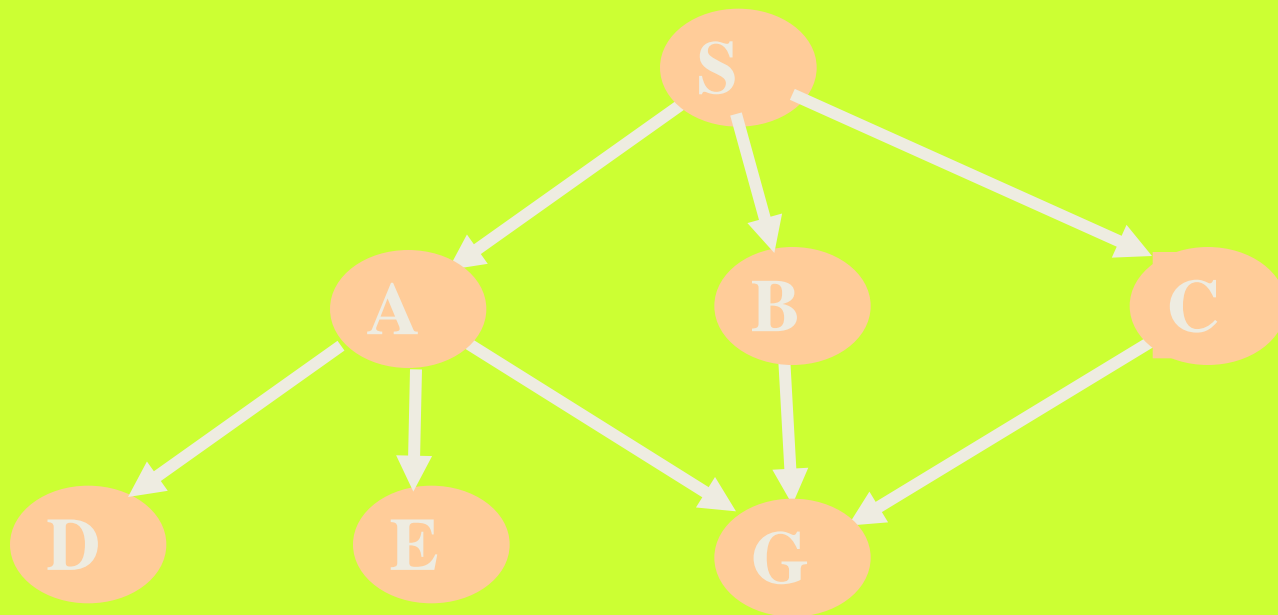
# Iterative Deepening Search

- $L = 1$



# Iterative Deepening Search

- $L = 2$



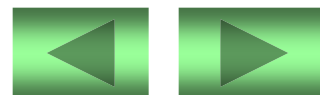
# Iterative Deepening Search: Properties

- Complete:
  - Yes
- Time:
  - $(d+1)b^0 + db^1 + (d-1)b^2 + \dots + b^d = O(b^d)$
- Space:
  - $O(b * d)$
- Optimal:
  - yes, if step cost = 1



# Comparing Uninformed Strategies

Criterion	BFS	Uniform -Cost	DFS	Depth- Limited	Iterative Deepening
Complete	Yes	Yes	No	No	Yes
Time	$O(b^d)$	$b^d$	$b^m$	$b^l$	$b^d$
Space	$O(b^d)$	$b^d$	$b * m$	$b * l$	$b * d$
Optimal	Yes	Yes	No	Yes, if $l \geq d$	Yes



# Comparing Uninformed strategies

## What strategy to use and When

- Breadth-First Search:
  - Solutions are expected to be shallow
- Uniform-Cost Search:
  - Tree search is costly
  - It is used to get least cost solution



# Comparing Uninformed strategies

## What strategy to use and When

- Depth-First Search:
  - Solutions are expected to be in depth
- Iterative-Deepening Search:
  - Space is limited and the shortest solution path is needed



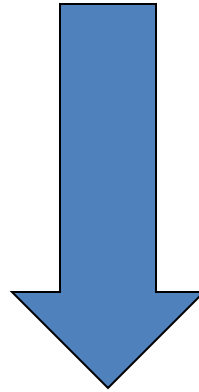
# Limitation of Uninformed Strategies

- Uninformed search strategies can find solutions to problems by systematically expanding the nodes till finding the goal
- The number of nodes to be explored is so high that the problem of complexity becomes critical. This leads to what is called **combinatorial explosion**



# Limitation of Uninformed Strategies

- Thus such strategies are inefficient in most cases



- Informed search strategies can find solution more efficiently

