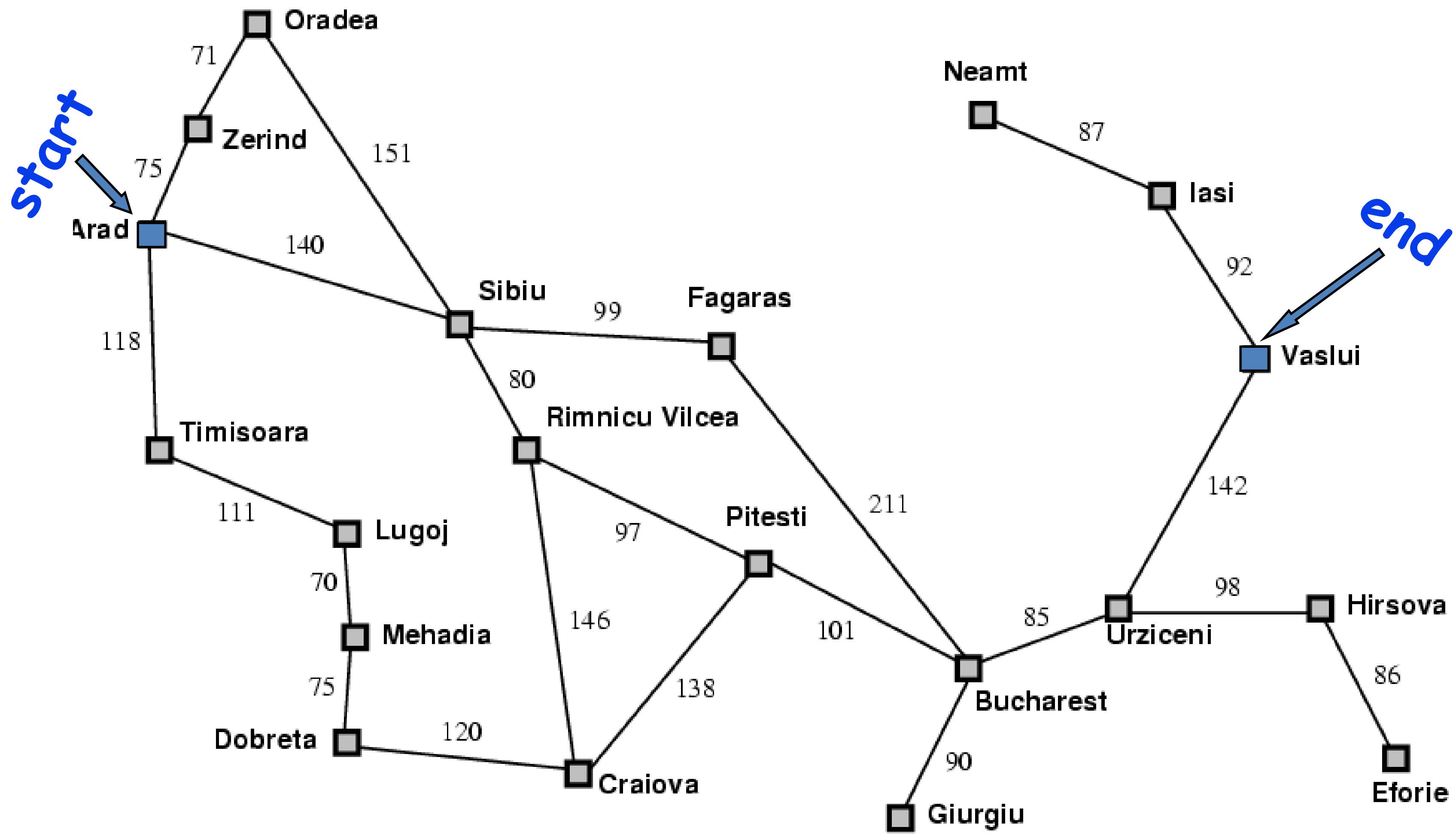
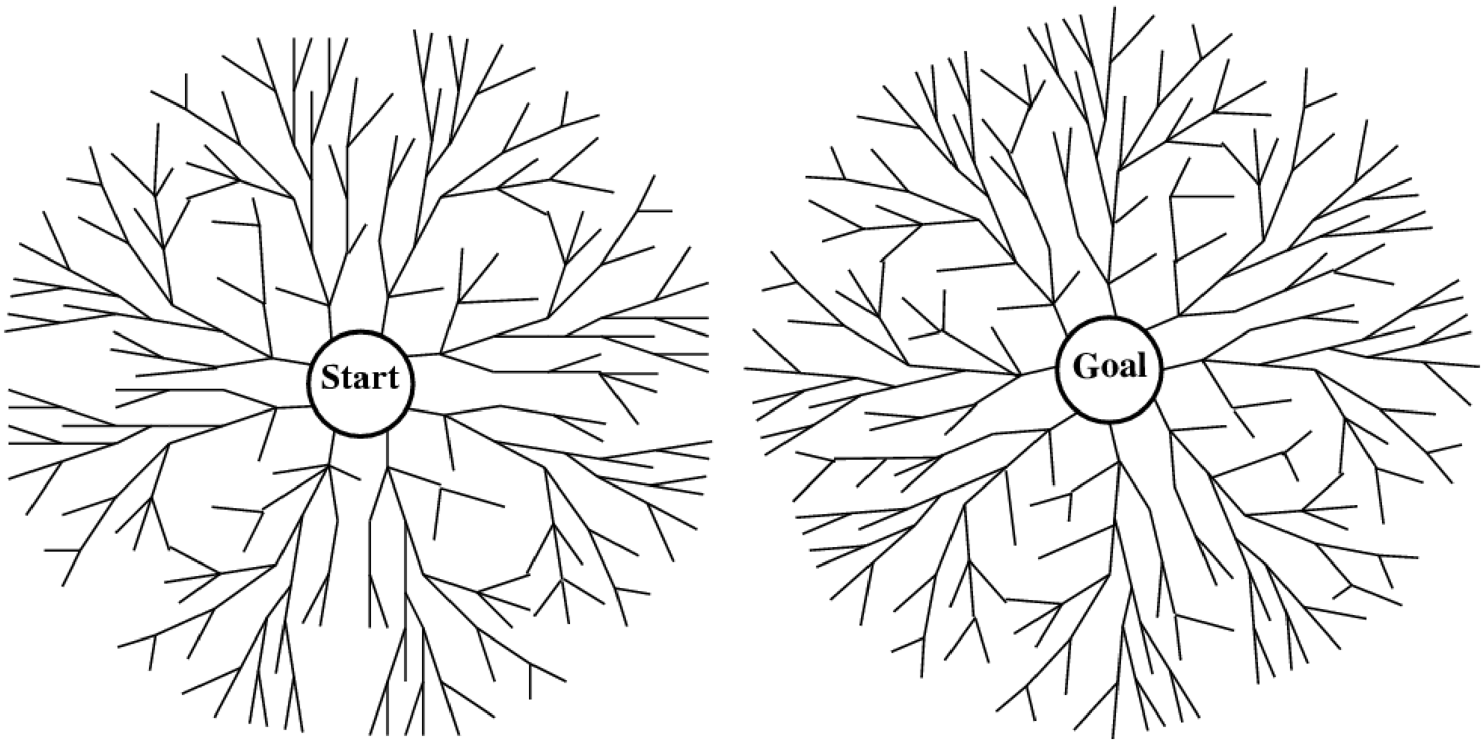
Forwards *vs.* Backwards



vs. Bidirectional



When is bidirectional search applicable?

- **Generating predecessors is easy**
- **Only 1 (or few) goal states**

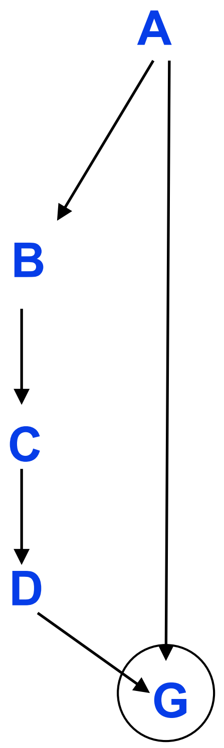
Bidirectional search

- <u>Complete?</u> Yes
- <u>Time?</u>
  - $O(b^{d/2})$
- <u>Space?</u>
  - $O(b^{d/2})$
- <u>Optimal?</u>
  - Yes if uniform cost search used in both directions

Summary of algorithms

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

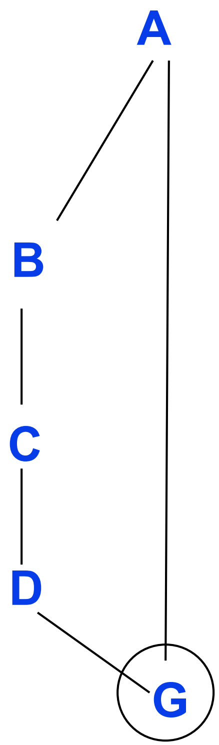


BFS: A,B,G

DFS: A,B,C,D,G

IDDFS: (A), (A, B, G)

Note that IDDFS can do fewer expansions than DFS on a graph shaped search space.



BFS: A,B,G

DFS: A,B,A,B,A,B,A,B,A,B

IDDFS: (A), (A, B, G)

Note that IDDFS can do fewer expansions than DFS on a graph shaped search space.

Search on undirected graphs or directed graphs with cycles…

Cycles galore…

Graph (instead of tree) Search: Handling repeated nodes

- Repeated expansions is a bigger issue for DFS than for BFS or IDDFS
  - Trying to remember all previously expanded nodes and comparing the new nodes with them is infeasible
  - Space becomes exponential
  - duplicate checking can also be expensive

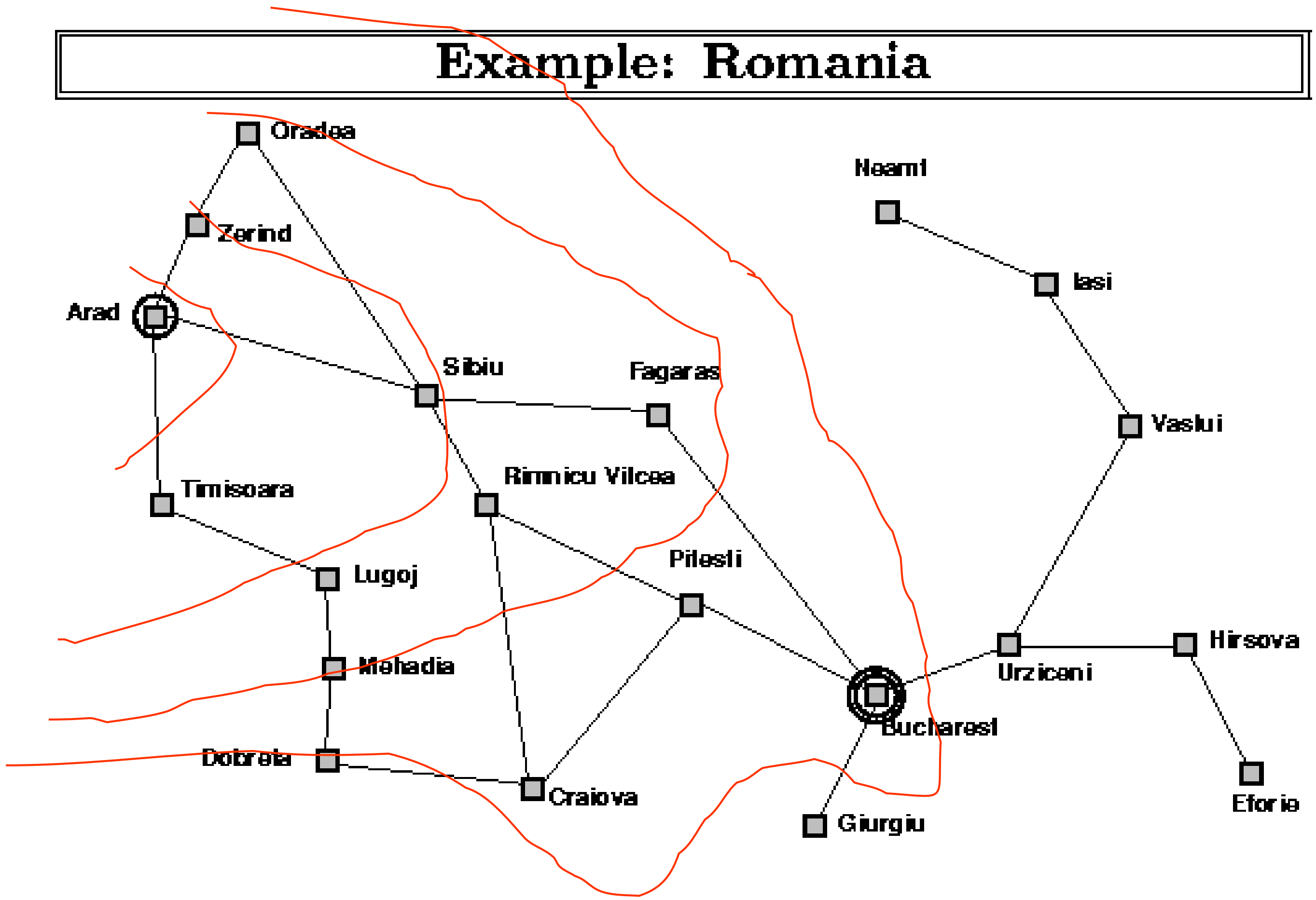
- Partial reduction in repeated expansion can be done by
  - Checking to see if any children of a node n have the same state as the parent of n
  - Checking to see if any children of a node n have the same state as any ancestor of n (at most d ancestors for n—where d is the depth of n)

chapter03 - GSview

File Edit Options View Orientation Media Help

File: chapter03 206, 460pt Page: "5" 5 of 63

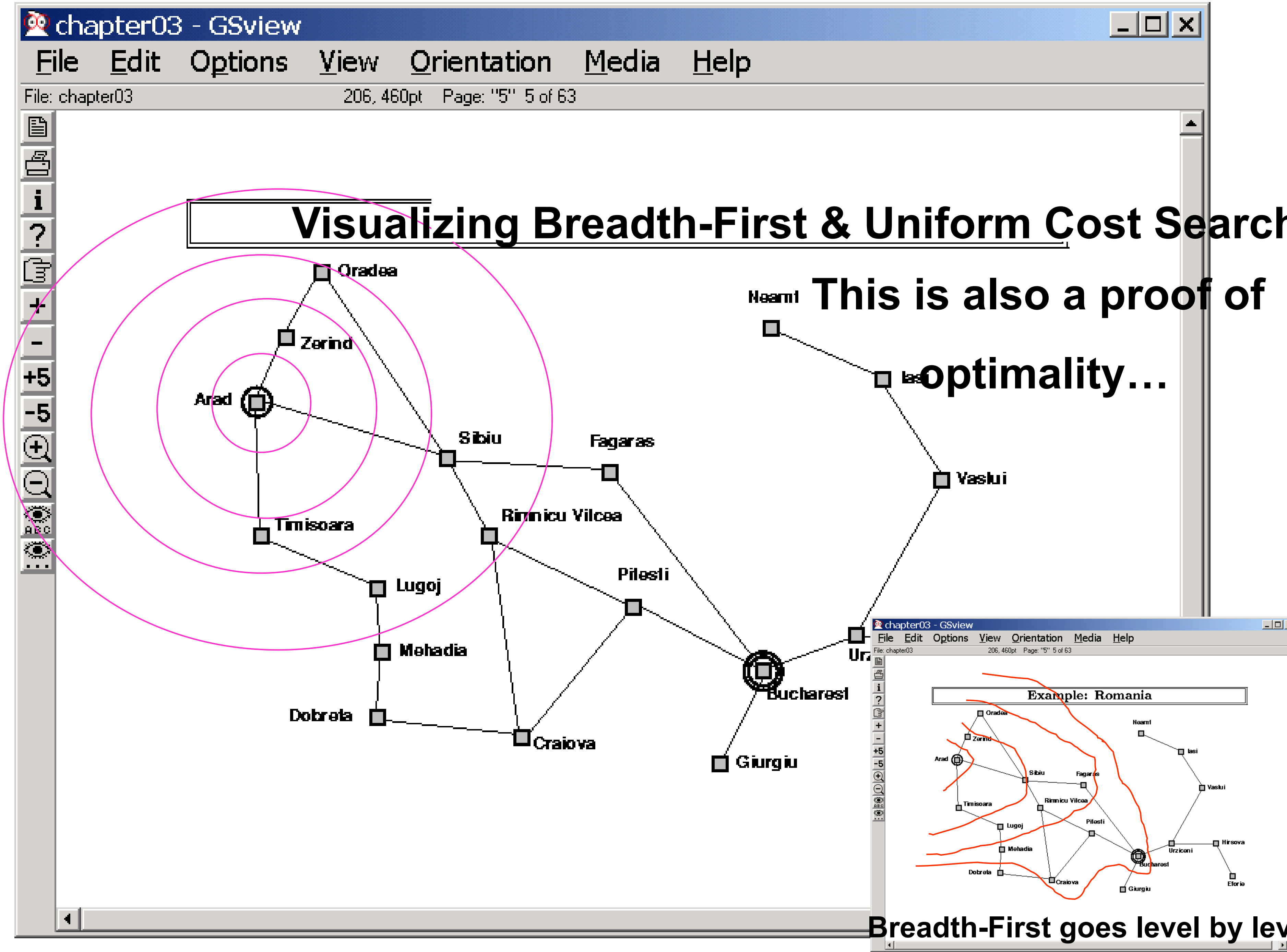
Example: Romania



Breadth-First goes level by level

Visualizing Breadth-First & Uniform Cost Search

This is also a proof of optimality…



Breadth-First goes level by leve

Problem

- All these methods are slow (blind)

© Jen Theodore

- Solution → add guidance ("heuristic estimate")
  → "informed search"