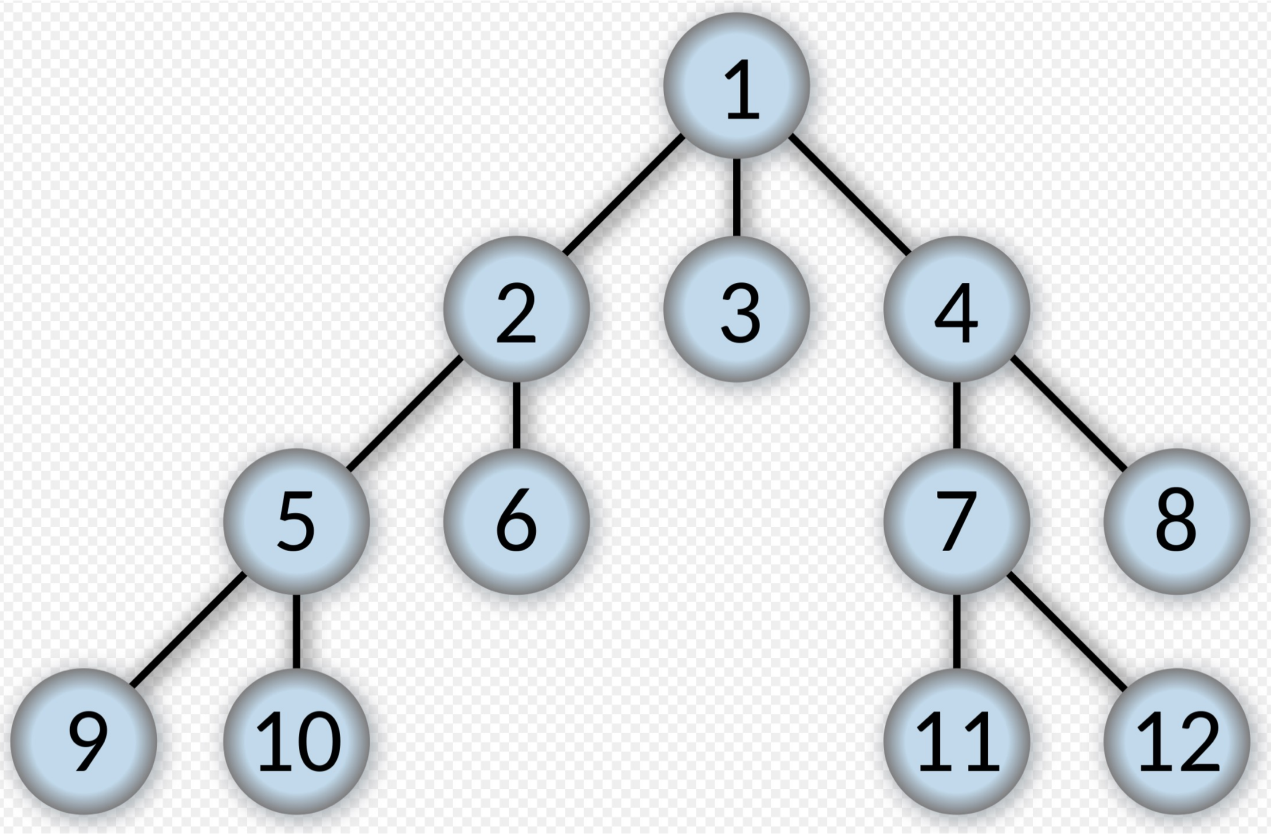
CSCI 3202: Intro to Artificial Intelligence

Lecture 6: Breadth-First Search (BFS)

Rhonda Hoenigman

Department of Computer Science



Assignment 1 posted on Canvas

Due: Feb 4 at 6pm.

Python notebook

Review – agents and states

action: move, suck up dirt

Agents: actions, percepts.

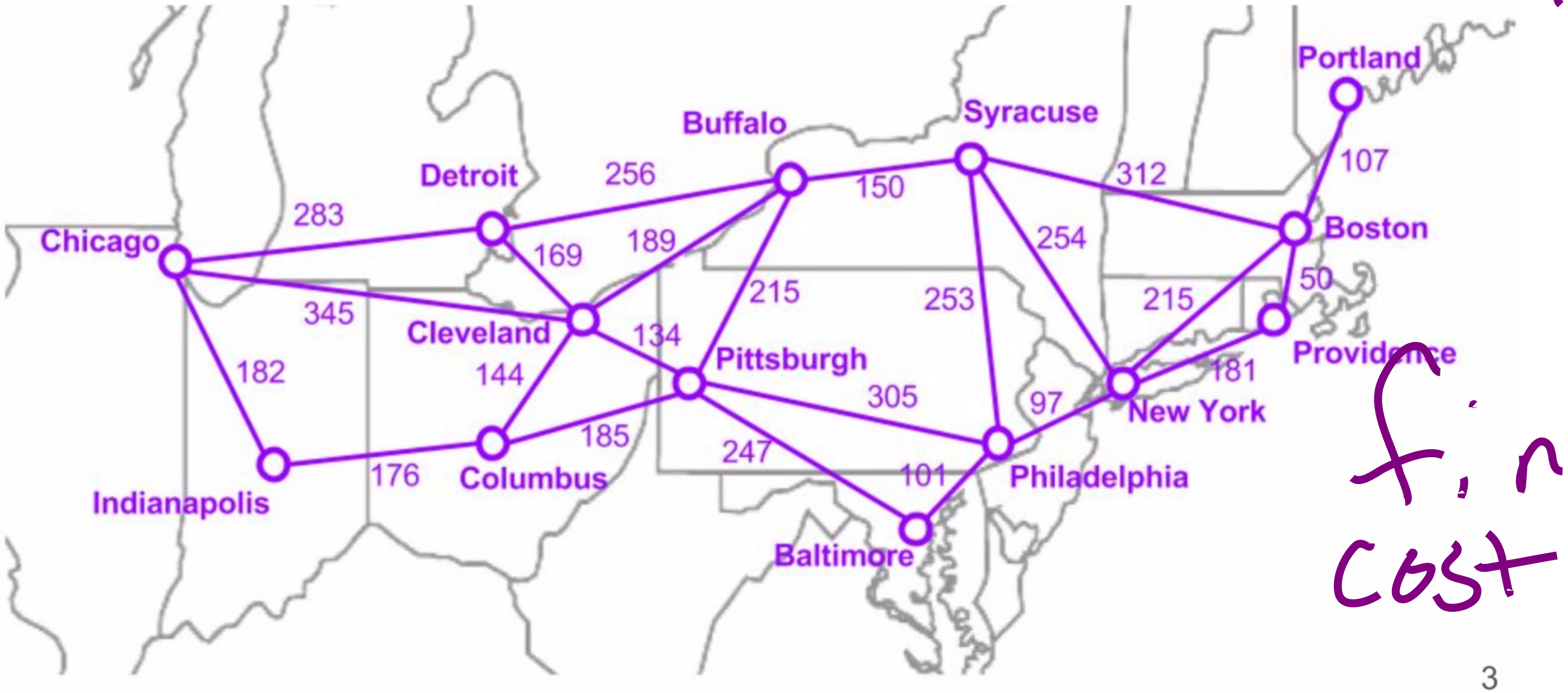
E.g. robot vaccum

states: location, dirty or clean

States: discrete configuration of the environment.

E.g positions of agents, dirty and clean cells

Search algorithms: find problem solution –sequence of actions to go from current state to goal state.



Path finding problem

find lowest cost path

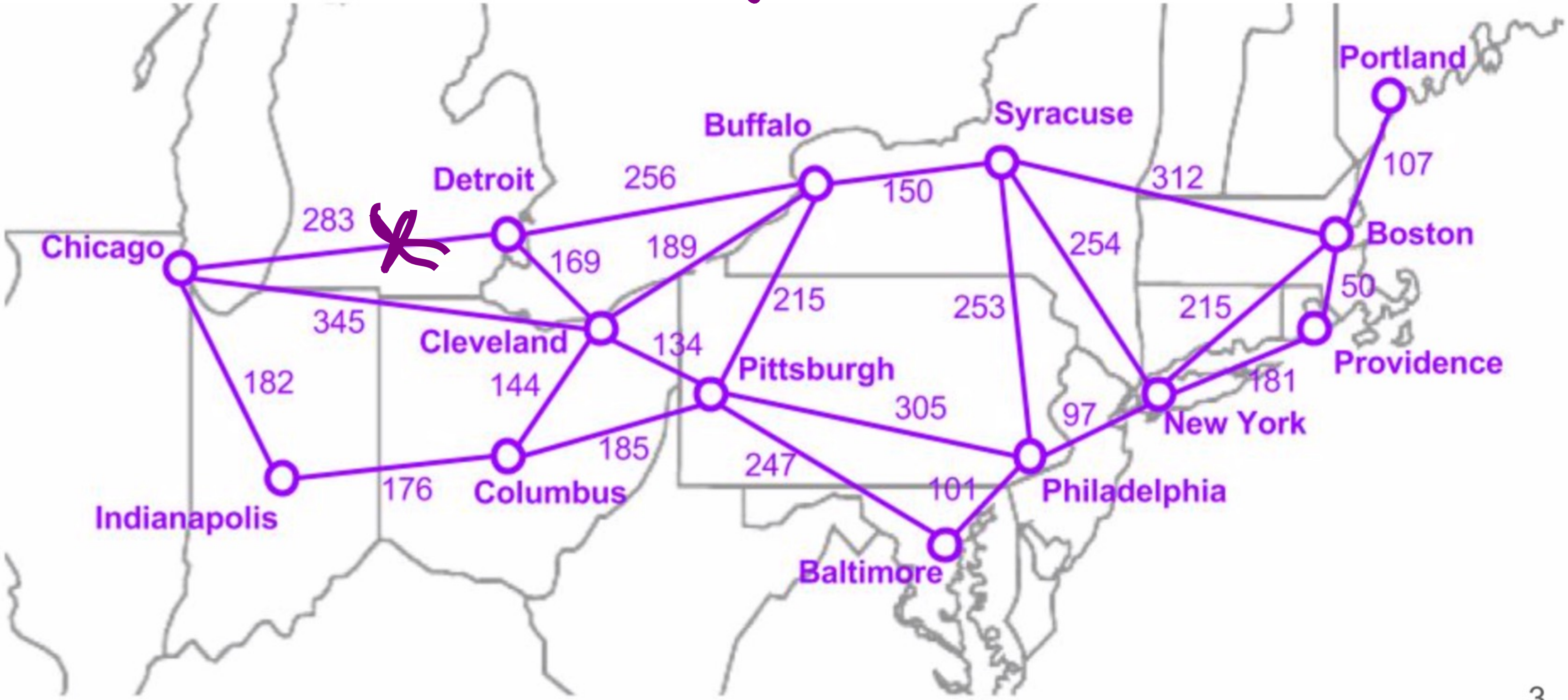
Search

Uninformed Search - no additional information about states beyond that in the problem definition

traffic jam

Informed Search - Some idea of which non-goal states are "more promising" than others

more information is better



Search strategies this week

Breadth-first search (BFS) – search across the tree before searching deeper into the tree.

Depth-first search (DFS) – search deeper into the tree before searching across the tree

Uniform Cost Search – BFS strategy with additional logic

Many, many, many search algorithms built on basic premise of BFS or DFS.

Search – building a search tree

A few variables:

Branching factor: ***b***, maximum number of successors of any node b = 5

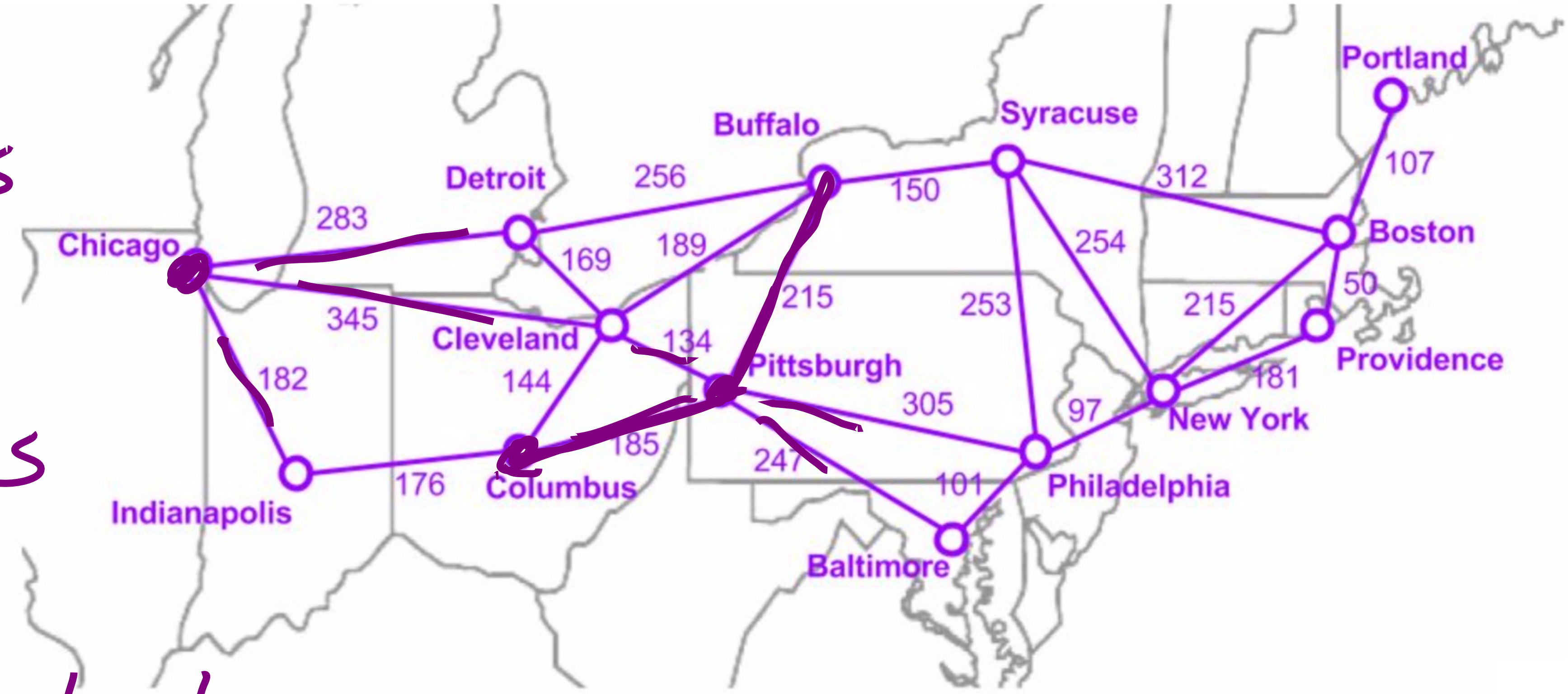
Depth: ***d***, the depth (in the search tree) of the shallowest goal node

Chicago:
3 successors

Pitts:
5 successors

Chicago

Ind, Cle, Detroit



Search Indy

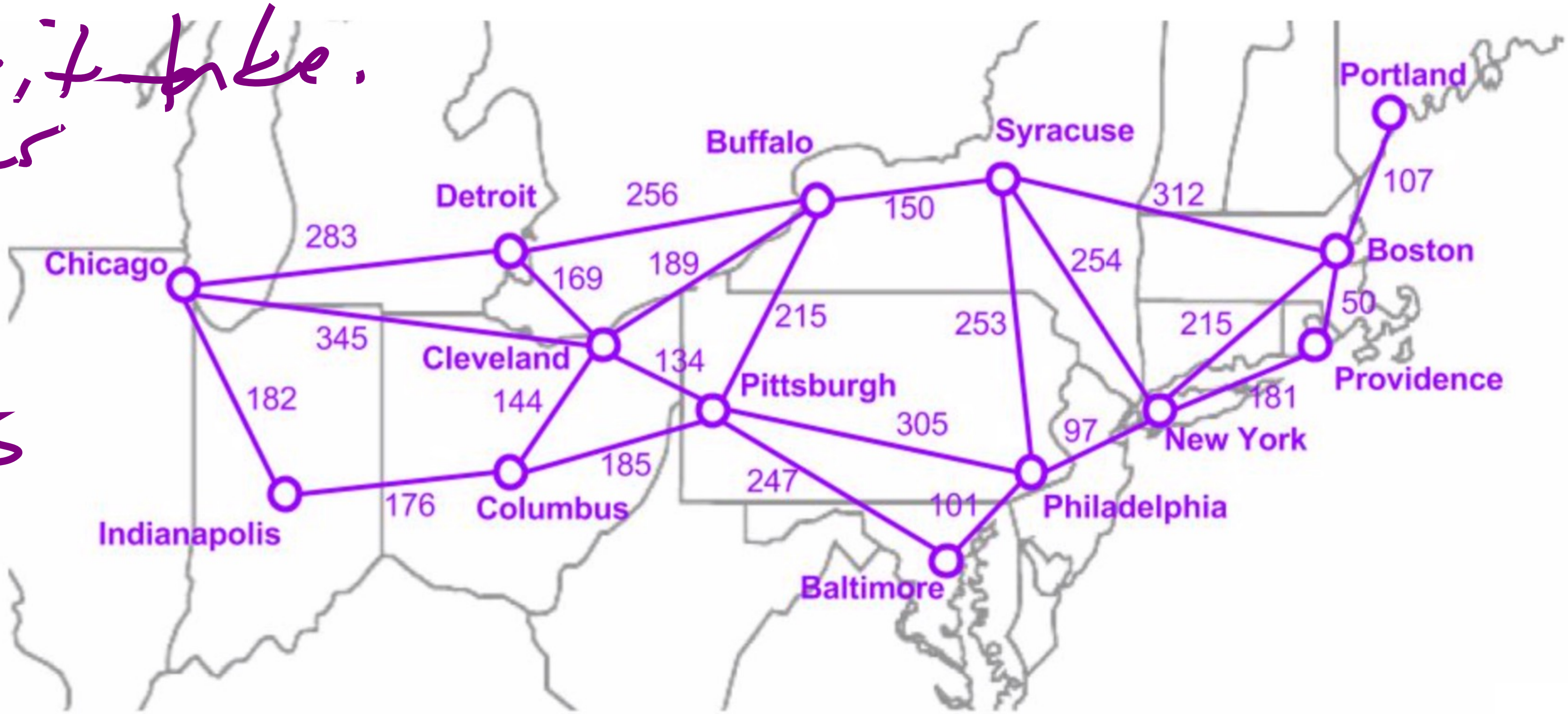
Things to think about:

Completeness: Alg. guaranteed to find a soln. if one exists

Optimality: Alg guaranteed to find optimal soln. lowest cost

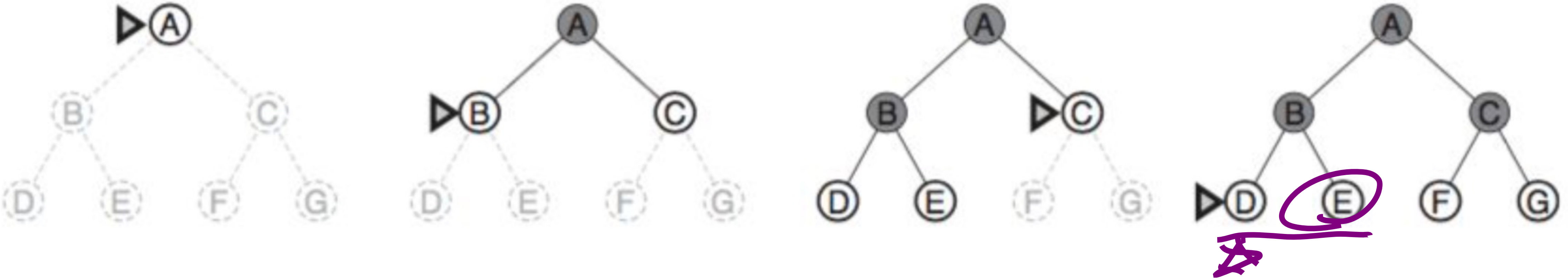
Time Complexity:
How long does it take.
# of operations

Space Complexity:
How much memory does it need



Breadth–first Search (BFS)

- Uninformed
- Expand all nodes at a given depth before proceeding into to the next layer (FIFO)
- Apply a goal test to each node



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

Breadth–first Search (BFS)

Example: Traveling in the US northeast

Start: Chicago

Goal: Pittsburgh

Ex: Ch:

Front: [Ind, Cle, Det]

Ex: [Chi, Cle]

Front: [Ind, Det, Col, Pitts, Buf]

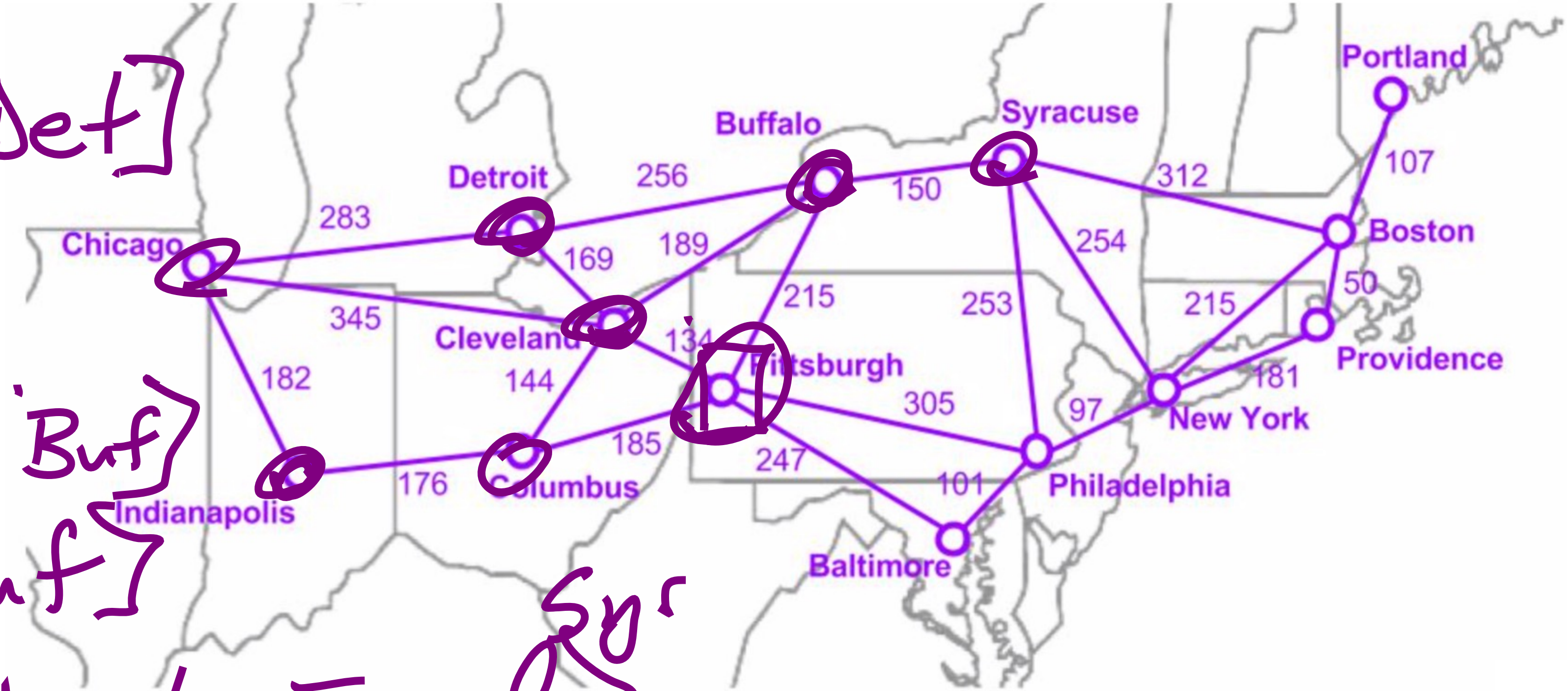
Ex: [Chi, Cle, Buf]

Ex: [Chi, Cle, Det, Ind] Syr

Ex: [

Define step costs:

- Number of cities to goal (unweighted)
- Miles between cities along major highways (weighted)
- Time to travel to next city (weighted)



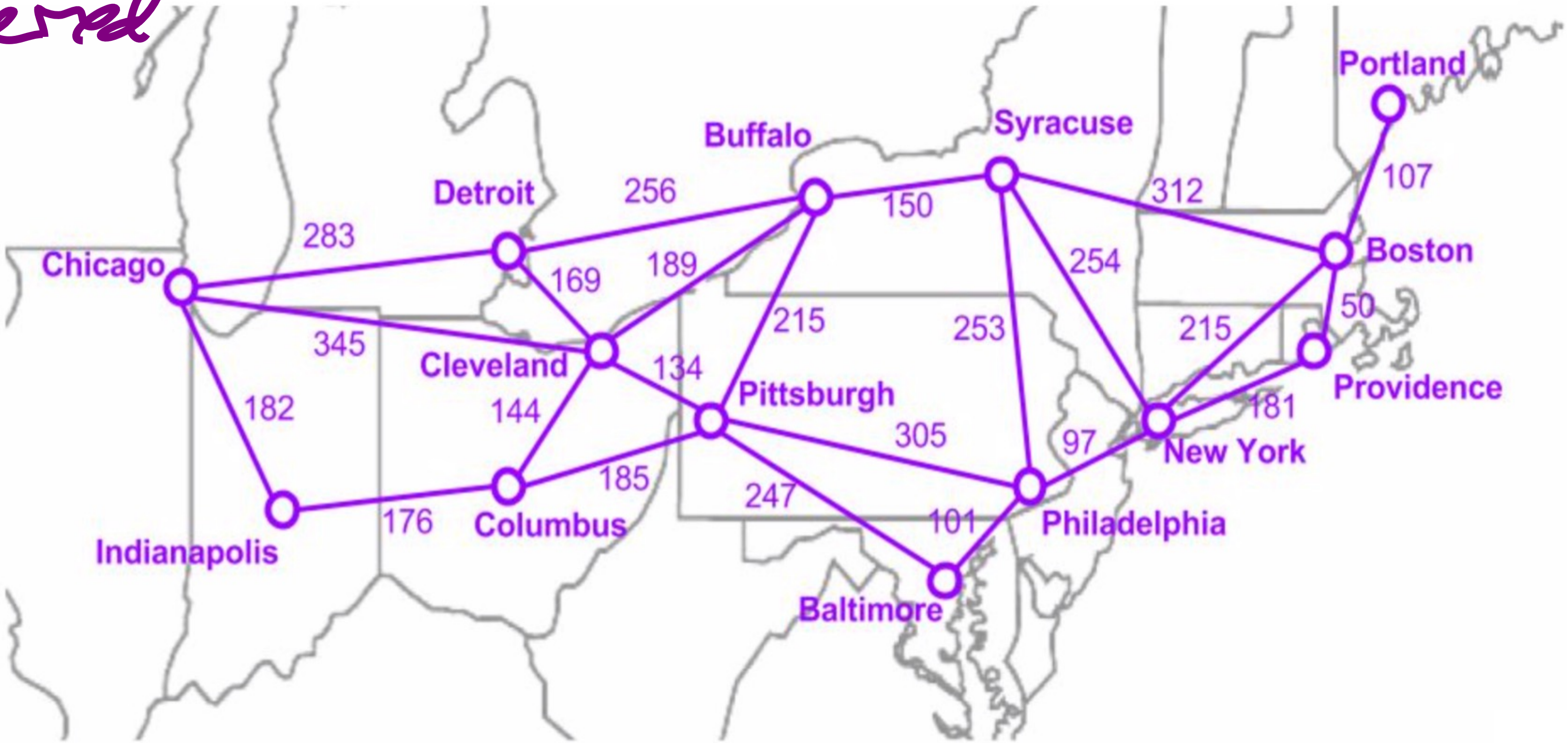
Breadth–first Search (BFS)

Example: Traveling in the US northeast

Question: Would changing the step cost function change our BFS result?
-Even if there is a major traffic jam in our route?

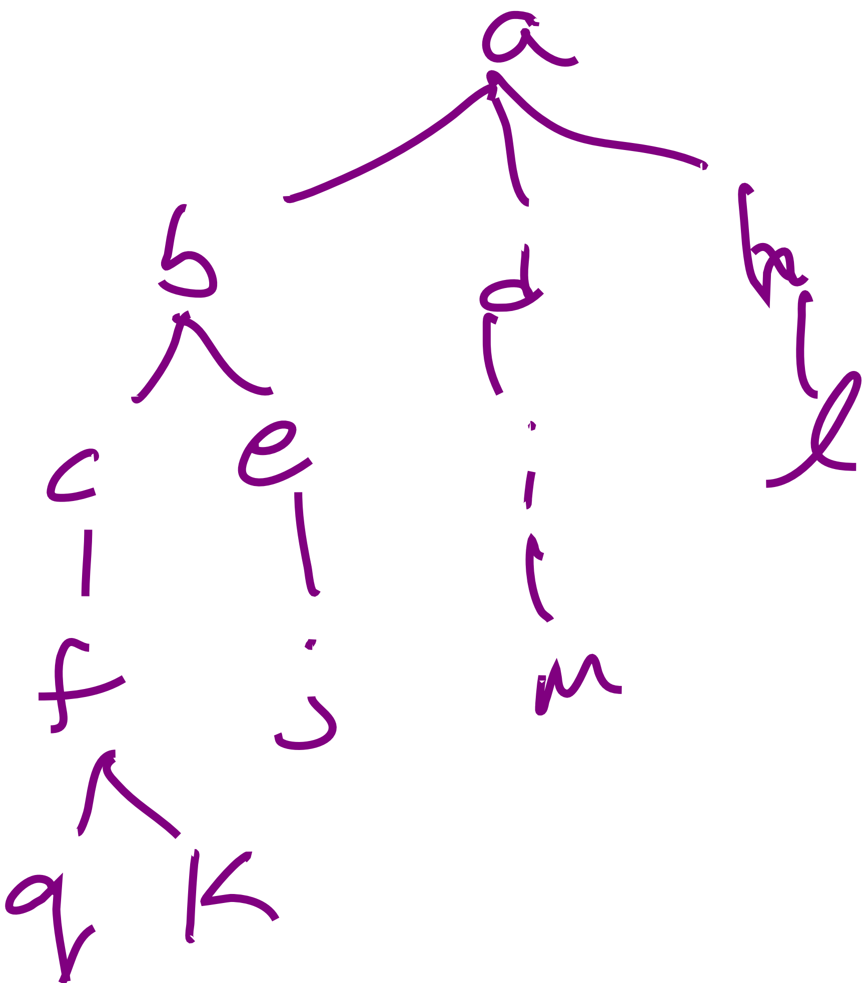
No. Step cost not considered

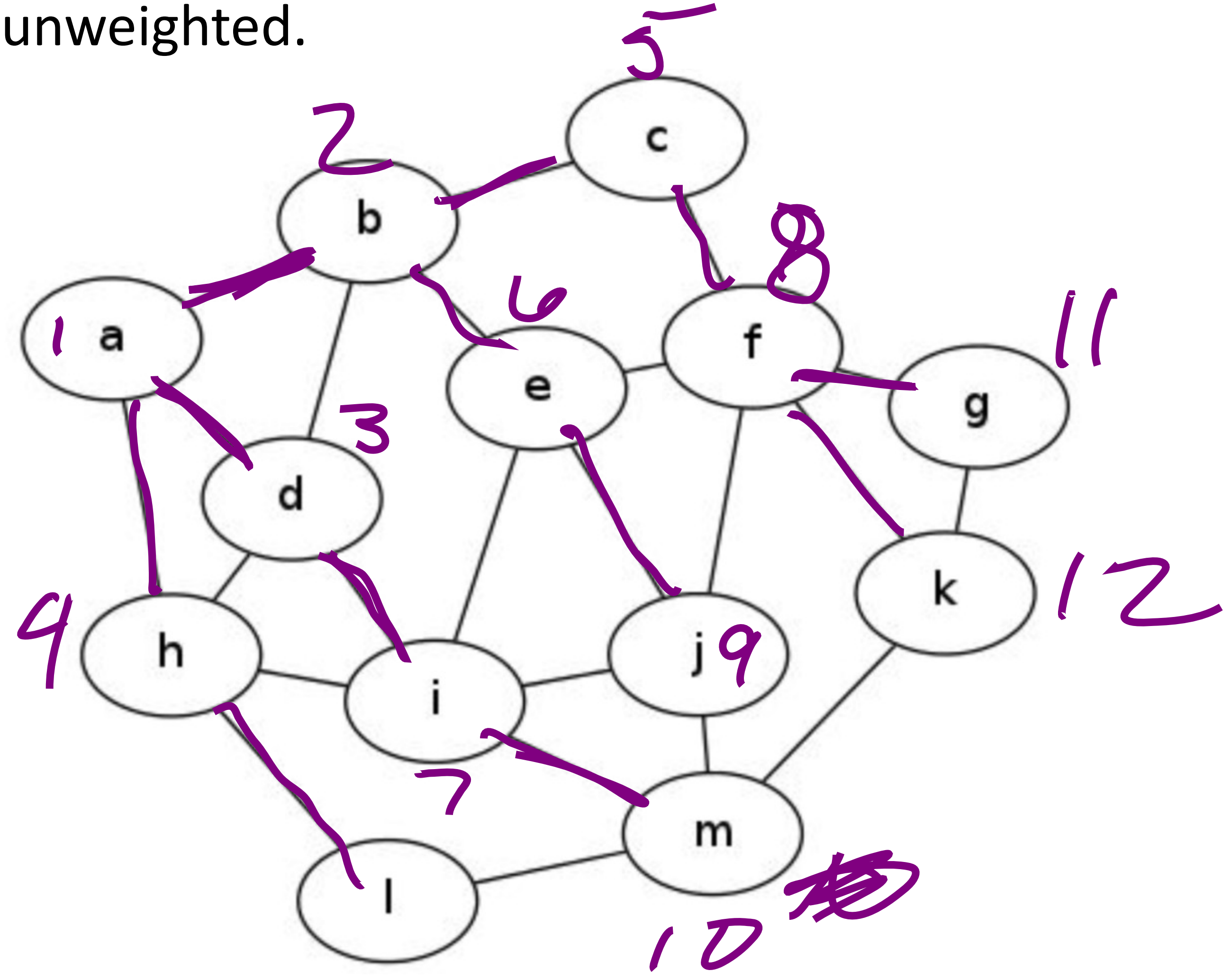
Step costs: estimated travel time (in minutes) along major highways



Breadth–first Search (BFS)

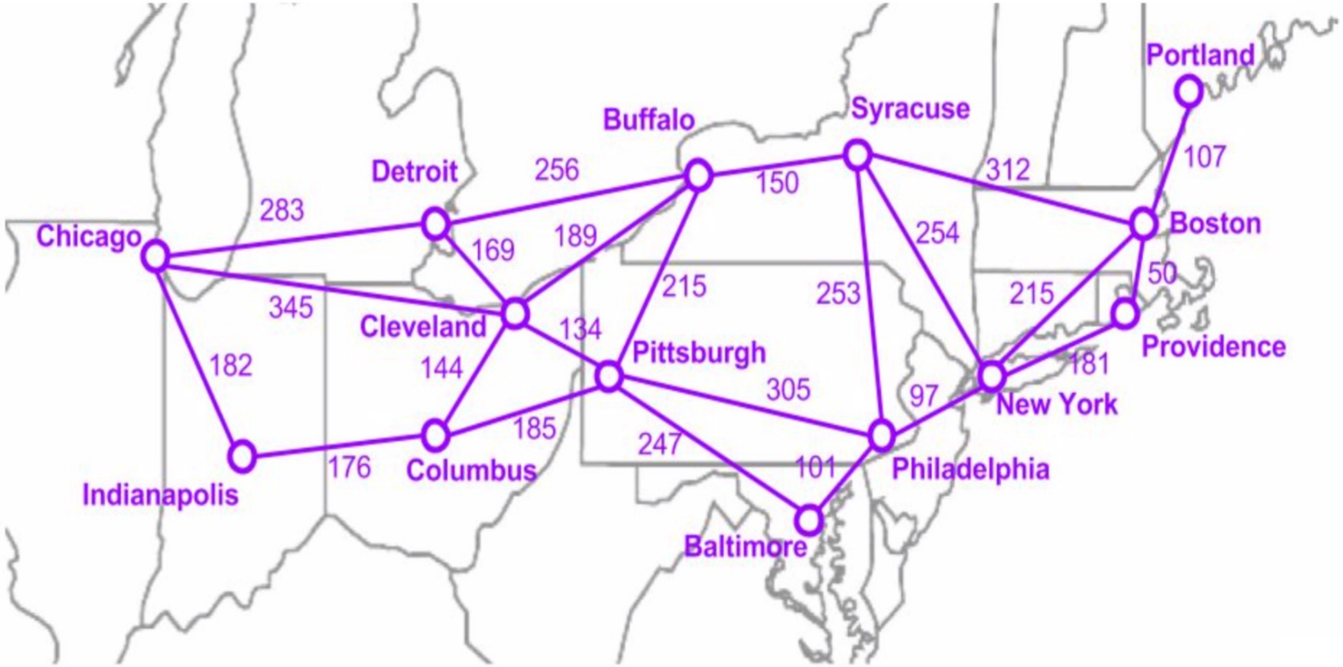
Example: Number the nodes in the search graph according to the order in which they would be expanded using BFS to find a path from a to k. Assume that nodes within a layer are expanded in alphabetical order. Edges are unweighted.





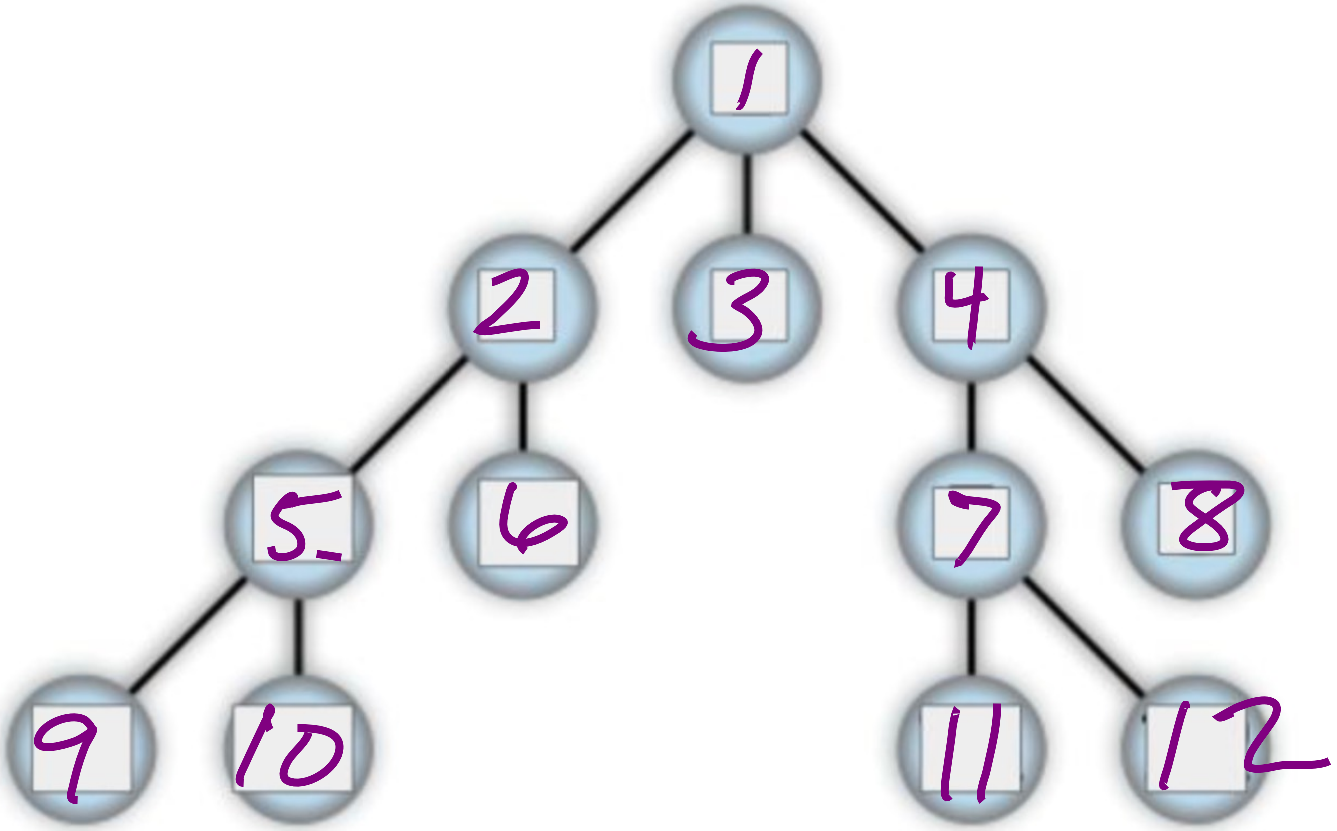
Breadth–first Search (BFS)

Example: Traveling in the northeast again. Sketch a search tree with Chicago as the initial state.



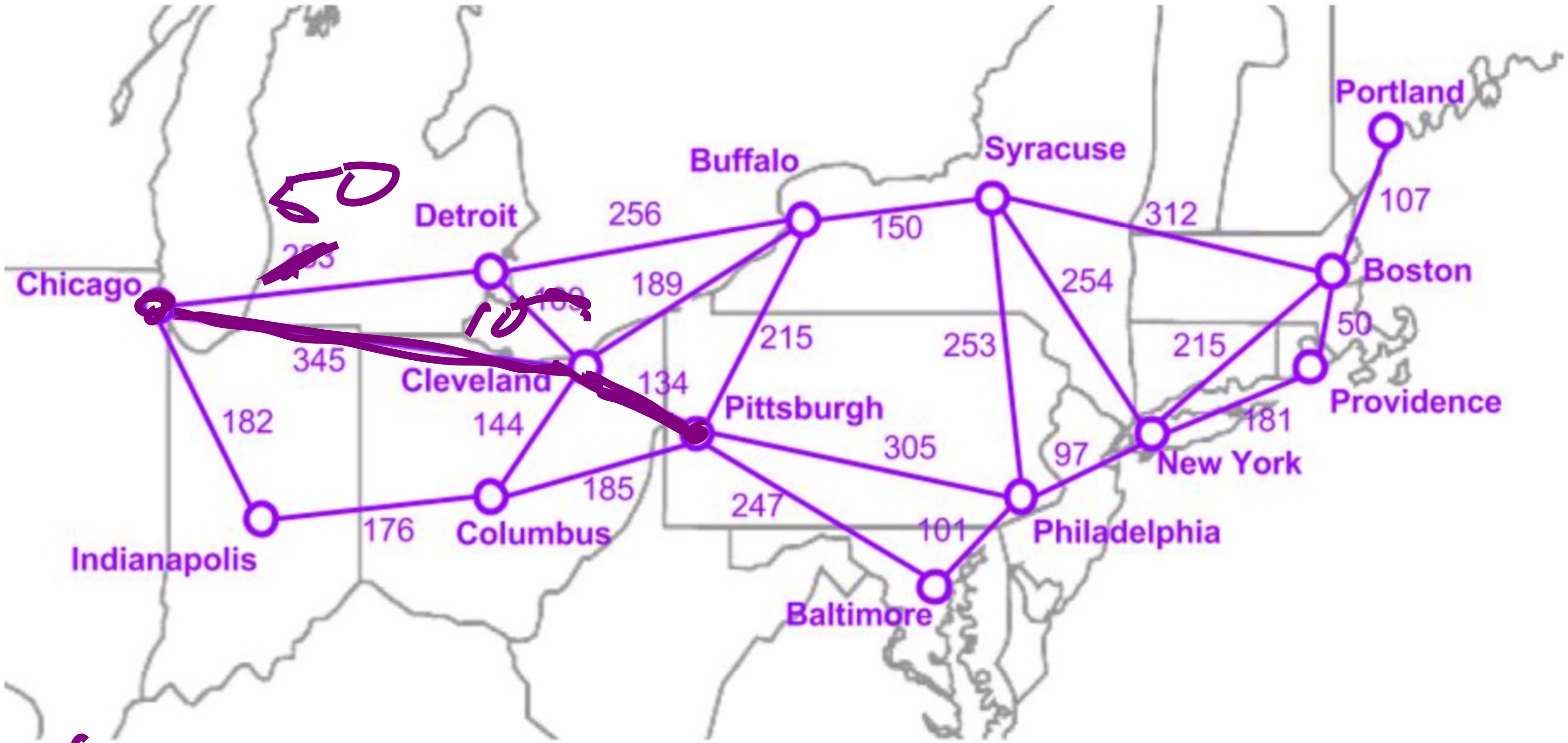
Breadth–first Search (BFS)

Example: Number the nodes in the search tree according to the order in which they would be expanded using BFS. Assume that the goal is not found, and nodes within a layer are expanded from left to right.



Breadth–first Search (BFS)

345+134
60+134



Complete? Yes

Optimal? In an unweighted graph, BFS is optimal. Not optimal with weighted graph

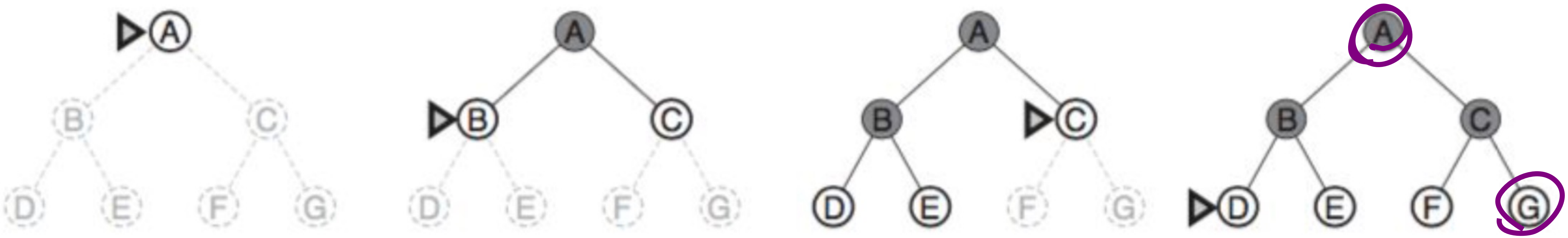
Breadth–first Search (BFS)

Time Complexity: Suppose that each layer generates b nodes (calling b the "branching factor") and the search problem has d total layers.

- layer 0 (root) generates $b^0 = 1$ node
- layer 1 generates $b^1 = b$ nodes
- layer 2 generates b^2 nodes

… and so on …

total: $\boldsymbol{1 + b + b^2 + b^3 + \ \ldots + b^d = \ \mathcal{O}\left(b^d\right)}$

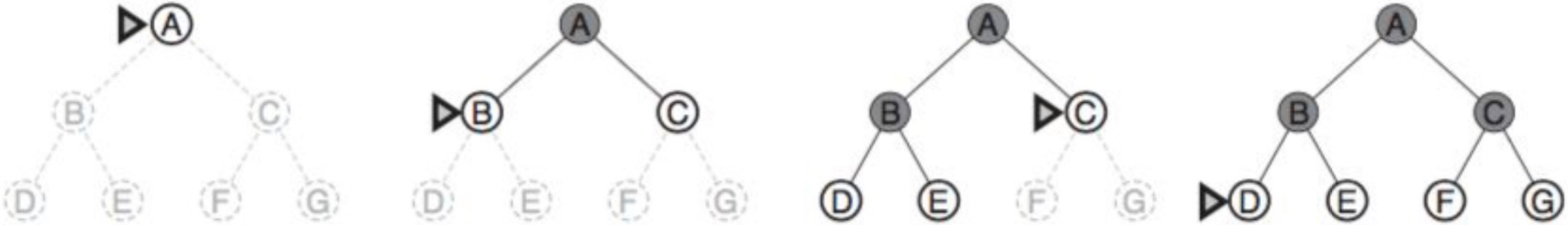


Breadth–first Search (BFS)

Space Complexity: assumes need to store every node in the explored set $= \mathcal{O}(b^{d-1})$

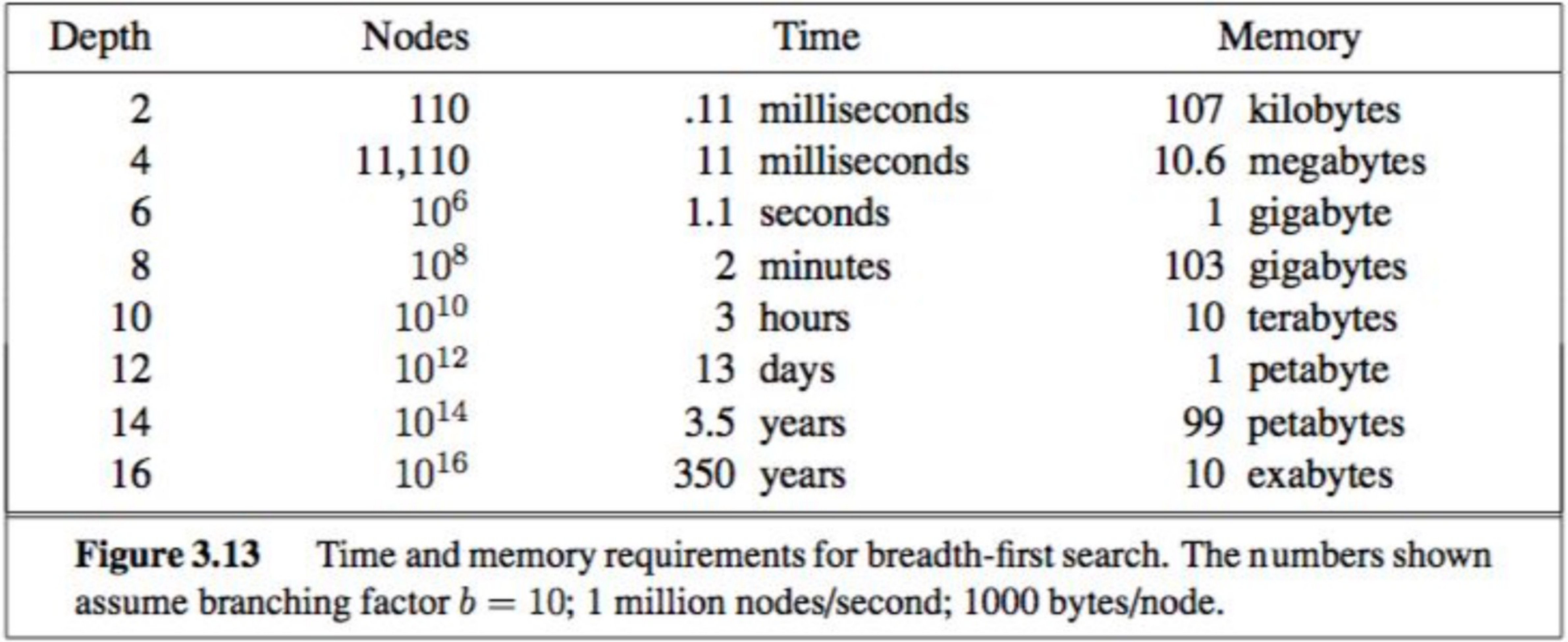
and every node on the frontier $= \mathcal{O}(b^{d})$

- ➢ $\mathcal{O}(b^{d})$



Breadth–first Search (BFS)

Memory requirements are a problem. So is Time if all nodes searched.



Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Figure 3.13 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 1 million nodes/second; 1000 bytes/node.

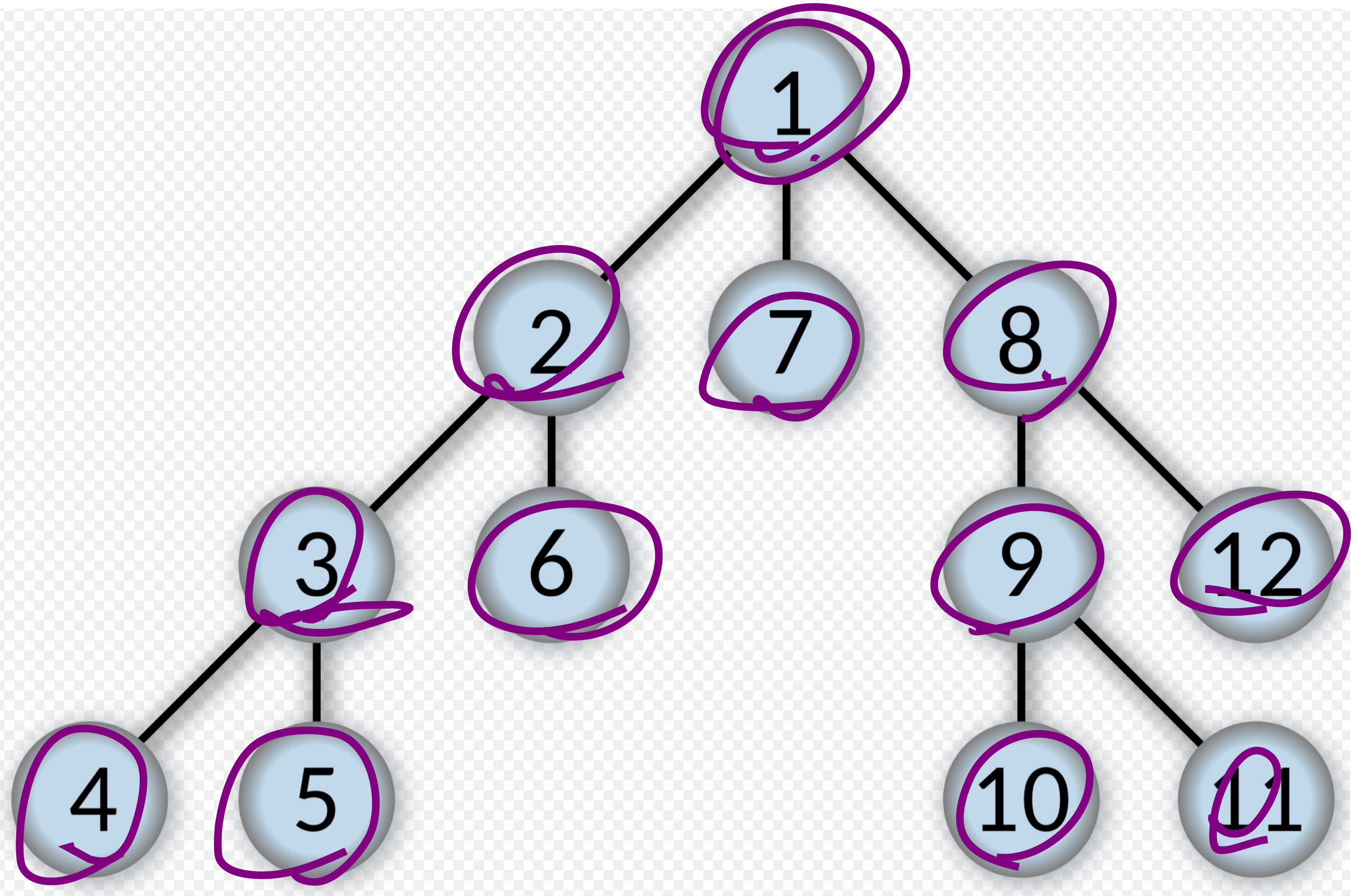
Next Time

Depth-first Search (DFS)

CSCI 3202: Intro to Artificial Intelligence
Lecture 7: Depth-First Search (DFS)

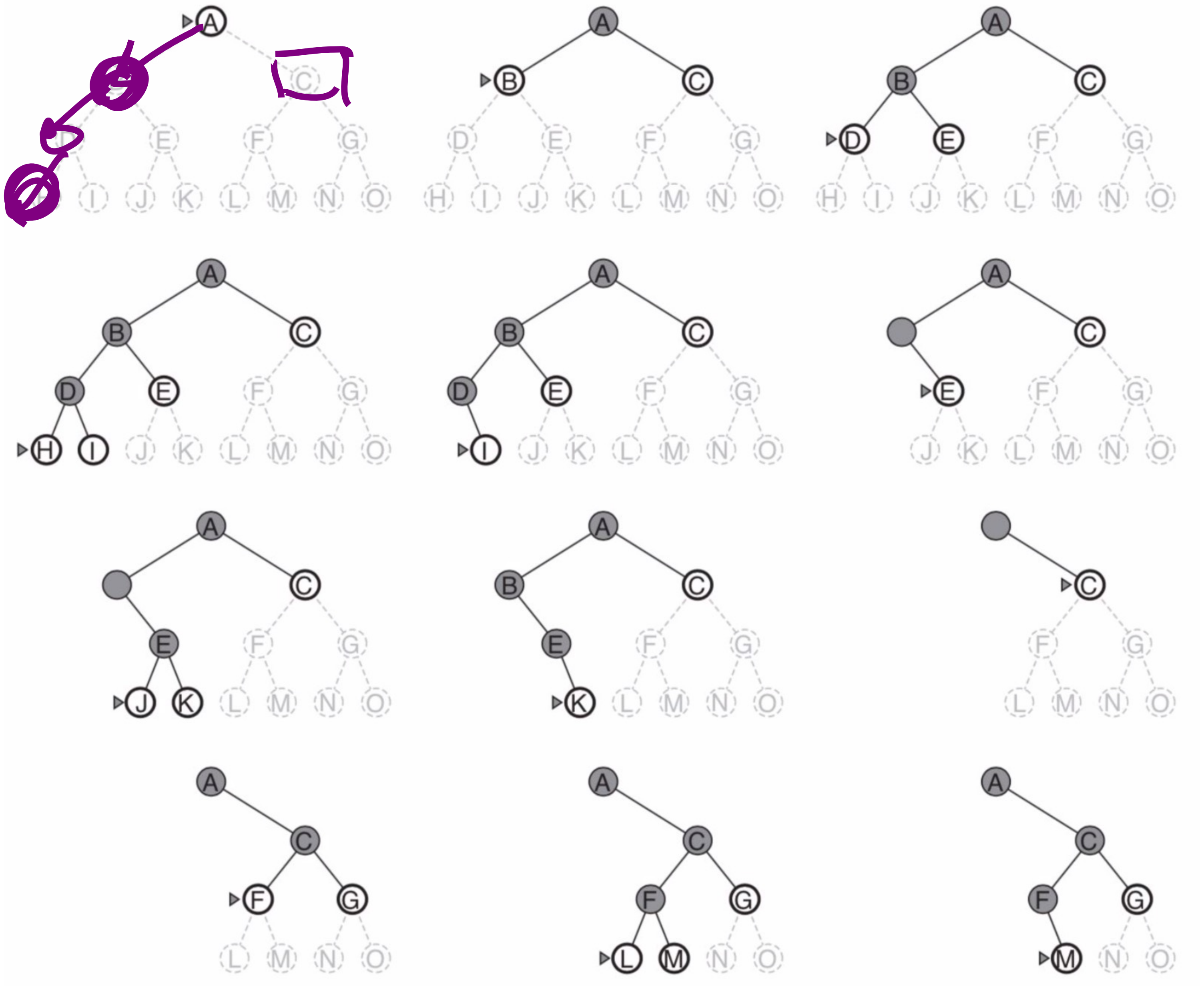
Rhonda Hoenigman

Department of Computer Science



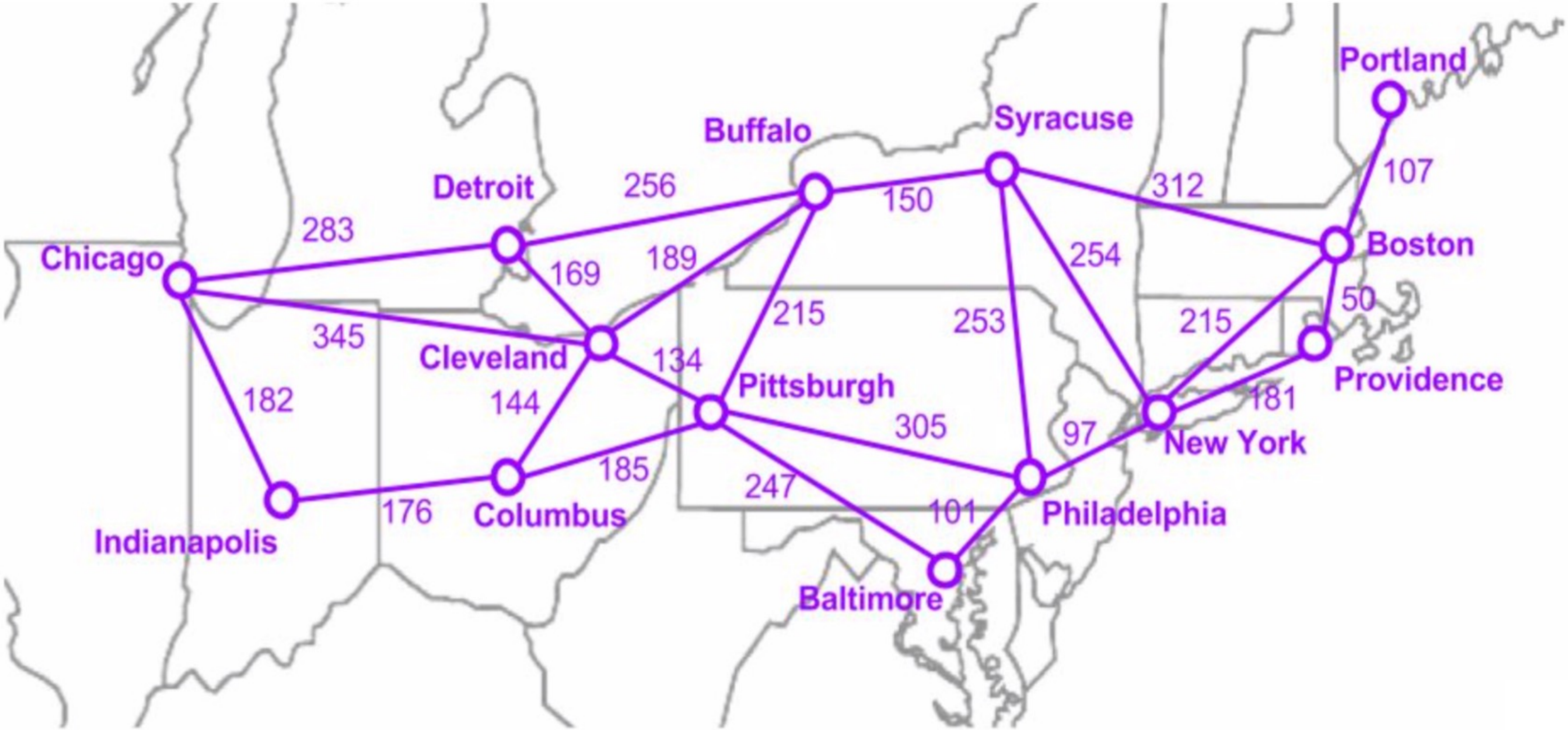
Depth–First Search（DFS）

- 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)

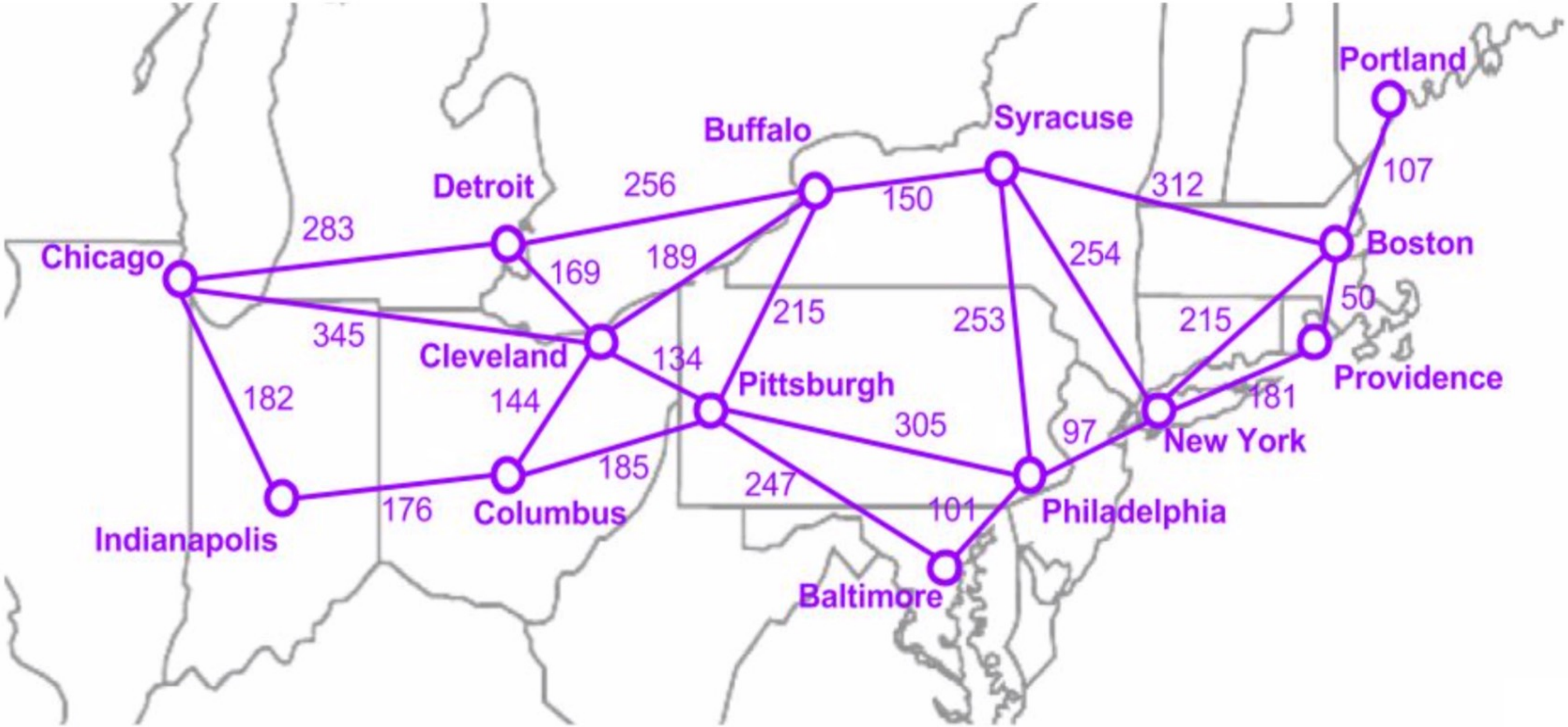
Example: Traveling in the US northeast
Step costs: miles between cities along major highways



Depth–First Search (DFS)

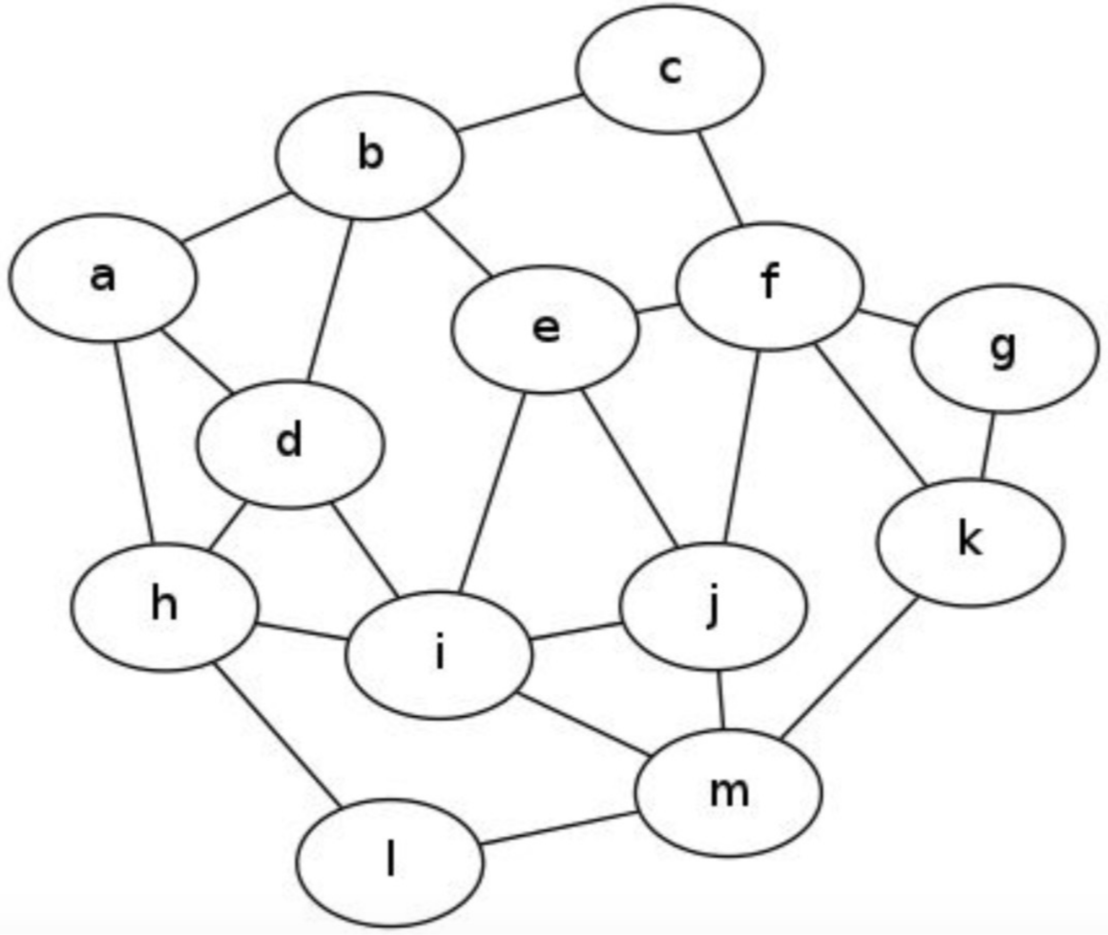
Example: Traveling in the US northeast. **Question**: Would changing the step cost function change our DFS result?

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?


c
b
a
f
e
g
d
k
h
i
j
m
l