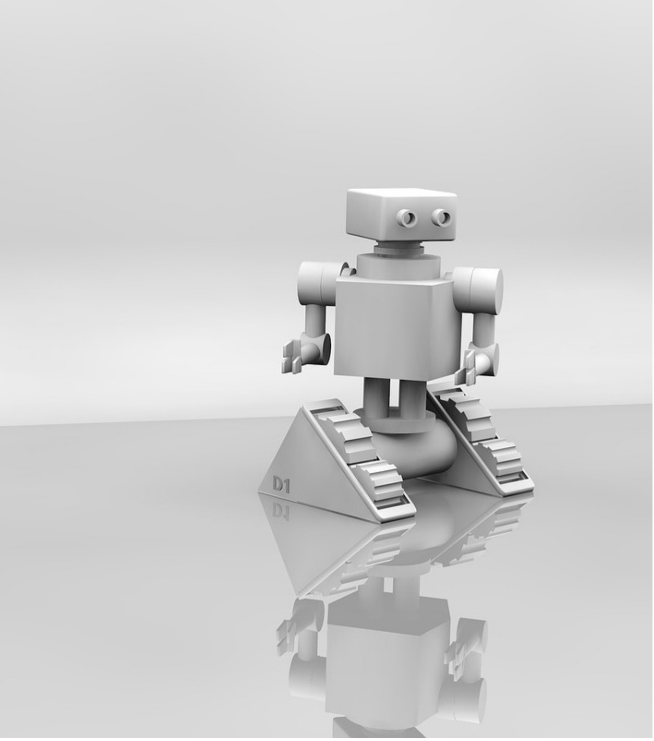


CPSC 481
Artificial Intelligence

Dr. Mira Kim
Mira.kim@fullerton.edu

What we will cover today

- Uninformed search
  - Depth first search
  - Breadth first search

Textbook: Chapter 3.4

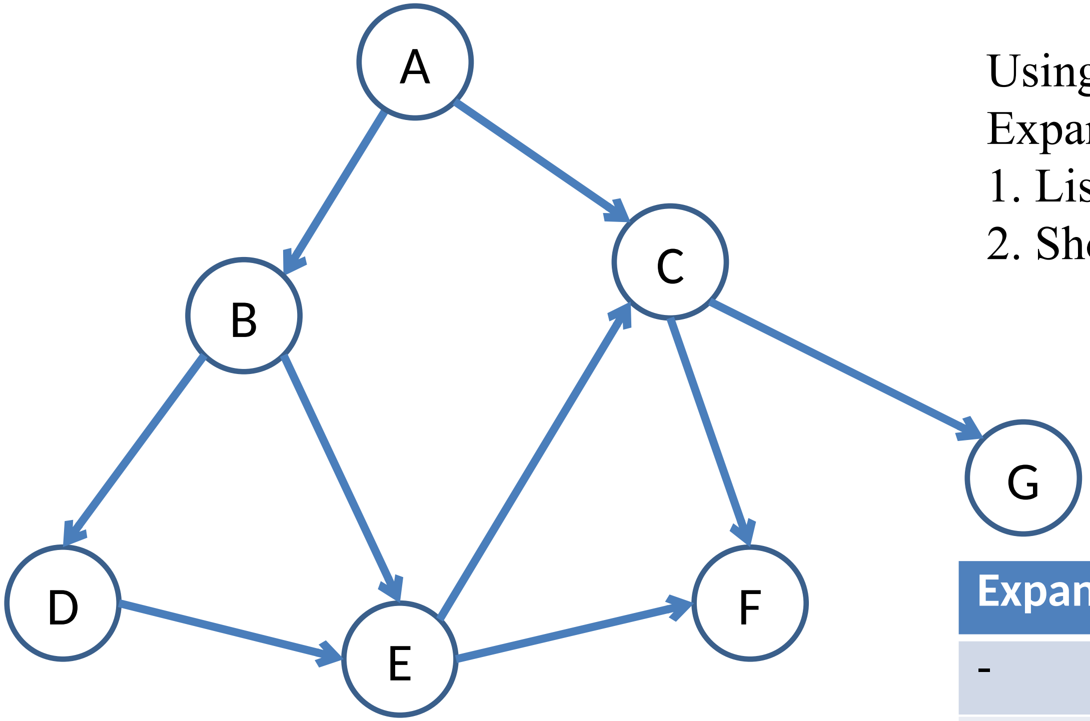
Uninformed search

- Does not have any knowledge about how close a state is to the goal
- Useful when the problem domain lacks additional information or heuristics to guide the search process
- Can be inefficient in complex search spaces since they do not exploit any domain-specific knowledge to guide the search direction

Depth-first TREE search

- Input: A problem
- Data structure: frontier
  - Also called “open”
  - Stack, LIFO queue

```
Initialize frontier with initial state
Loop do
    IF the frontier is empty RETURN FAILURE
    Choose top node and remove it from frontier
    IF top node is goal RETURN SUCCESS
    expand top node: pushing child nodes to the frontier
```



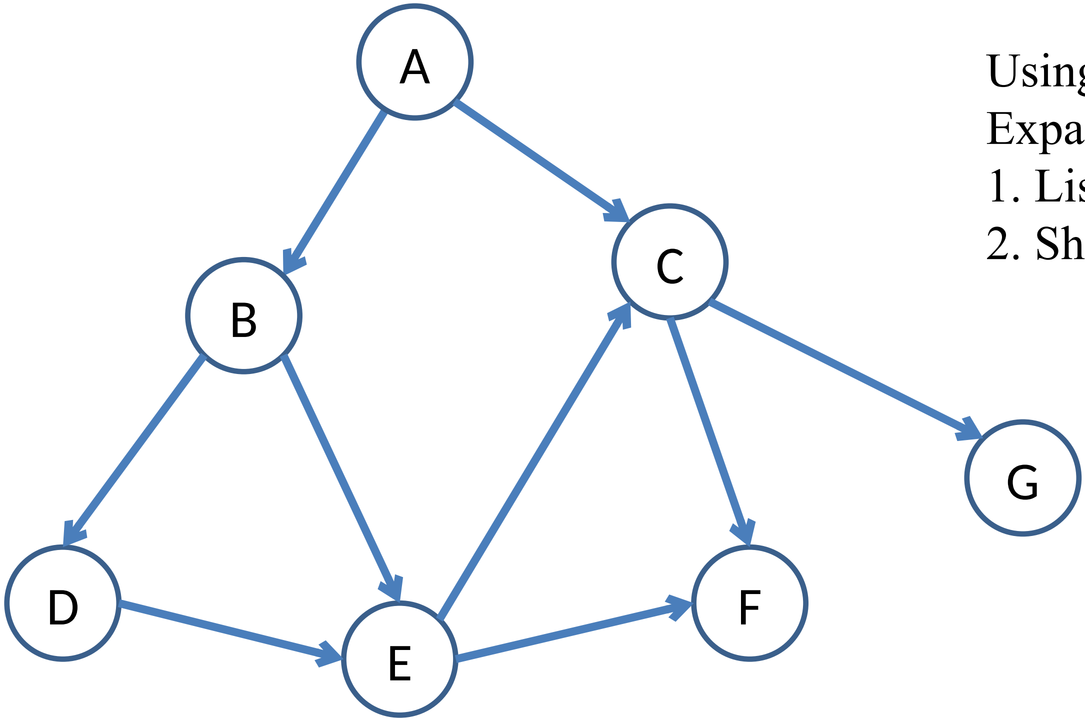
Initial state: A
Goal state: D

Using DFS (Tree)
Expand child nodes in alphabetical order
1. List states as they are expanded
2. Show frontier/stack at every step

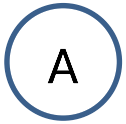
Expanded node	Frontier
-	A

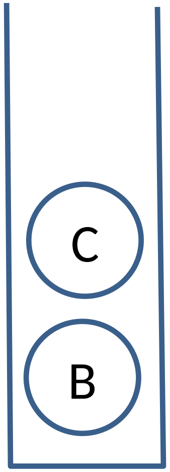
Initial state: A
Goal state: D

Using DFS (Tree)
Expand child nodes in alphabetical order
1. List states as they are expanded
2. Show frontier/stack at every step



List of states:

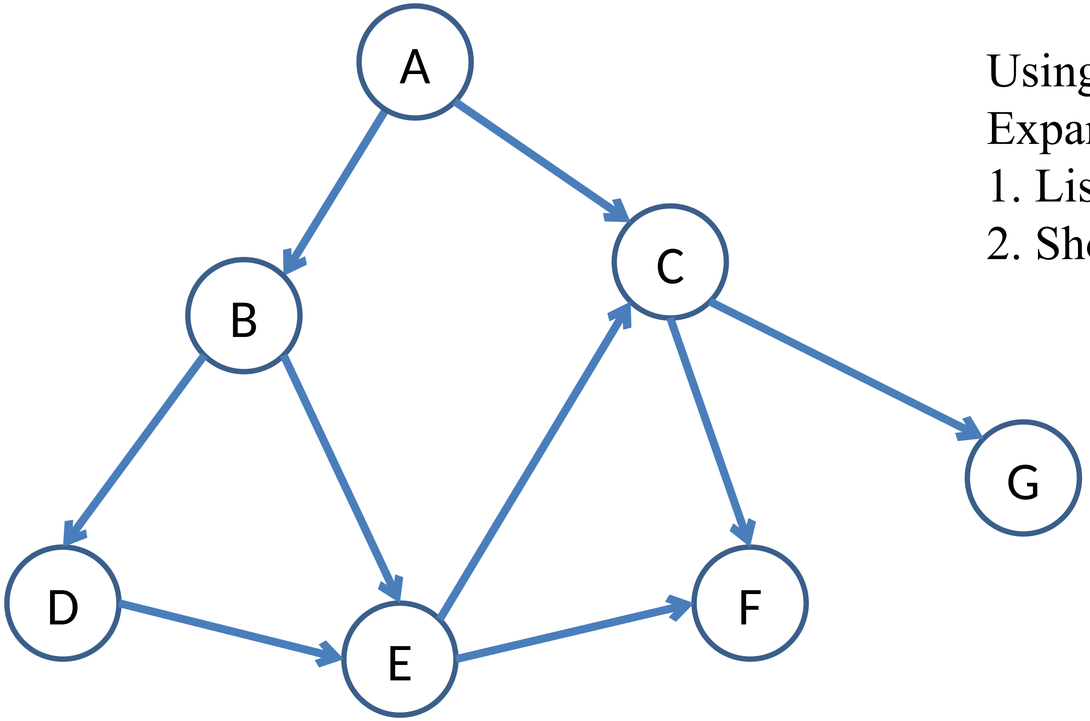




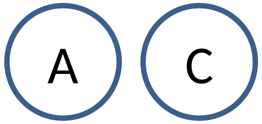
Frontier

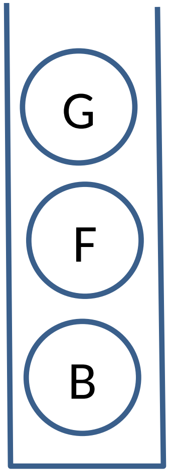
Initial state: A
Goal state: D

Using DFS (Tree)
Expand child nodes in alphabetical order
1. List states as they are expanded
2. Show frontier/stack at every step



List of states:

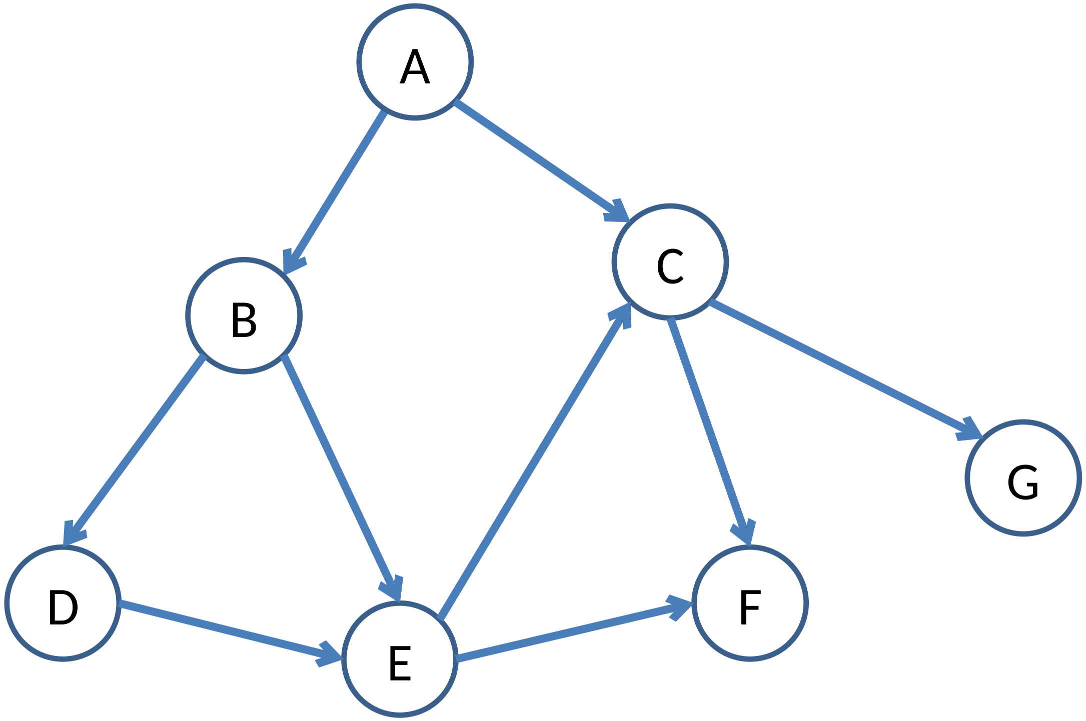




Frontier

In-class exercise

- Work on the rest of the iterations
  - Show expanded node and frontier in a table using DFS



Expanded node	Frontier
-	A
A	B, C
C	B, F, G
G	B, F
F	B
B	D, E
E	D, C, F
F	D, C
C	D, F, G
G	D, F
F	D
D (Goal)	-

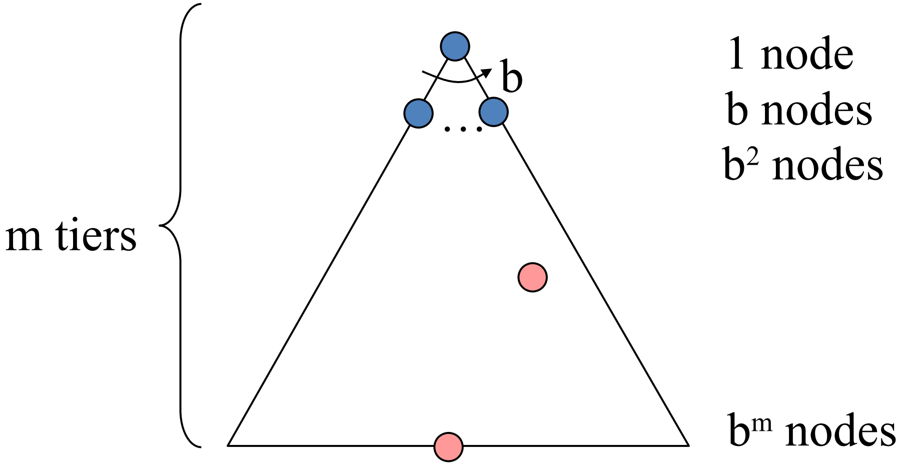
Depth-first TREE search

- Redundant paths
  - Some states can be reached in more than one way

Search Algorithm Properties

- Complete: Guaranteed to find a solution if one exists?
- Optimal: Guaranteed to find the least cost path?
- Time complexity
- Space complexity

- Visualization of search tree:
  - **b** is the branching factor
  - **m** is the maximum depth
  - solutions at various depths



m tiers

b

…

1 node

b nodes

b^2 nodes

b^m nodes

- Number of nodes in entire tree?
  - $1 + b + b^2 + …. b^m = b^{m+1} = O(b^m)$

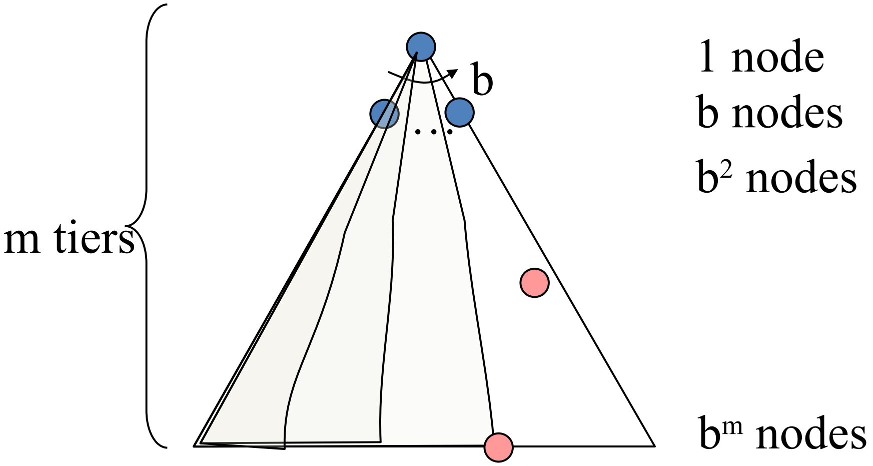
Depth-First Search (DFS) Properties

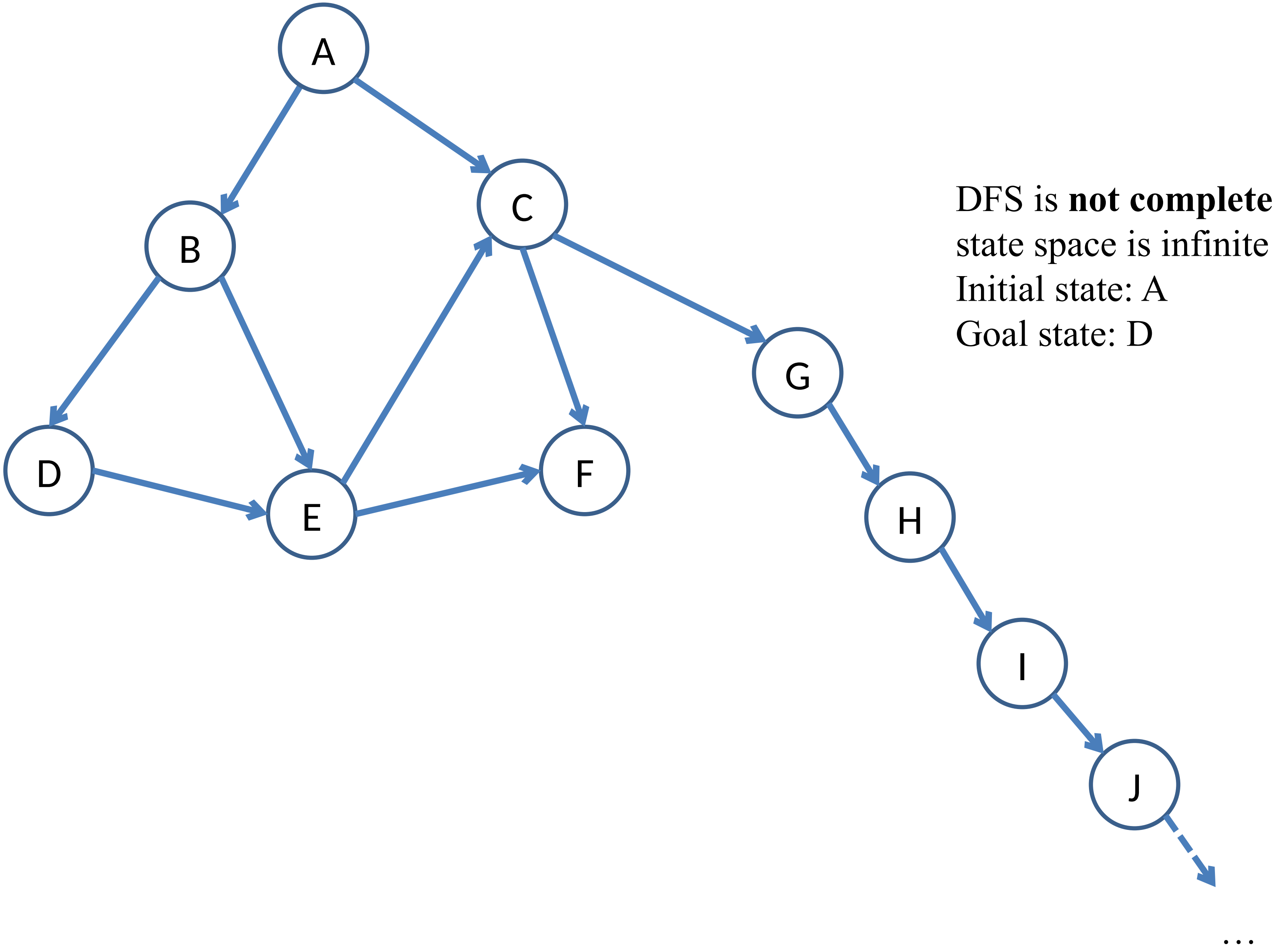
- Time complexity
  - Some left/right prefix of the tree.
  - Could process the whole tree!
  - If m is finite, takes time $O(b^m)$

- Space complexity
  - Only has nodes on path to root + siblings for each: so,

- Is it complete?
  - State space can have loops, or could be infinite, so no

- Is it optimal?
  - No, solution not always the shortest path





DFS is **not complete** if the state space is infinite

Initial state: A

Goal state: D

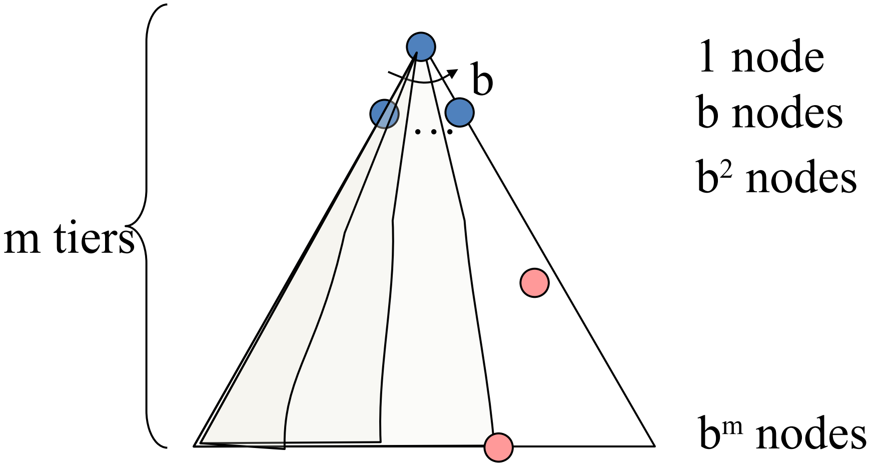
Depth-first TREE search with loop checking

- Input: A problem
- Data structures:
- frontier
  - Stack, LIFO queue
- path from root to current node

```
Initialize frontier with initial state
Loop do
    IF the frontier is empty RETURN FAILURE
    Choose top node and remove it from frontier
    IF top node is goal RETURN SUCCESS
    expand top node
    push resulting nodes to the frontier IF they are not already on the path
```

Depth-First Search with loop checking Properties

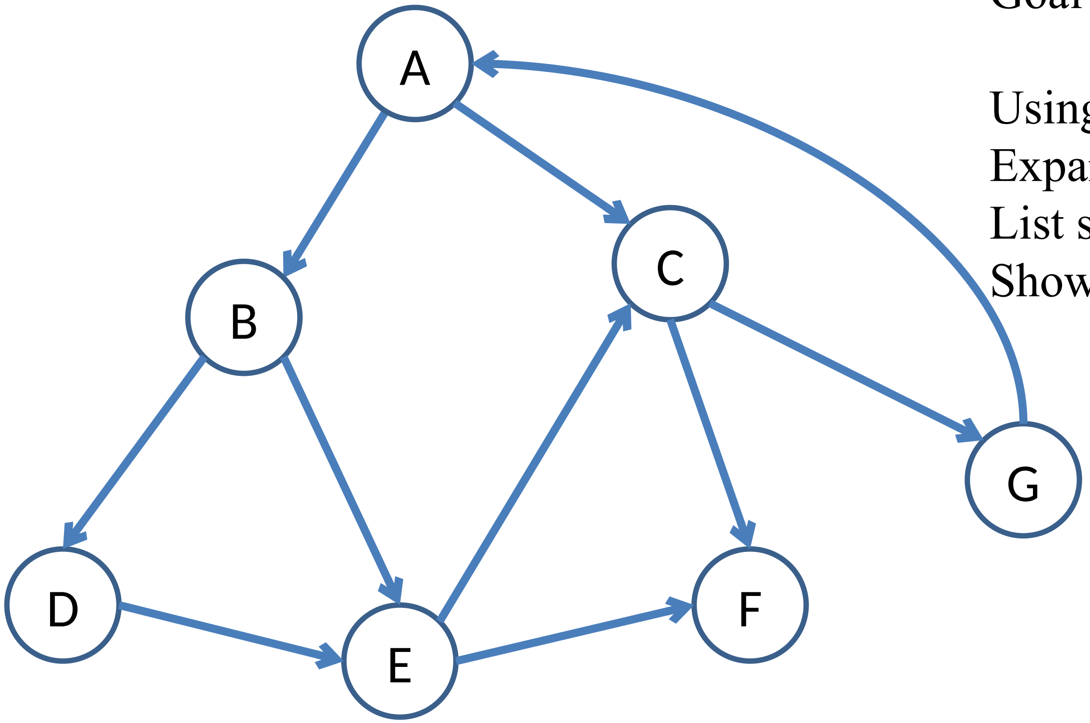
- Time complexity
  - Some left prefix of the tree.
  - Could process the whole tree!
  - If m is finite, takes time O(b^m)
- Space complexity
  - Only has nodes on path to root + siblings for each,
  - Path is of length
  - so,
- Is it complete?
  - Yes, in finite state spaces
  - But still not in infinite state spaces
- Is it optimal?
  - No, does not find the shortest path solution



Depth-first GRAPH search

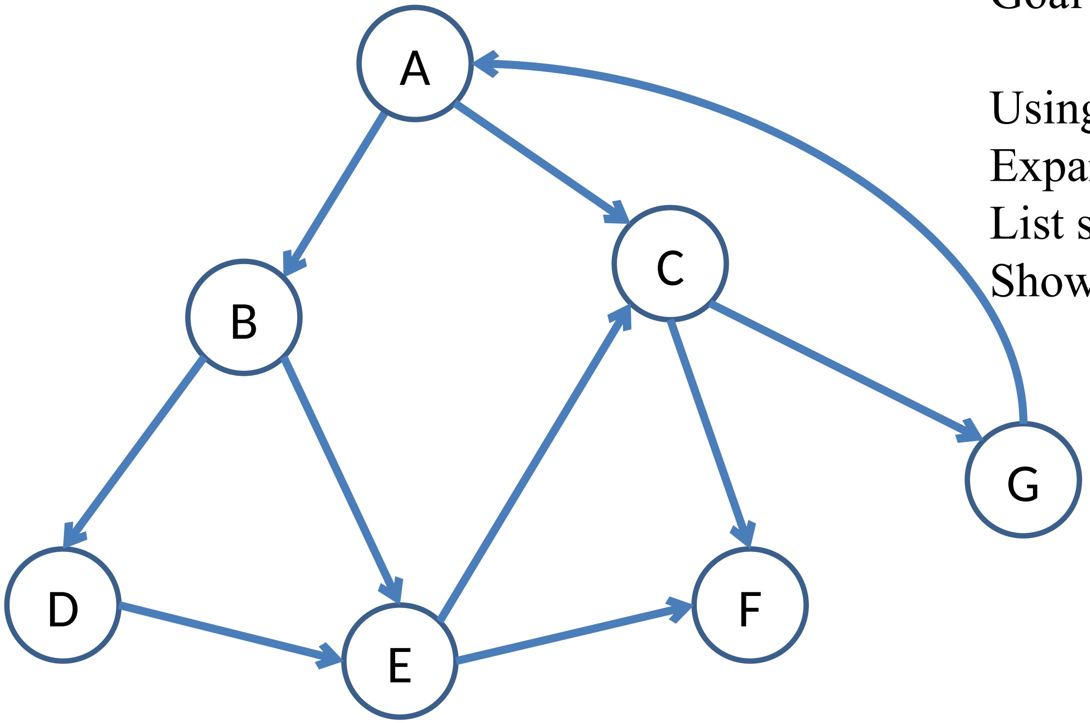
- Input: A problem
- Data structure:
  - Frontier (also called “open”)
    - Stack, LIFO queue
  - Explored (also called “closed”)
    - Set, for efficiency

```
Initialize frontier with initial state
Initialize explored to empty
Loop do
    IF the frontier is empty RETURN FAILURE
    Choose top node from frontier and remove it
    IF top node is goal RETURN SUCCESS
    Add node to explored
    expand node, pushing resulting nodes to the frontier only
    if not already on frontier or explored
```



Initial state: A
Goal state: D

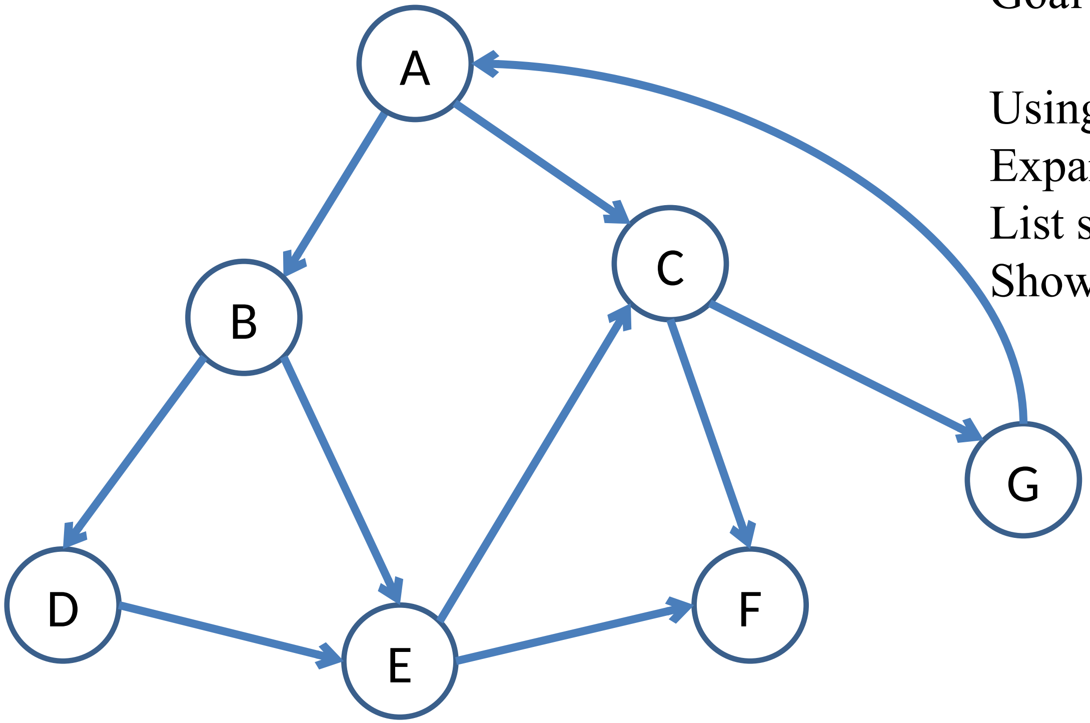
Using DFS (Graph version)
Expand child nodes in alphabetical order
List states as they are expanded?
Show frontier, explored at every step?



Initial state: A
Goal state: D

Using DFS (Graph version)
Expand child nodes in alphabetical order
List states as they are expanded?
Show frontier, explored at every step?

Frontier: [A]
Explored: {}



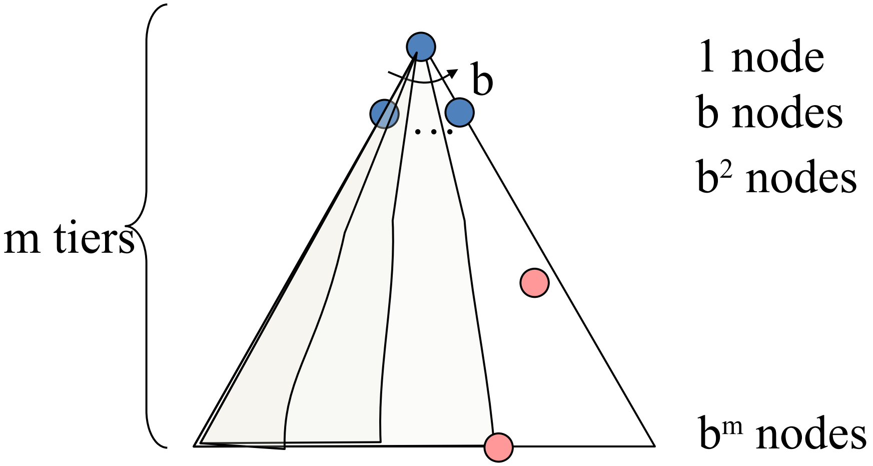
Initial state: A
Goal state: D

Using DFS (Graph version)
Expand child nodes in alphabetical order
List states as they are expanded?
Show frontier, explored at every step?

Frontier: []
Explored: {A, C, G, F, B, E, D}

Depth-First Search (Graph version) Properties

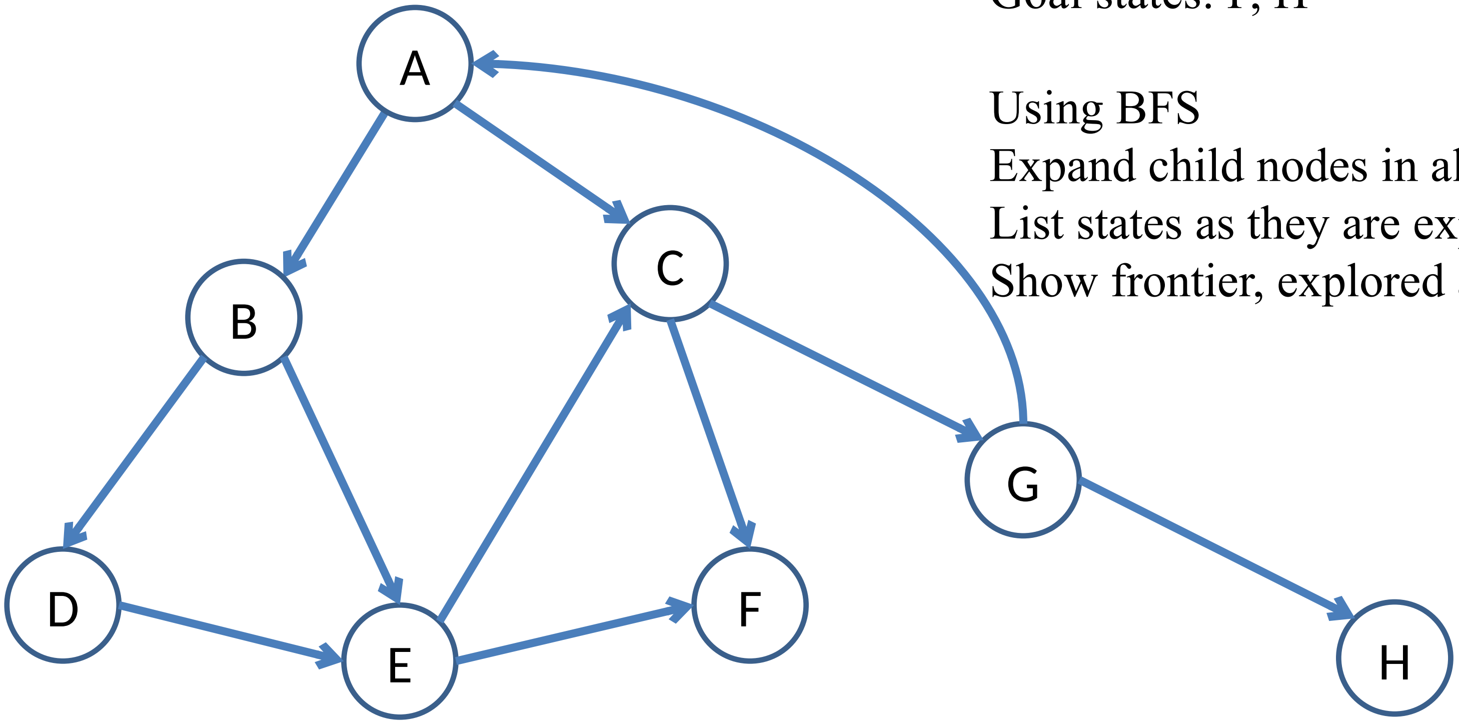
- Time complexity
  - Limited by the state space
  - Can be much smaller than $O(b^m)$
- Space complexity
  - All expanded nodes will be added to explored
  - so, $O(b^m)$
- Is it complete?
  - Yes, in finite state spaces
  - But still not in infinite state spaces
- Is it optimal?
  - No, it finds the "leftmost" solution, regardless of depth or cost



Breadth-first graph search

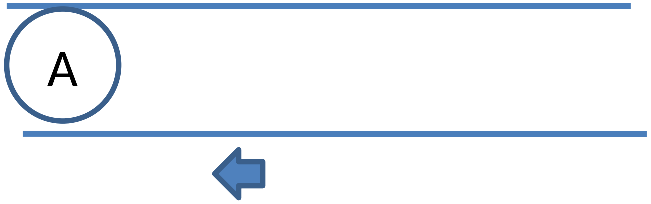
- Input: A problem
- Data structures:
  - Frontier (also called “open”)
    - **Queue, FIFO queue**
  - Explored (also called “closed”)
    - Set, for efficiency

```
Initialize frontier with initial state
Initialize explored to empty
Loop do
    IF the frontier is empty RETURN FAILURE
    Choose front-node from frontier and remove it
    Add front-node to explored
    FOR every child-node of front-node
      IF child-node not already on frontier or explored
        IF child-node is goal RETURN SUCCESS
        push child-node to frontier
```



Initial state: A
Goal states: F, H

Using BFS
Expand child nodes in alphabetical order
List states as they are expanded?
Show frontier, explored at every step?



Initial state: A

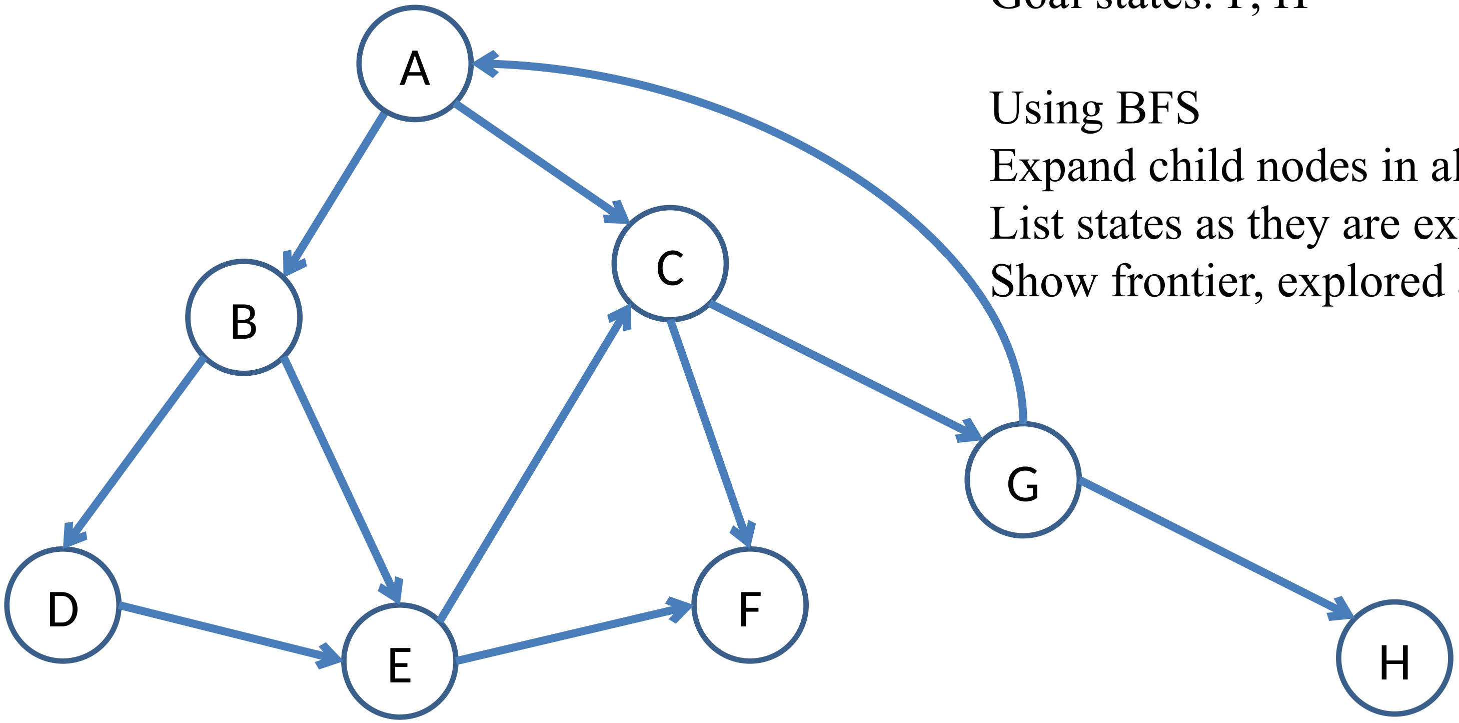
Goal states: F, H

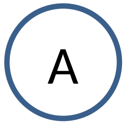
Using BFS

Expand child nodes in alphabetical order

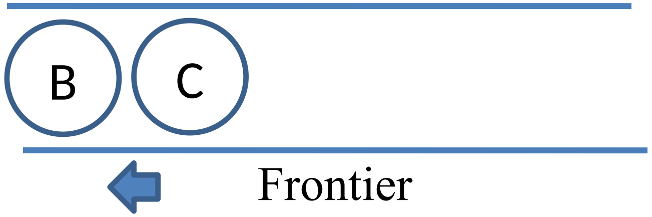
List states as they are expanded?

Show frontier, explored at every step?





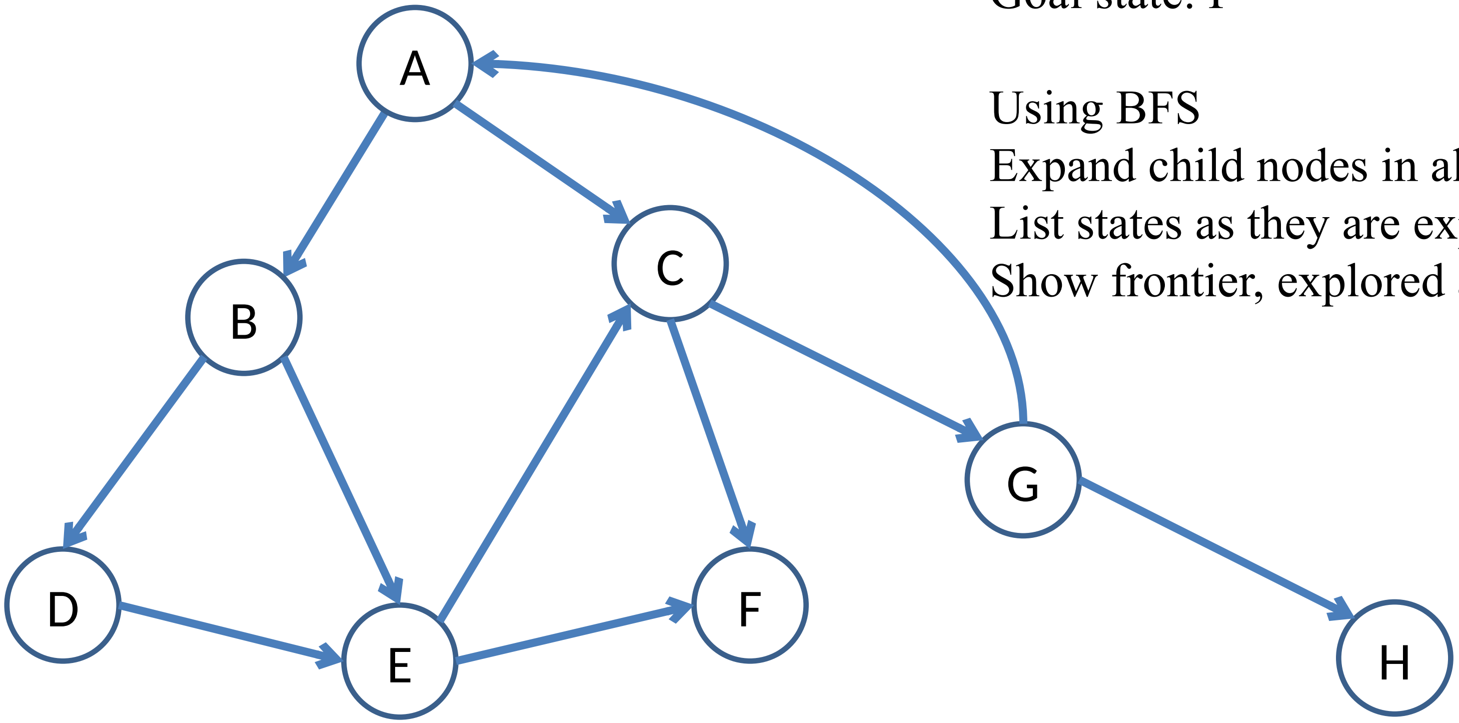
List of states:

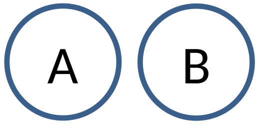


Frontier

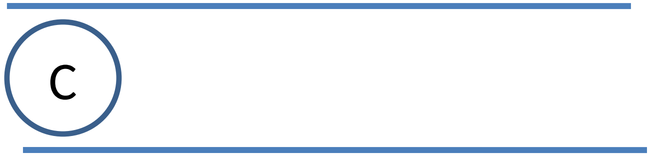
Initial state: A
Goal state: F

Using BFS
Expand child nodes in alphabetical order
List states as they are expanded?
Show frontier, explored at every step?





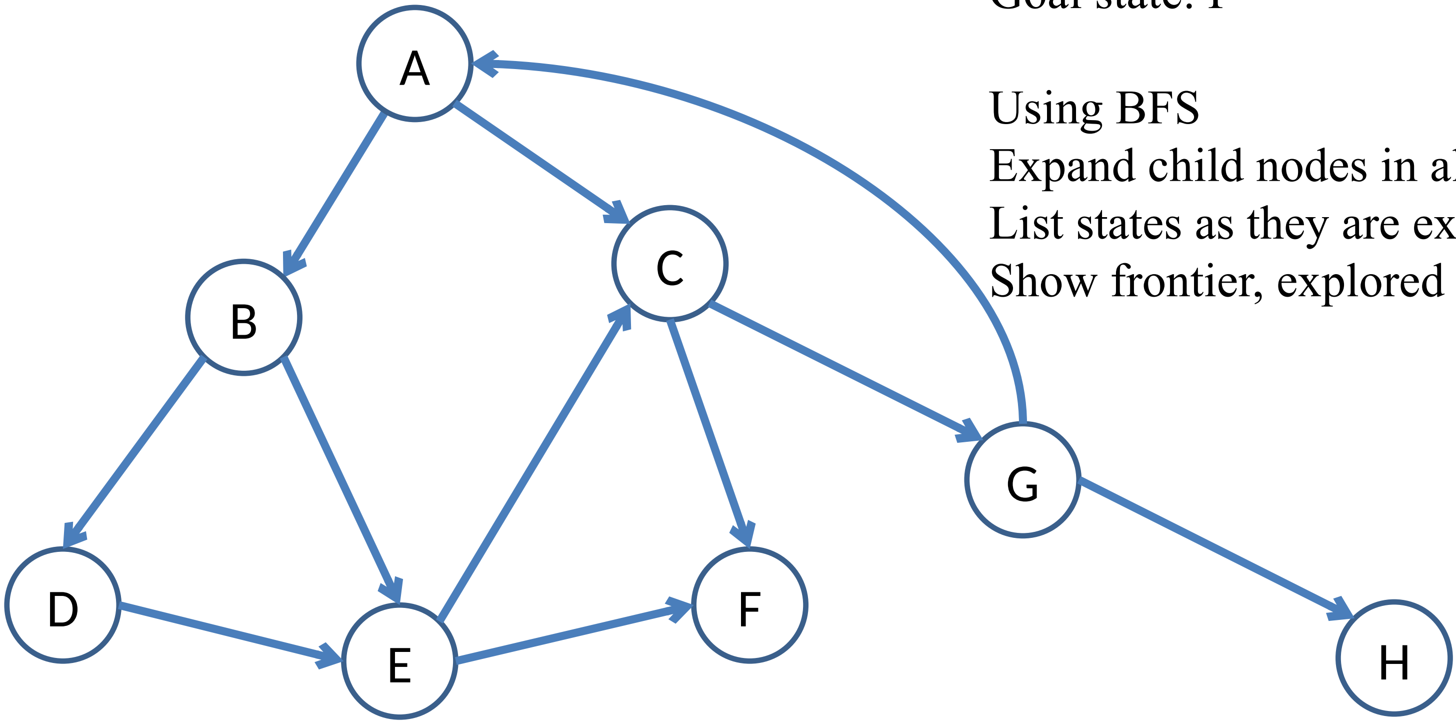
List of states:



Frontier

Initial state: A
Goal state: F

Using BFS
Expand child nodes in alphabetical order
List states as they are expanded?
Show frontier, explored at every step?



A B

List of states:

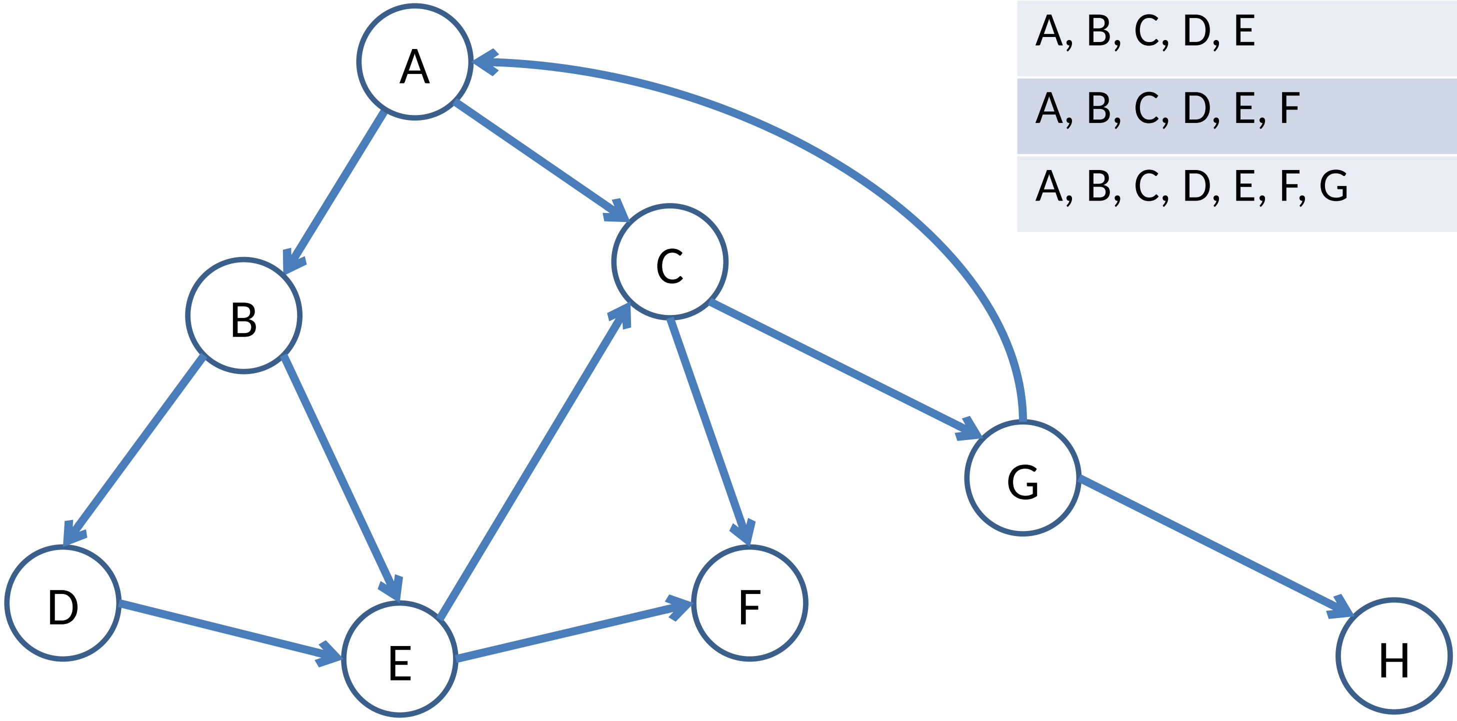
C D E

Frontier

In-class exercise

- Work on the rest of the iterations
  - Show expanded node and frontier in a table using BFS

Explored	Frontier
-	A
A	B, C
A,B	C, D, E
A, B, C	D, E, F, G
A, B, C, D	E, F, G
A, B, C, D, E	F, G – goal state
A, B, C, D, E, F	G
A, B, C, D, E, F, G	H – goal state



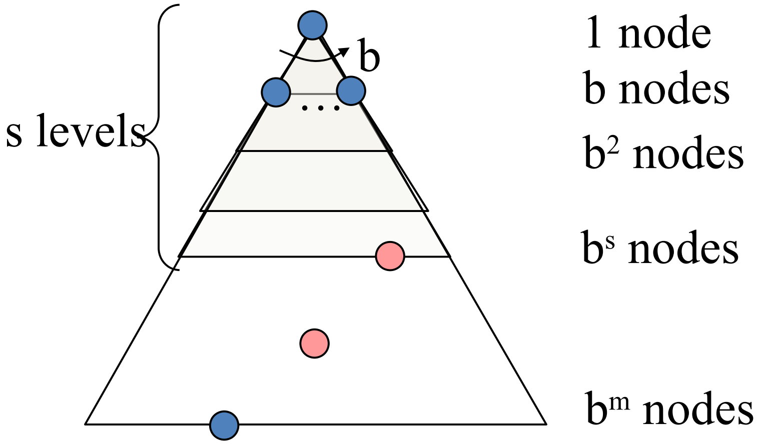
Breadth-First Search (BFS) Properties

- What nodes does BFS expand?
  - Processes all nodes above shallowest solution
  - Let depth of shallowest solution be s
  - Search takes time O(b^s)

- How much space does the frontier take?
  - Has roughly the last tier, so O(b^s)

- Is it complete?
  - s must be finite if a solution exists, so yes!

- Is it optimal?
  - **Yes**, if costs are all equal
  - more on costs later



Acknowledgement

- https://inst.eecs.berkeley.edu/~cs188/su20/

References

- Russel and Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Prentice Hall, 2010.