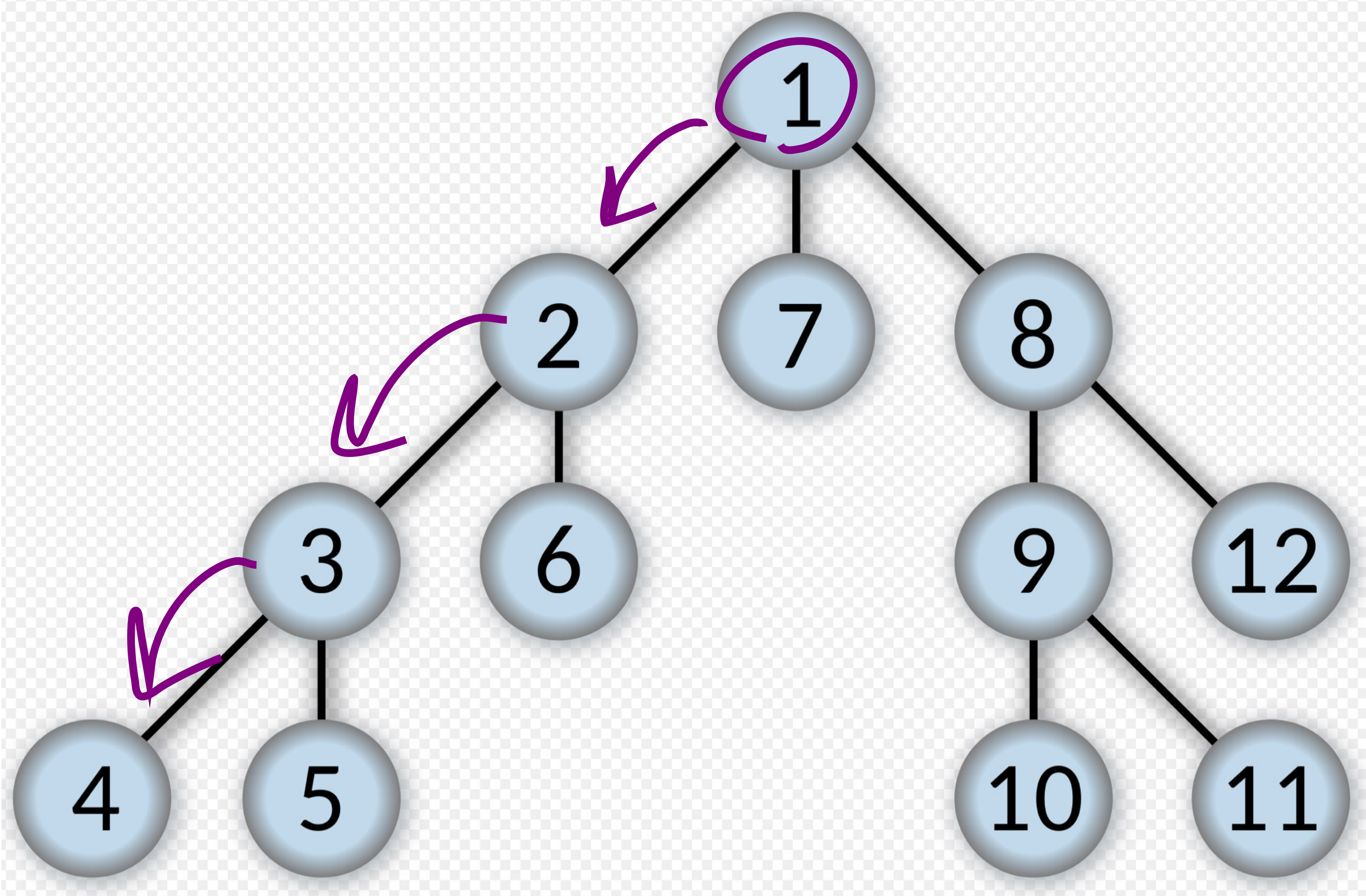
CSCI 3202: Intro to Artificial Intelligence

Lecture 7: Depth-First Search (DFS)

Uniform Cost Search

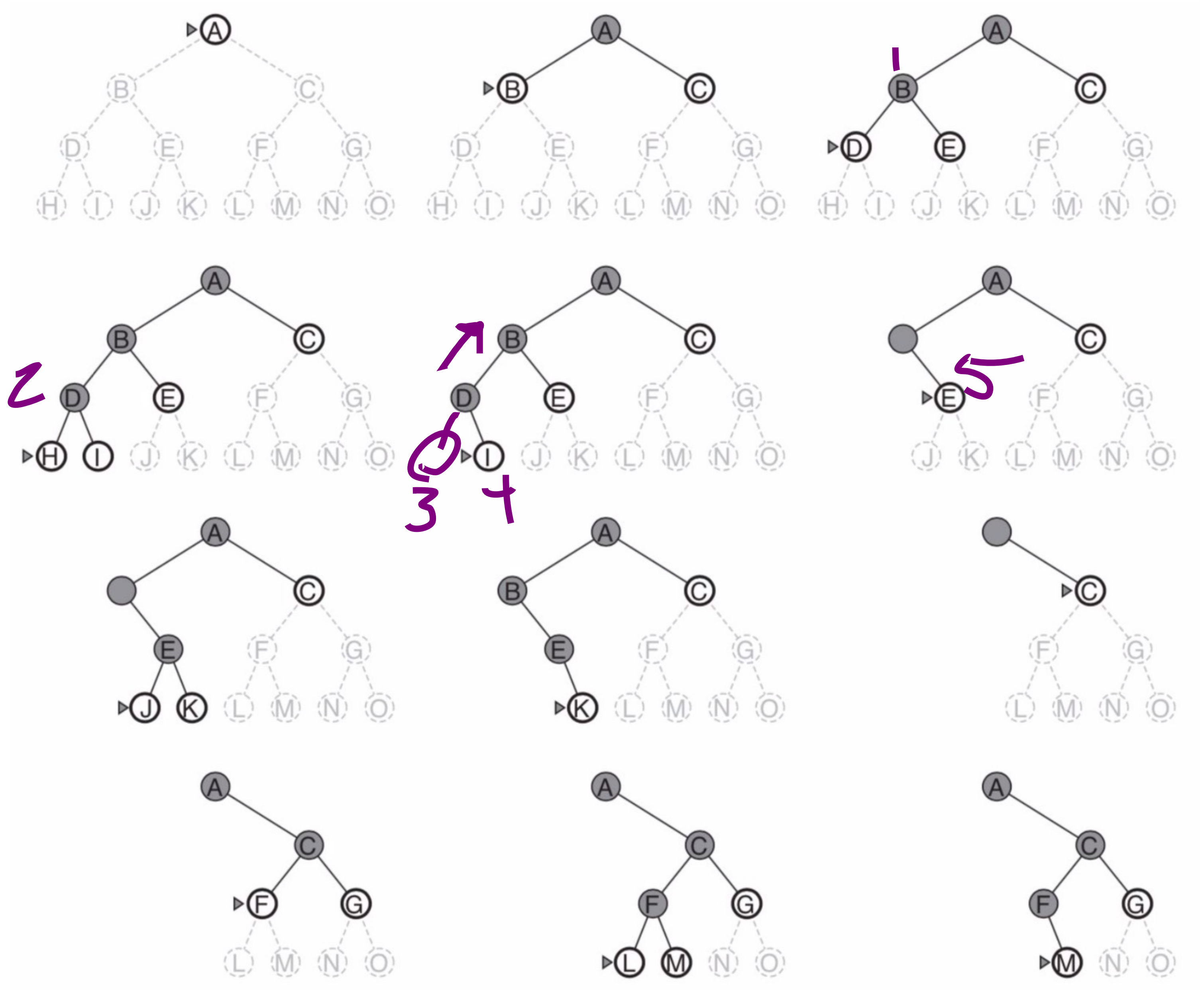
Rhonda Hoenigman

Department of Computer Science



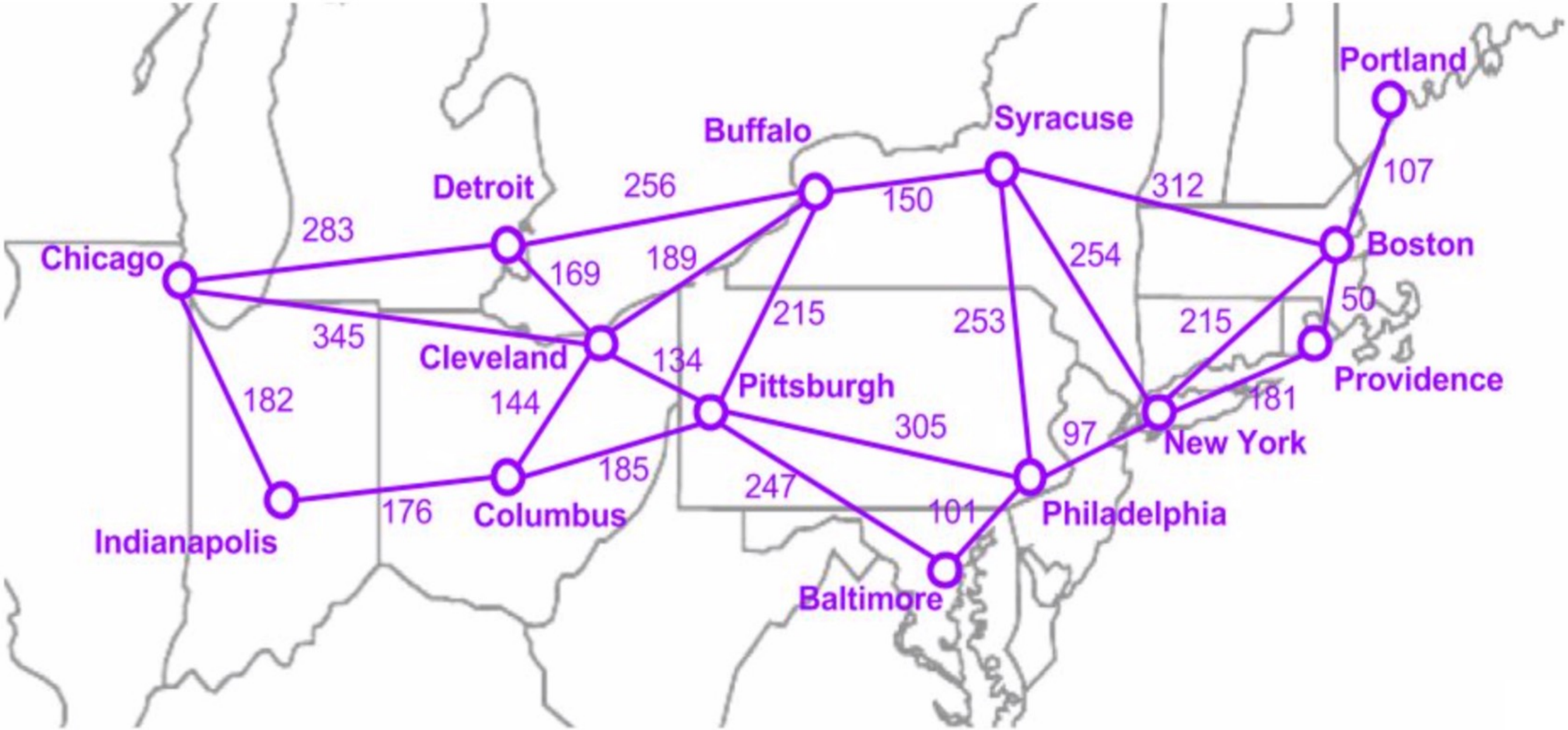
Depth–First Search (DFS)

- Uninformed
- Expand deepest node first (LIFO)
- “Back up” to next-deepest node with unexplored successors
- Implementation determines nodes explored and known
  - Iterative and recursive versions



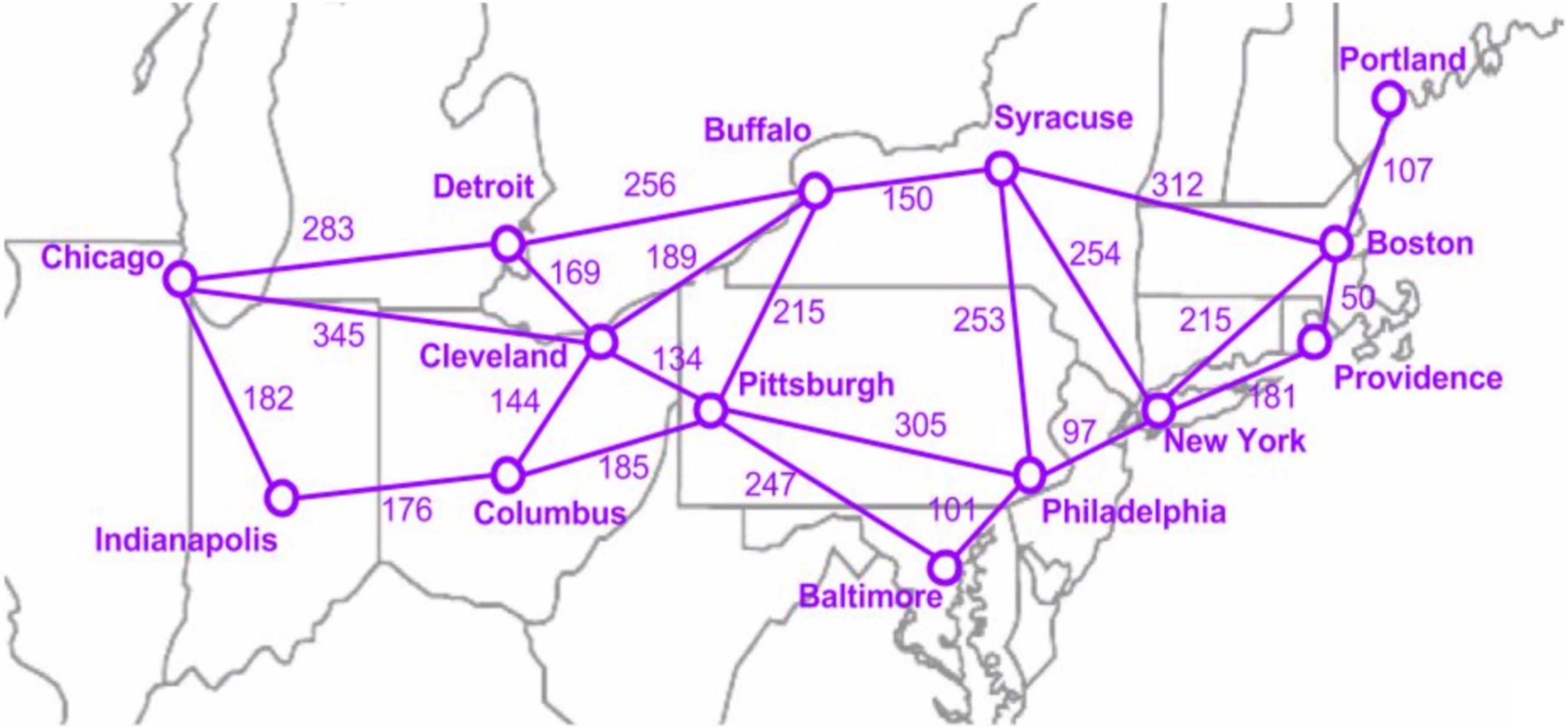
Depth–First Search (DFS)

Example: Traveling in the US northeast
Step costs: miles between cities along major highways



Depth–First Search (DFS)

Example: Traveling in the US northeast. **Question**: Would changing the step cost function change our DFS result?
Step costs: estimated travel time (minutes) along major highways at 5PM east coast time on a Friday

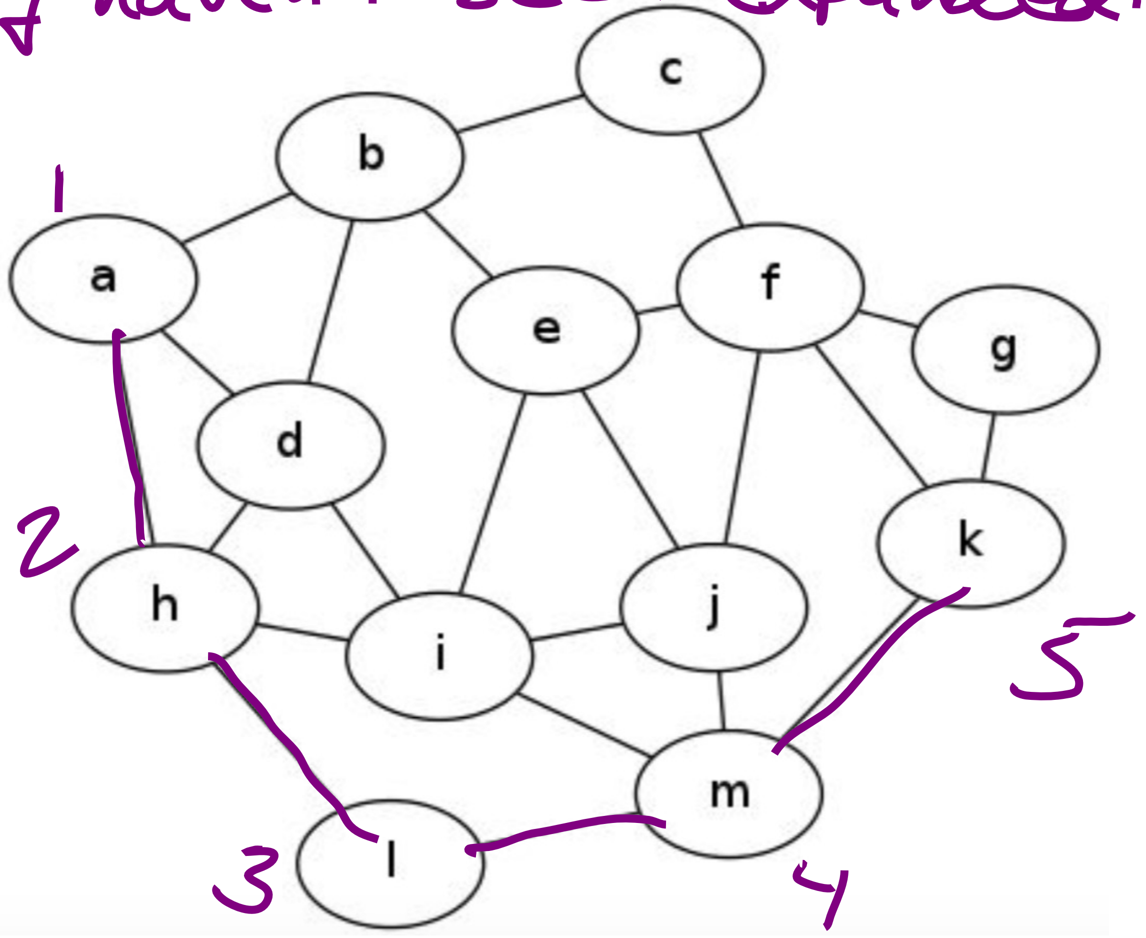


Depth–First Search (DFS)

Example: Number the nodes in the search graph according to the order in which they would be expanded using DFS to find a path from a to k. Assume that nodes within a layer are added to the stack by alphabetical order. What is the route that DFS yields? Nodes added to frontier stack if they haven't been expanded.

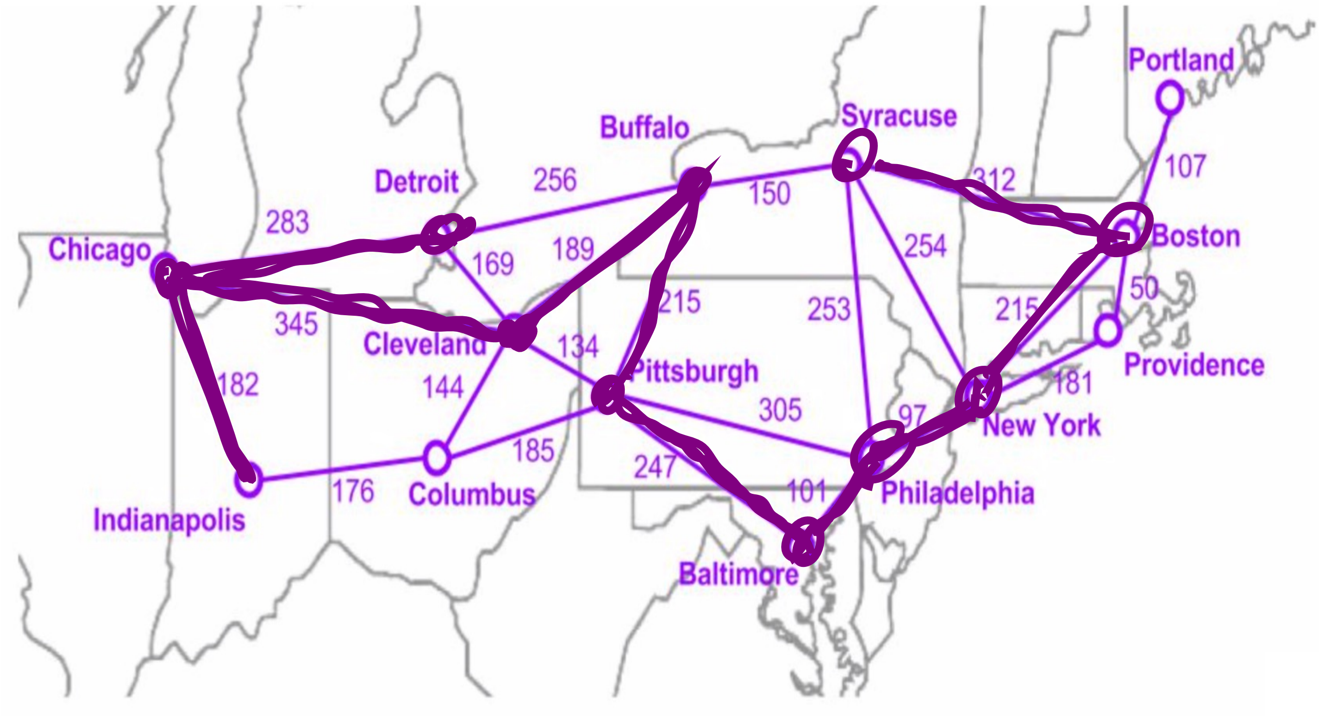
Exp	Stack
a	b, d, h
h	b, d, d, i, l
l	b, d, d, i, m
m	b, d, d, i, i, j, k
k	

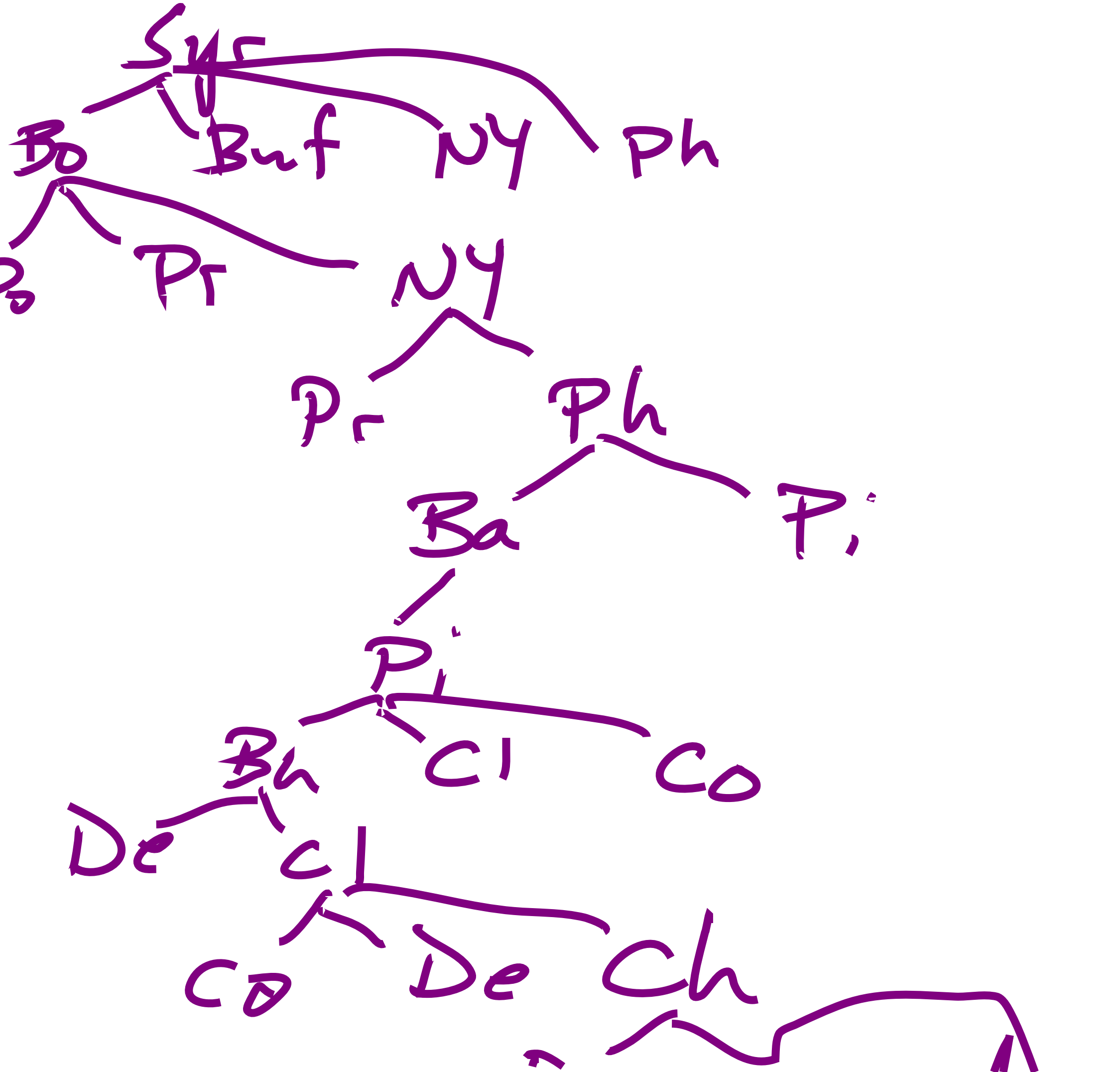
$\begin{array}{|c|}\hline h \\ \hline d \\ \hline b \\ \hline\end{array}$



Depth_First Search (DFS)

Example: Draw the search tree for the graph below. Include successor nodes not ~~explored~~ expanded. ~~Explore~~ Expand in alphabetical order. Start: Syracuse. Goal: Indianapolis.



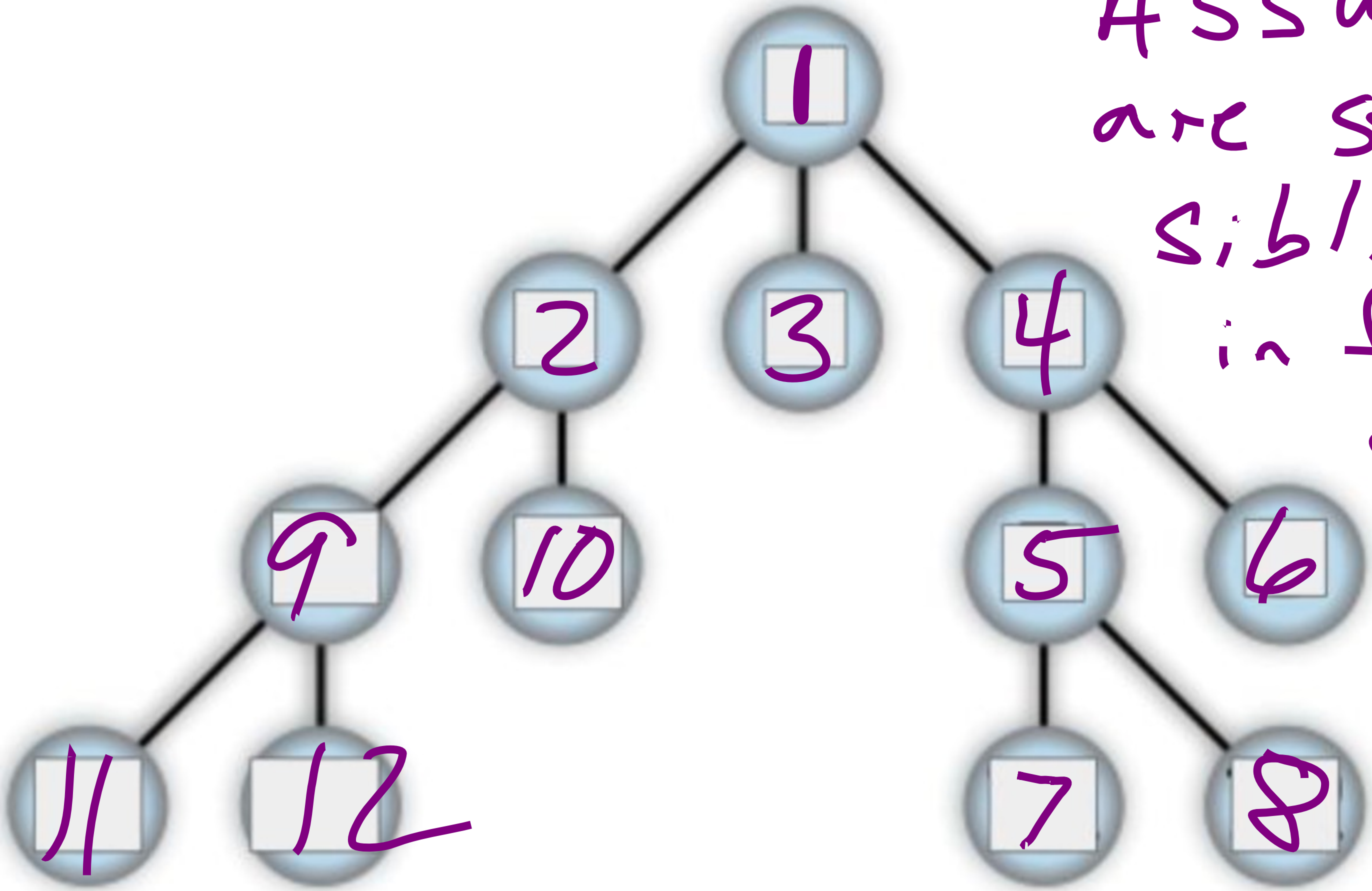


End

Depth–First Search (DFS)

Example: Number the nodes in the search tree according to the order in which they would be added to frontier using DFS. Assume that the goal is not found, and nodes are added to the frontier stack from left to right.

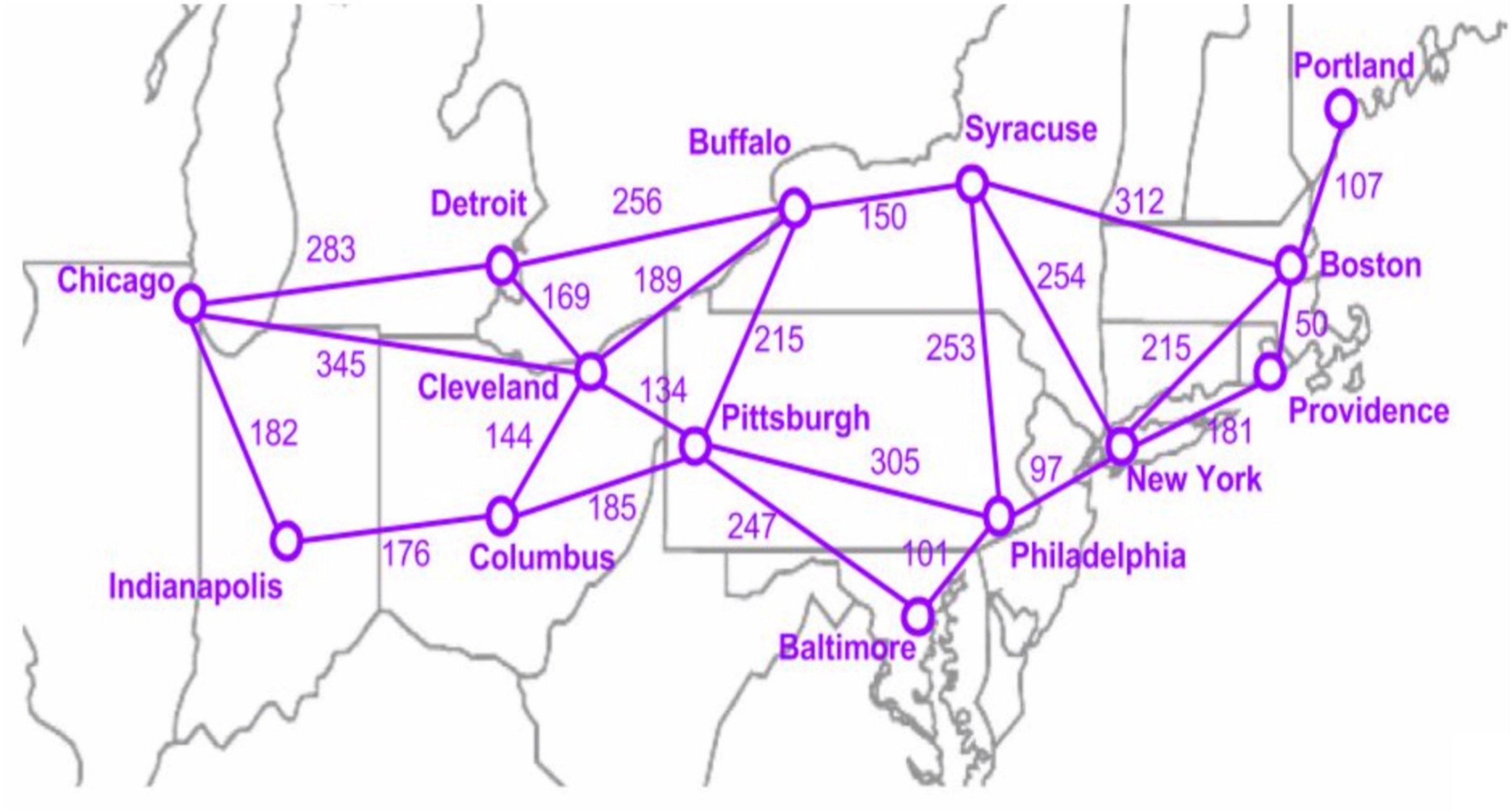
stack



Assumes we are storing sibling nodes in frontier. Purely recursive solution would have different

Depth_First Search (DFS)

order



Complete? Yes. If solution exists

Optimal? No. not even in unweighted graph

Depth-First Search (DFS)

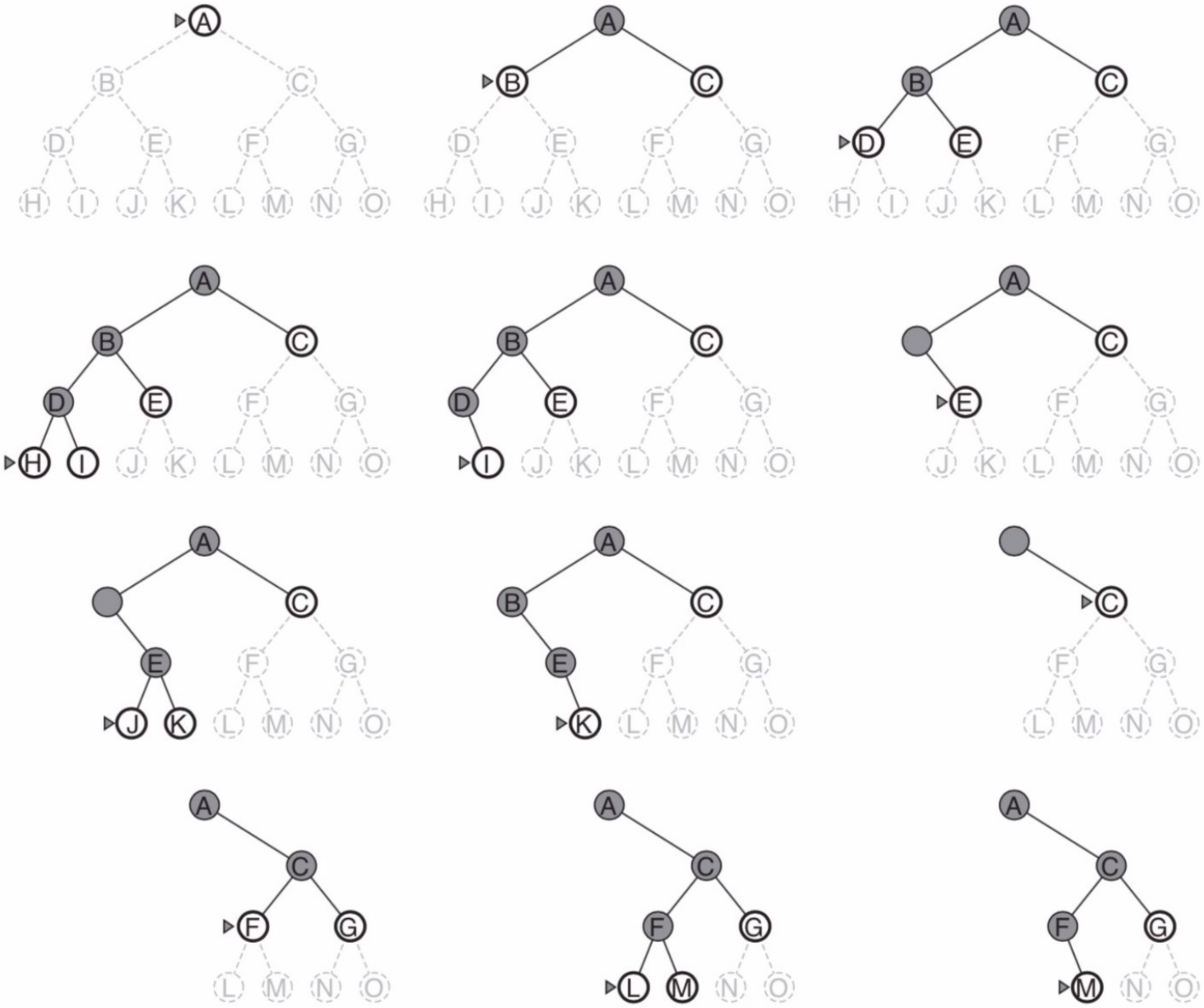
Time Complexity: Same as BFS

- branching factor b
- maximal depth of m layers
- shallowest goal state in layer d

➢ might need to generate all b^m states

➢ could be substantially more than just going to shallowest goal state b^d

➢ total worst case: $\mathcal{O}(b^m)$



Depth–First Search (DFS)

Space Complexity:

- branching factor b
- maximal depth of m layers
- shallowest goal state in layer d

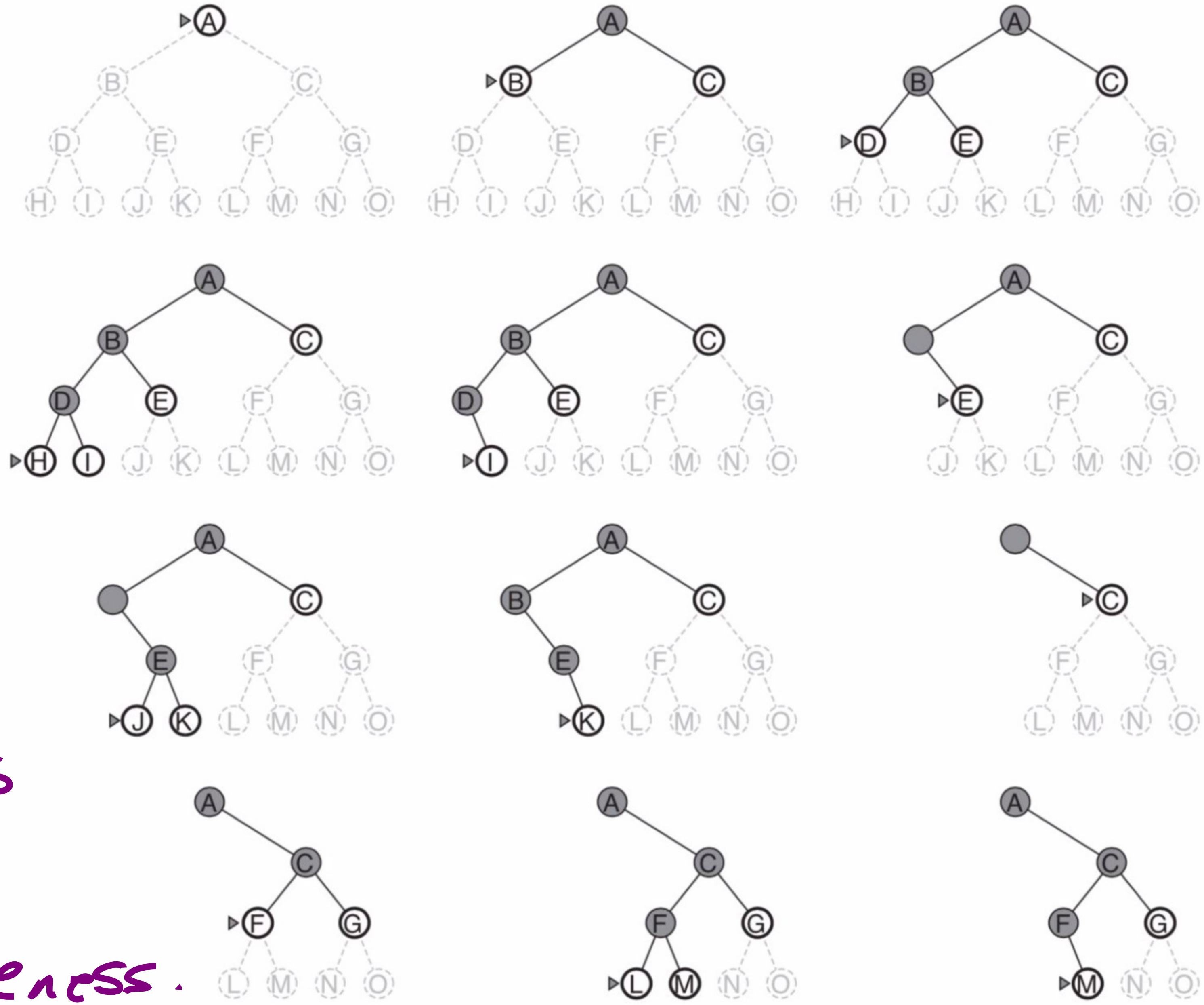
➢ If all nodes stored in frontier: $\mathcal{O}(b^m)$, same as BFS

➢ Recursive: only need to have one branch expanded at a time: b
… for each of m layers.

➢ total worst case: $\mathcal{O}(mb)$

Better than BFS

➢ Potential failure in infinite state spaces. Needs finite graph for completeness.



not found if DFS stuck in infinite branch

Depth–First Search (DFS)

Offshoots:

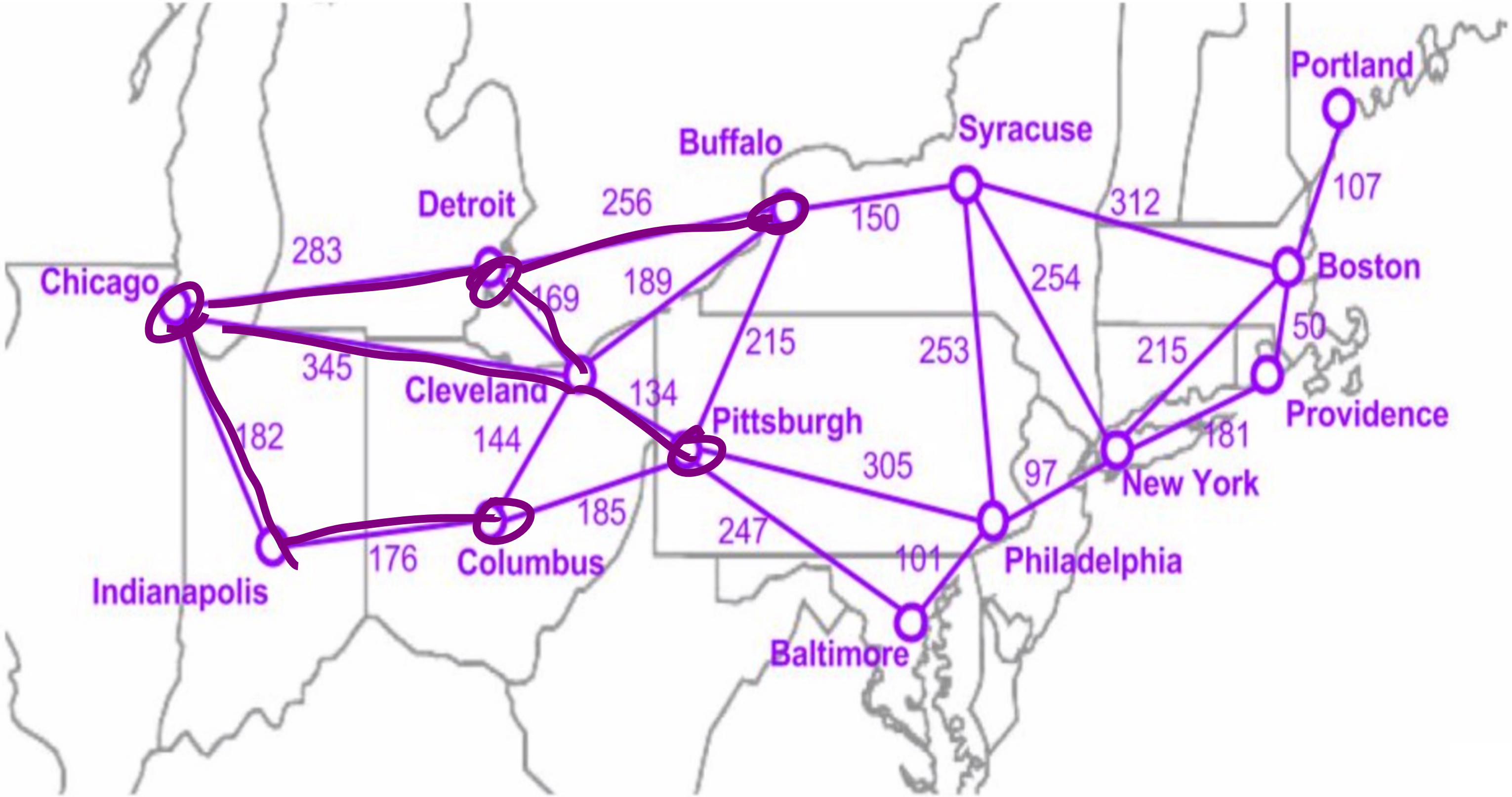
Depth-limited search:

- Search only to maximal depth l

Iterative deepening search:

- Gradually increase l until you find a solution

- Blend of elements of BFS and DFS

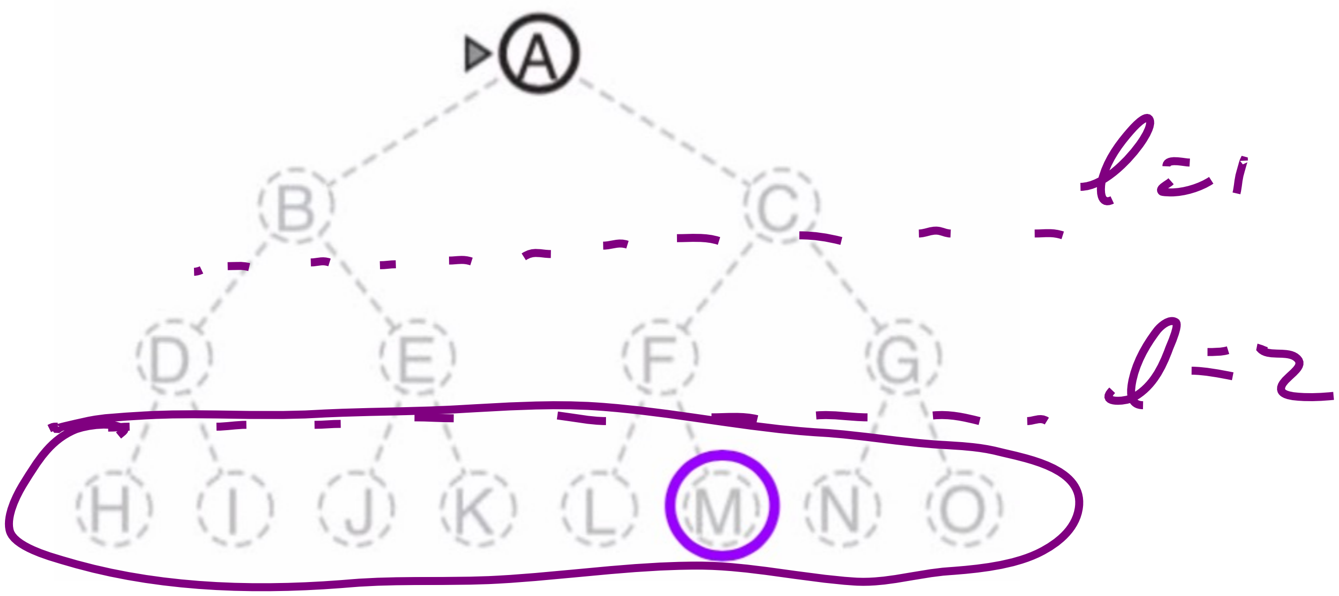


Depth–First Search (DFS)

Iterative deepening search:

- Gradually increase l until you find a solution
- Blend of elements of BFS and DFS
- Question: Why isn't this massively computationally intensive?

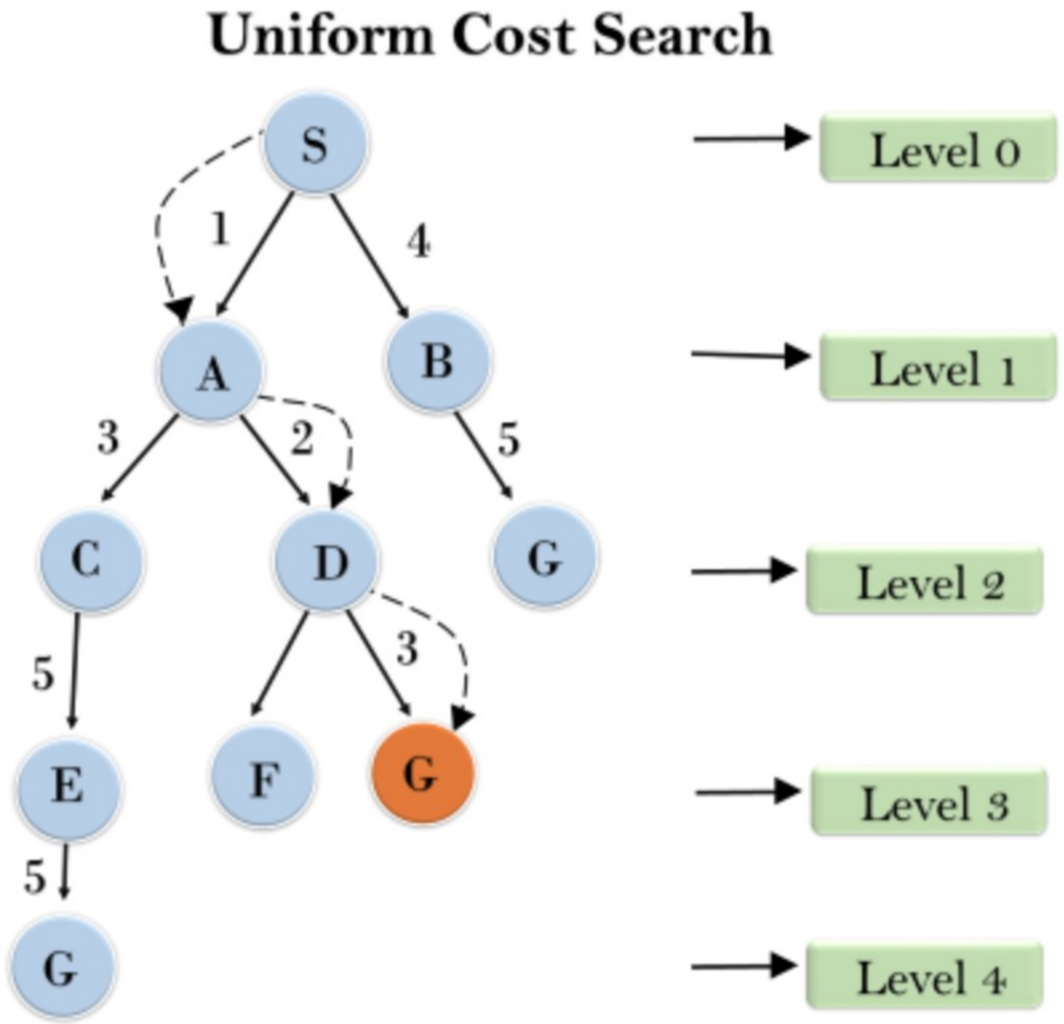
# of nodes to search increases as we go deeper into graph. Nodes at top of the search tree can be searched quickly.



CSCI 3202: Intro to Artificial Intelligence

Uniform Cost Search (UCS)

Rhonda Hoenigman
Department of Computer Science



Uniform–cost Search (UCS)

- BFS strategy
- Expand cheapest node first (lowest path cost)
- Frontier is a priority queue
- Cost function sets priority

Chicago → Cleveland

Chi → Ind

Start

Chi - 0

Det - 283

$\boxed{\text{Cle - 345}}$

~~Ind - 182~~

Col - 358

Buf - 283+256

Cle - 283+169,

2

182

Uniform–cost Search (UCS)

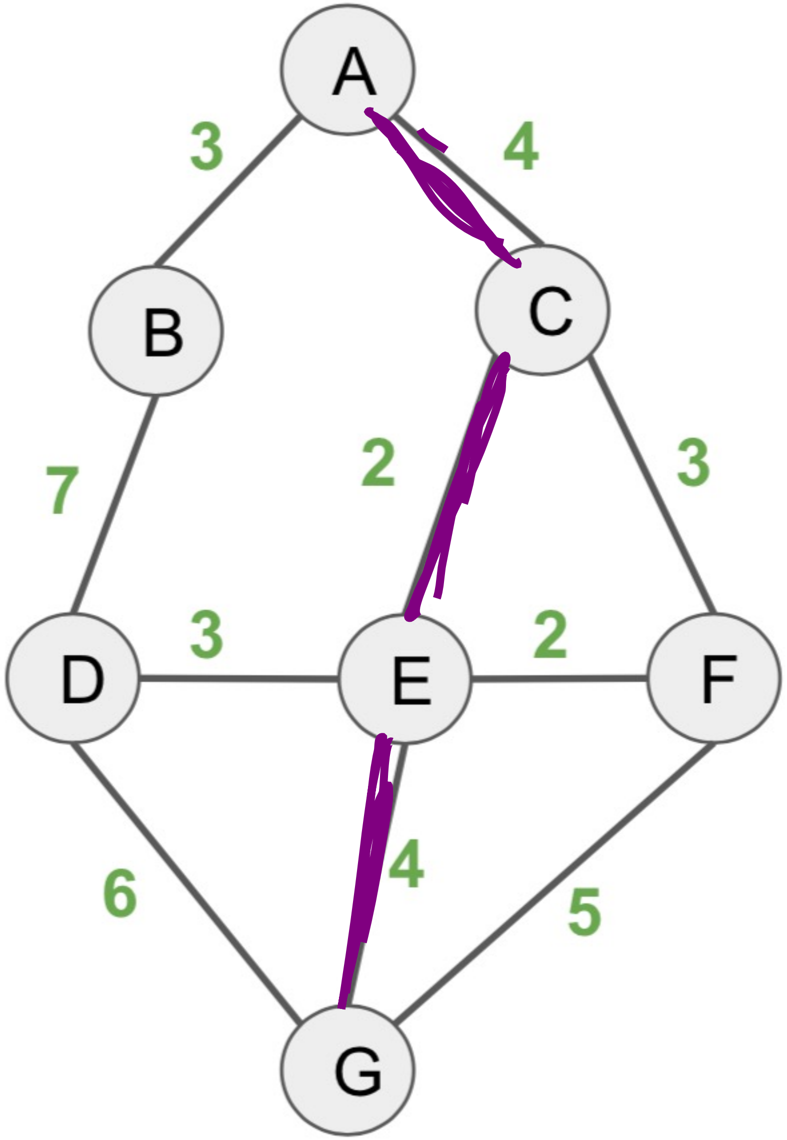
3 5 8

- ***Goal test occurs when node is selected for expansion***

- Because we know we've taken the cheapest path to get there, UCS is **optimal if all edge weights > 0**

- It is also **complete** because it's a more general form of BFS (which is complete)

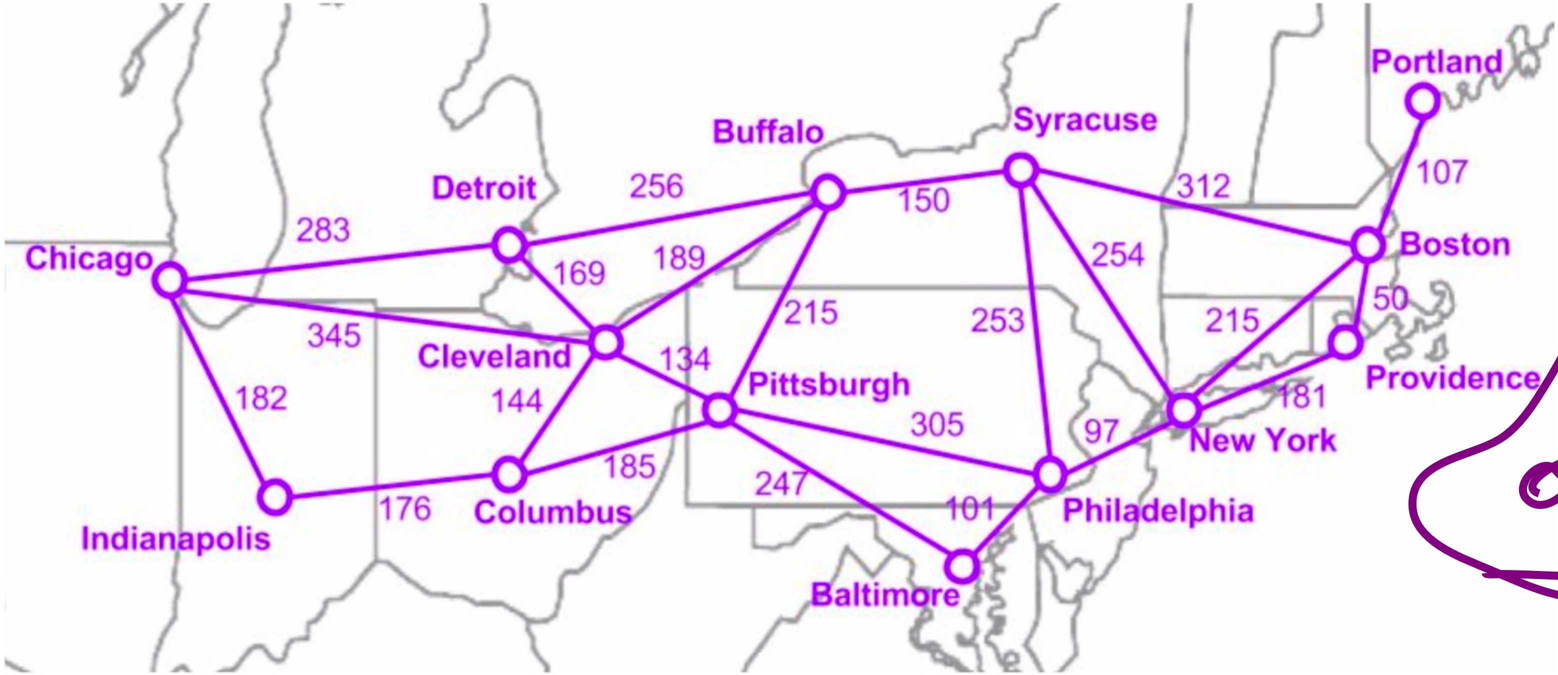
Uniform–cost Search (UCS)



Example: Perform a UCS on the graph below. A is the starting point; G is the goal.

Exp	Frontier
A	(B, 3), (C, 4)
B	(C, 4), (D, 10)
C	(D, 10), (E, 6), (F, 7)
E	(D, 9), (F, 7), (G, 10)
F	(D, 9), (G, 10)
D	(G, 10)
G	

Uniform-cost Search (UCS)



Start here on Friday.

Example: Use UCS to find a route from Detroit to Philadelphia.

Uniform–cost Search (UCS)

- Can get stuck if there are sequences of no-cost actions. Optimality requires positive edge weights

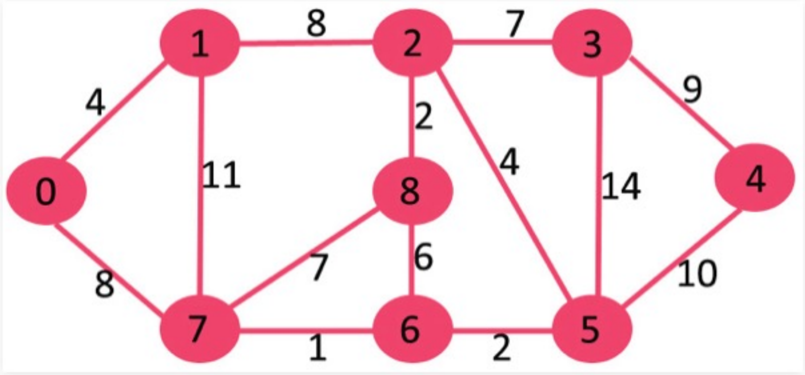
- Worst-case in time and space complexity: $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$

  - C* is cost of optimal solution
  - ϵ is minimal action cost

- Potential inefficiency: Explores in every “direction”

Dijkstra's Shortest Path Algorithm

❖ Uniform Cost Search is a variant of Dijkstra's shortest path algorithm.



Example: Use Dijkstra's algorithm to find the shortest path from 0 **to all other nodes** (Shortest Path Tree)

Next Time

A* Search