Introduction to AI

Lecture 5: Uninformed search strategies

Mona Taghavi

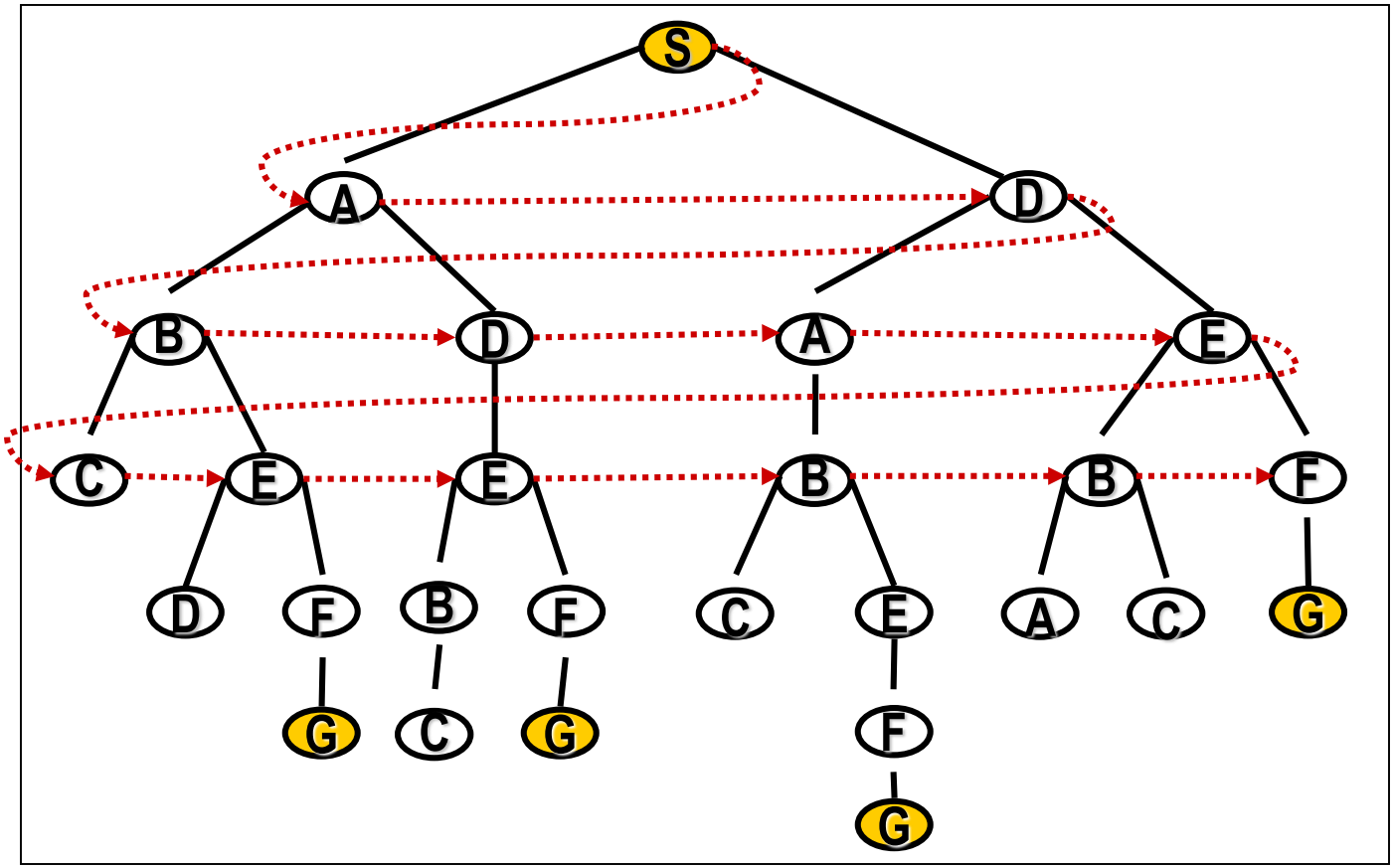
LaSalle College
Montréal

Uninformed search strategies

Use only information available in the problem formulation

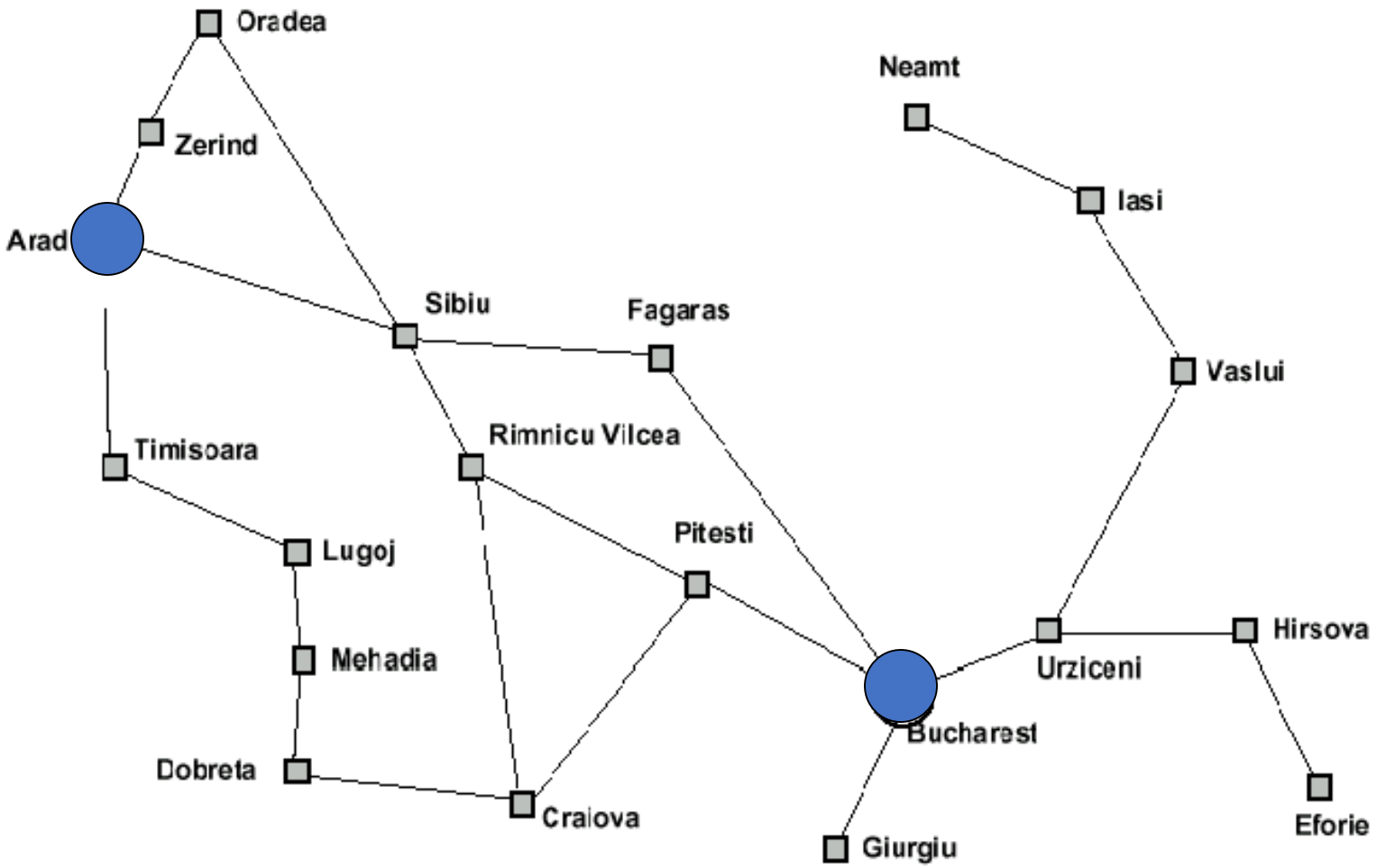
- Breadth-first
- Depth-first
- Uniform-cost
- Depth-limited (bounded-depth)
- Iterative deepening

Breadth-first search

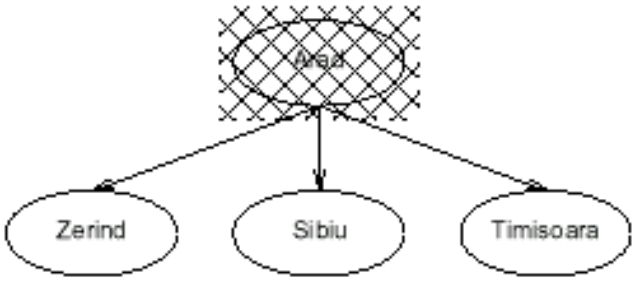


Move downwards, level by level, until goal is reached.

Example: Traveling from Arad To Bucharest



Breadth-first search

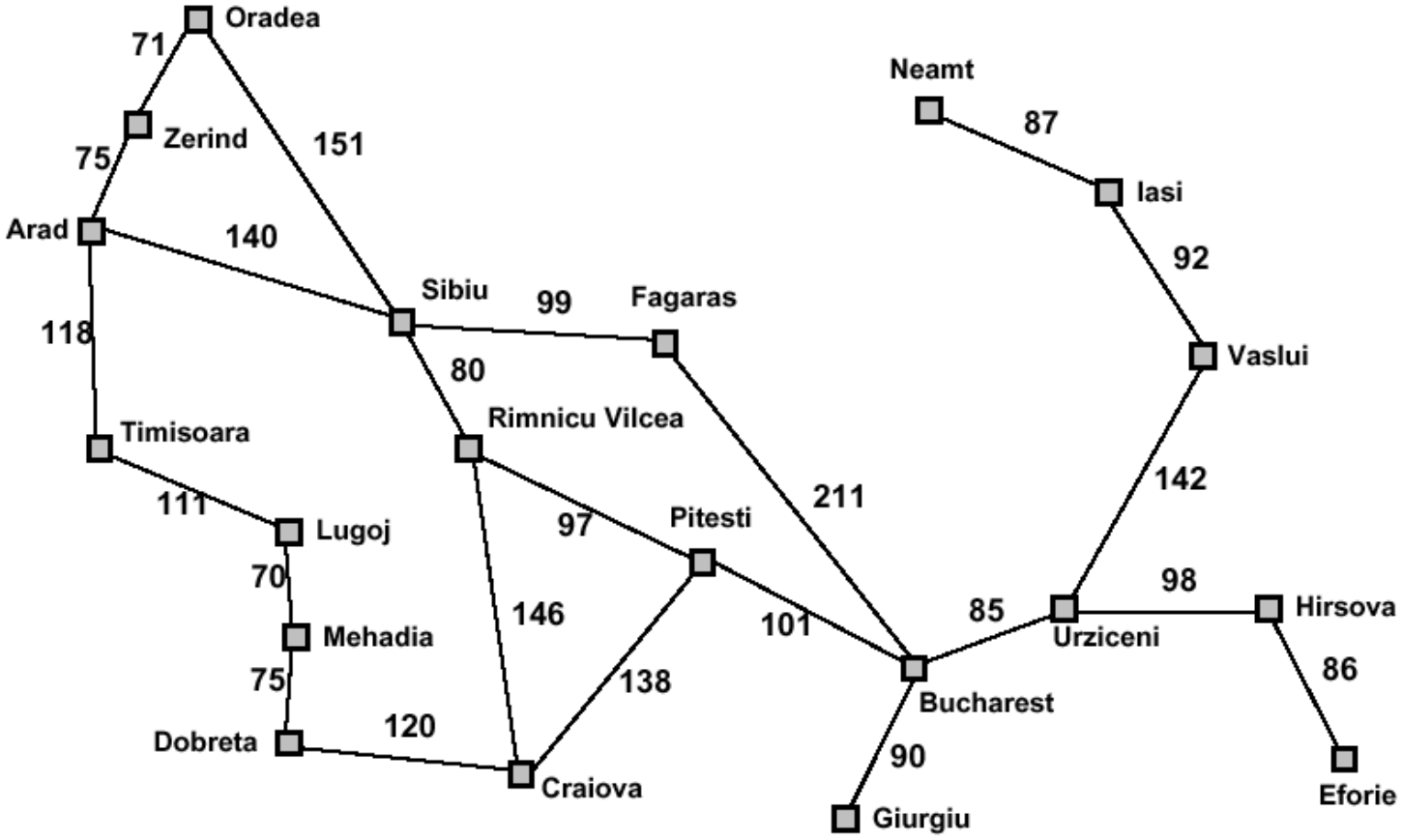


Uniform-cost search

A refinement of the breadth-first strategy:

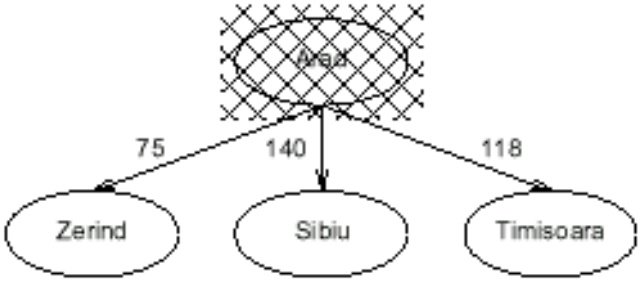
Breadth-first = uniform-cost with path cost = node depth

Romania with step costs in km

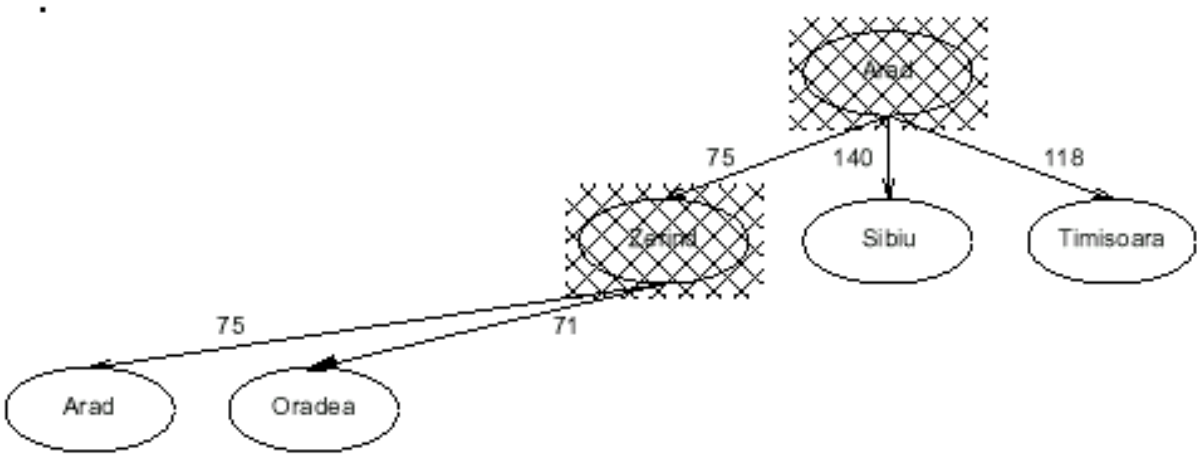


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

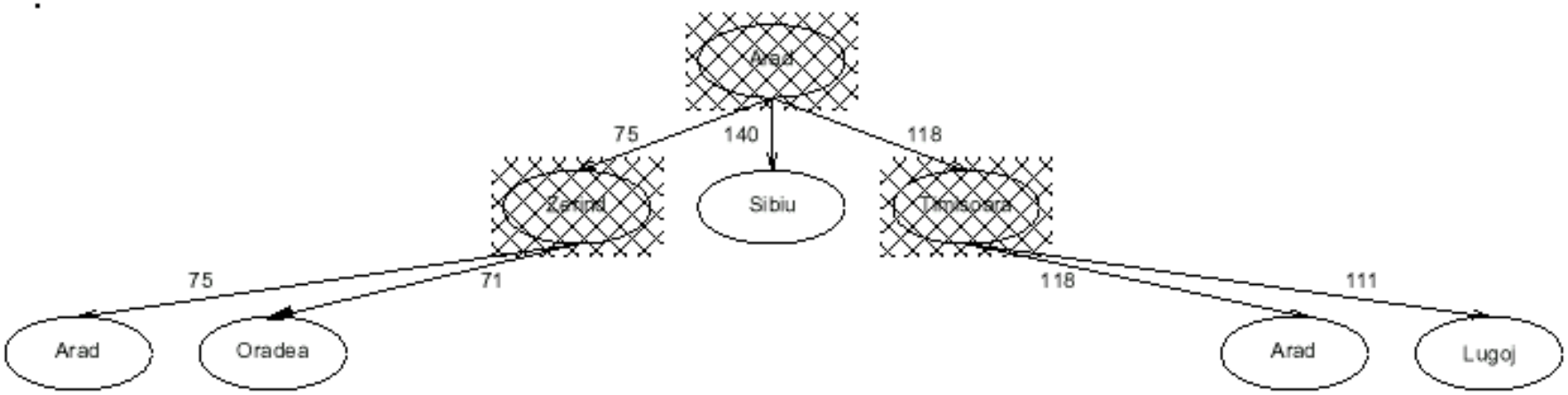
Uniform-cost search



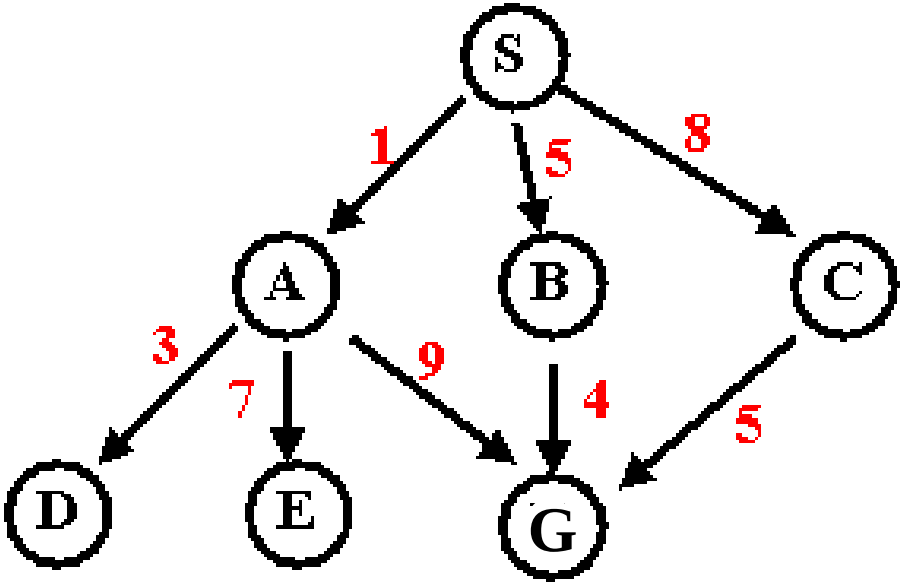
Uniform-cost search



Uniform-cost search



Example


Example Illustrating Uninformed Search Strategies
S
A
B
C
D
E
G
1
5
8
3
7
9
4
5

Breadth-First Search Solution

Breadth-First Search

return GENERAL-SEARCH(problem, ENQUEUE-AT-END)

exp. node	nodes list
	{ S }
S	{ A B C }
A	{ B C D E G }
B	{ C D E G G' }
C	{ D E G G' G" }
D	{ E G G' G" }
E	{ G G' G" }
G	{ G' G" }

Solution path found is S A G <-- this G also has cost 10

Number of nodes expanded (including goal node) = 7

Uniform-Cost Search Solution

Uniform-Cost Search

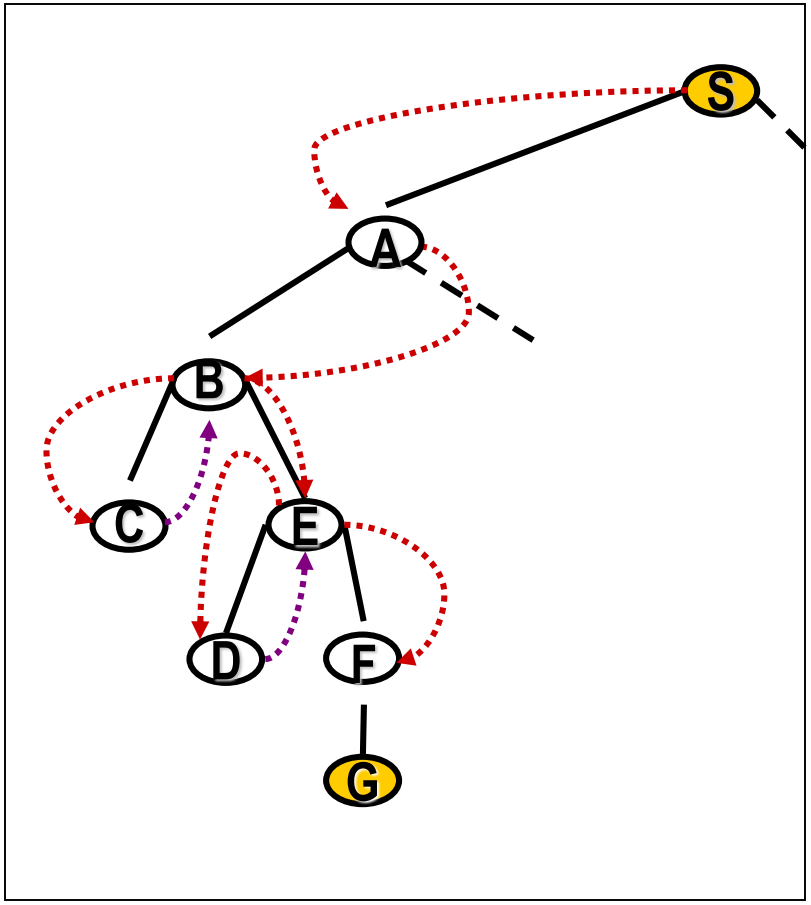
GENERAL-SEARCH(problem, ENQUEUE-BY-PATH-COST)

exp. node	nodes list
	{ S }
S	{ A(1) B(5) C(8) }
A	{ D(4) B(5) C(8) E(8) G(10) } (NB, we don't return G)
D	{ B(5) C(8) E(8) G(10) }
B	{ C(8) E(8) G(9) G(10) }
C	{ E(8) G(9) G(10) G(13) }
E	{ G(9) G(10) G(13) }
G	{ }

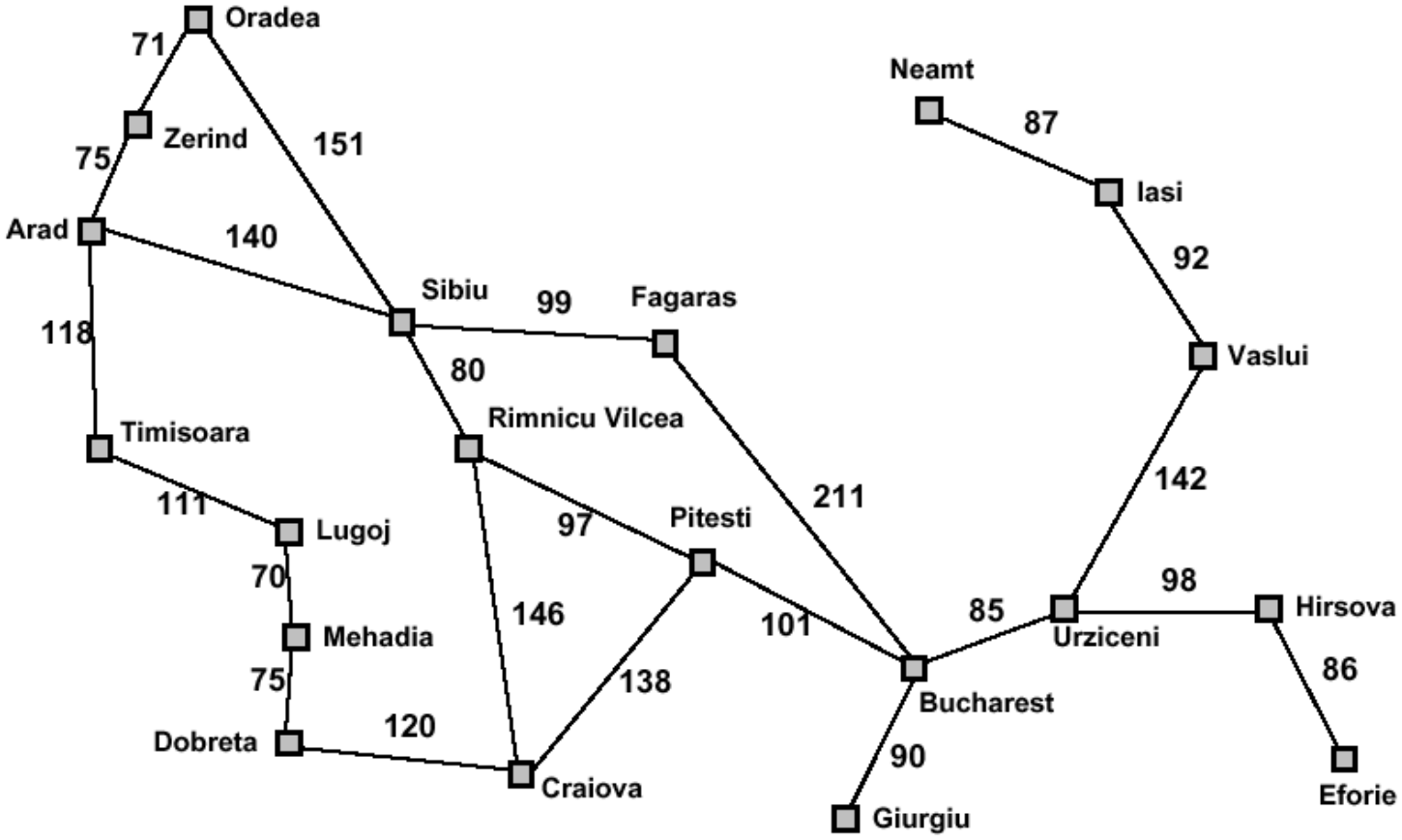
Solution path found is S B G <-- this G has cost 9, not 10

Number of nodes expanded (including goal node) = 7

Depth First Search

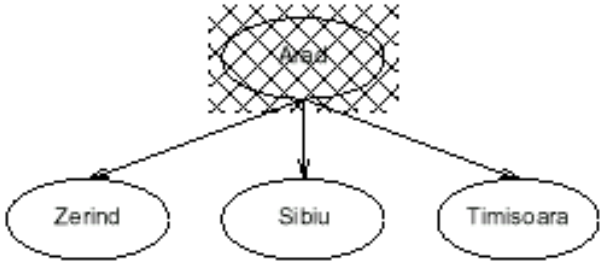


Romania with step costs in km

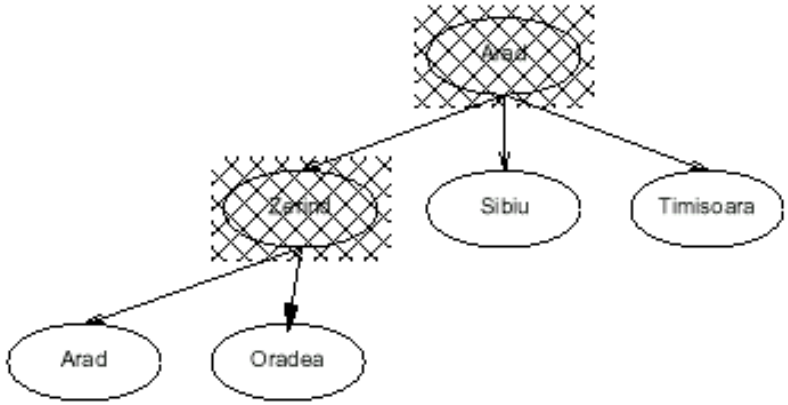


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

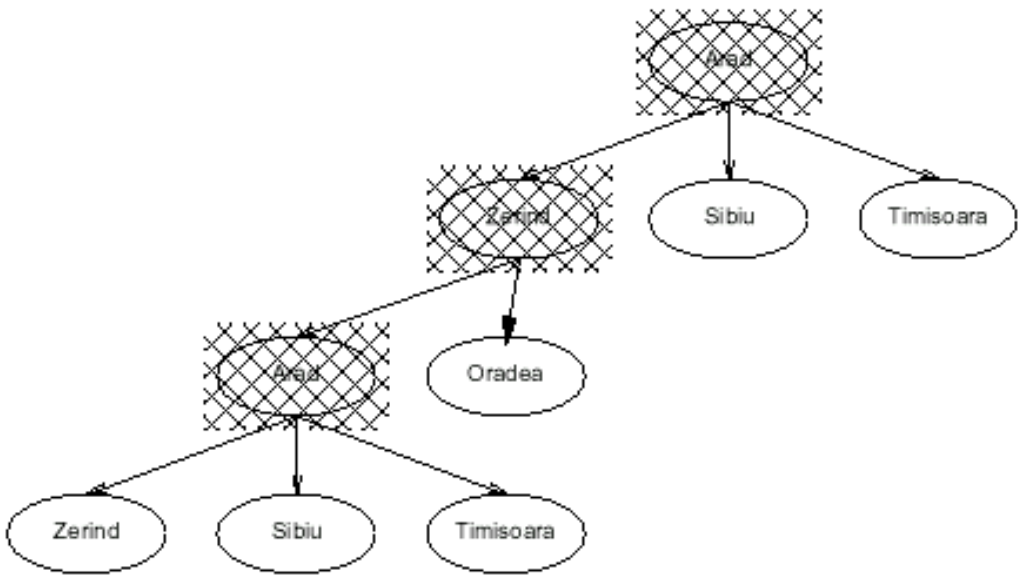
Depth-first search



Depth-first search



Depth-first search



I.e., depth-first search can perform infinite cyclic excursions
Need a finite, non-cyclic search space (or repeated-state checking)

Avoiding repeated states

In increasing order of effectiveness and computational overhead:

- do not return to state we come from, i.e., expand function will skip possible successors that are in same state as node's parent.
- do not create paths with cycles, i.e., expand function will skip possible successors that are in same state as any of node's ancestors.
- do not generate any state that was ever generated before, by keeping track (in memory) of every state generated, unless the cost of reaching that state is lower than last time we reached it.

Depth-limited search

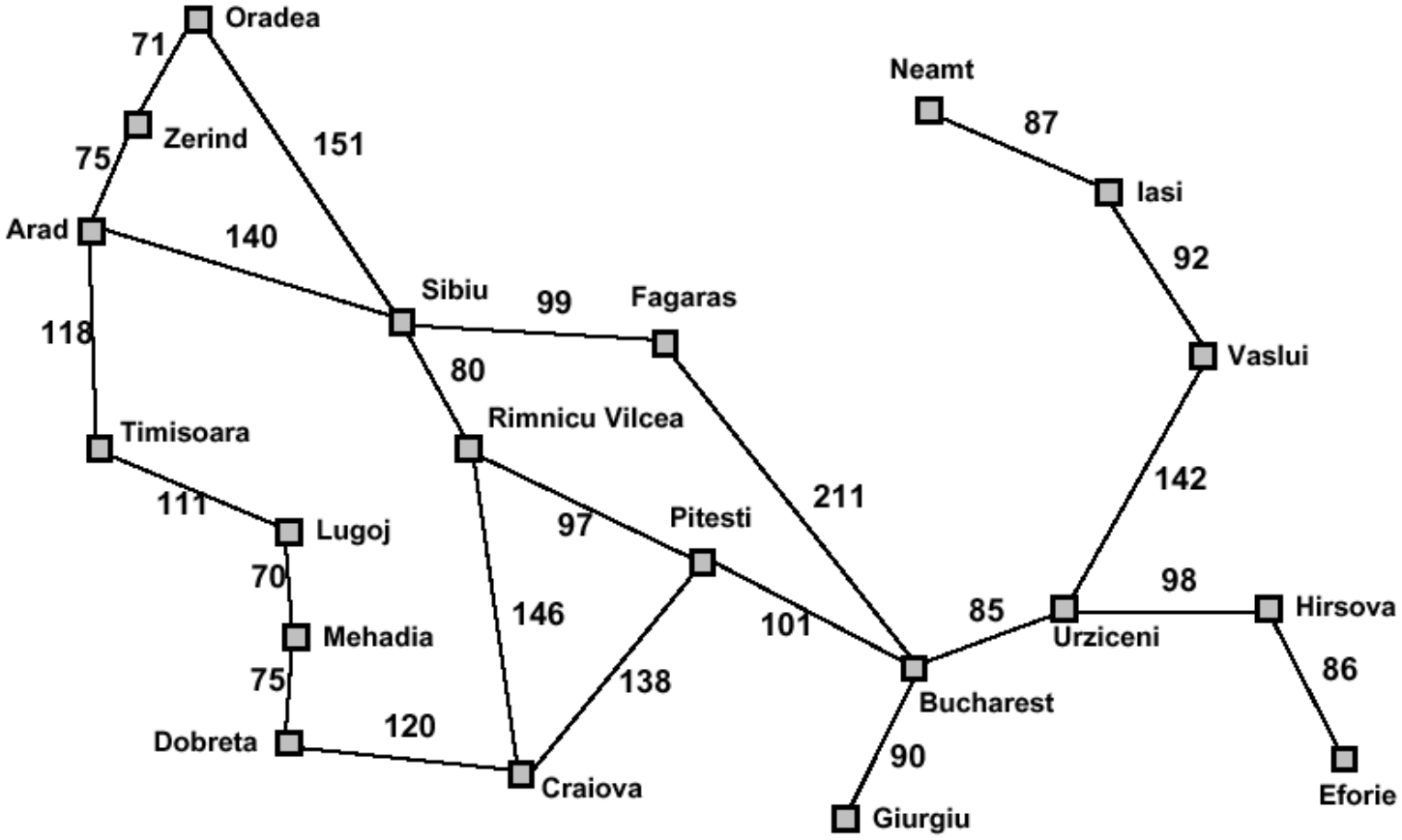
Is a depth-first search with depth limit l

Implementation:
Nodes at depth l have no successors.

Complete: if cutoff chosen appropriately then it is guaranteed to find a solution.

Optimal: it does not guarantee to find the least-cost solution

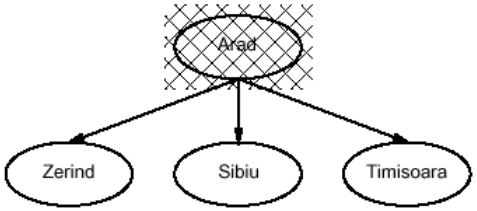
Romania with step costs in km



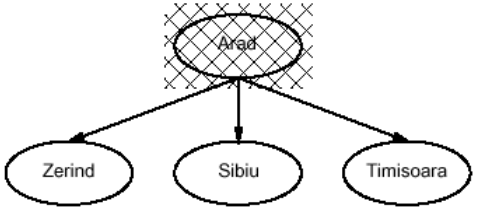
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

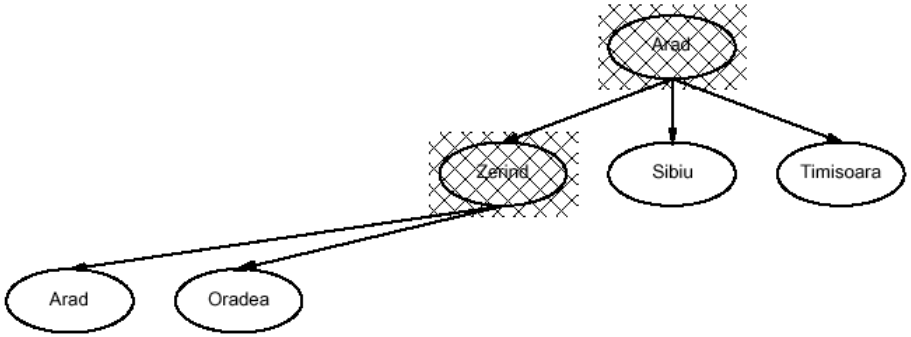
Iterative deepening search $l = 0$

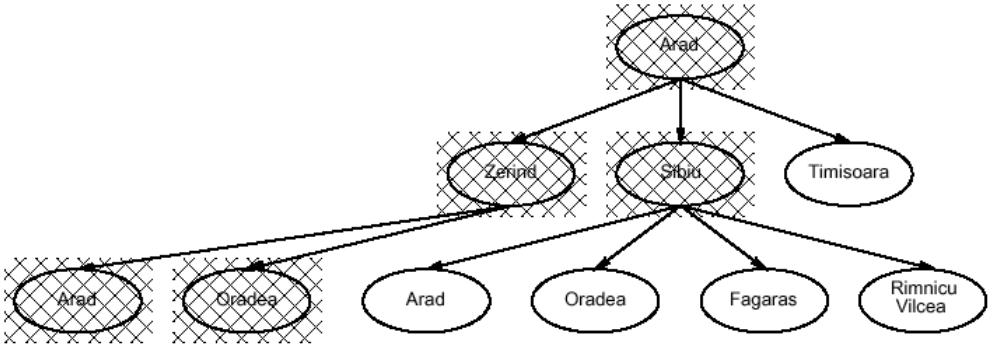
Iterative deepening search $l = 1$



Iterative deepening search $l = 2$






Arad
Zerind
Sibiu
Timisoara
Arad
Oradea
Arad
Oradea
Fagaras
Rimnicu
Vilcea