Uninformed Search

CS161

Prof. Guy Van den Broeck

What is uninformed search?

- Structure: Search ‘engine’
- State space vs. search tree (containing nodes)
- Two important notions:
  - Expanding a state
  - Generating a state
- Fringe/frontier: nodes not yet expanded
- Essence of search: which state to expand next?

  Leads to different search strategies
- Blind/uninformed vs. Heuristic/informed search

How Many Nodes in Search Tree?

- Branching factor b; depth d; node count $N(b,d)$

$$N(b,d) = 1 + b + b^2 + \cdots + b^d$$

$$b \cdot N(b,d) = b + b^2 + b^3 + \cdots + b^{d+1}$$

$$b \cdot N(b,d) - N(b,d) = b^{d+1} - 1$$

$$N(b,d) = \frac{b^{d+1} - 1}{b - 1} \approx \frac{b^{d+1}}{b - 1}$$

$$= b^d \left(\frac{b}{b-1}\right)$$

$$= O(b^d)$$

Tree Search Algorithm

```
frontier = {initial state}
loop do
    if frontier is empty return fail
    node = choose leaf to remove from frontier
    if node is goal state return node's state
    frontier += node.expand()
```

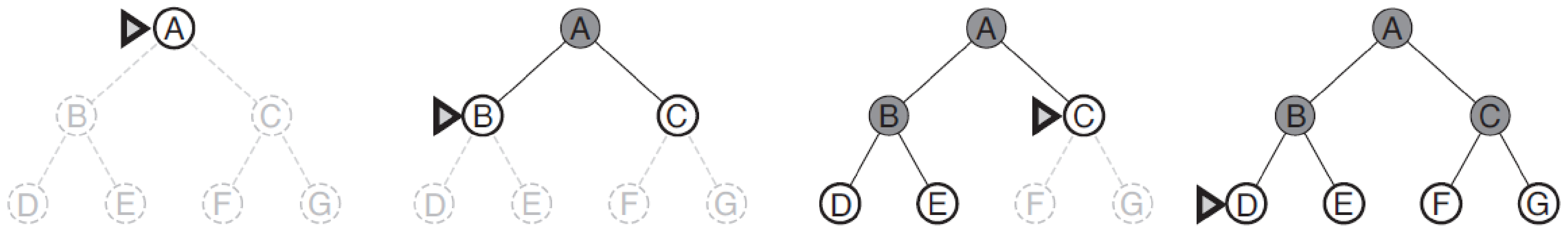
Graph Search Algorithm

```
frontier = {initial state}
loop do
    if frontier is empty return fail
    node = choose leaf to remove from frontier
    if node is goal state return node's state
    frontier += node.expand()
```

+ Add chosen leaf to explored set (initially empty)
+ Add only new nodes to frontier

    Nodes not in frontier or explored set

Breadth-First Search



Properties

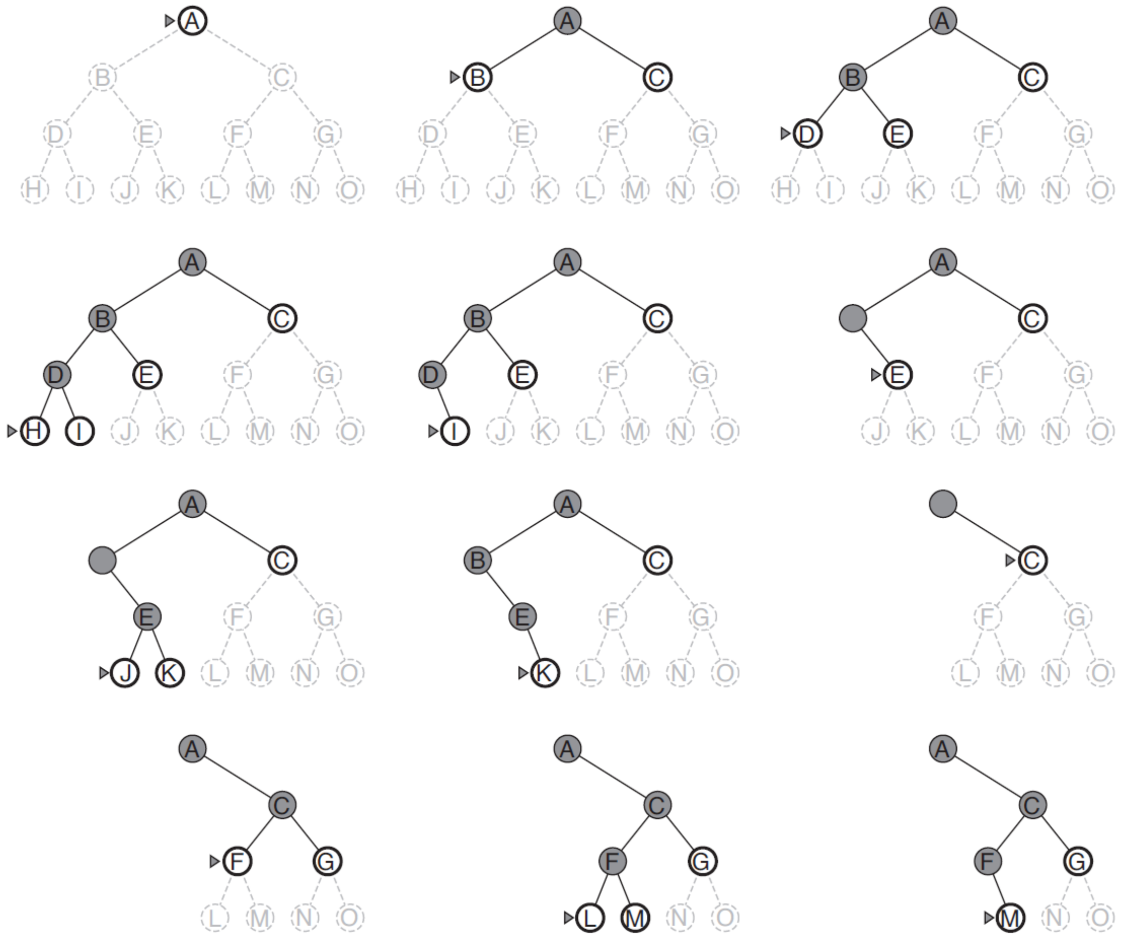
- Complete? Yes
- Optimal? Yes

  Frontier separates explored from unexplored states

- Time and space complexity depend on whether goal test happens on expand or on generate.
  - On expand: O(b^d)
  - On generate: O(b^{d+1})

  (d is optimal solution depth, b is branching factor)

Depth-First Search



Properties

- Complete? No
  - Fail is infinite-depth spaces:
    - Loops (run in circles)
    - Infinite state space (e.g., Knuth's problem)
- Optimal? No
- Time complexity: O(b^m)

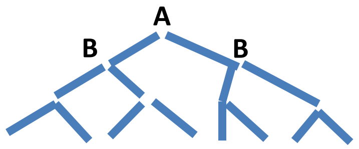
  When is this good/bad?
- Space complexity: O(bm)

  (m is max depth of search tree)

What about repeated states?

- Can turn a linear problem into exponential
  - E.g.:

$$\begin{tikzcd} A \arrow[r, bend left] \arrow[r, bend right] & B \arrow[r, bend left] \arrow[r, bend right] & C \arrow[r, bend left] \arrow[r, bend right] & D \arrow[r, bend left] \arrow[r, bend right] & E \end{tikzcd}$$



  - E.g.: moving in a grid
    - Search tree at depth d has O(4^d) nodes
    - Only O(d^2) distinct leaves (a circle in the grid)
    - For d=20: trillion vs. hundred nodes.

  - DFS is not for free!

How to deal with repeated states?

1. Check your parents
2. Check your ancestors
3. Check every node visited already

- (1-2) is cheap with DFS
- (3) is graph search instead of tree search
  - Complexity O(s) where s is number of states
  - Can be less than O(b^d) but more than O(bd)

Can we make DFS terminate?

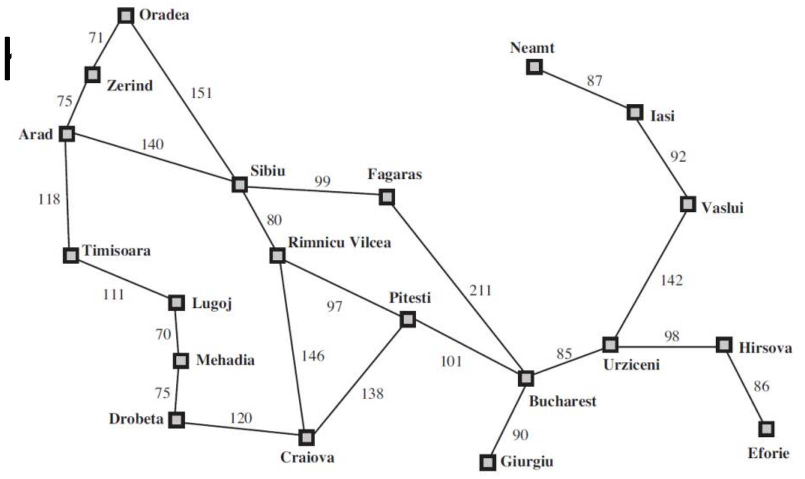
Depth-Limited Search

- Pretend search tree stops at depth limit l
  Nodes at depth l have no successor
- Run DFS

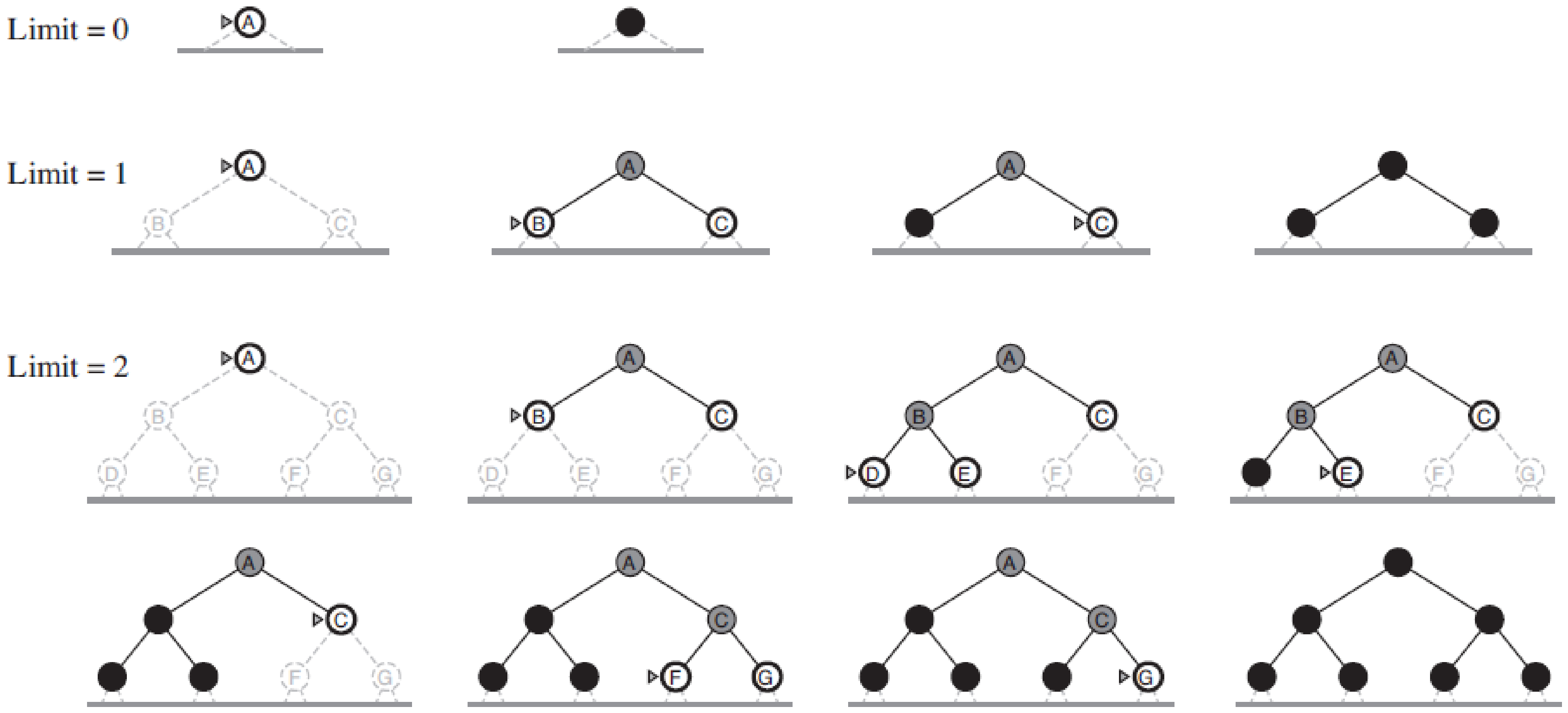
- Complete? Yes, if $l \geq d$.
- Optimal? No
- Time complexity? O(b^l)
- Space complexity? O(bl)

Can we make DLS complete?

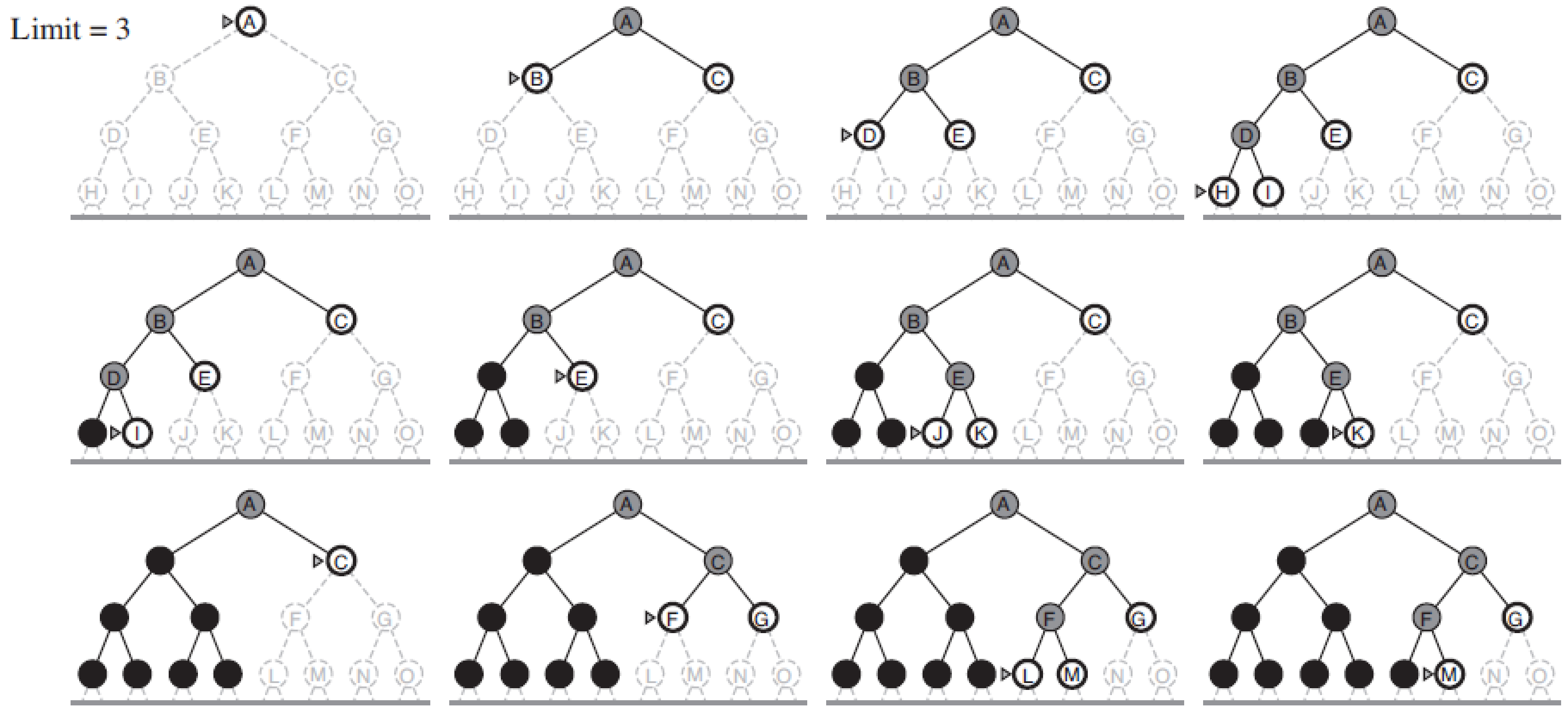
- How do we set depth limit l?
  - Maximum length path without repeated states
  - Compute the <u>diameter</u> of search space:
    Maximum shortest path between any two nodes
  - E.g., Romania has 20 cities:
    - Depth limit 19 is trivial
    - Diameter is 9, therefore deptl
- More general solution?



Iterative Deepening



Iterative Deepening



Properties

- Complete? Yes
- Optimal? Yes
- Space? O(bd)
- Time?

ID Time Complexity

- Time?

  - # nodes to depth d $\approx b^d\left(\frac{b}{b-1}\right)$
  - # nodes explored in total

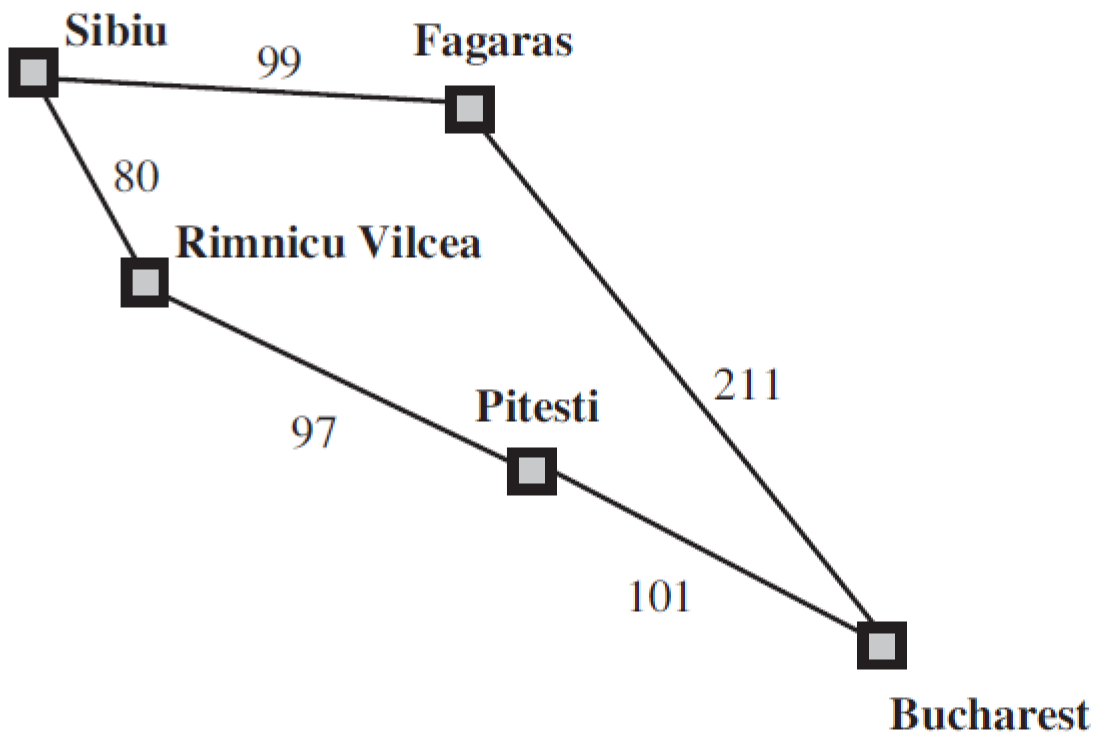
    $$\approx b^0\left(\frac{b}{b-1}\right) + b^1\left(\frac{b}{b-1}\right) + \cdots \qquad = b^d\left(\frac{b}{b-1}\right)^2$$

  - ID is $\left(\frac{b}{b-1}\right)$ times slower than BFS (at most 2 times)
  - b=10,d=5: 111k nodes for BFS/DFS, 123k for ID
  - Can ID be better than BFS?

    Yes if BFS goal test on expand: 1111k nodes for BFS!

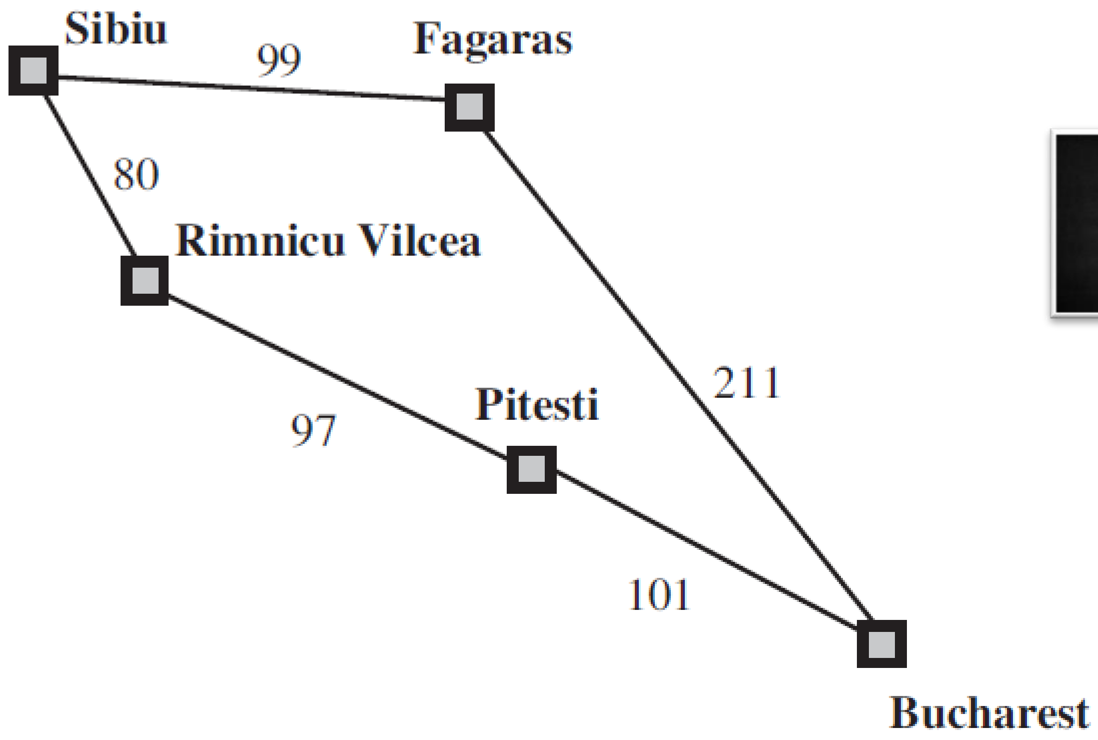
Bidirectional Search

What if costs not 1 per action?



Uniform-Cost Search

f(n) is distance g(n) from start



Properties

- Optimality: Careful about goal test!!!
- Complexity analysis
  - C^* is best solution cost
  - ϵ is cheapest action cost