ADAM PURTEE
UNIVERSITY OF ROCHESTER
CSC 442: ARTIFICIAL INTELLIGENCE

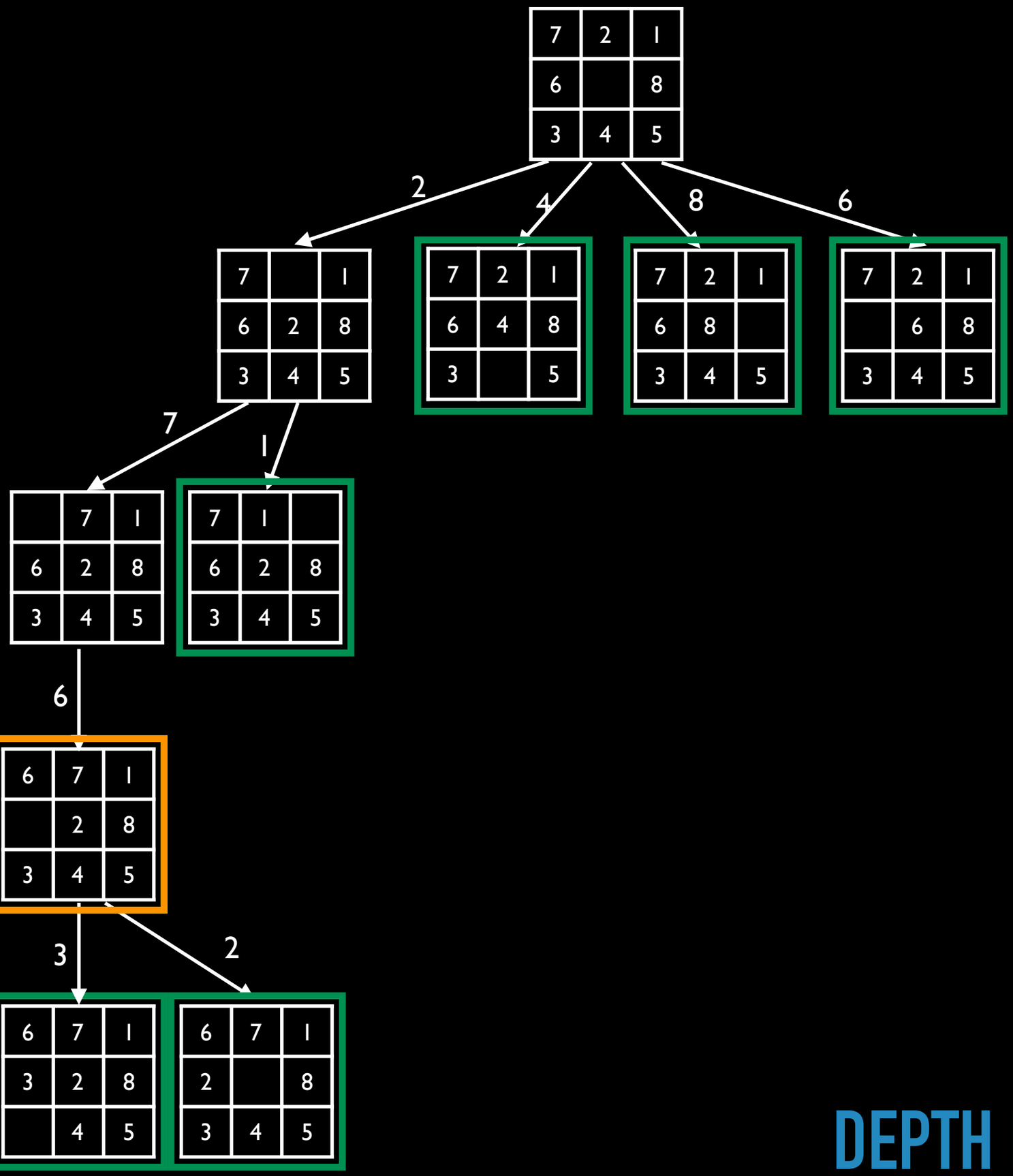
SEARCH ALGORITHMS PT 2

ANNOUNCEMENTS

- Quiz on Monday! (Online, Search)
- Project 1 begins Monday (adversarial search)

LAST TIME

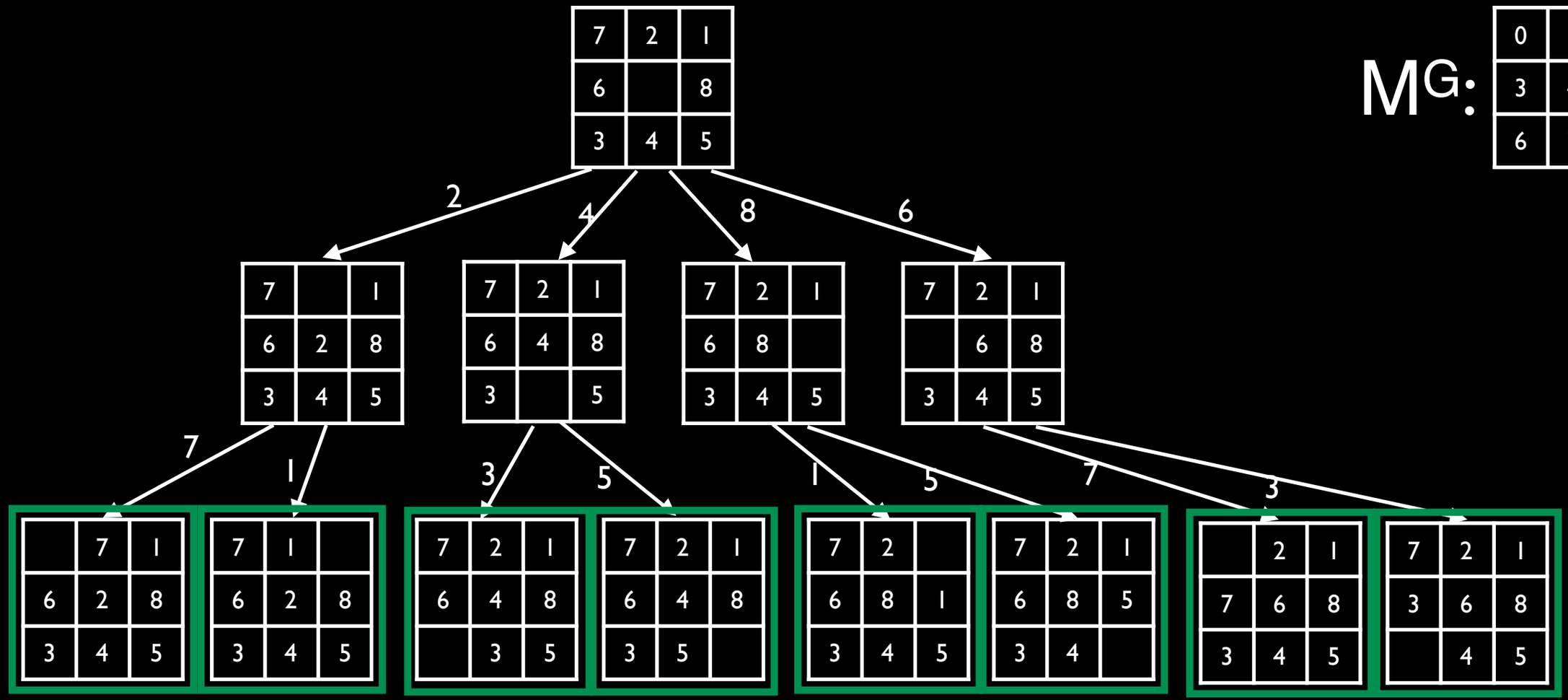
- DFS, BFS, IDS
- Completeness, Optimality, Time/Space Complexity



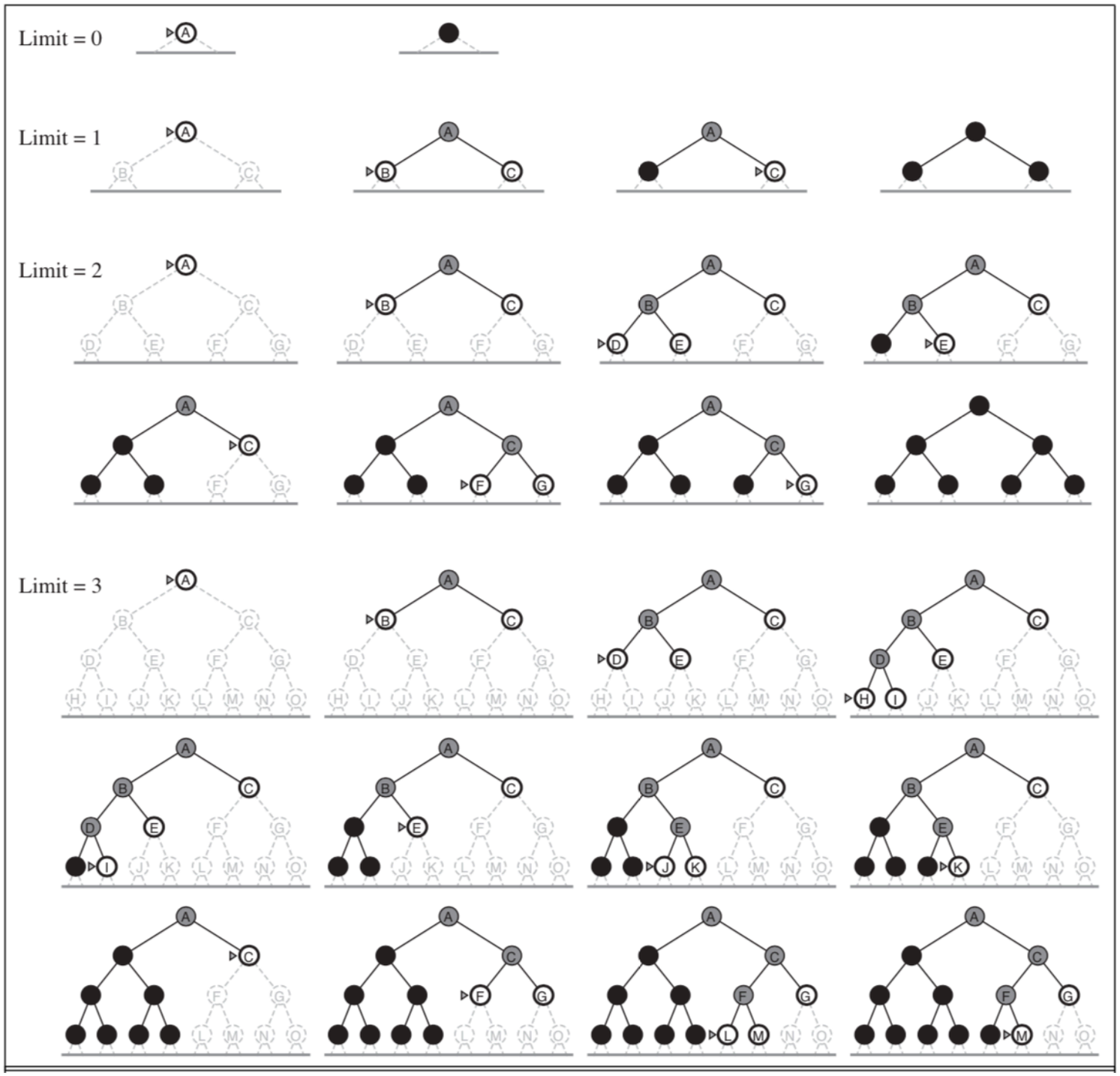
MG:

0	1	2
3	4	5
6	7	8

DEPTH FIRST (GRAPH) SEARCH



BREADTH FIRST SEARCH



AIMA Figure 3.19, p89

UNINFORMED STRATEGIES

	BFS	DFGS	DFTS	IDS
Complete?	✓	✓	✗	✓
Optimal?	✓*	✗	✗	✓*
Time	$O(b^d)$	$O(b^m)$	$O(b^m)$	$O(b^d)$
Space	$O(b^d)$	$O(b^m)$	$O(bm)$	$O(bd)$

* If step costs are identical (see book)

AIMA

"Iterative deepening is the preferred uninformed search method when the search space is large and the depth of the solution is not known."

AIMA

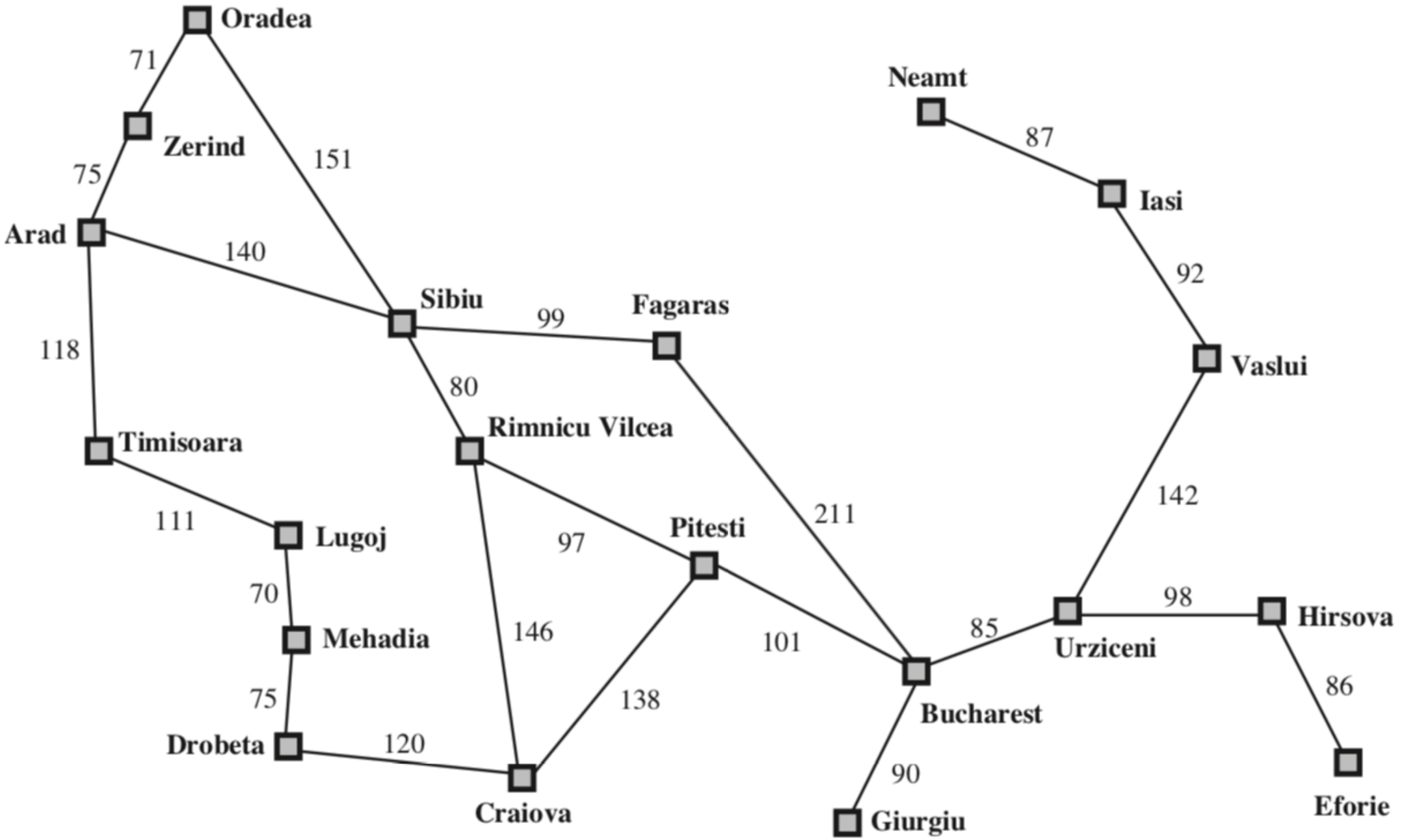
"Exponential complexity search problems cannot be solved by uninformed methods for any but the smallest instances."

NEW ALGORITHMS TODAY

- Uniform Cost Search (Dijkstra's Algorithm)
- Greedy Best First Search
- A* Search

- What about problems where we're not interested in the number of steps in the solution, but instead we want to minimize the total **cost**.

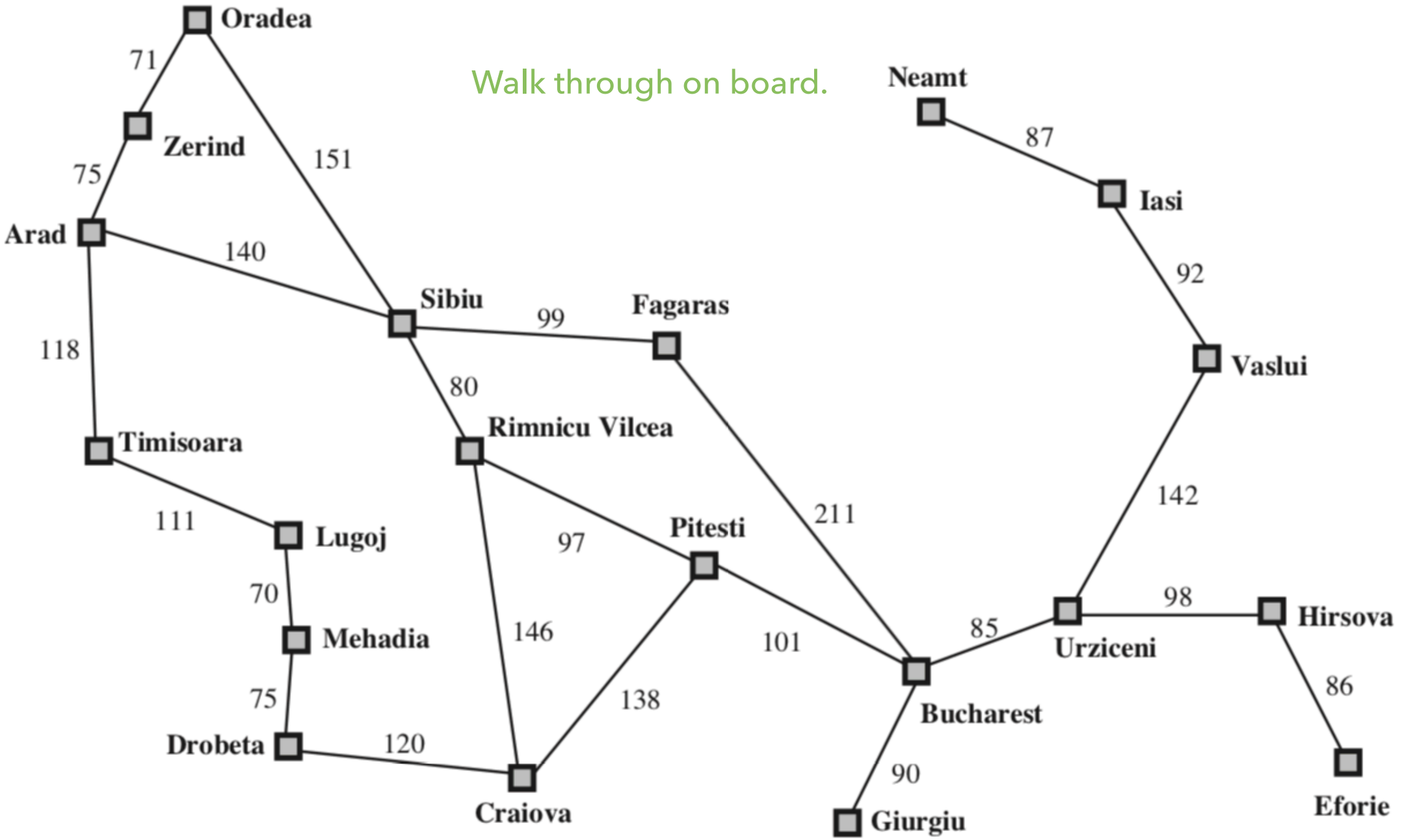
ROUTE FINDING



UNIFORM COST SEARCH

- Maintain a priority queue of states and always expand in the order from least-expensive to most-expensive.
- Very similar to BFS!

ROUTE FINDING



Walk through on board.

UNIFORM COST SEARCH

```
def search(problem):
    let q be a new queue
    let e be the explored set
    add Node(initial state, 0, nil) to q
    add initial state to e
while q is not empty:
    n = front(q)
    if goal(n.state):
        return solution(n)
    add n.state to explored
    for a in actions(s):
        s' = transition(n.s, a)
        add Node(s', n.c + cost(a, s), a) to q
 return nil
```

dfs	lifo
bfs	fifo
ucs	priority
a*	priority

node — state + cost + parent

cost(a,s) - cost of action a in state s

solution(n) - recursively extracted sequence of actions,

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure

    node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
    frontier ← a priority queue ordered by PATH-COST, with node as the only element
    explored ← an empty set
    loop do
        if EMPTY?(frontier) then return failure
        node ← POP(frontier)   /* chooses the lowest-cost node in frontier */
        if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
        add node.STATE to explored
        for each action in problem.ACTIONS(node.STATE) do
            child ← CHILD-NODE(problem, node, action)
            if child.STATE is not in explored or frontier then
                frontier ← INSERT(child, frontier)
            else if child.STATE is in frontier with higher PATH-COST then
                replace that frontier node with child
```

AIMA Figure 3.14, p84

SEARCH ALGORITHMS — UCS

- Complete… Yes! ✓ (b finite, same as bfs) (all steps cost at least ϵ)
- Optimal… Yes! ✓
- Time complexity … $O(b^{1+\lfloor C^*/\epsilon \rfloor})$
- Space complexity … $O(b^{1+\lfloor C^*/\epsilon \rfloor})$

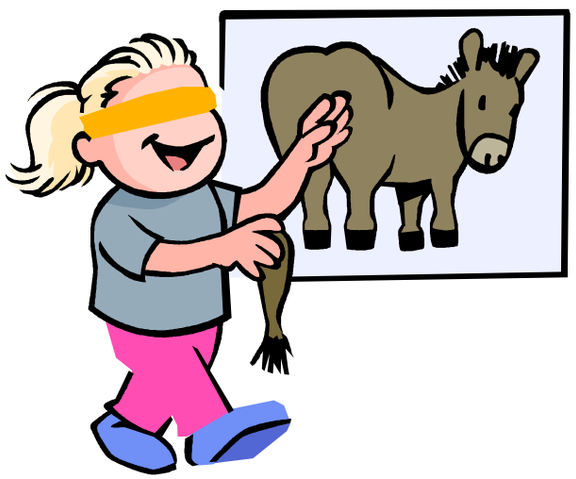
Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes^{a}	Yes^{a,b}	No	No	Yes^{a}	Yes^{a,d}
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon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

AIMA Figure 3.21, p92

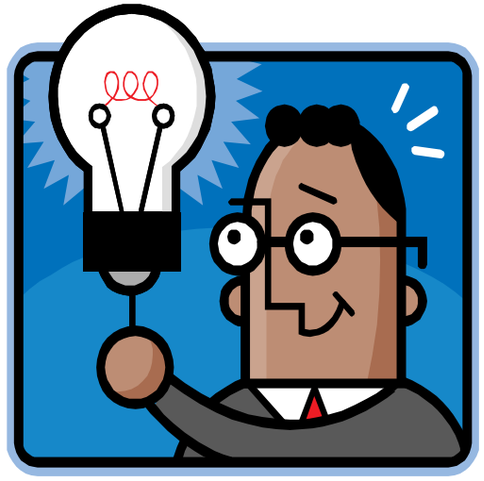
SEARCH STRATEGIES

Uninformed



No additional information about states

Informed (Heuristic)



Can identify "promising" states

SEARCH STRATEGIES

- DFS/BFS/IDS/UCS are examples of **uniformed** or **brute-force** search which explore the state space blindly.
- Sometimes, however, we can do better!

SEARCH STRATEGIES

- Next, we'll look at two variations of UCS which make use of **heuristics** to (hopefully) **expand fewer nodes** while still finding the goal.

- Best First Search – Always choose the "best" route.
- A* Search – Choose the best option, based on where we've been.

HEURISTICS

- A **heuristic** - $h(n)$ - is a special kind of function which estimates the cost of an optimal solution through a particular node n.

- A heuristic is **admissible** if it never over-estimates the true cost of a solution.

- A heuristic is **consistent** if it obeys the triangle inequality:

  - $h(n) \leq c(n, a, n') + h(n')$

EVALUATION VS HEURISTIC FUNCTIONS

- $f(n)$ - Cost of cheapest path from n to a goal node
- $h(n)$ - **Estimated** cost of cheapest path from n to goal

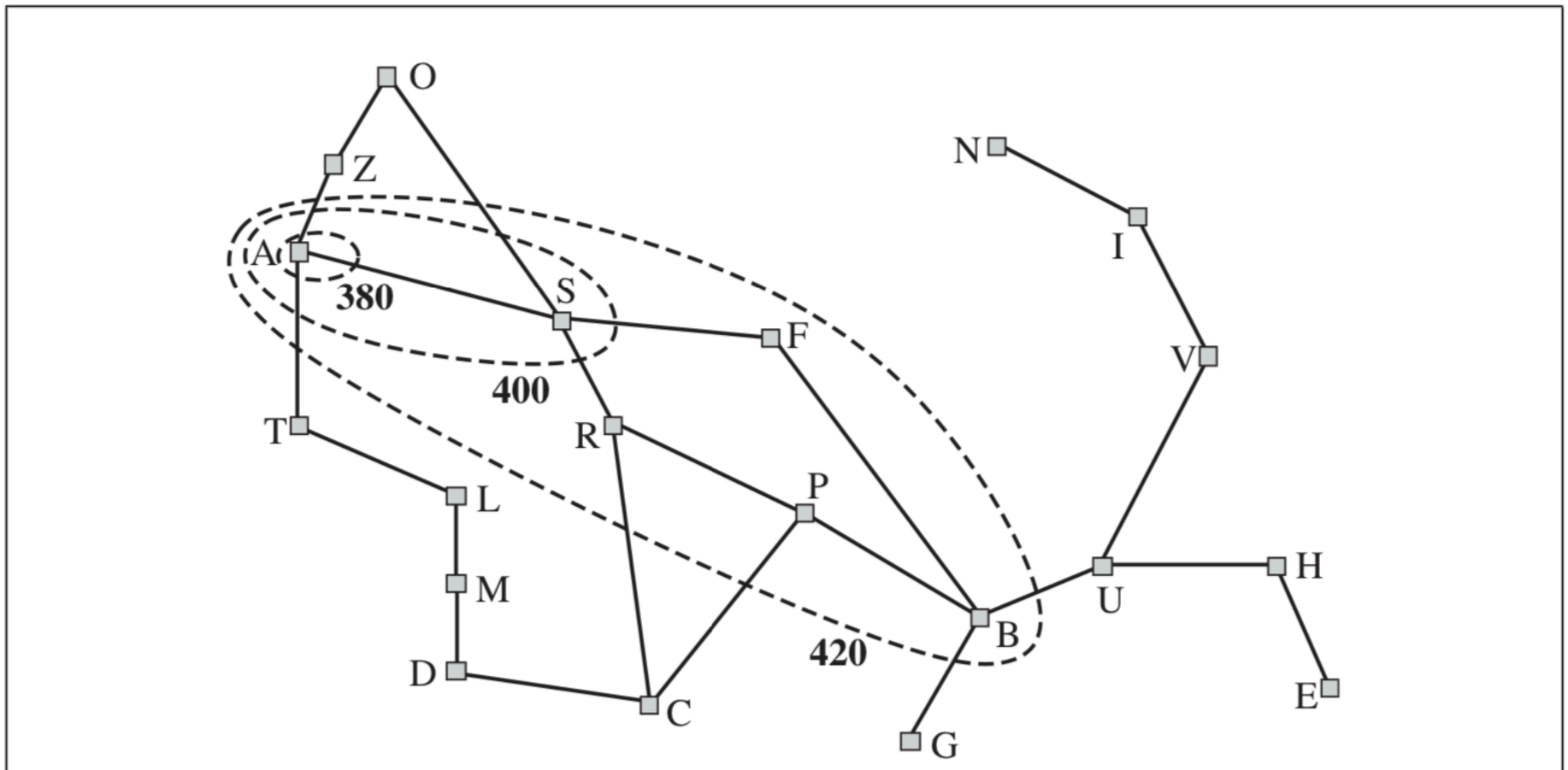
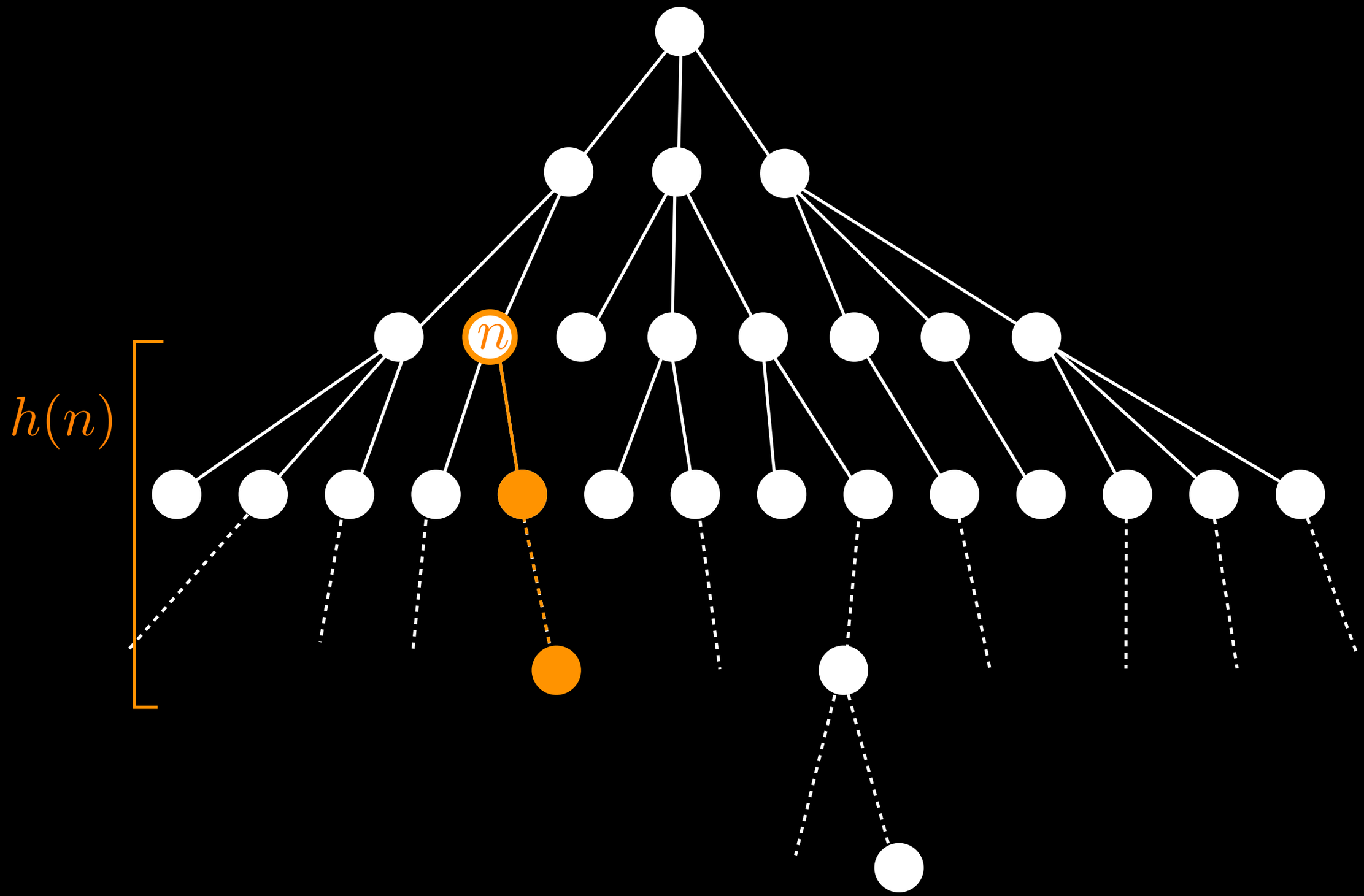


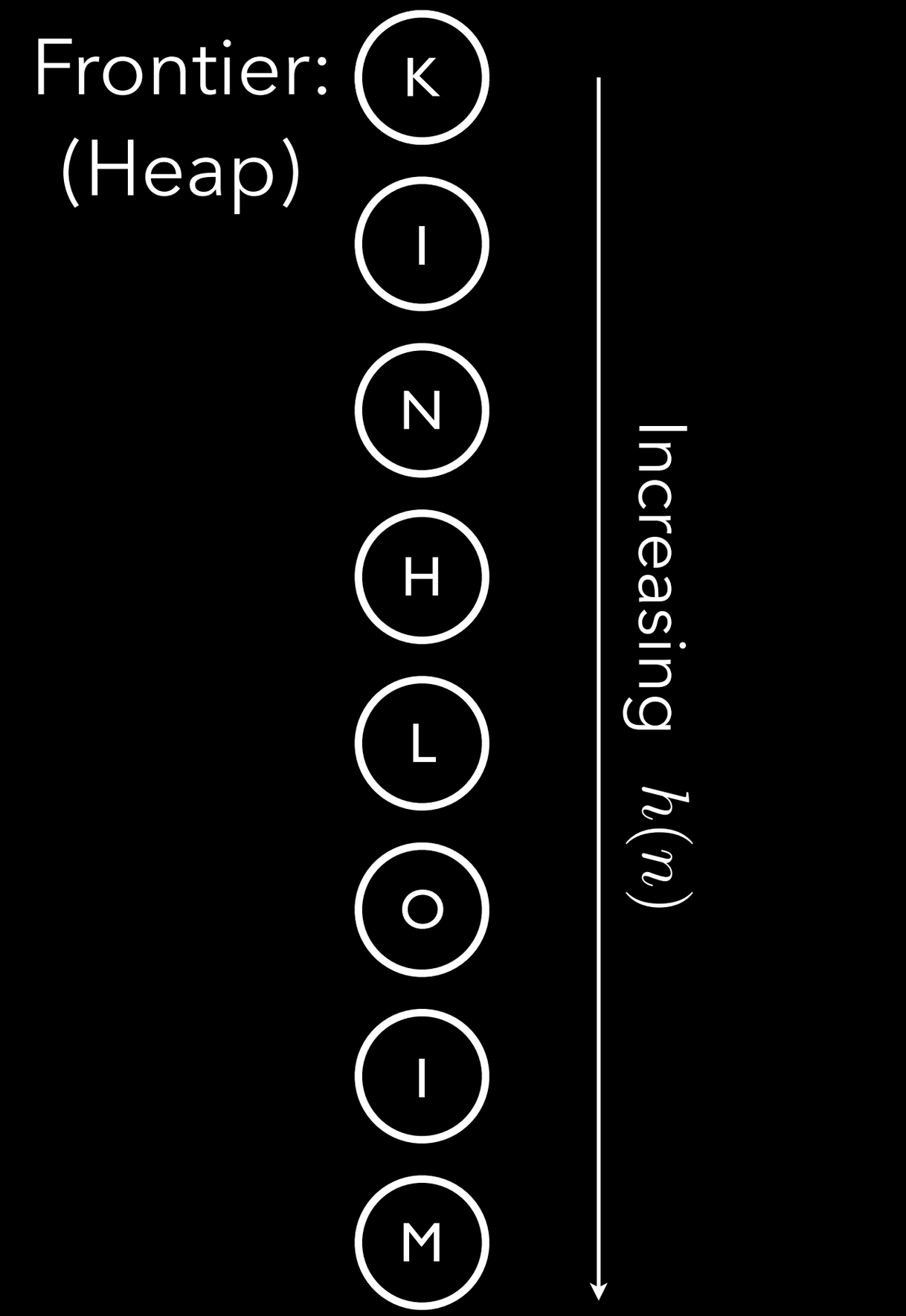
Figure 3.25 Map of Romania showing contours at $f = 380$, $f = 400$, and $f = 420$, with Arad as the start state. Nodes inside a given contour have f-costs less than or equal to the contour value.

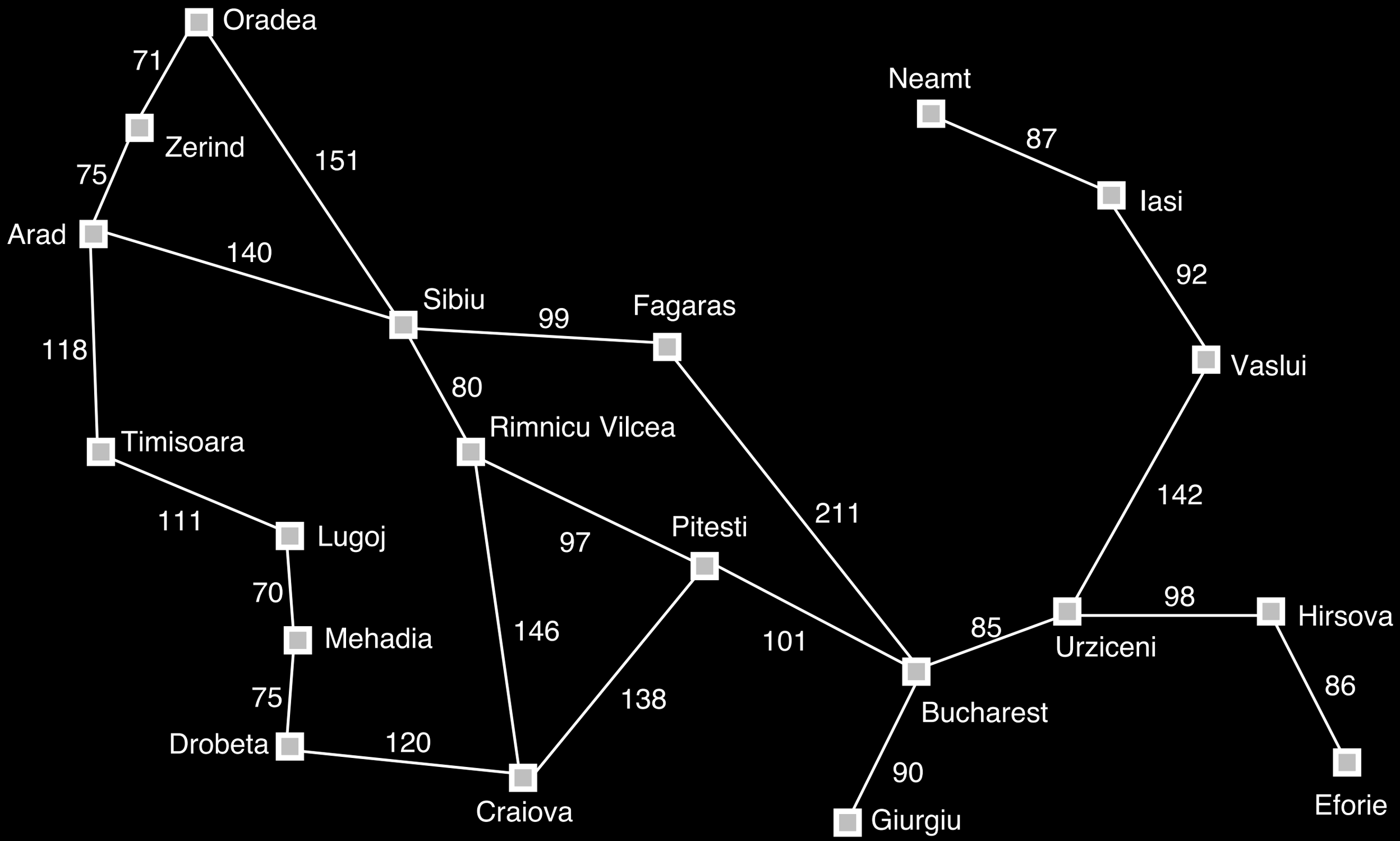

$h(n)$
n

Heuristic function

$h(n)$

Estimated cost of cheapest path from *n* to a goal node




Oradea
71
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
85
Urziceni
98
Hirsova
75
138
Bucharest
86
Drobeta
120
90
Craiova
Giurgiu
Eforie
Neamt
87
Iasi
92
Vaslui
142

h_{SLD}
(straight-line distance to Bucharest)

Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244

Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Frontier: A

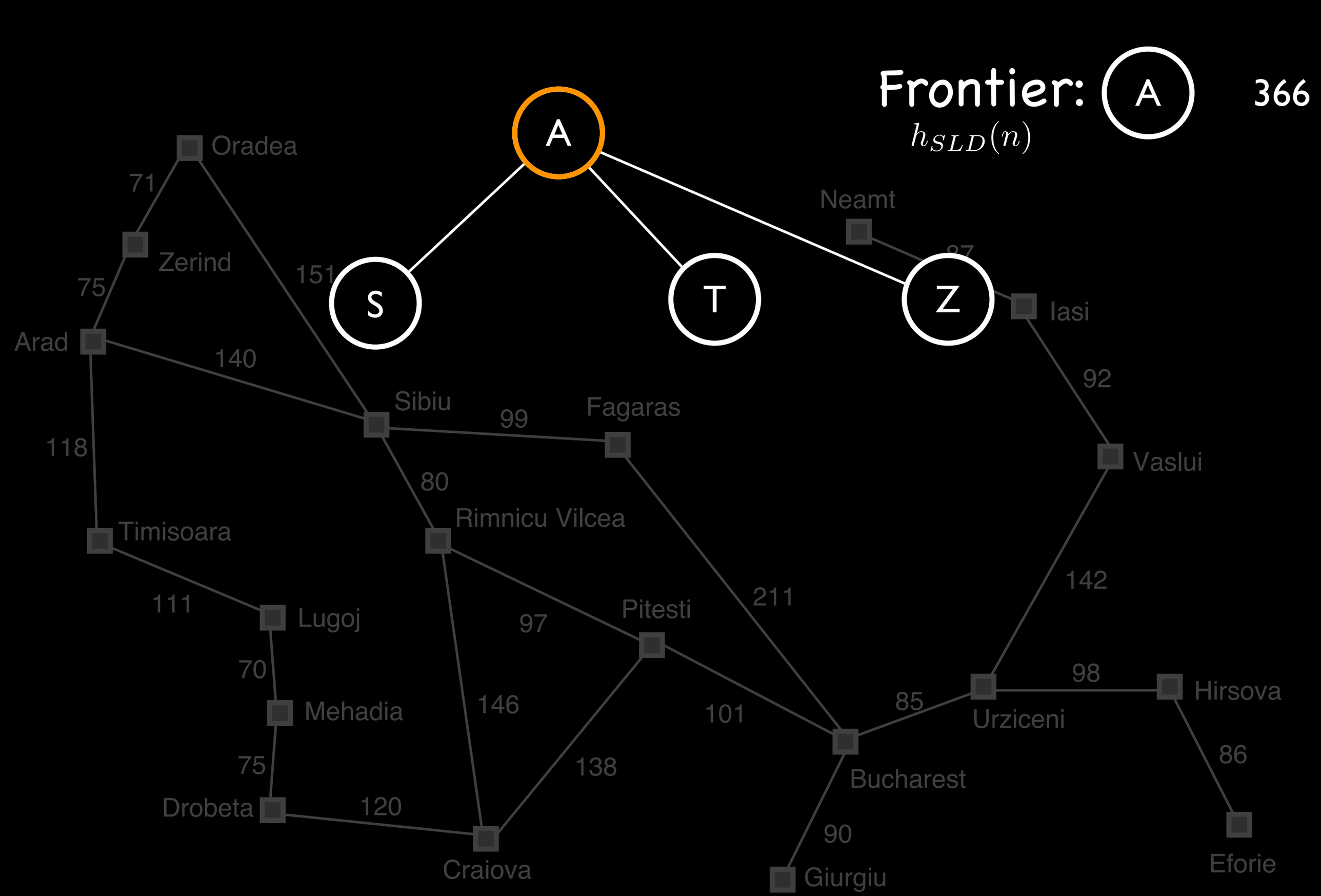
366

$h_{SLD}(n)$

Oradea
71
Zerind
151
75
Arad
140
Sibiu
99
Fagaras
118
80
Timisoara
Rimnicu Vilcea
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
98
Hirsova
85
Urziceni
86
Bucharest
90
Giurgiu
Eforie

Frontier: A 366

$h_{SLD}(n)$



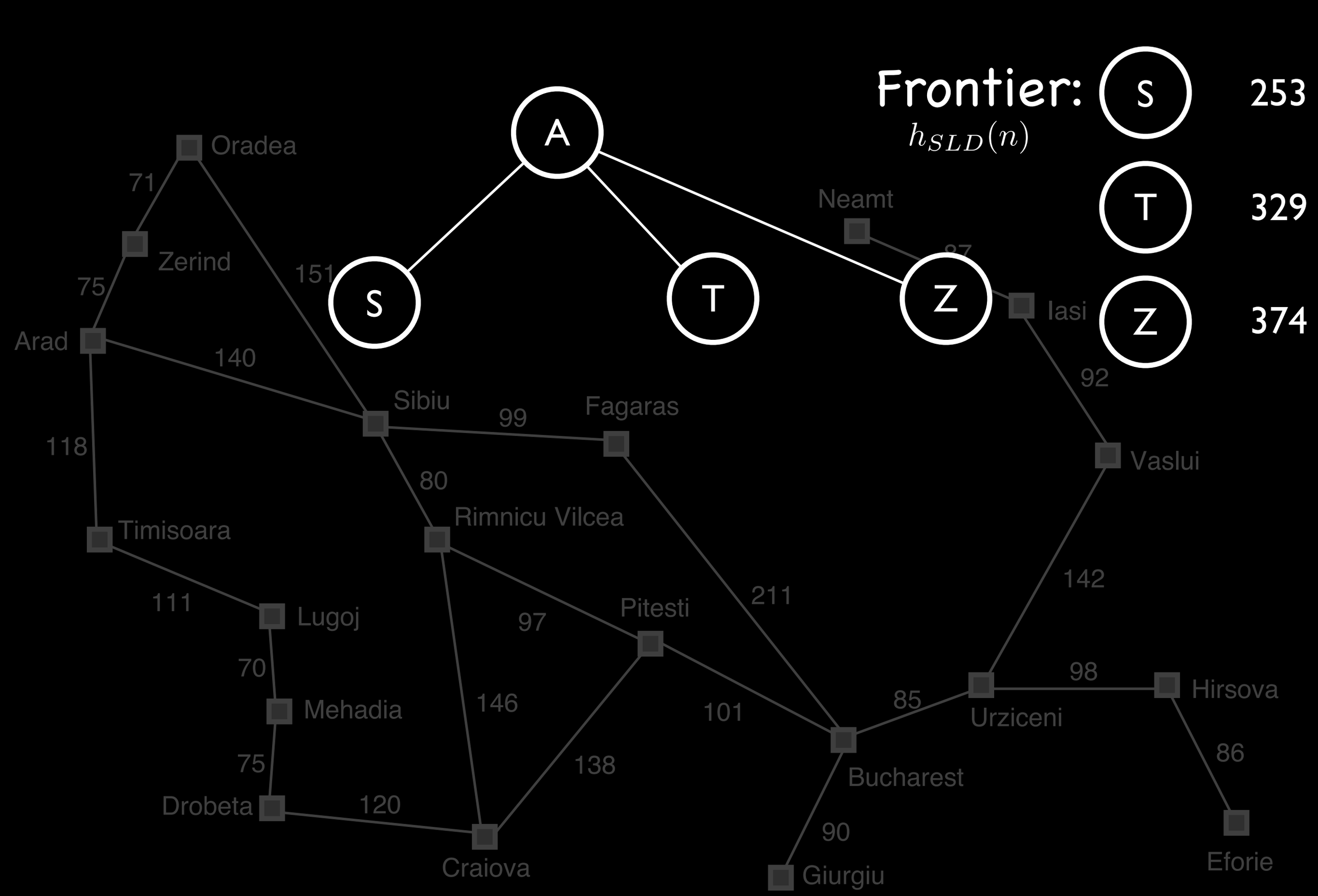
Frontier:

$h_{SLD}(n)$

S 253

T 329

Z 374



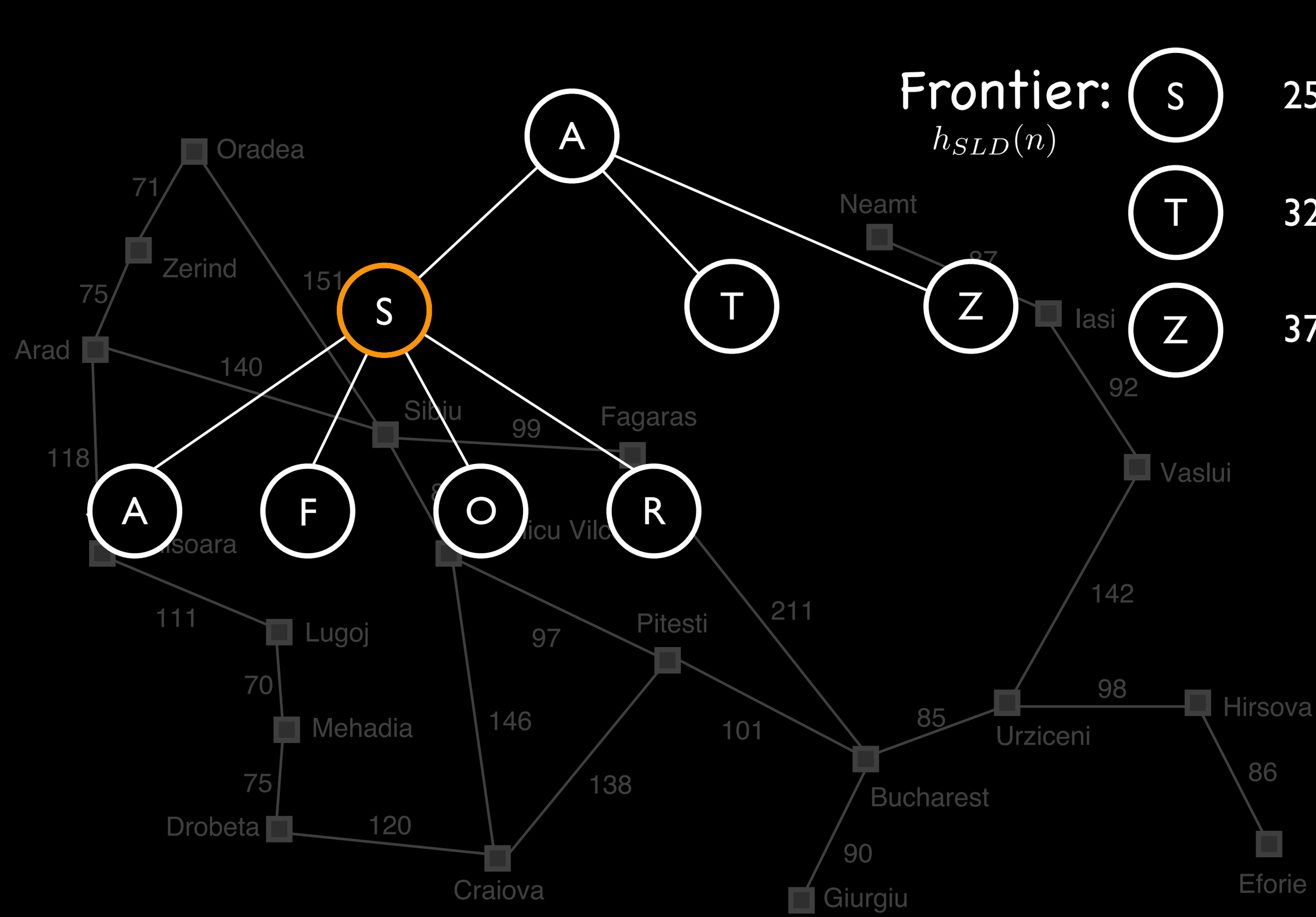
Frontier:

$h_{SLD}(n)$

S 253

T 329

Z 374



Frontier:

$h_{SLD}(n)$

F 320

R 393

A 366

A

F

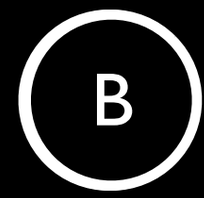
O

R

Oradea
71
Neamt
Zerind
87
75
151
S
T
Z
Iasi
Arad
140
92
Sibiu
99
Fagaras
118
Rimnicu Vilcea
Timisoara
142
211
111
Lugoj
Pitesti
97
98
O 380
Hirsova
70
Mehadia
146
101
85
Urziceni
75
138
86
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Eforie

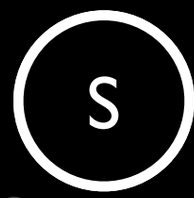
Frontier:

$h_{SLD}(n)$

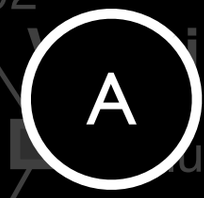


076

193



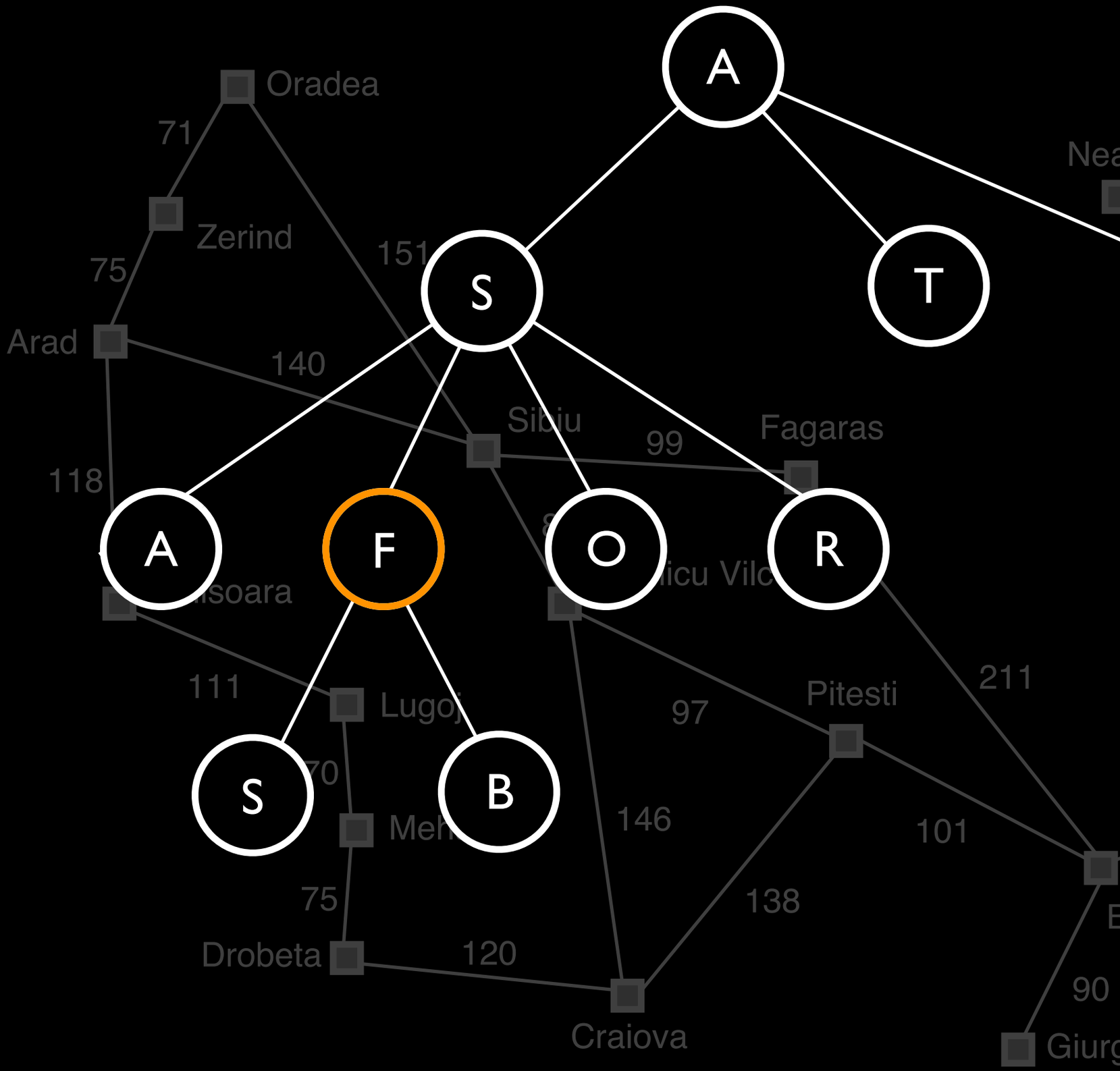
329



366

374

380



Frontier:

$h_{SLD}(n)$

B 0

R 193

S 253

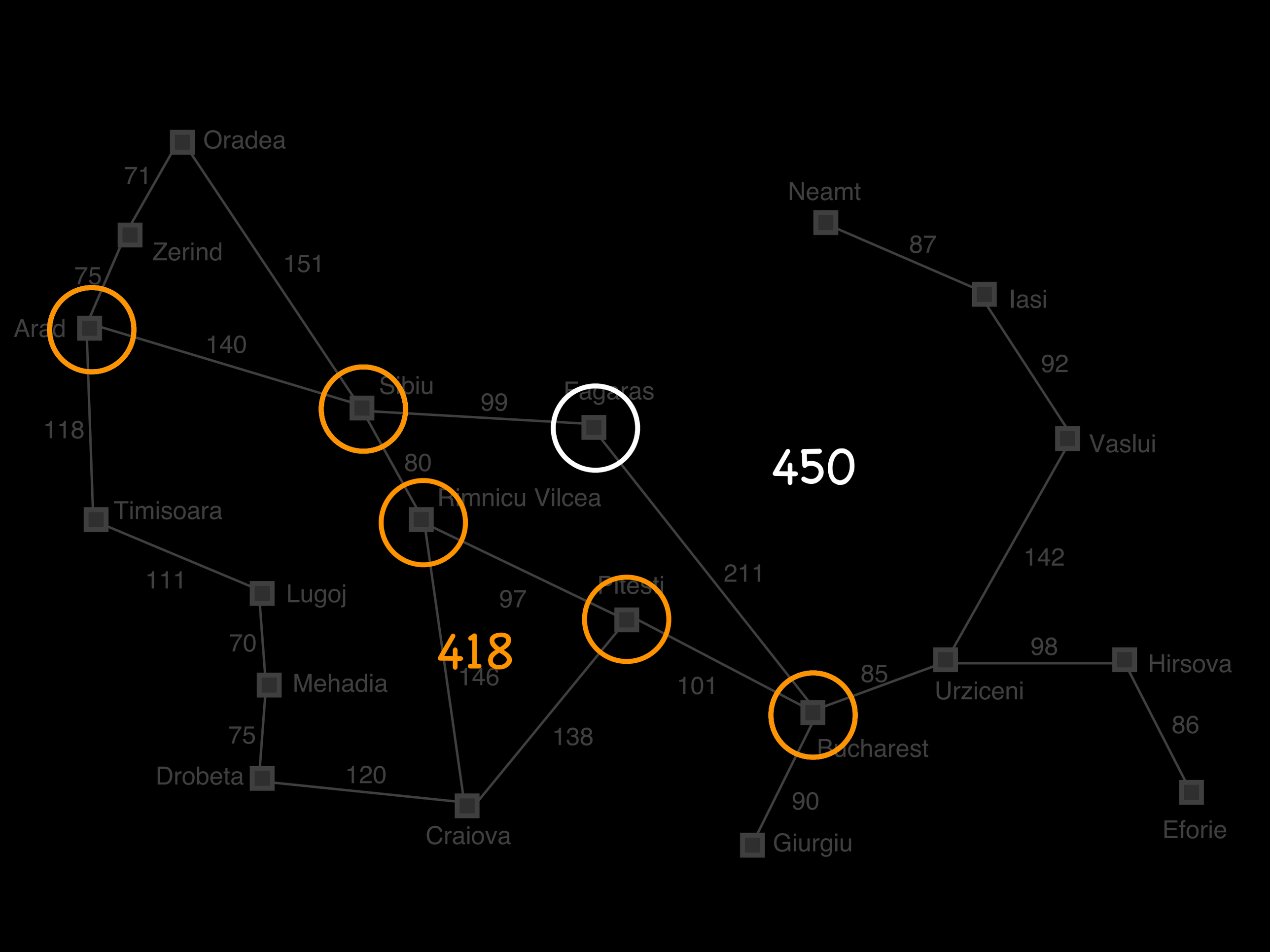
T 329

A 366

Z 374

O 380

Oradea
71
Zerind
75
Arad
151
140
Sibiu
99
Fagaras
118
Timisoara
111
Lugoj
70
Mehadia
75
Drobeta
120
Craiova
146
97
Pitesti
138
211
101
Neamt
87
Iasi
92
142
98
Hirsova
86
Eforie
85
Urziceni
Bucharest
90
Giurgiu


Oradea
71
Neamt
87
Zerind
151
Iasi
75
Arad
140
92
Sibiu
99
Fagaras
Vaslui
118
80
450
Rimnicu Vilcea
Timisoara
142
111
Lugoj
211
Pitesti
97
418
98
Hirsova
70
146
Mehadia
85
101
Urziceni
75
138
86
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Eforie

Evaluation function

$$f(n)$$

Evaluation function

$$f(n) = h(n)$$

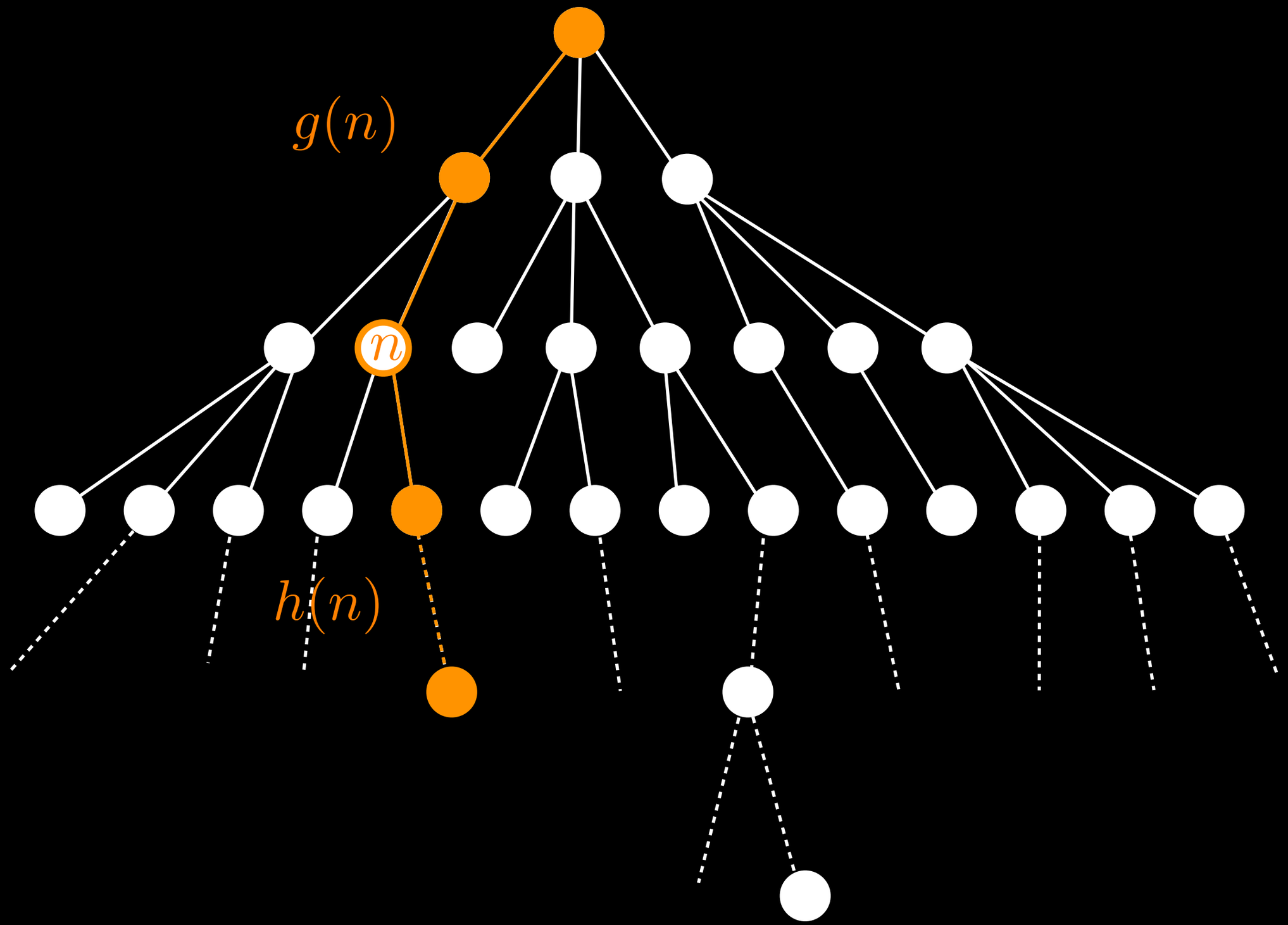
Estimated cost of cheapest path from n to a goal node

Evaluation function

$$f(n) = g(n) + h(n)$$

True cost of path from start node to node n

Estimated cost of cheapest path from n to a goal node


$g(n)$
n
$h(n)$

Evaluation function

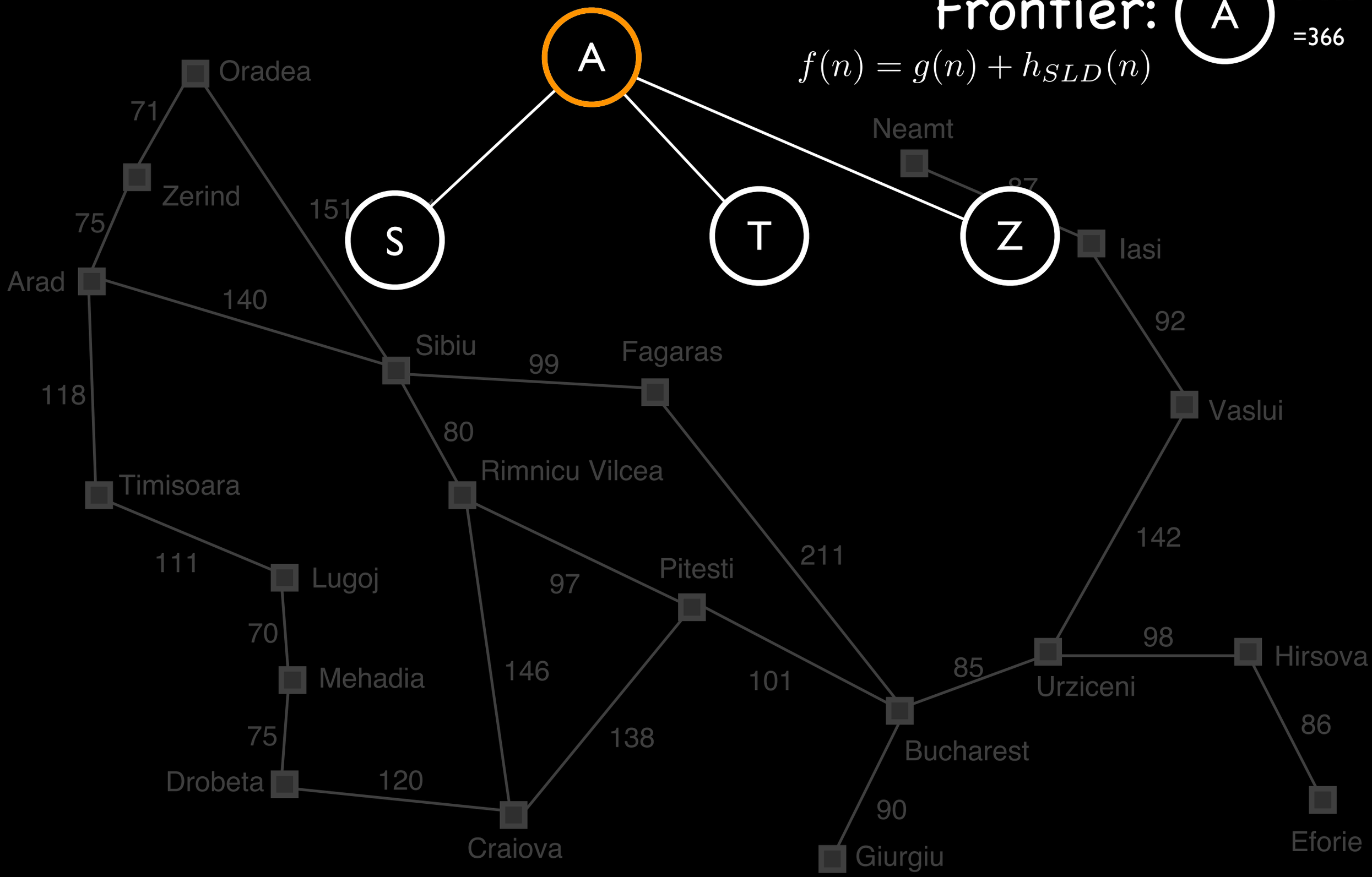
$$f(n) = g(n) + h(n)$$

$\equiv$ Estimated cost of cheapest solution through n

Frontier: A

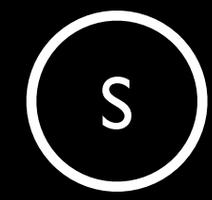
0+366
=366

$$f(n) = g(n) + h_{SLD}(n)$$

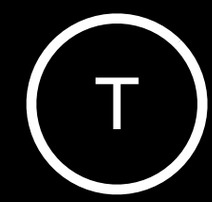


Frontier:

$$f(n) = g(n) + h_{SLD}(n)$$



140+253
= 393



118+329
= 447

75+374
=449

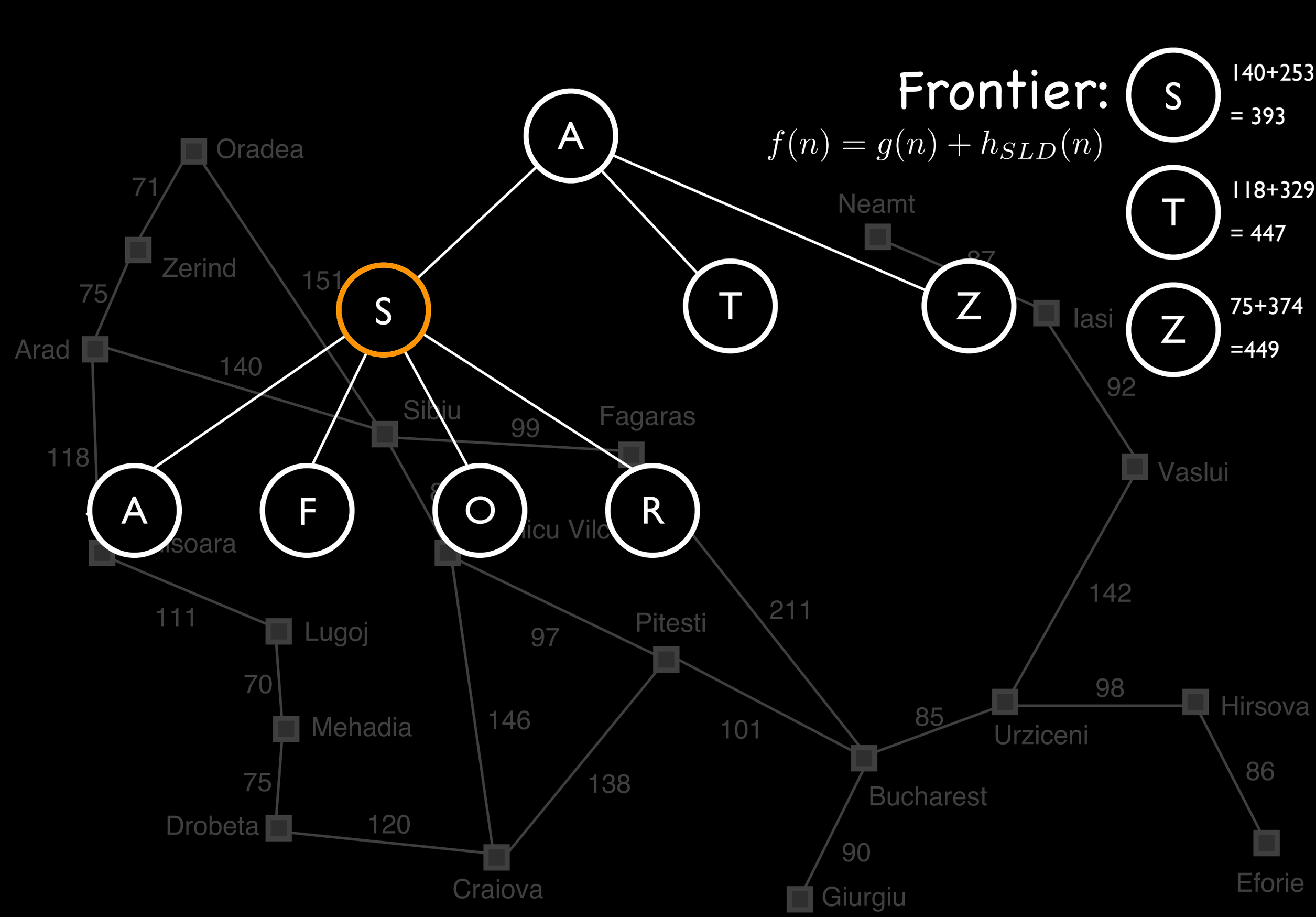
Frontier:

$$f(n) = g(n) + h_{SLD}(n)$$

S 140+253 = 393

T 118+329 = 447

Z 75+374 =449



Frontier:

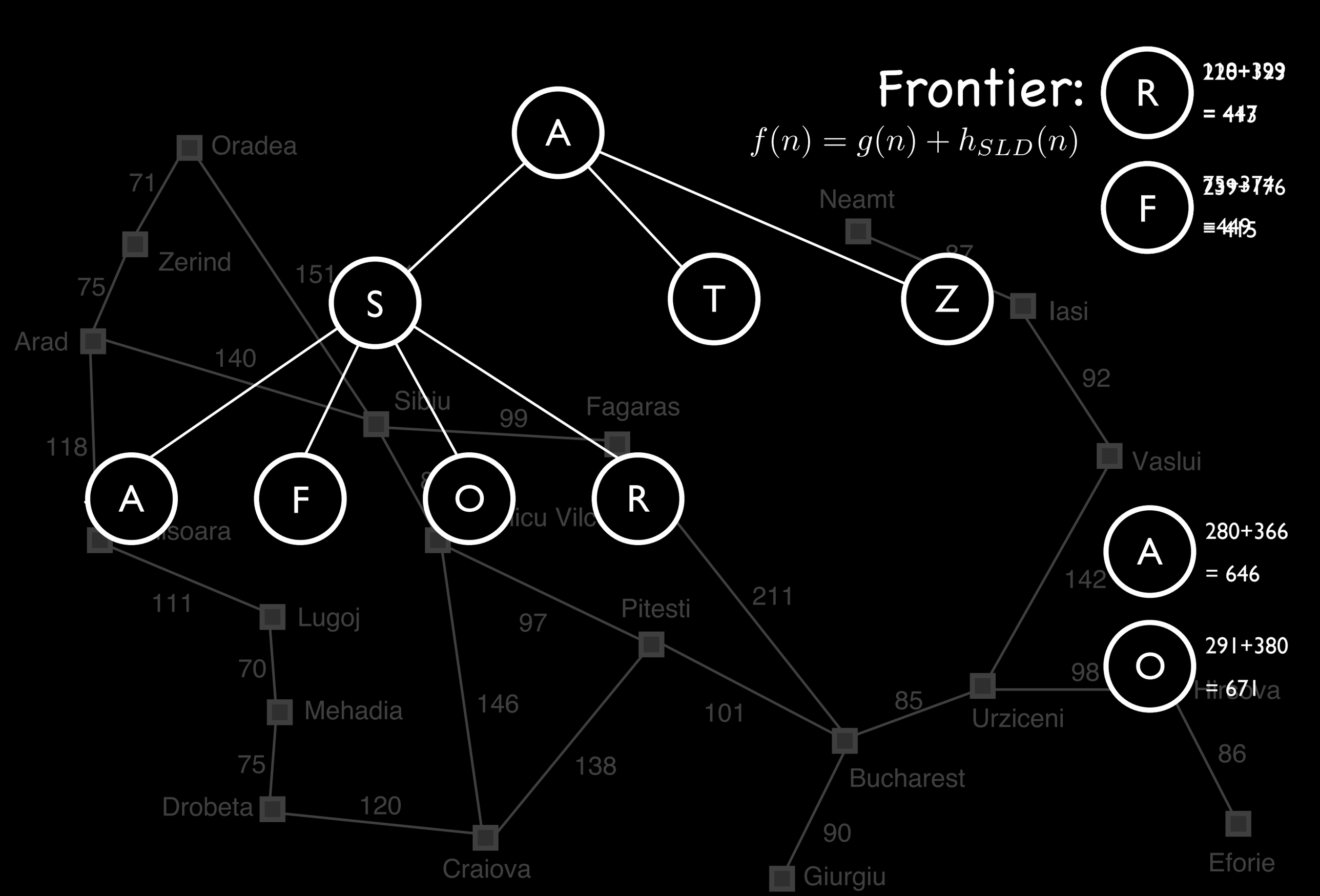
$$f(n) = g(n) + h_{SLD}(n)$$

R
220+329
= 449

F
239+176
= 415

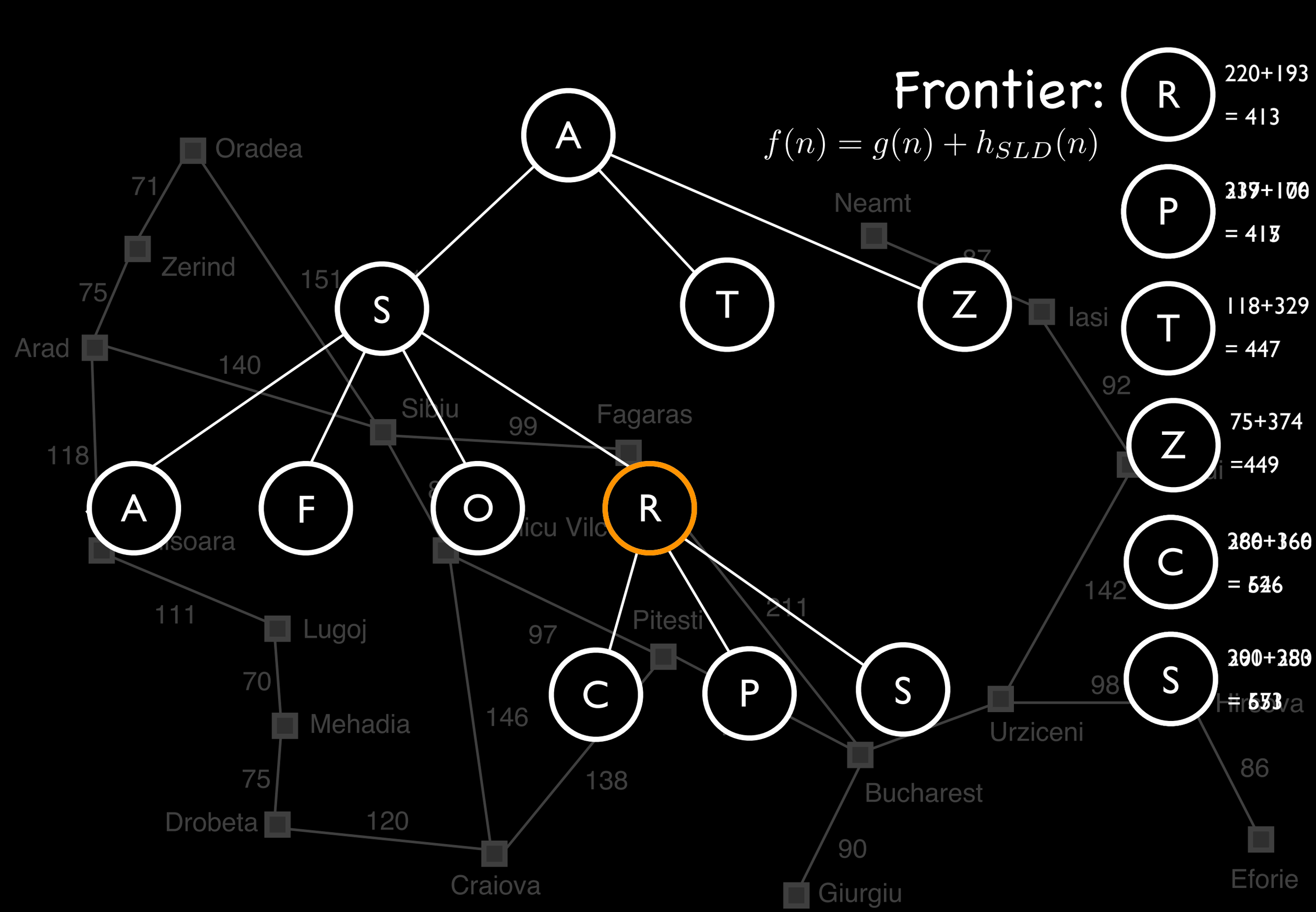
A
280+366
= 646

O
291+380
= 671



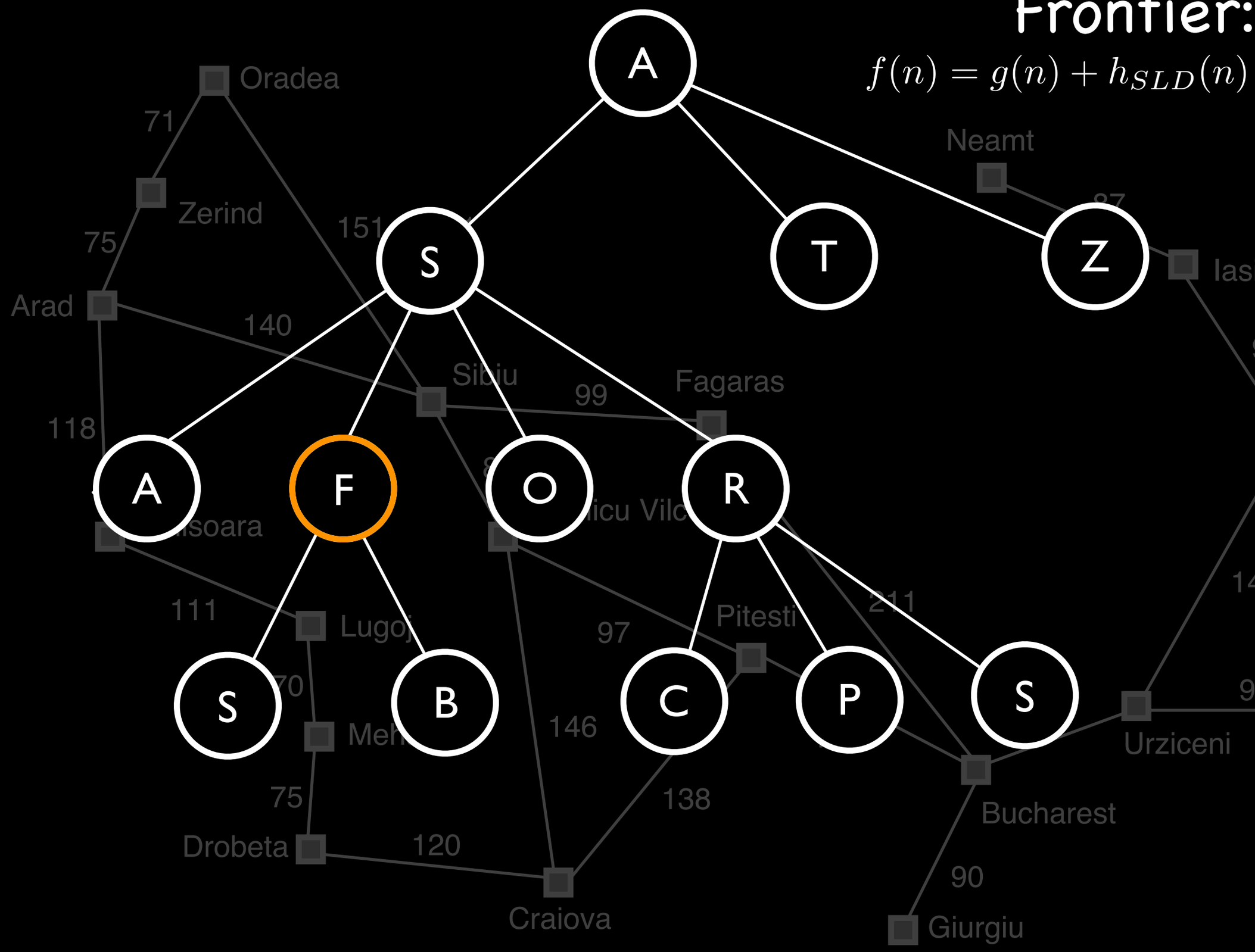
Frontier:

$$f(n) = g(n) + h_{SLD}(n)$$



Frontier:

$$f(n) = g(n) + h_{SLD}(n)$$



F 239+176 = 415

P 317+100 = 417

T 118+329 = 447

B ~~450~~ ~~374~~ ~~415~~ ~~450~~

C 366+160 = 526

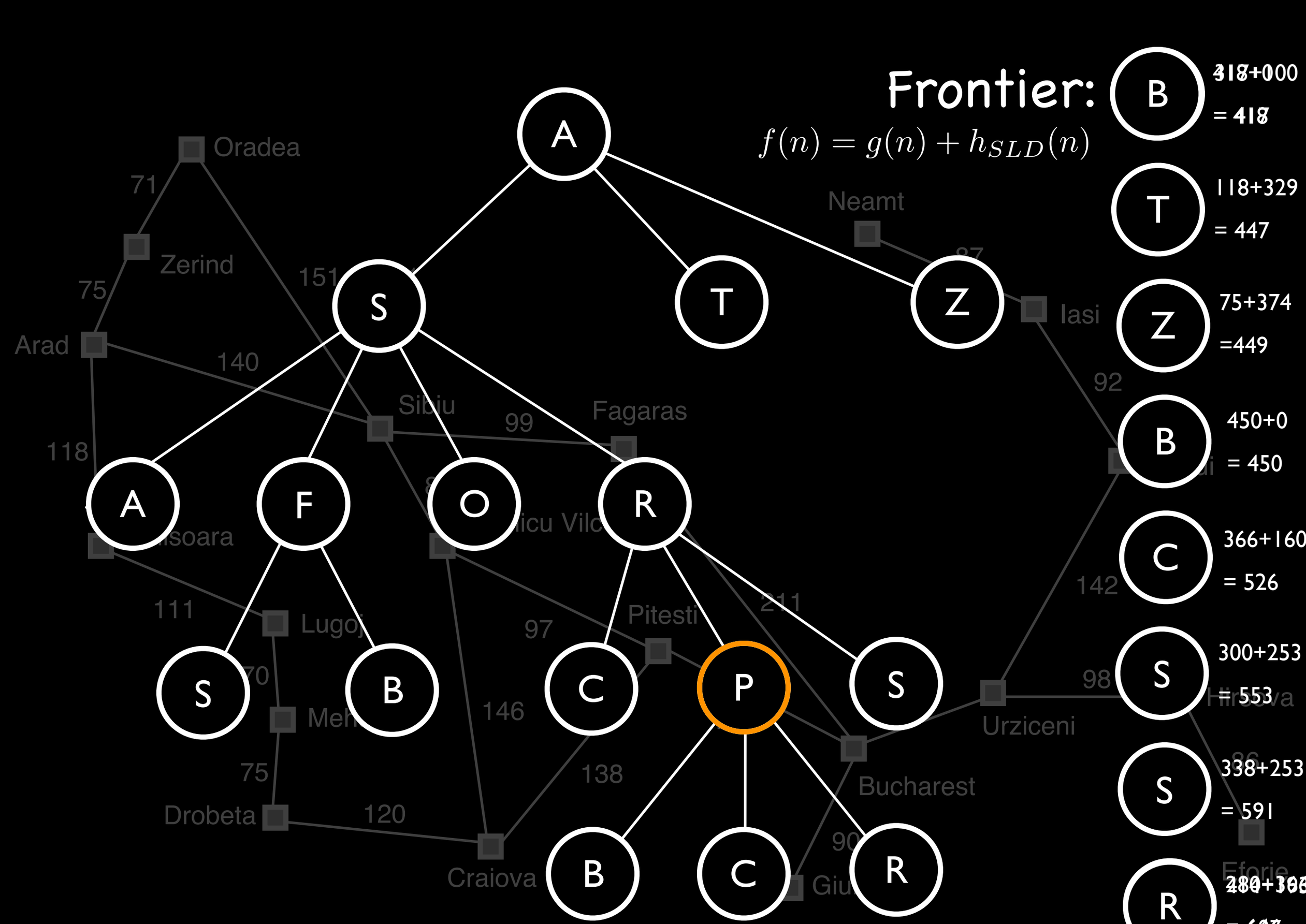
S 300+253 = 553

S ~~280+253~~ = ~~533~~ 596

O 291+380 = 671

Frontier:

$$f(n) = g(n) + h_{SLD}(n)$$



B 418+000 = 418

T 118+329 = 447

Z 75+374 =449

B 450+0 = 450

C 366+160 = 526

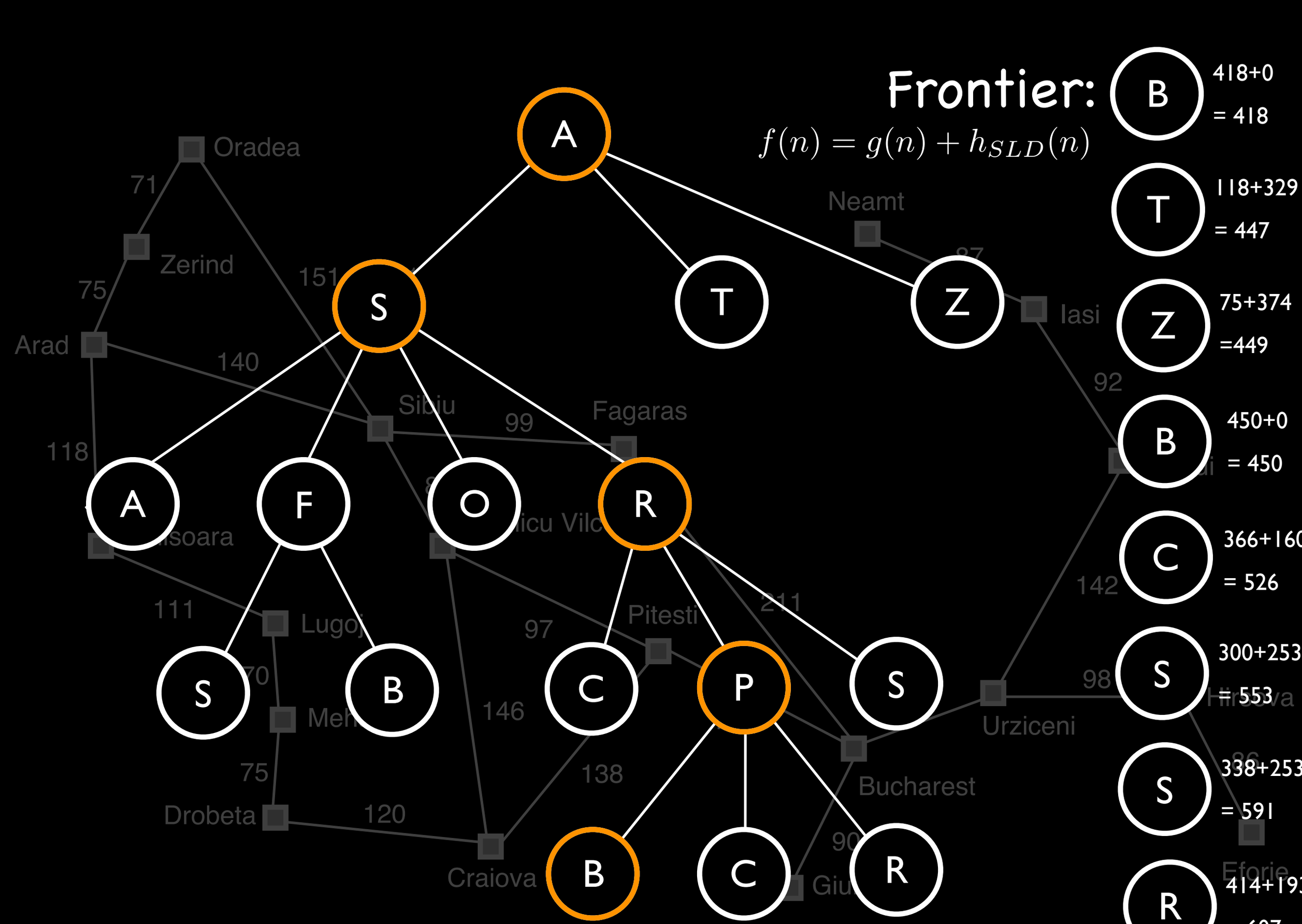
S 300+253 = 553

S 338+253 = 591

R 480+308 = 607

Frontier:

$$f(n) = g(n) + h_{SLD}(n)$$



Evaluation function

$$f(n) = g(n) + h(n)$$

True cost of path from start node to node *n*

Estimated cost of cheapest path from *n* to a goal node

If $h(n)$
is admissible

Completeness

Optimality

Never overestimates the true cost
of a solution

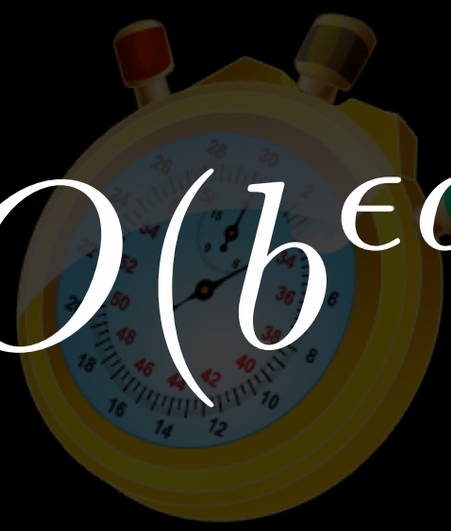
$$f(n) = g(n) + h_{SLD}(n)$$

A* SEARCH

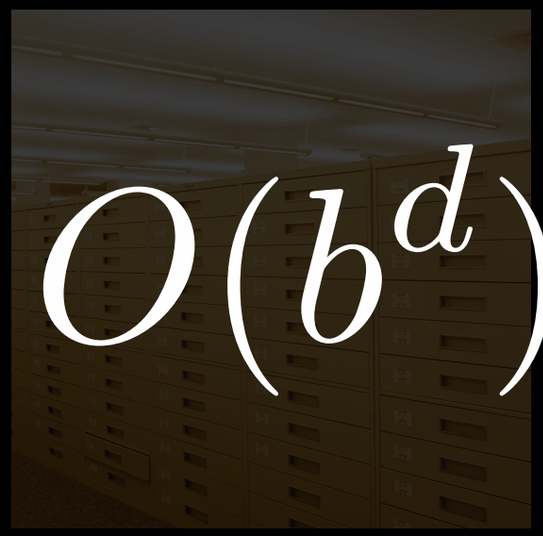
If $h(n)$ is admissible

Completeness

Optimality



$O(b^{\epsilon d})$



$O(b^{d})$

Time Complexity

Space Complexity

HEURISTIC FUNCTIONS

Where do good heuristic functions come from?

Good question...

(See Section 3.6 for some ideas)

ACKNOWLEDGEMENTS

- Thanks to Prof George Ferguson for the awesome animated slides on A*!

NEXT TIME

- Next time we'll begin discussing adversarial search.
- Check out the minimax algorithm from Chapter 5.
- We will also have an *online* quiz on monday.

A* HISTORICAL NOTES

- Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics
- Objective: derive an optimal search algorithm which expands the fewest number of nodes necessary.

SHAKEY THE ROBOT

- Circa 1966-1972
- First mobile robot capable of reasoning about its own actions.
- "We worked for a month trying to find a good name for it, ranging from Greek names to whatnot, and then one of us said, 'Hey, it shakes like hell and moves around, let's just call it Shakey.'" — Charles Rosen