Chapter 3

Lecture 2
Uninformed Search Algorithms

Jim Rehg

Review: Problem Solving Agents

- Problem Solving = get to a goal
- Goal = particular state(s) in the world
- Problem Formulation = init state, transition model, goal test, path cost
- Solution = sequence of actions to goal
- Optimal Solution = lowest cost sequence

Review: Tree Search

- Root node is init state
- Transition model tells next states
- Goal test? yes -> done, no->expand more

Search Strategy: Which leaf node to expand 1st?

Uninformed Search

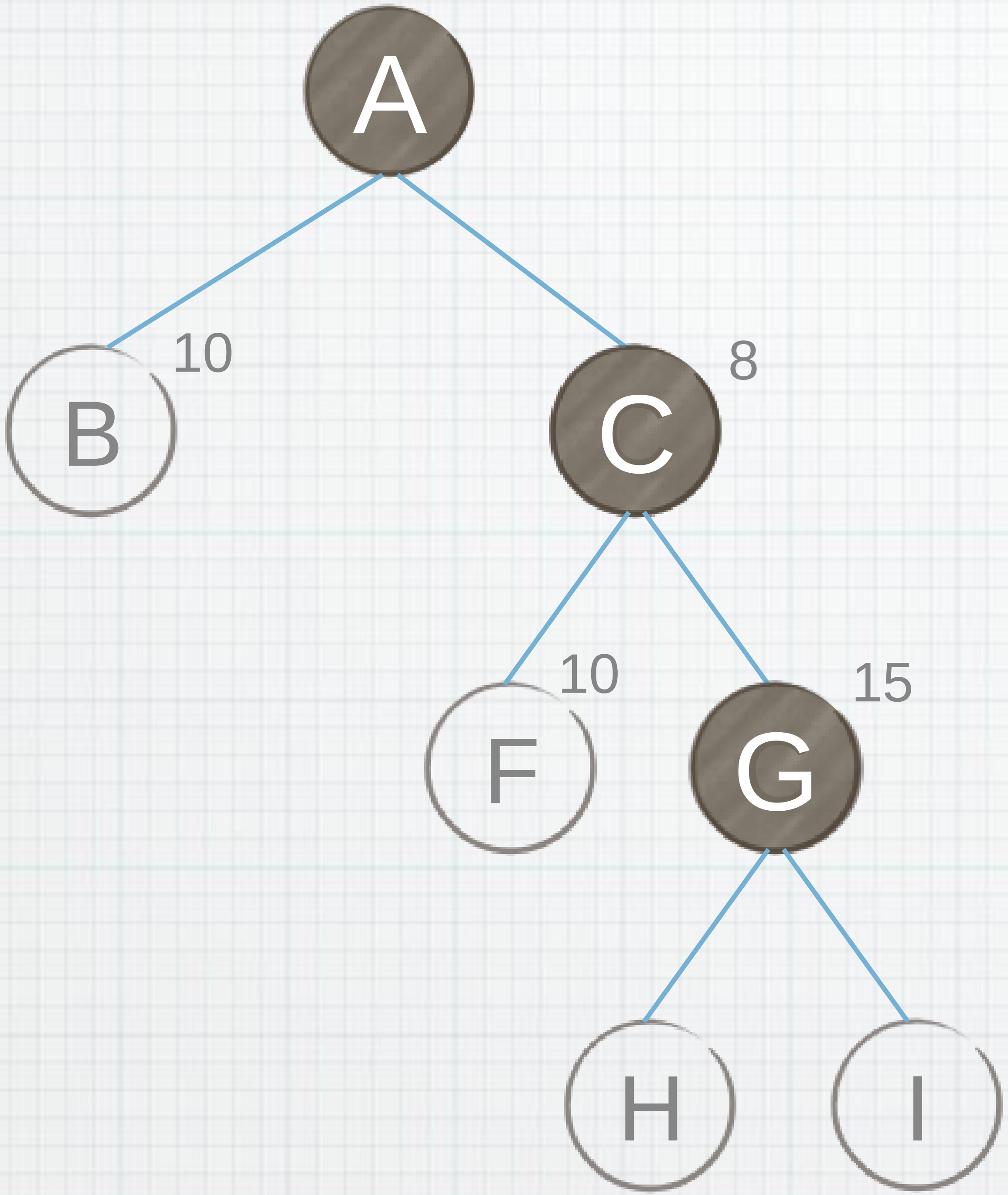
Use only the info in the problem definition

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative-deepening search

Identify the Type of Search

- Breadth-first search
- Uniform-cost search
- Depth-first search

What Kind of Search?



Frontier

~~A~~

~~C~~

~~G~~

I

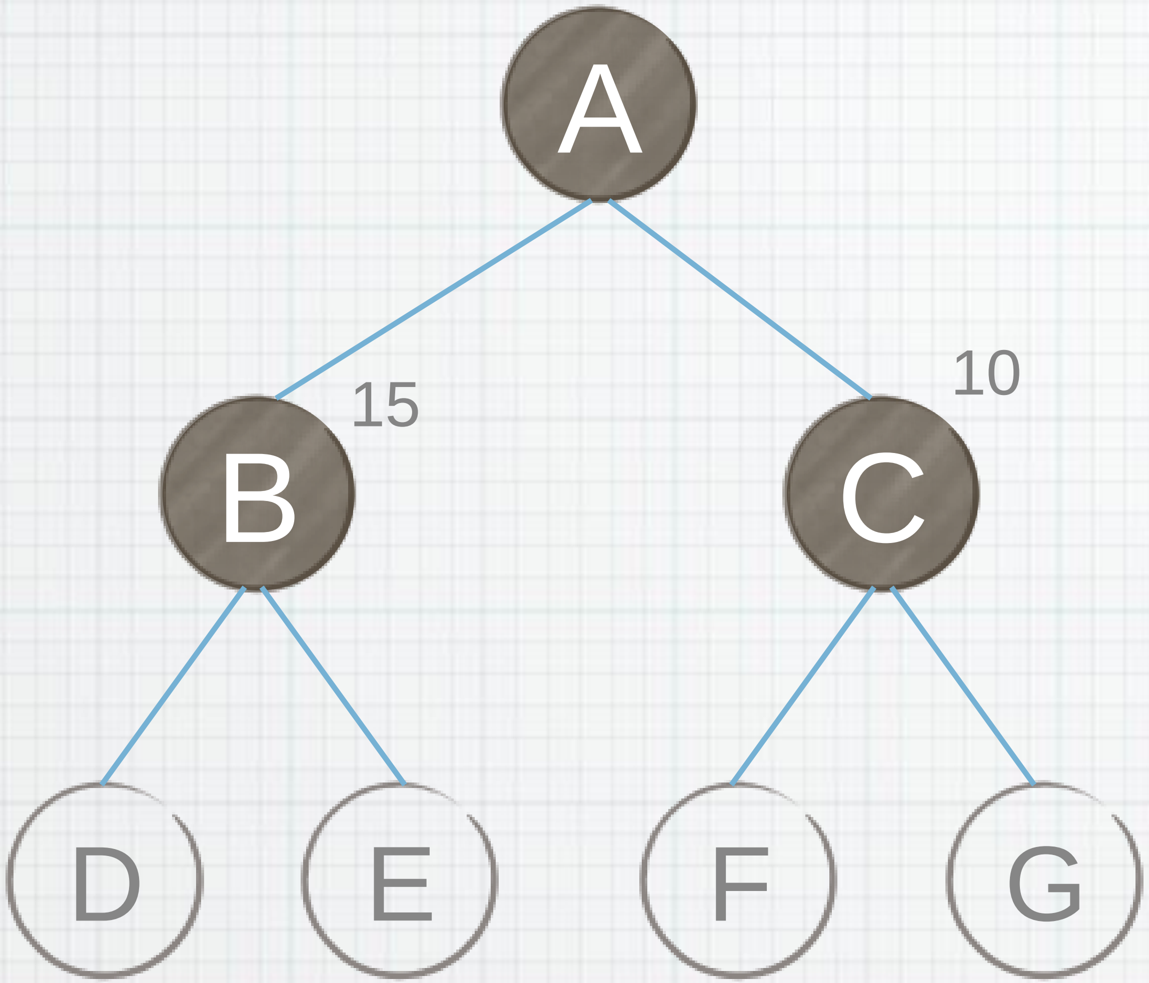
H

F

B

Depth First Search

What Kind of Search?



Frontier

~~A~~

~~B~~

~~C~~

D

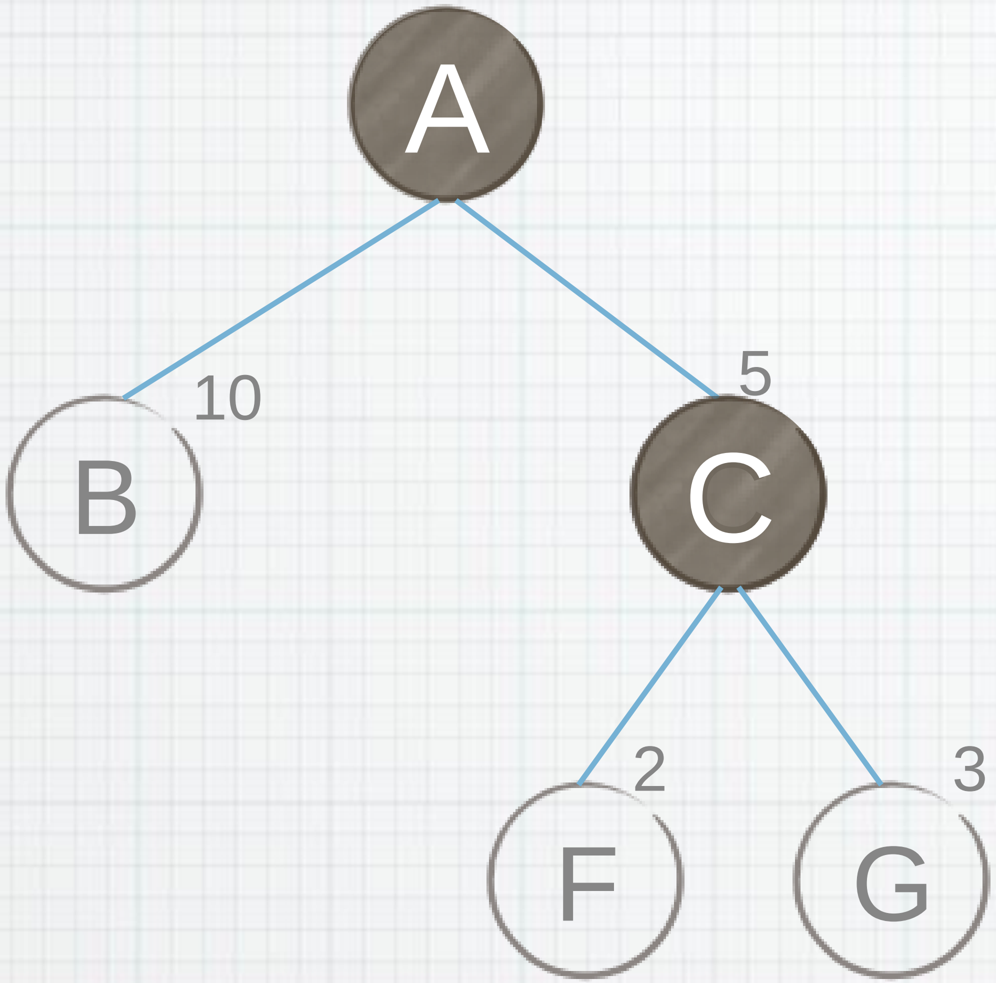
E

F

G

Breadth First Search

What Kind of Search?



Frontier

~~A~~

~~C~~

F

G

B

Uniform Cost Search

Uninformed Search

Use only the info in the problem definition

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative-deepening search

Depth-limited Search

- Implementation: DFS with limit l, on max depth to expand into frontier
- Complete: No.… limit may be < depth of goal
- Optimal: No
- Time: $O(b^l)$
- Space: $O(bl)$

Uninformed Search

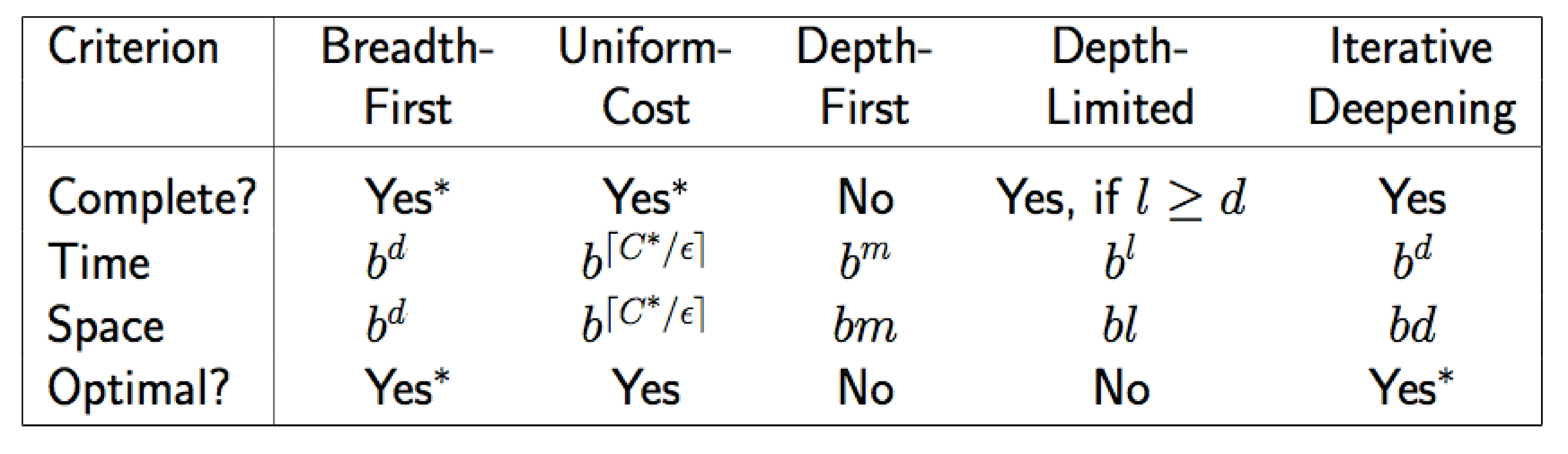
Use only the info in the problem definition

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative-deepening search

Iterative-Deepening Search

- Implementation: do DFS for l=1,2,3,4...
- Complete: Yes, will find the shallowest goal
- Optimal: No, shallowest not necessarily optimal
- Time: $O(b^d)$
- Space: O(bd)

Comparison of Algs

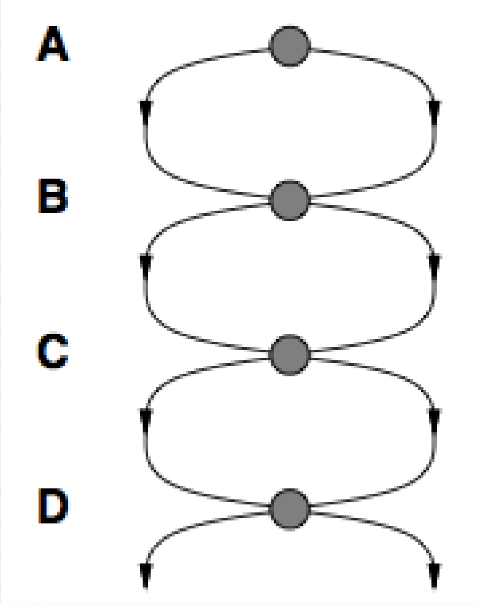


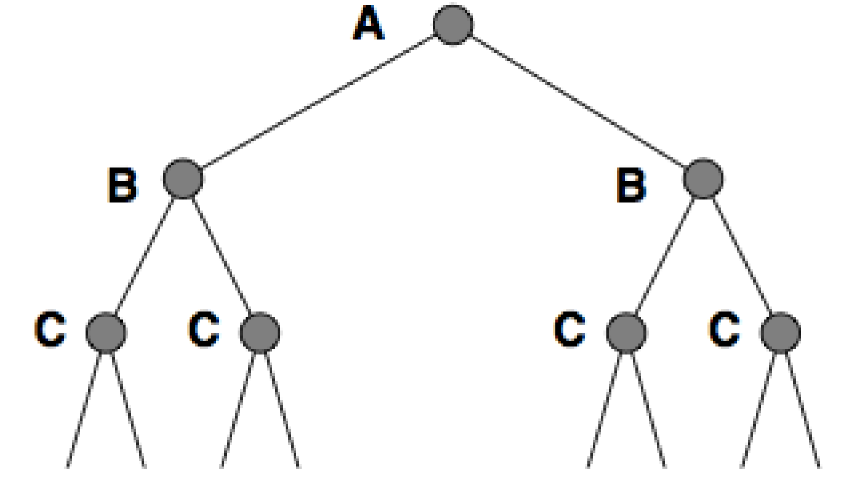
Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes*	Yes*	No	Yes, if $l \geq d$	Yes
Time	b^d	$b^{\lceil C^*/\epsilon \rceil}$	b^m	b^l	b^d
Space	b^d	$b^{\lceil C^*/\epsilon \rceil}$	bm	bl	bd
Optimal?	Yes*	Yes	No	No	Yes*

- BFS vs. DFS
  - memory
- DFS vs. D-limited vs. Iterative-D
  - time

Avoiding Repeat States

- Don’t want to waste time going somewhere twice
- When does this happen?
  - 2 paths to a state
  - Actions are reversible
- Linear problem, exponentially larger!





Graph Search

- Implementation: keep closed list (where you’ve been) only put in queue if not on Closed or Frontier

Frontier	Closed
~~A~~	A
~~C~~	C
~~G~~	G
I	
H	
F	
B	

Graph Search

- Optimal: Not necessarily, discards newly discovered paths, uniform-cost version fixes this
- Time: better since no repeats
- Space: more since storing 2nd list

Summary: Uninformed Search

- Uses only information in problem def
- Variety of uninformed search strategies
- Iterative deepening uses only linear space and not much more time than other algorithms
- Graph search can be exponentially more efficient than tree search

Informed Search

- What if you know more...
  - Designer knows something about the problem to help the agent
  - Domain knowledge
- Use this to expand the BEST node first

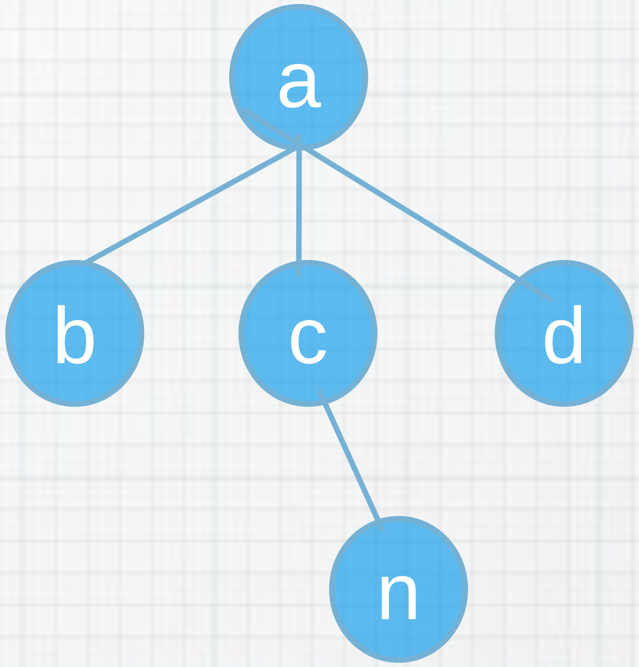
Evaluation Function

- f(n) = desirability of node n
- Best-First Search:
  Tree search + Evaluation Function f(n)

Search Strategy: How to define eval function

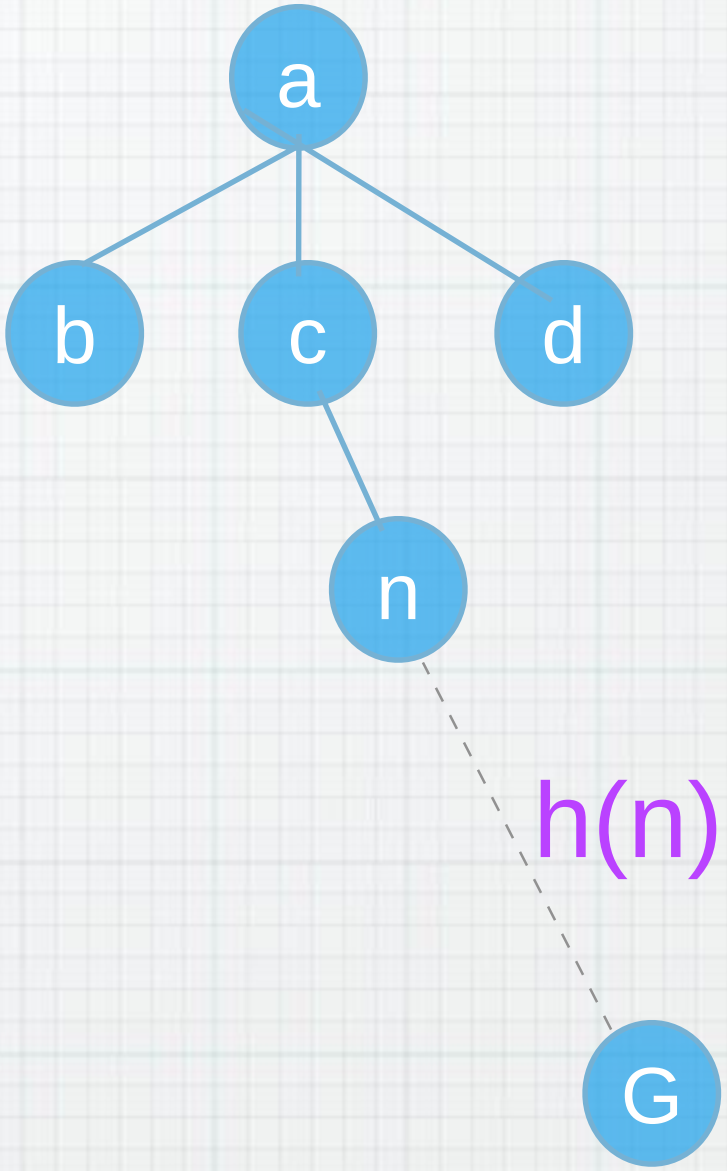
Heuristic Function

- Key to BFS algorithms is the heuristic h(n)
  - estimated cheapest path, n to goal
  - estimated future path cost from n



Heuristic Function

- Key to BFS algorithms is the heuristic h(n)
  - estimated cheapest path, n to goal
  - estimated future path cost from n

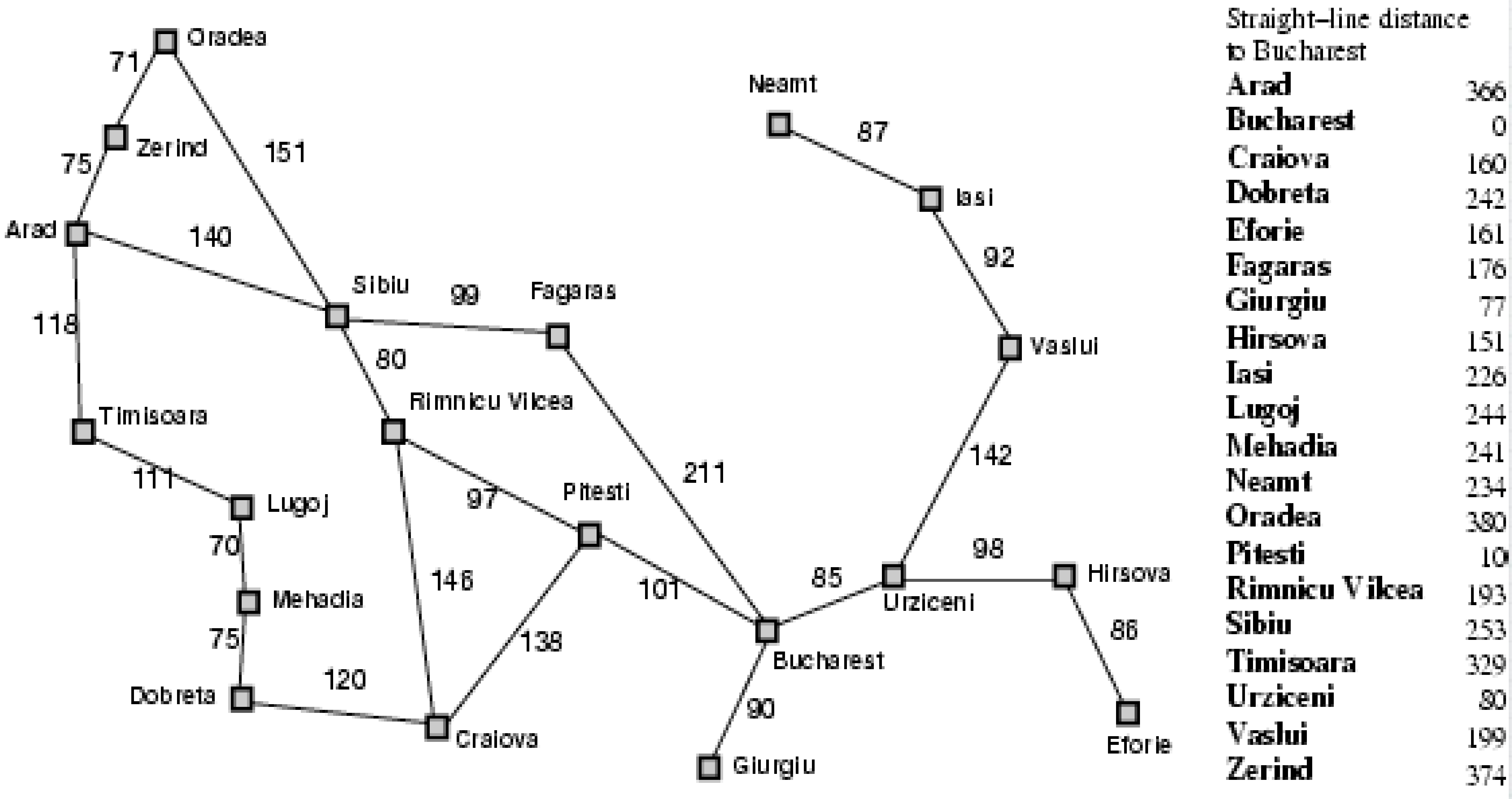


Greedy Best-First

- f(n) = h(n): expand node that appears to be closest from here
- Example — Route planning — a common heuristic is straight line distance to goal

Greedy Best-First

- step costs in km; h(n) = straight line distance

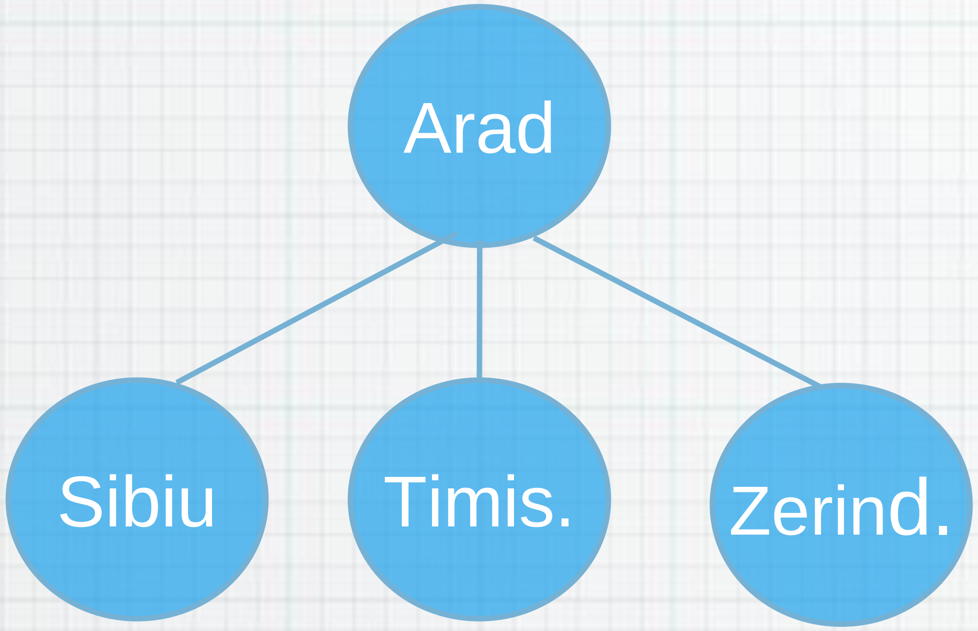


Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best-First

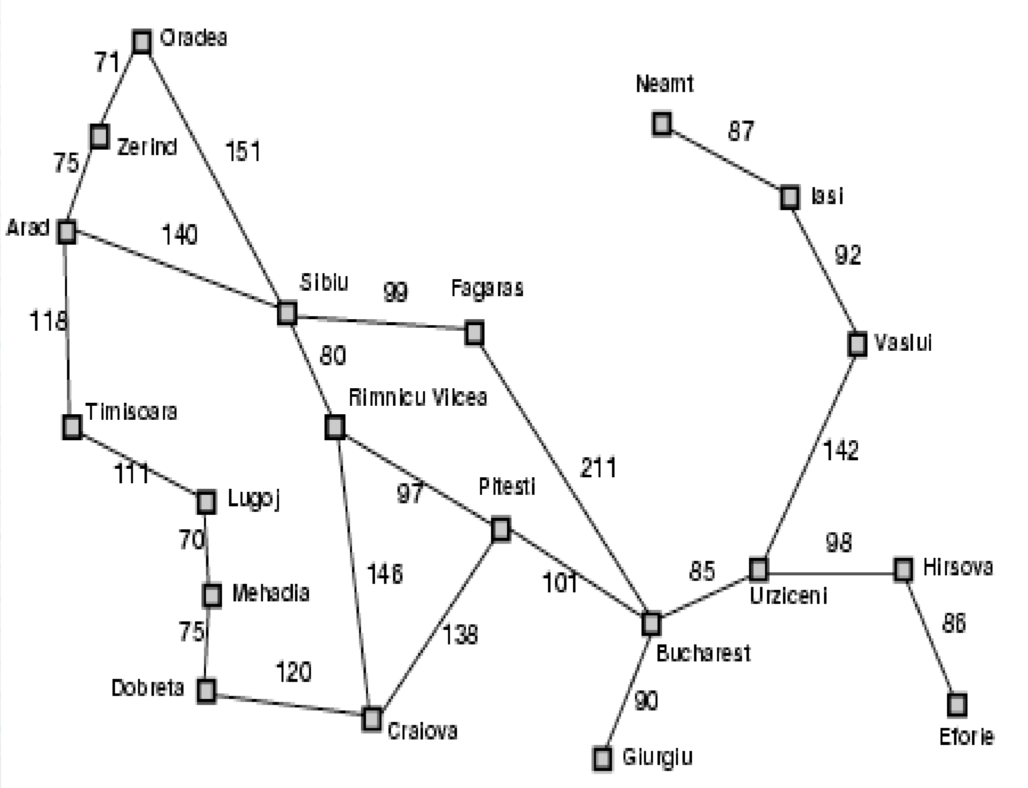
goal = Bucharest

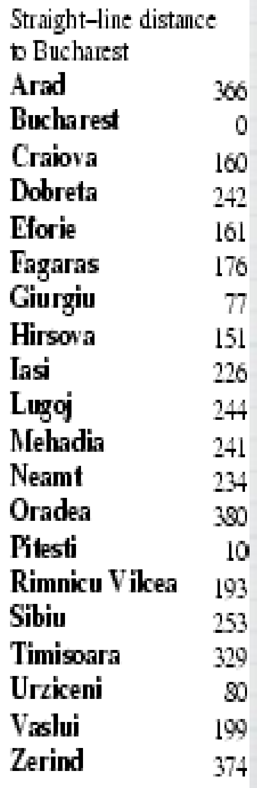


h(Sibiu) = 253

h(Timis.) = 329

h(Zerind) = 374

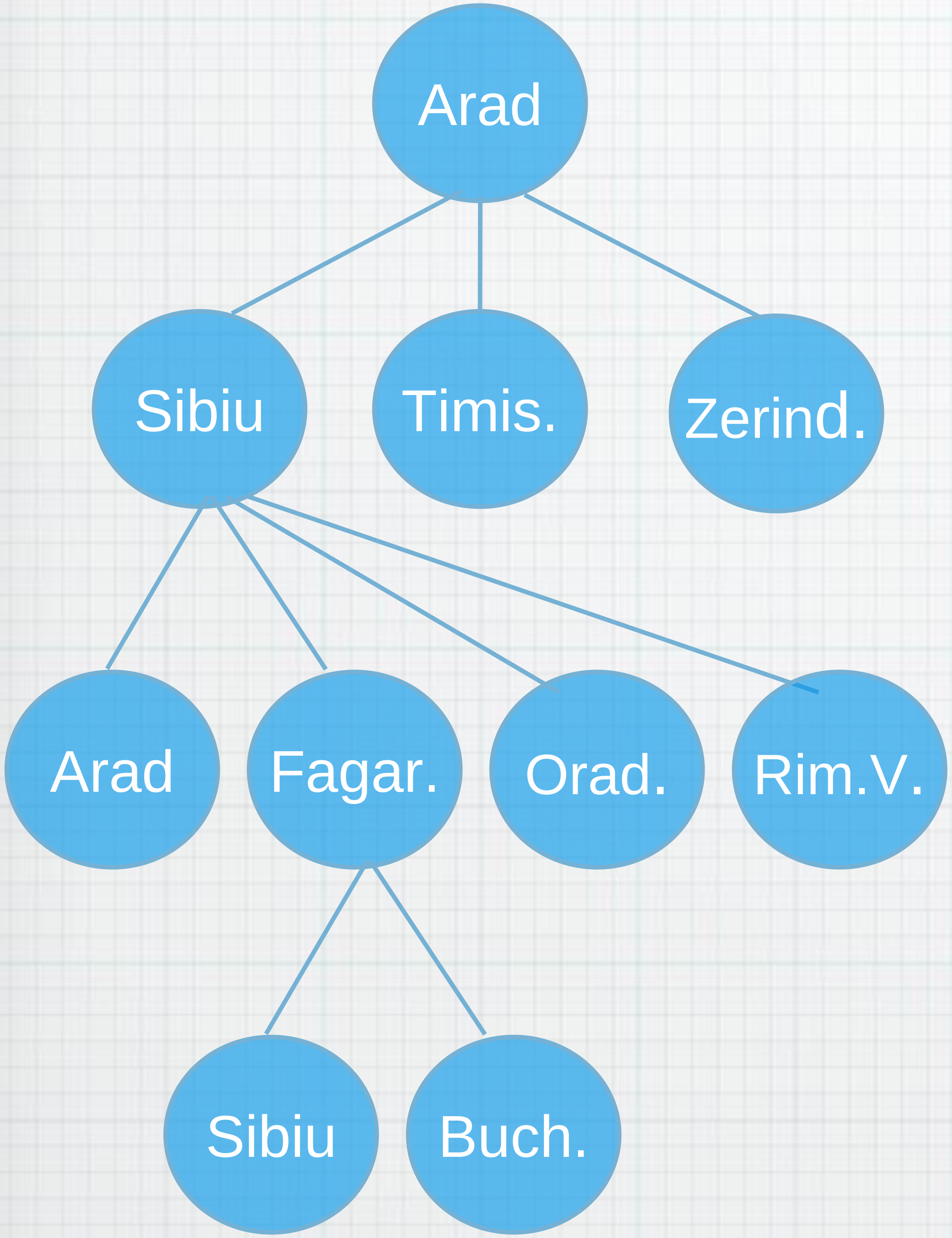




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best-First

goal = Bucharest

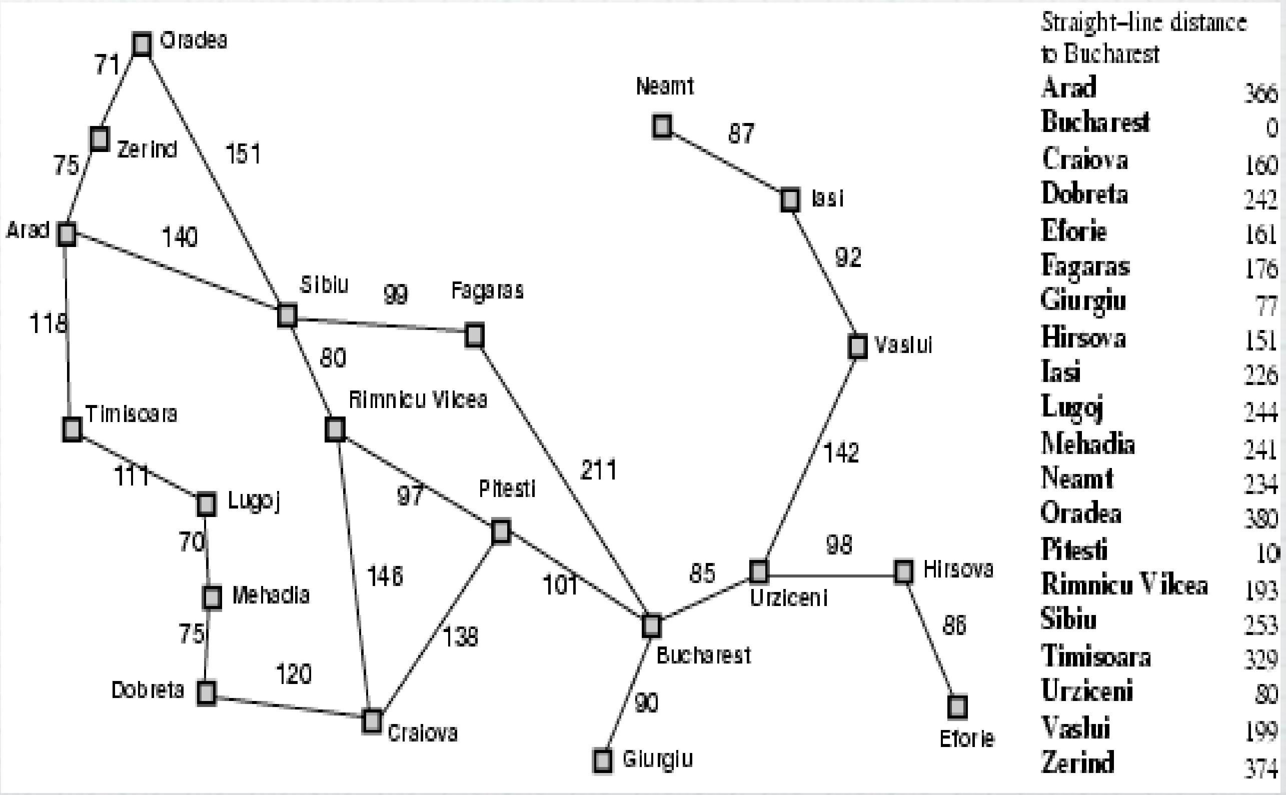


h(Arad) = 366

h(Fagaras) = 176

h(Oradea) = 380

h(Rim.Vic.) = 193

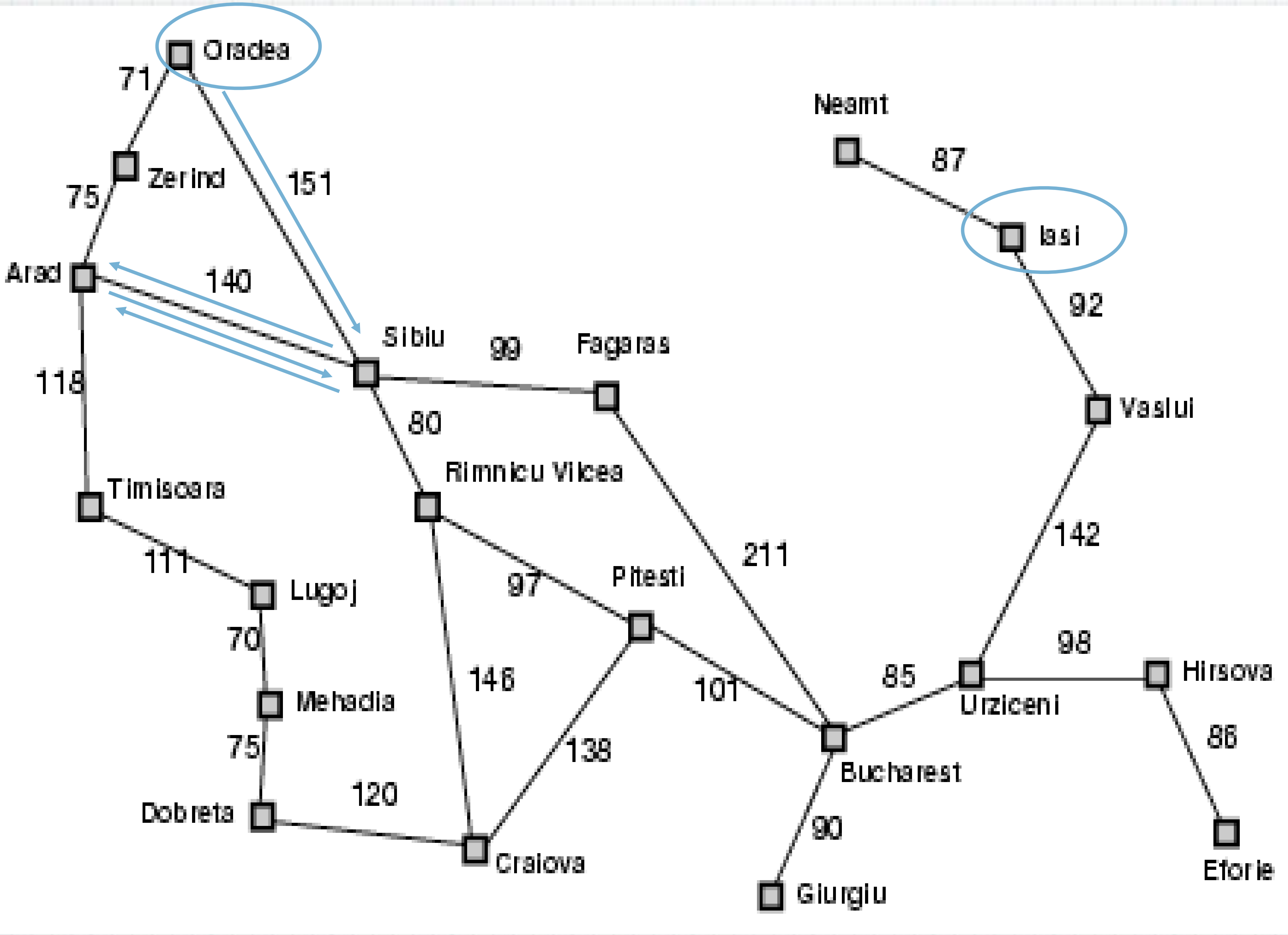


Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best-First

- Complete?
    - No, can get stuck in loops



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vikea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best-First

- Complete?
    - No, can get stuck in loops
    - Yes, if graph-search version
- Optimal?
    - No, only pays attention to future not how costly it was to get here