Lecture 5:
Breadth-first search

Artificial Intelligence
CS-GY-6613-I

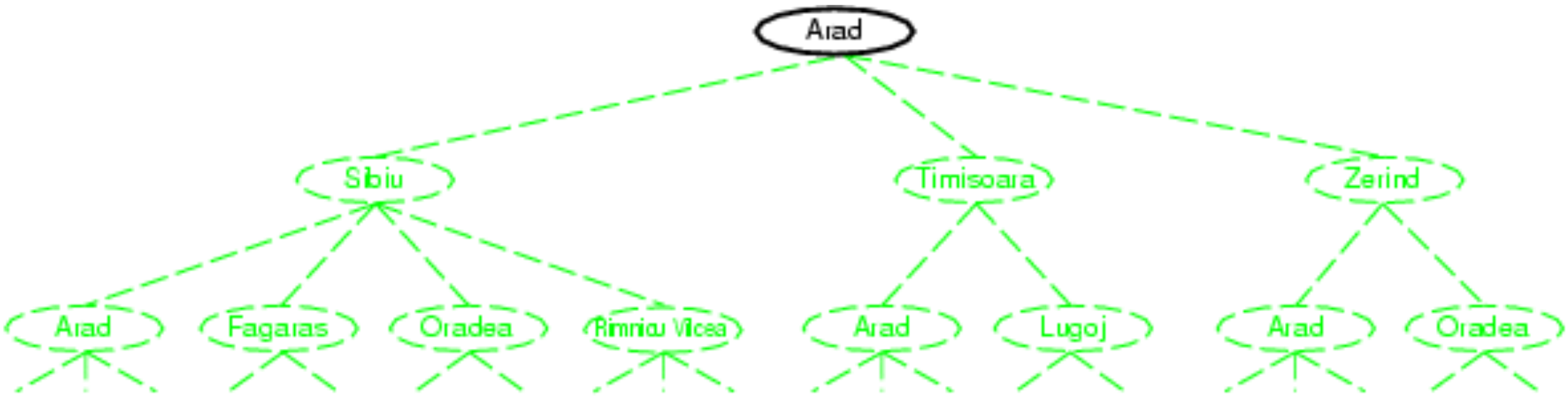
Julian Togelius
julian.togelius@nyu.edu

Tree search

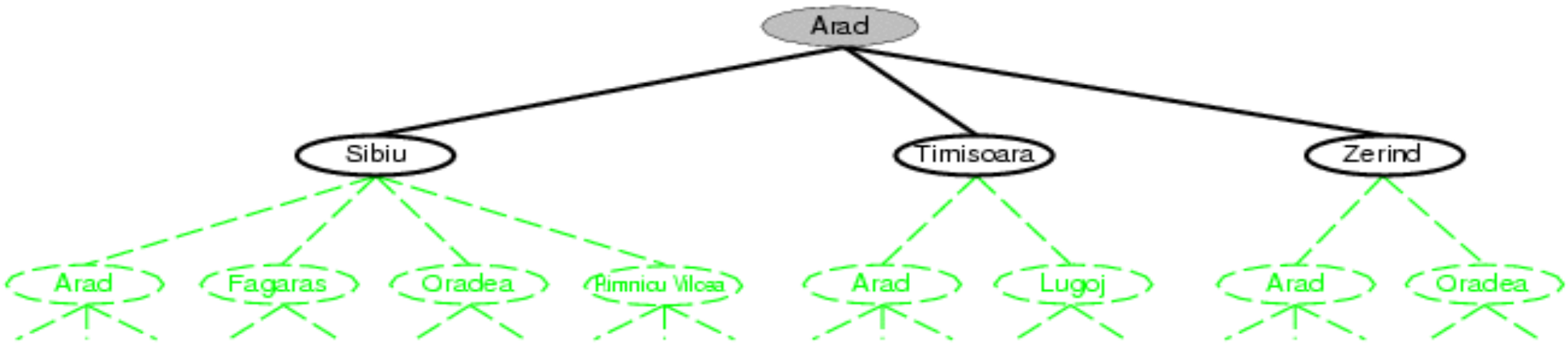
- offline, simulated exploration of state space by generating successors of already-explored states (a.k.a.~expanding states)

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
```

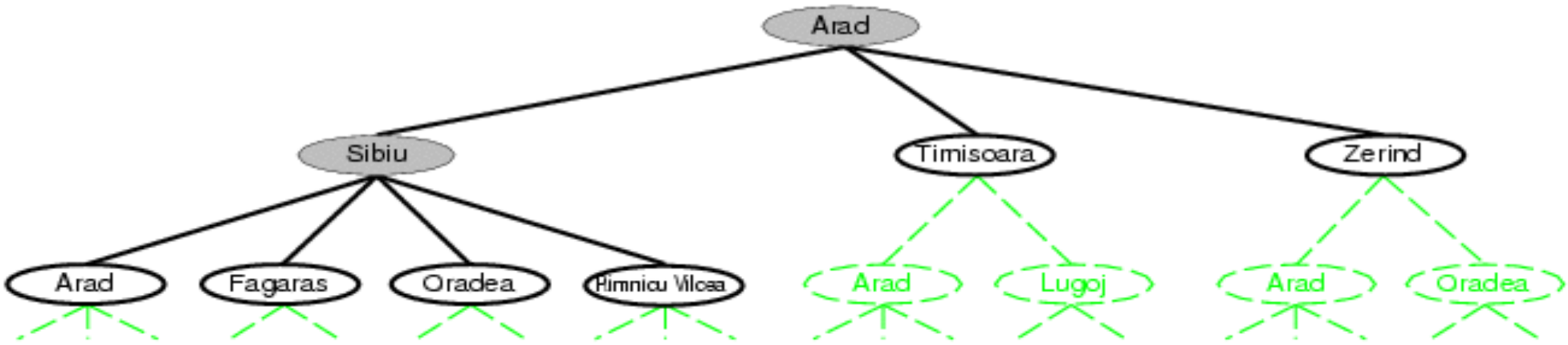
Tree search


Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea

Tree search


Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea

Tree search


Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea

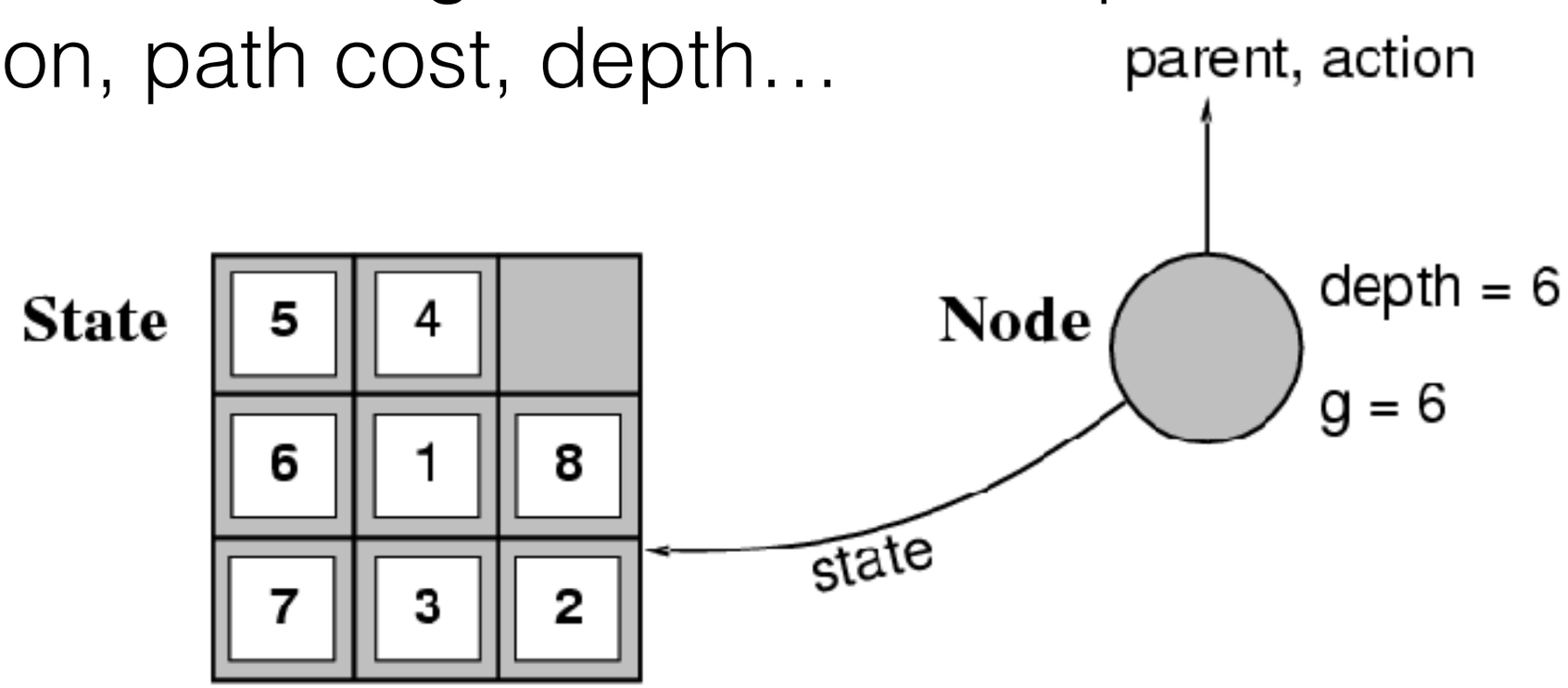
```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
        fringe ← INSERTALL(EXPAND(node, problem), fringe)

---

function EXPAND( node, problem) returns a set of nodes
    successors ← the empty set
    for each action, result in SUCCESSOR-FN[problem](STATE[node]) do
        s ← a new NODE
        PARENT-NODE[s] ← node;  ACTION[s] ← action;  STATE[s] ← result
        PATH-COST[s] ← PATH-COST[node] + STEP-COST(node, action, s)
        DEPTH[s] ← DEPTH[node] + 1
        add s to successors
    return successors
```

Nodes versus states

- A state is a configuration of an environment/agent
- A node is a data structure constituting part of a search tree, might include state, parent node, action, path cost, depth…

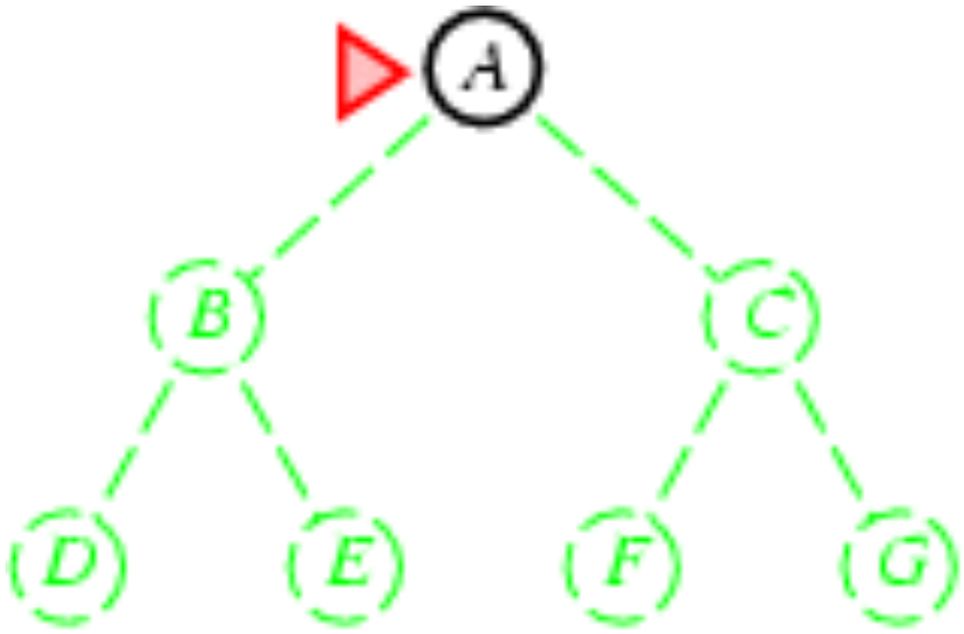


Uninformed search

- Uninformed search strategies use only the information available in the problem definition
- Breadth-first search
- Uniform cost search
- Depth-first search
- Depth-limited search
- Iterative deepening search

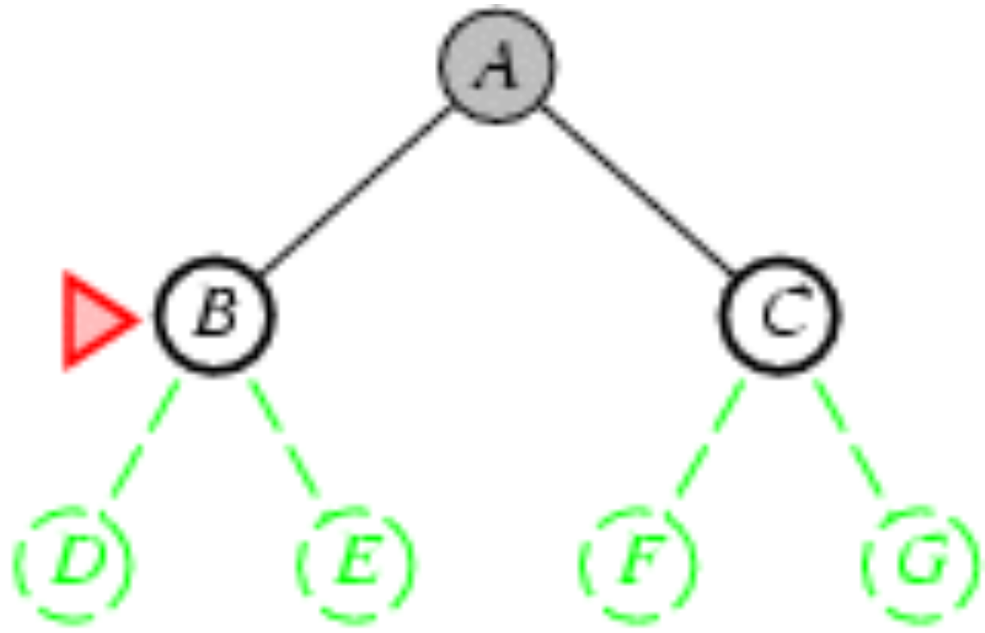
Breadth-first search

- Expand shallowest unexpanded node
- Implementation: new nodes added to a FIFO queue



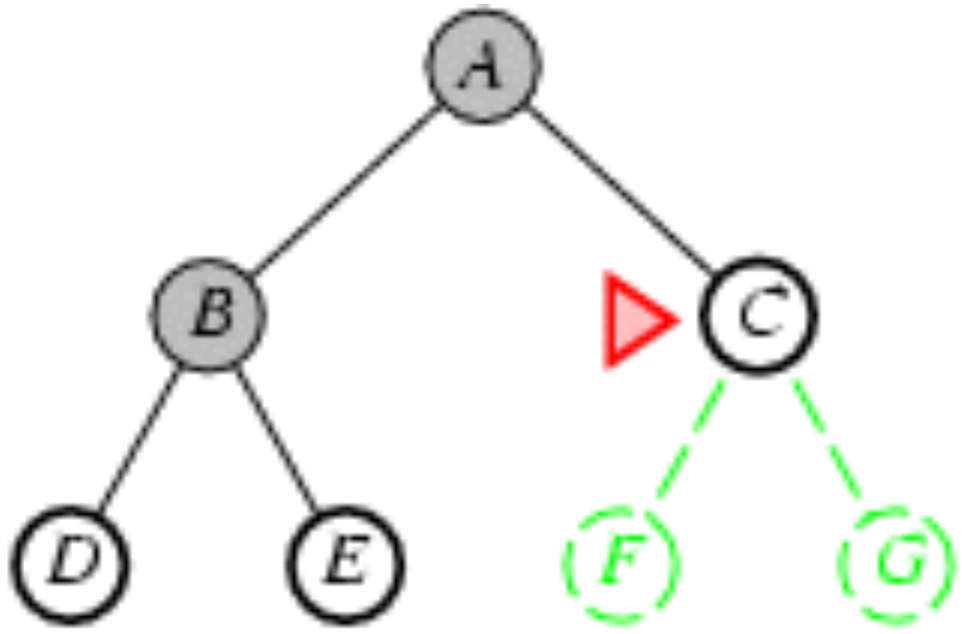
Breadth-first search

- Expand shallowest unexpanded node
- Implementation: new nodes added to a FIFO queue



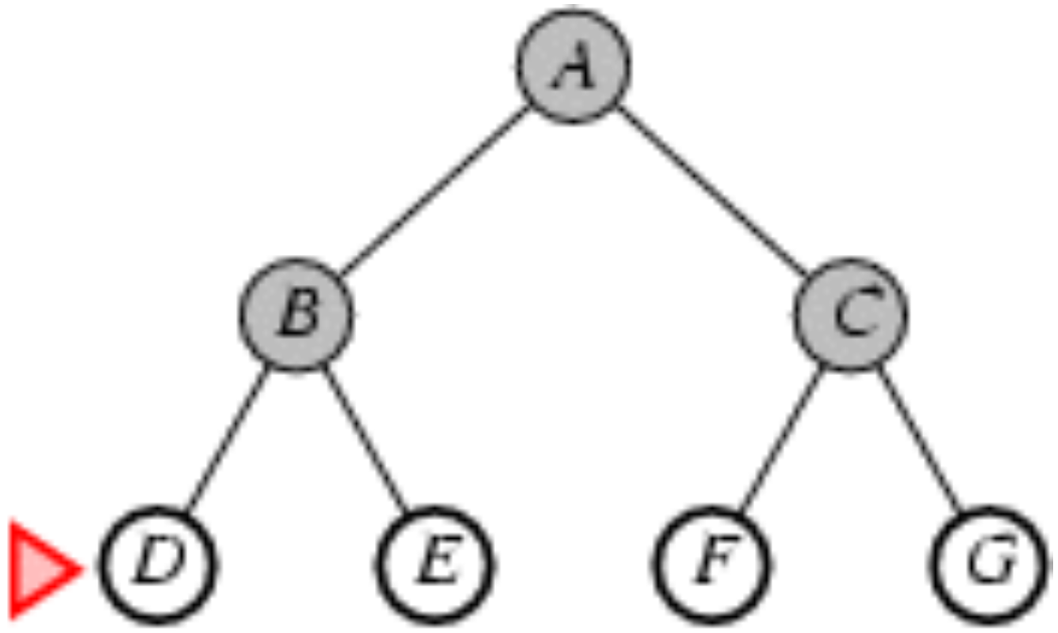
Breadth-first search

- Expand shallowest unexpanded node
- Implementation: new nodes added to a FIFO queue



Breadth-first search

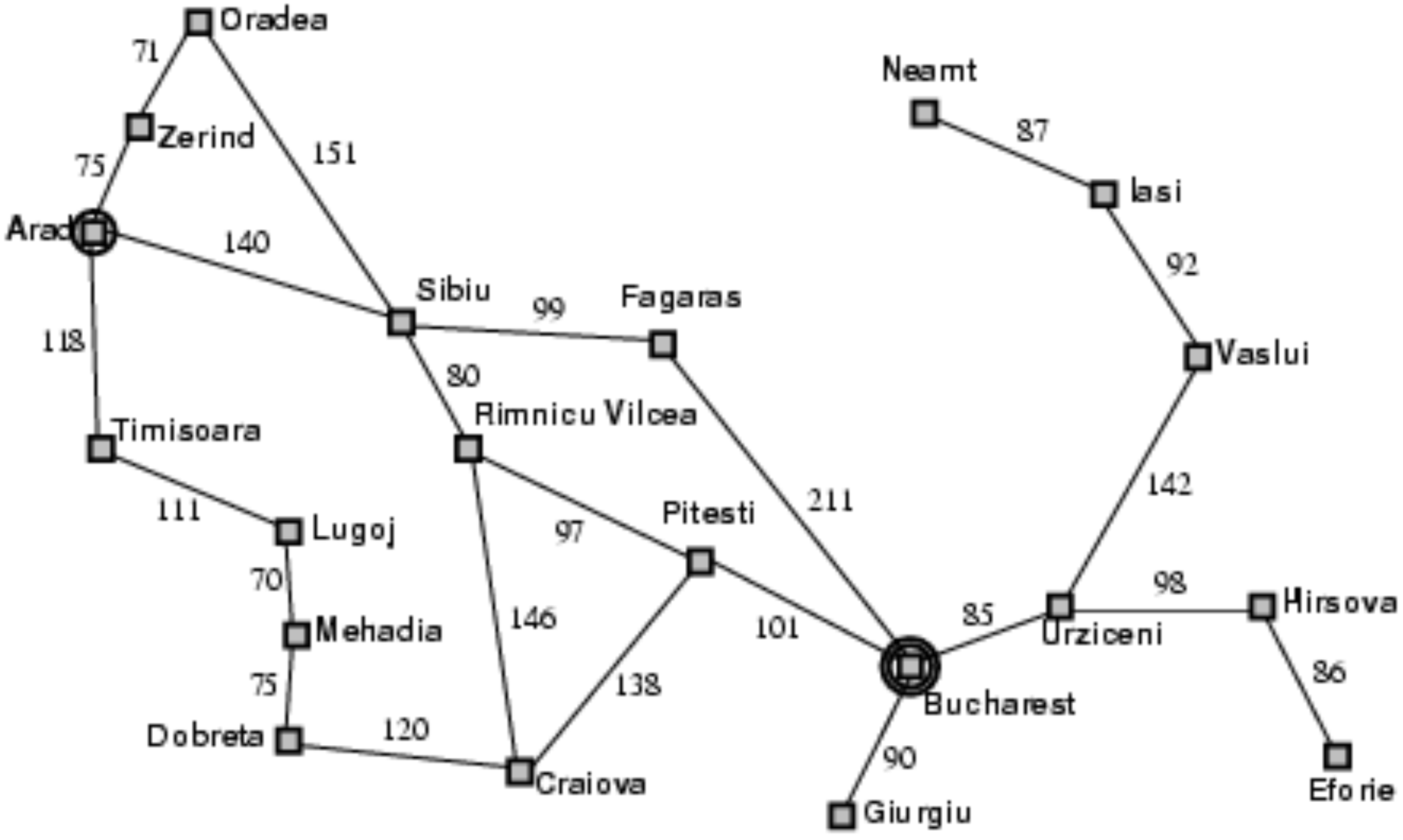
- Expand shallowest unexpanded node
- Implementation: new nodes added to a FIFO queue



Breadth-first search

- Complete? Yes (if b is finite)
- Time? $1+b+b^2+b^3+\ldots +b^d + b(b^d-1) = O(b^{d+1})$
- Space? $O(b^d+1)$ (keeps every node in memory)
- Optimal? Yes (if cost = 1 per step)

From Arad to Bucharest



Uniform-cost search

- Expand least-cost unexpanded node
- Equivalent to breadth-first if step costs all equal
- Complete and optimal