# CSC242: Intro to AI

Lecture 3

Search Strategies

```
Solution graphSearch(Problem p) {
  Set<Node> frontier = new Set<Node>(p.getInitialState());
  Set<Node> explored = new Set<Node>();
  while (true) {
    if (frontier.isEmpty()) {
      return false;
    }
    Node node = frontier.selectOne();
    if (p.isGoalState(node.getState())) {
      return node.getSolution();
    }
    explored.add(node);
    for (Node n : node.expand()) {
      if (!explored.contains(n)) {
        frontier.add(n);
      }
    }
  }
}
```

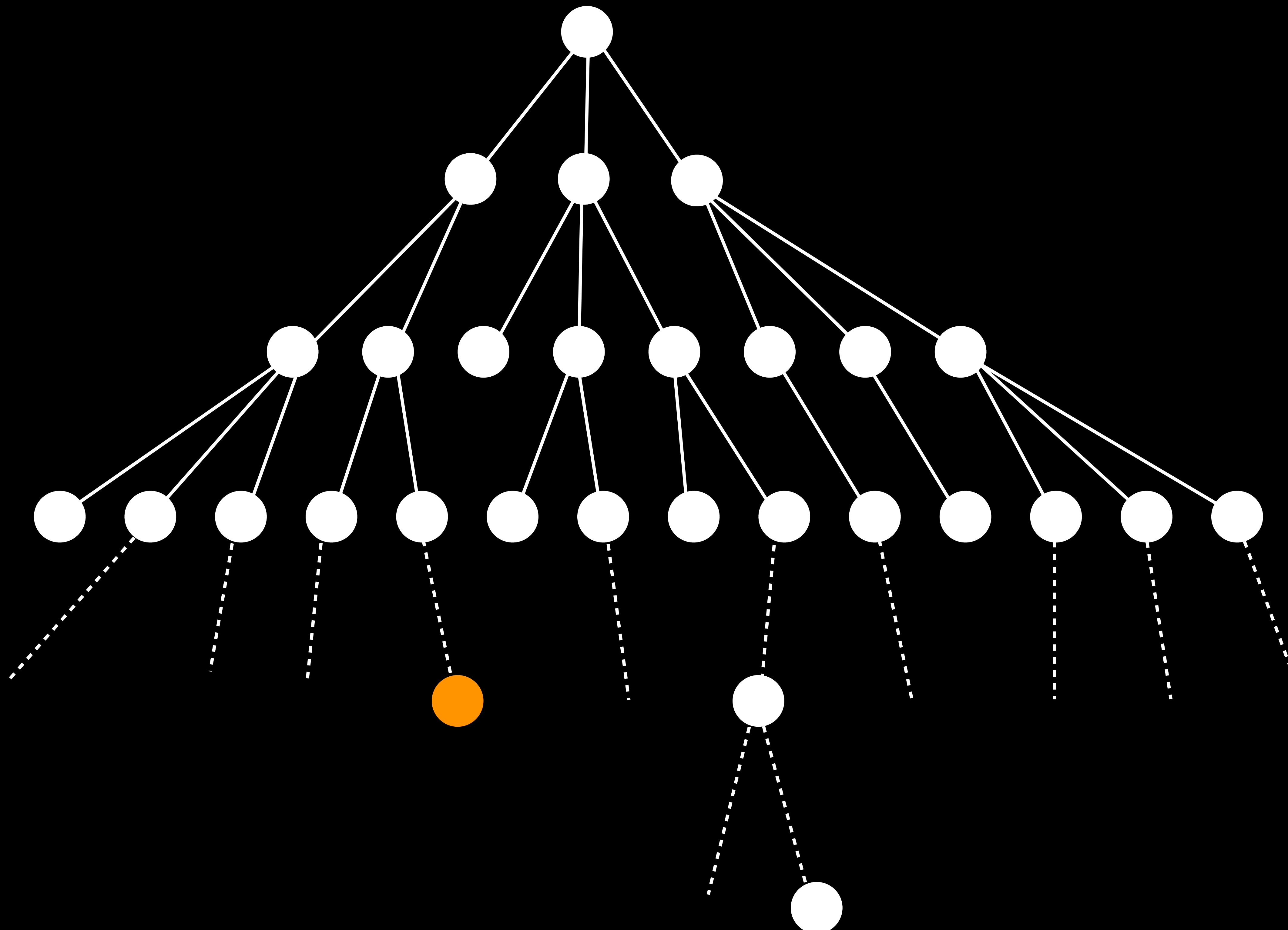
# Kinds of Strategies

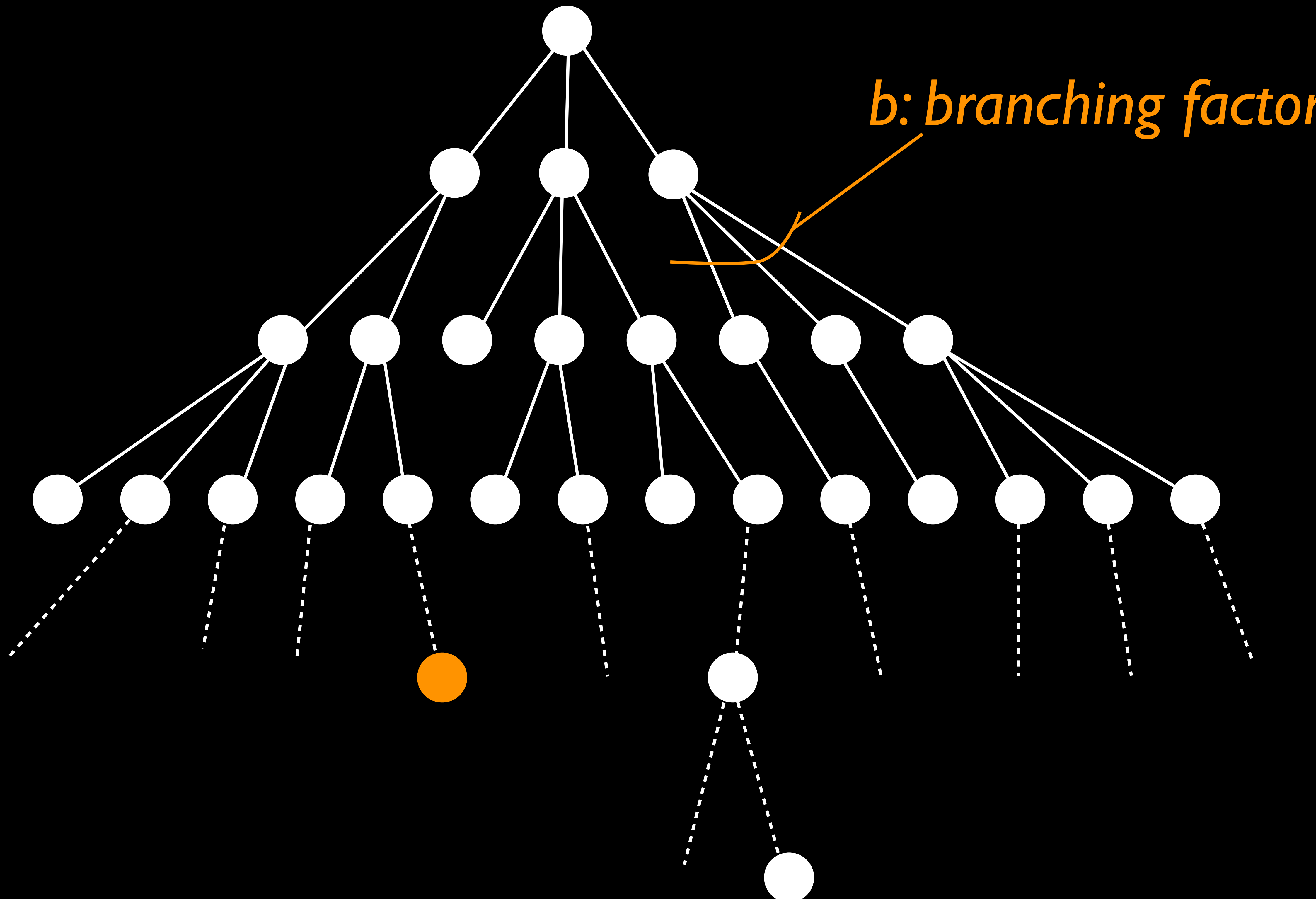
# Kinds of Strategies

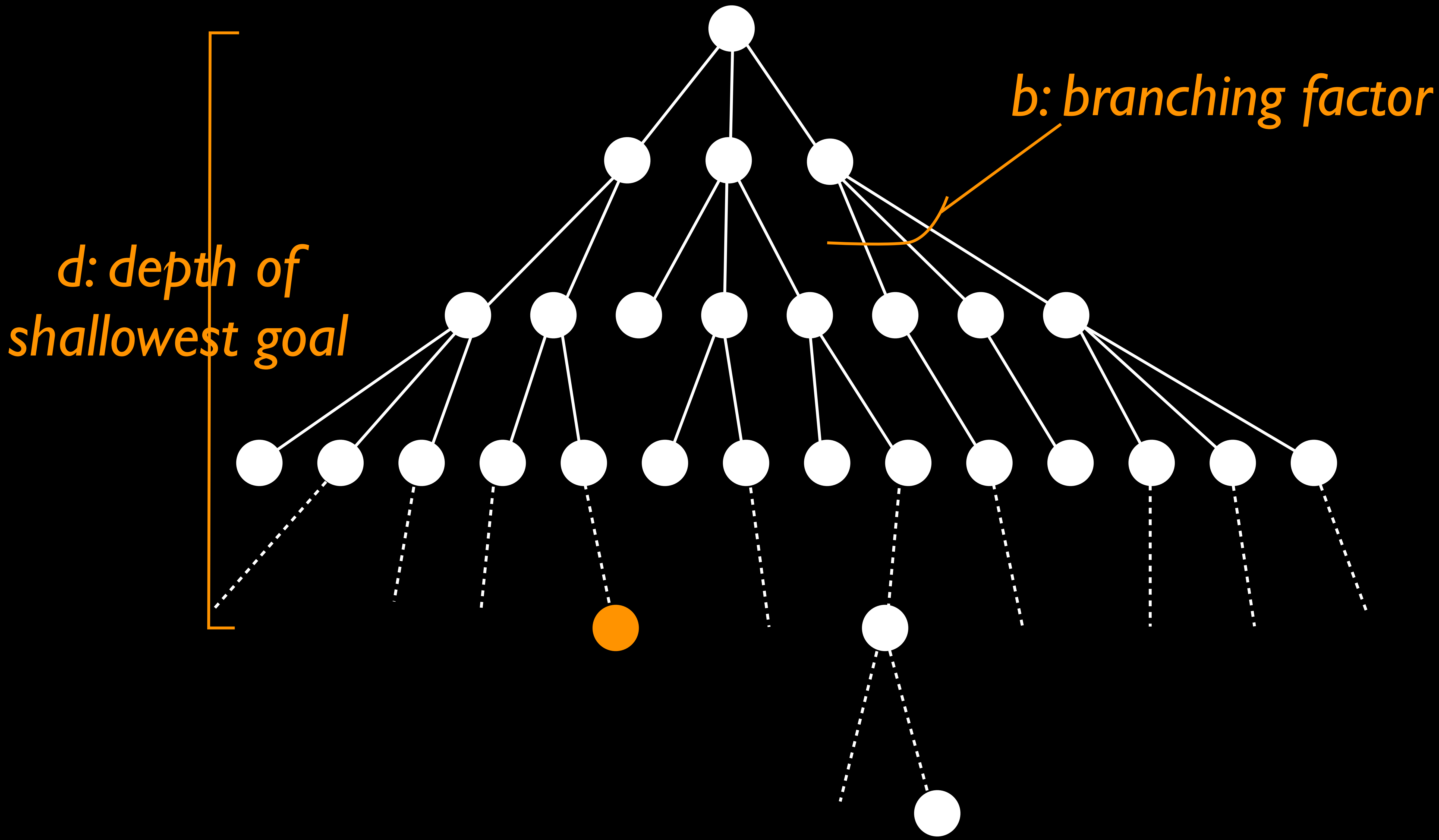
- Uninformed
  - Only considers the structure of the search space given by the transition function
  - Does not consider the state information associated with a node

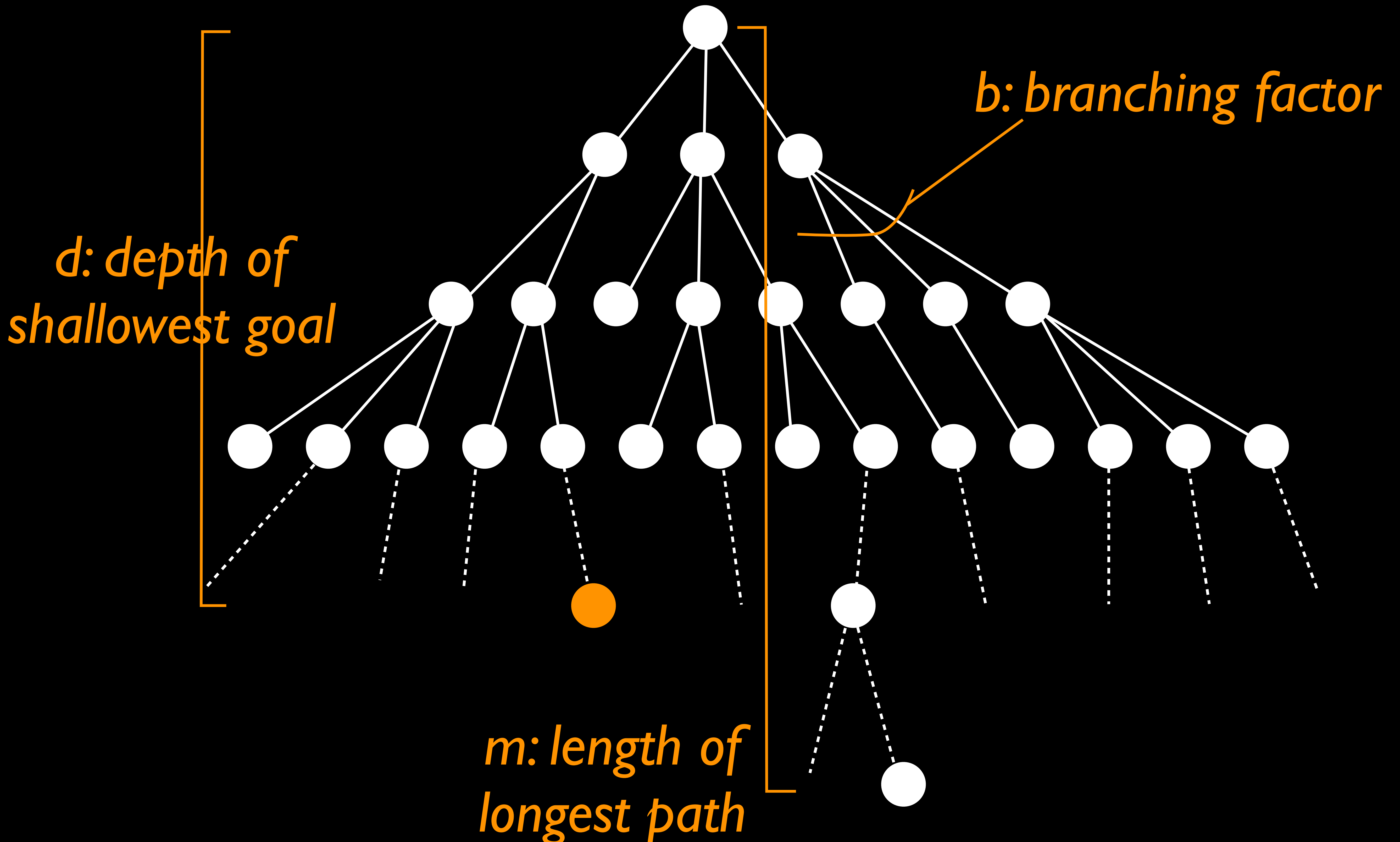
# Kinds of Strategies

- Uninformed
  - Only considers the structure of the search space given by the transition function
  - Does not consider the state information associated with a node
- Heuristic
  - Uses the state information to help select promising nodes to expand




b: branching factor


b: branching factor
d: depth of
shallowest goal


b: branching factor
d: depth of shallowest goal
m: length of longest path

# Uninformed Strategies

# Uninformed Strategies

Credit: Ben Hider/Getty Images

# $100

# $100

- A kind of uninformed search that
  - Is guaranteed to find a goal state if one exists, and the branching factor is finite
  - Is guaranteed to find a goal that has smallest depth

# $100

- A kind of uninformed search that
  - Is guaranteed to find a goal state if one exists, and the branching factor is finite
  - Is guaranteed to find a goal that has smallest depth
- What is breadth-first search?

# $200

# $200

- A data structure used to represent the frontier in breadth-first search that allows "select_one" to be performed in constant time.

# $200

- A data structure used to represent the frontier in breadth-first search that allows “select_one” to be performed in constant time.
- What is a FIFO (first-in first-out) queue?

# $300

# $300

- The time complexity of breadth first-search, where b is the maximum branching factor and d is the depth of the shallowest goal

# $300

- The time complexity of breadth first-search, where b is the maximum branching factor and d is the depth of the shallowest goal
- What is $O(b^d)$?

# $400

# $400

- The space complexity of breadth first-search, where b is the maximum branching factor and d is the depth of the shallowest goal

# $400

- The space complexity of breadth first-search, where b is the maximum branching factor and d is the depth of the shallowest goal
- What is $O(b^d)$?

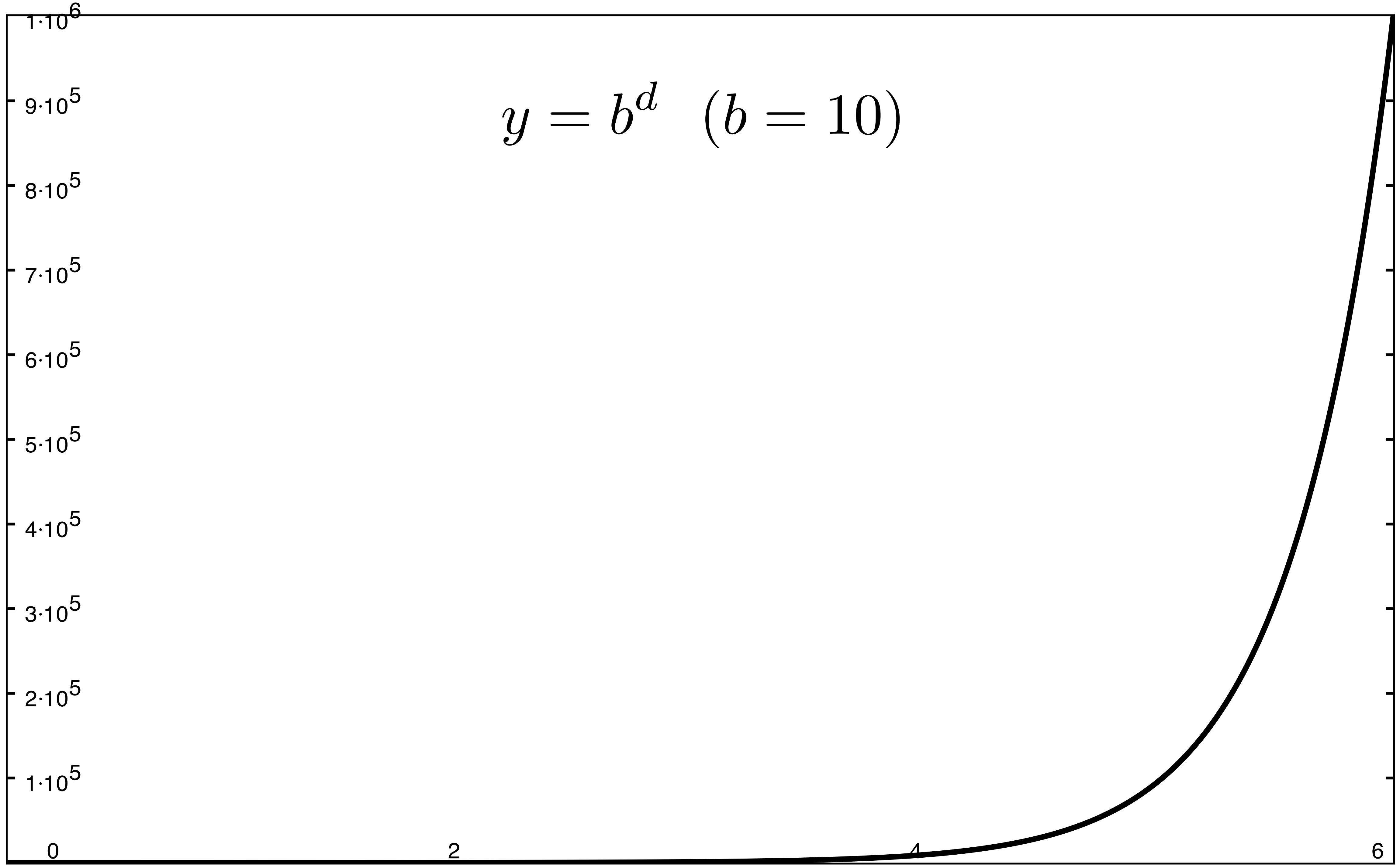
# $500

# $500

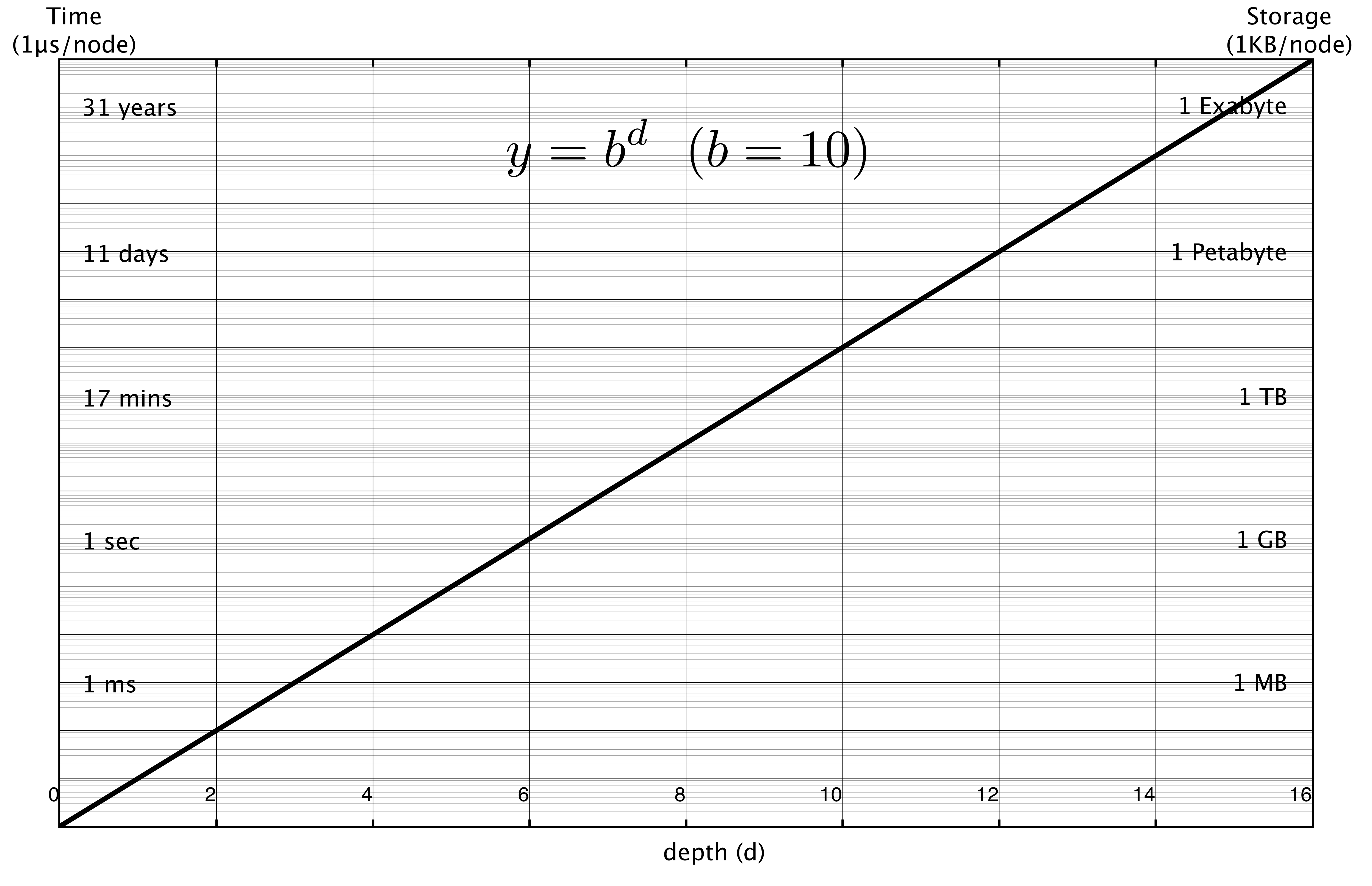
- If the state space has this property, then breadth-first search may fail to find a solution, even when one exists

# $500

- If the state space has this property, then breadth-first search may fail to find a solution, even when one exists
- What is having an infinite branching factor?



$y = b^d \quad (b = 10)$
$1{\cdot}10^6$
$9{\cdot}10^5$
$8{\cdot}10^5$
$7{\cdot}10^5$
$6{\cdot}10^5$
$5{\cdot}10^5$
$4{\cdot}10^5$
$3{\cdot}10^5$
$2{\cdot}10^5$
$1{\cdot}10^5$
0
2
4
6



Time
(1μs/node)
Storage
(1KB/node)
31 years
1 Exabyte
$y = b^d \quad (b = 10)$
11 days
1 Petabyte
17 mins
1 TB
1 sec
1 GB
1 ms
1 MB
0
2
4
6
8
10
12
14
16
depth (d)

# $100

# $100

- A kind of uninformed search that
  - Is guaranteed to find a goal state only if the maximum depth of the tree (m) is finite
  - The maximum size of the frontier is mb, where b is the maximum branching factor

# $100

- A kind of uninformed search that
  - Is guaranteed to find a goal state only if the maximum depth of the tree (m) is finite
  - The maximum size of the frontier is mb, where b is the maximum branching factor
- What is depth-first search?

# $200

# $200

- A data structure used to represent the frontier in depth-first search that allows "select_one" to be performed in constant time.

# $200

- A data structure used to represent the frontier in depth-first search that allows "select_one" to be performed in constant time.
- What is a stack (or what is a LIFO last-in first-out queue)?

# $300

# $300

- The time complexity of depth first-search, where b is the branching factor, d is the depth of the shallowest goal, and m is the maximum depth of node

# $300

- The time complexity of depth first-search, where b is the branching factor, d is the depth of the shallowest goal, and m is the maximum depth of node
- What is $O(b^m)$?

# $400

# $400

- You are usually off performing tree depth first search instead of graph depth first search because this data structure can grow to size $b^m$

# $400

- You are usually off performing tree depth first search instead of graph depth first search because this data structure can grow to size $b^m$
- What is the explored list?

```
Solution graphSearch(Problem p) {
  Set<Node> frontier = new Set<Node>(p.getInitialState());
  Set<Node> explored = new Set<Node>();
  while (true) {
    if (frontier.isEmpty()) {
      return false;
    }
    Node node = frontier.selectOne();
    if (p.isGoalState(node.getState())) {
      return node.getSolution();
    }
    explored.add(node);
    for (Node n : node.expand()) {
      if (!explored.contains(n)) {
        frontier.add(n);
      }
    }
  }
}
```

```
Solution graphSearch(Problem p) {
  Set<Node> frontier = new Set<Node>(p.getInitialState());

  while (true) {
    if (frontier.isEmpty()) {
      return false;
    }
    Node node = frontier.selectOne();
    if (p.isGoalState(node.getState())) {
      return node.getSolution();
    }

    for (Node n : node.expand()) {

        frontier.add(n);

    }
  }
}
```

# $500

# $500

- This simple trick enables breadth-first search to be complete, even if the maximum depth m is infinite

# $500

- This simple trick enables breadth-first search to be complete, even if the maximum depth m is infinite
- What is iterative deepening?

# Iterative Deepening

# Iterative Deepening

For d = 1, 2, 3, ...

    Do DFS to depth d

Until a goal is found.

| Depth | Number of nodes expanded |
|---|---|
| 1 | $b$ |
| 2 | $b + b^2$ |
| 3 | $b + b^2 + b^3$ |
| $d$ | $b + b^2 + b^3 + \cdots b^d$ |

| Depth | Number of nodes expanded |
|---|---|
| 1 | $b$ |
| 2 | $b + b^2$ |
| 3 | $b + b^2 + b^3$ |
| $d$ | $b + b^2 + b^3 + \cdots b^d$ |

$$(d)b + (d-1)b^2 + (d-2)b^3 + \cdots + (1)b^d$$

| Depth | Number of nodes expanded |
|---|---|
| 1 | $b$ |
| 2 | $b + b^2$ |
| 3 | $b + b^2 + b^3$ |
| $d$ | $b + b^2 + b^3 + \cdots b^d$ |

$$(d)b + (d-1)b^2 + (d-2)b^3 + \cdots + (1)b^d = O(b^d)$$

# Uninformed Strategies

| | BFS | DFS (tree) | IDS (tree) |
|---|---|---|---|
| Complete? | ✓ | ✗ | ✓ |
| Optimal? | ✓ | ✗ | ✓ |
| Time | $O(b^d)$ | $O(b^m)$ | $O(b^d)$ |
| Space | $O(b^d)$ | $O(bm)$ | $O(bd)$ |

* If step costs are identical (see book)

# Daily Double

# Daily Double

- Replacing the queue in breadth-first search with this data structure enables it find a shortest path to a goal where edges have weights

# Daily Double

- Replacing the queue in breadth-first search with this data structure enables it find a shortest path to a goal where edges have weights
- What is a heap, where the key is the cost of the path from the start node?

```
Solution graphSearch(Problem p) {
  Set<Node> frontier = new Set<Node>(p.getInitialState());
  Set<Node> explored = new Set<Node>();
  while (true) {
    if (frontier.isEmpty()) {
      return false;
    }
    Node node = frontier.selectOne();
    if (p.isGoalState(node.getState())) {
      return node.getSolution();
    }
    explored.add(node);
    for (Node n : node.expand()) {
      if (!explored.contains(n)) {
        frontier.add(n);
      }
    }
  }
}
```

```
Solution graphSearch(Problem p) {
  Set<Node> frontier = new Heap<Node>(p.getInitialState(), 0);
  Set<Node> explored = new Set<Node>();
  while (true) {
   if (frontier.isEmpty()) {
      return false;
   }
   node = frontier.selectOne();
   pathcost = node.cost;
   if (p.isGoalState(node.getState())) {
      return node.getSolution()
   }
   explored.add(node);
   for ((n, c) in node.expand()) {
      newcost = pathcost + c;
     if (!explored.contains(n) OR newcost < n.cost){
         n.cost = newcost
         if (frontier.contains(n))
            frontier.changekey(n, newcost)
         else frontier.add(n, newcost)
     }
    }
  }
}
```

# AIMA

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

# Heuristic Strategies

heuristic |hyoōˈristik|
adjective

1. Enabling a person to discover or learn something for themselves : *a "hands-on" or interactive heuristic approach to learning*.
2. Computing proceeding to a solution by trial and error or by rules that are only loosely defined.

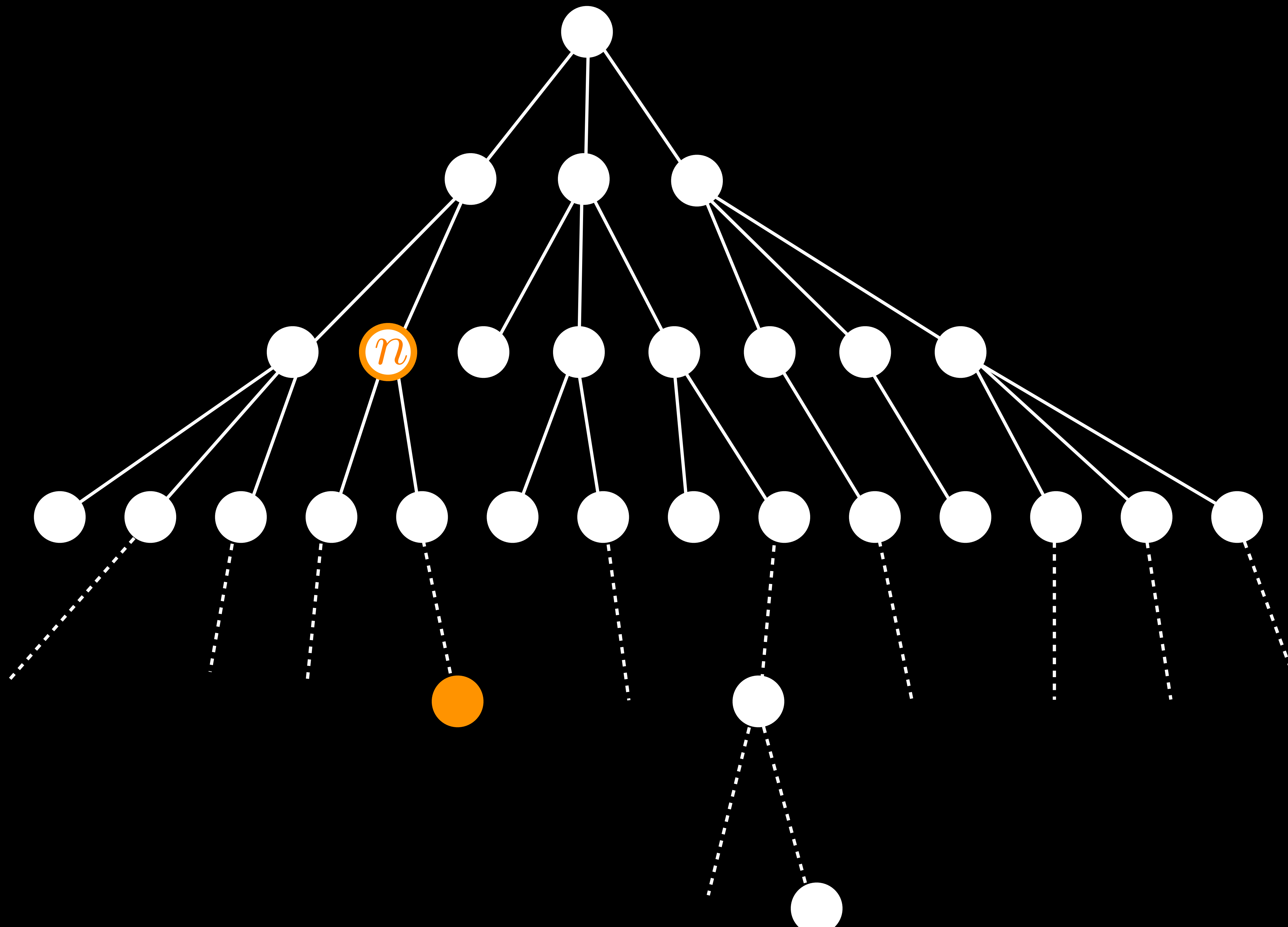
ORIGIN early 19th cent.: formed irregularly from Greek ***heuriskein 'find'***

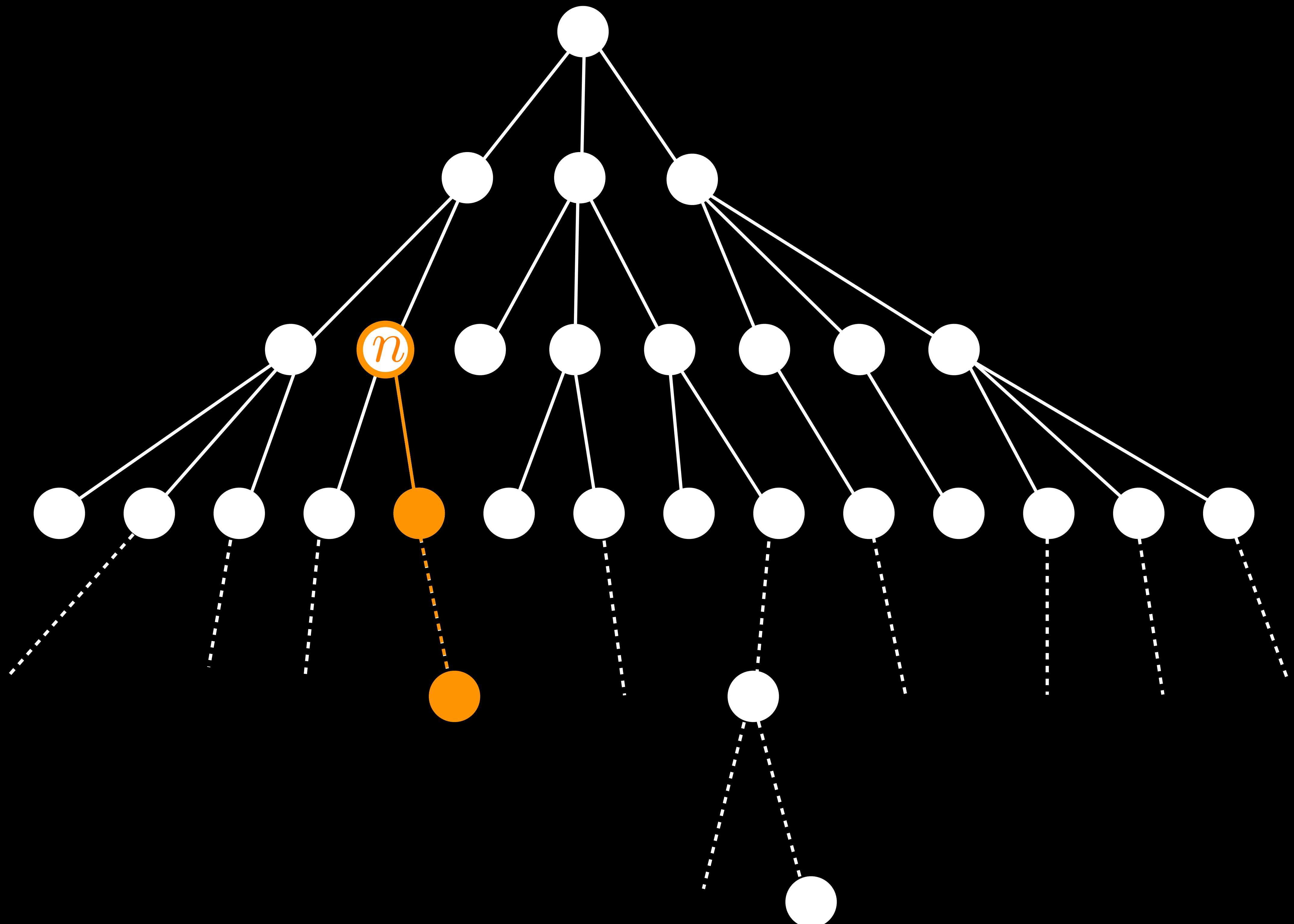
```
Solution graphSearch(Problem p) {
  Set<Node> frontier = new Set<Node>(p.getInitialState());
  Set<Node> explored = new Set<Node>();
  while (true) {
    if (frontier.isEmpty()) {
      return false;
    }
    Node node = frontier.selectOne();
    if (p.isGoalState(node.getState())) {
      return n.getSolution();
    }
    explored.add(node);
    for (Node n : node.expand()) {
      if (!explored.contains(n)) {
        frontier.add(n);
      }
    }
  }
}
```

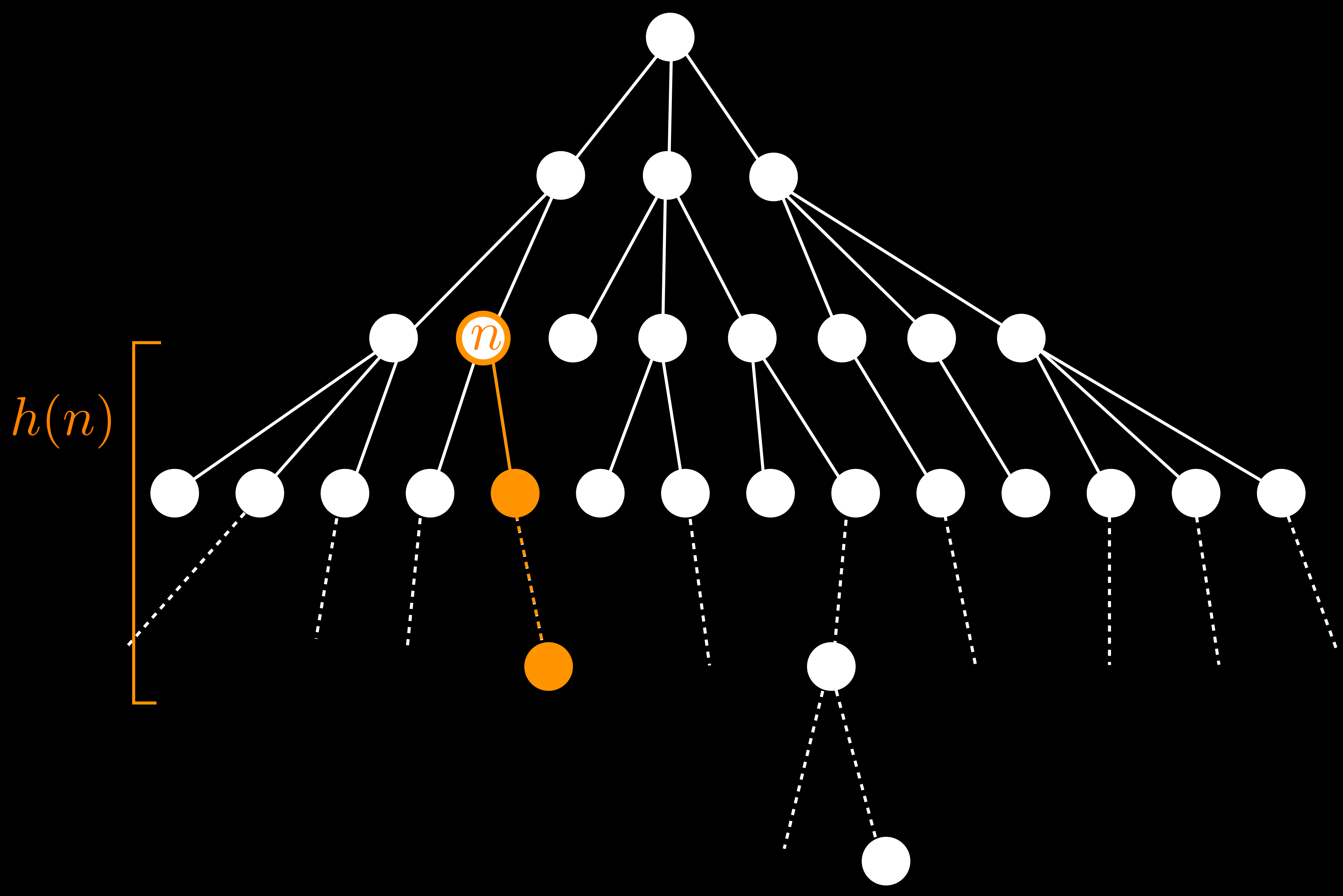
# Heuristic Function

$$h(n)$$

Estimated cost of cheapest path from n to a goal node


n


n



$h(n)$
$n$

Heuristic function

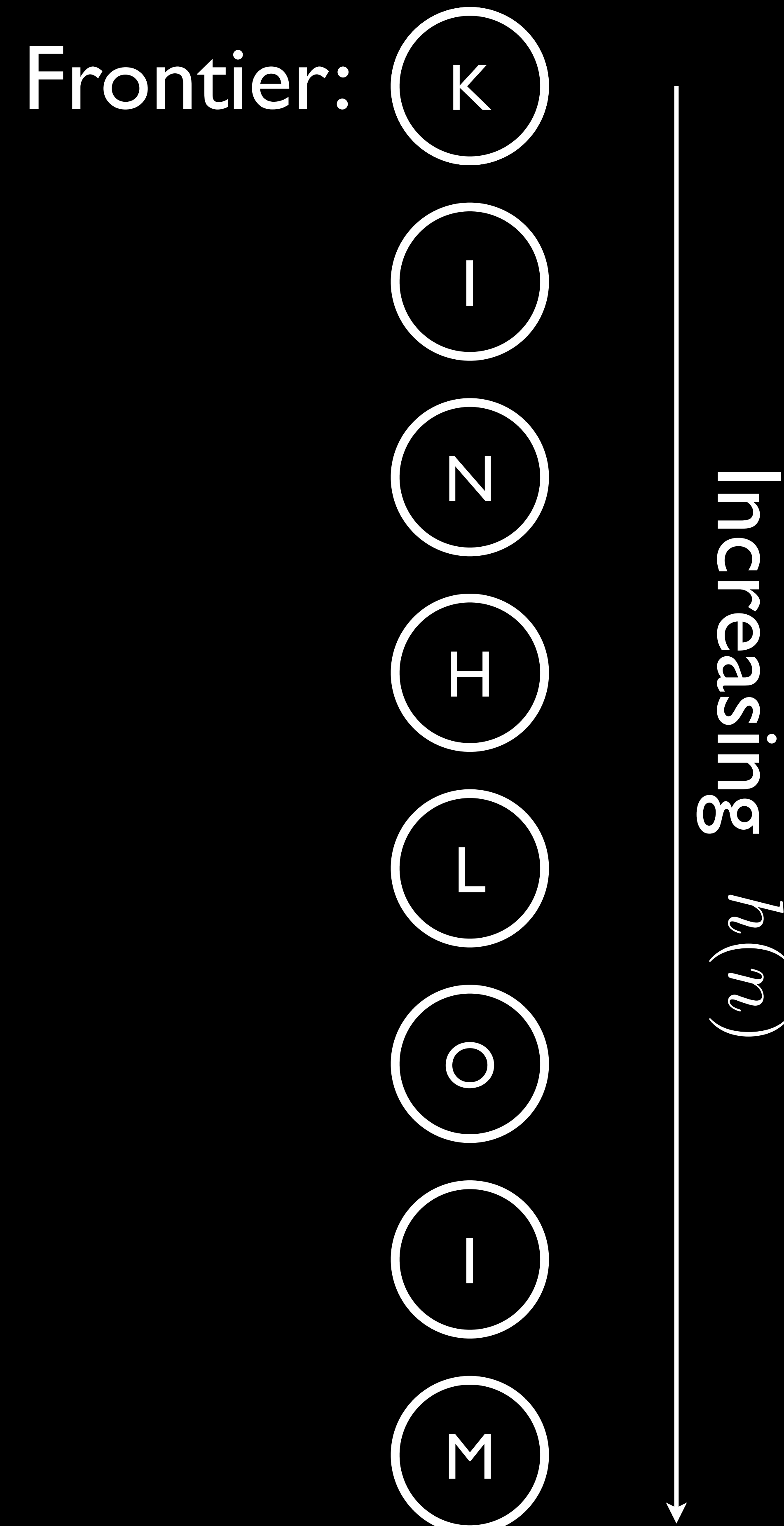
$$h(n)$$

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

Heuristic function

$$h(n)$$

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

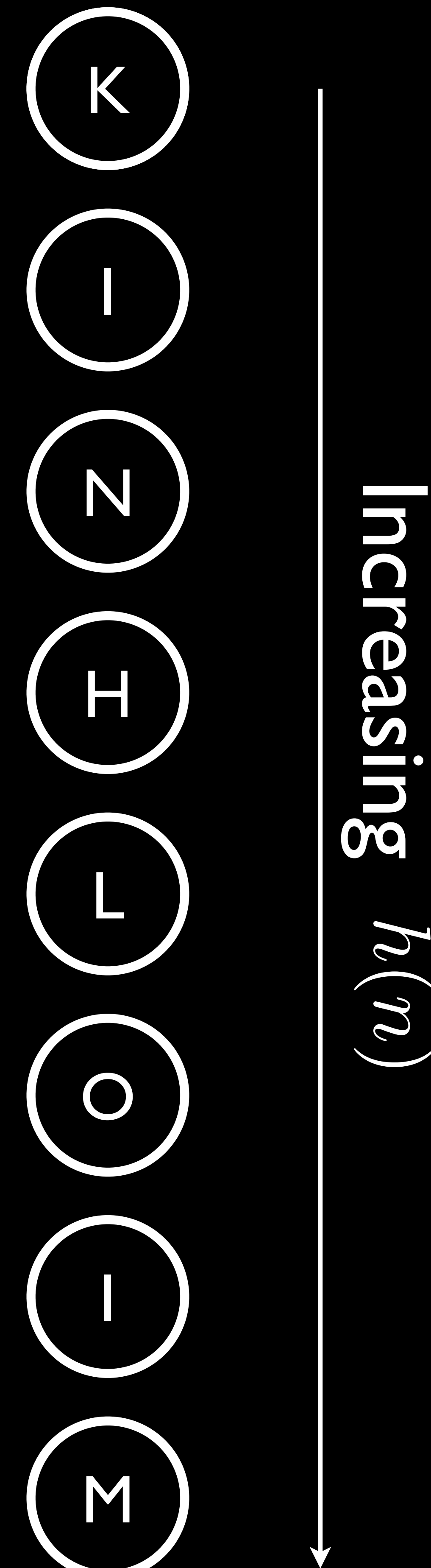


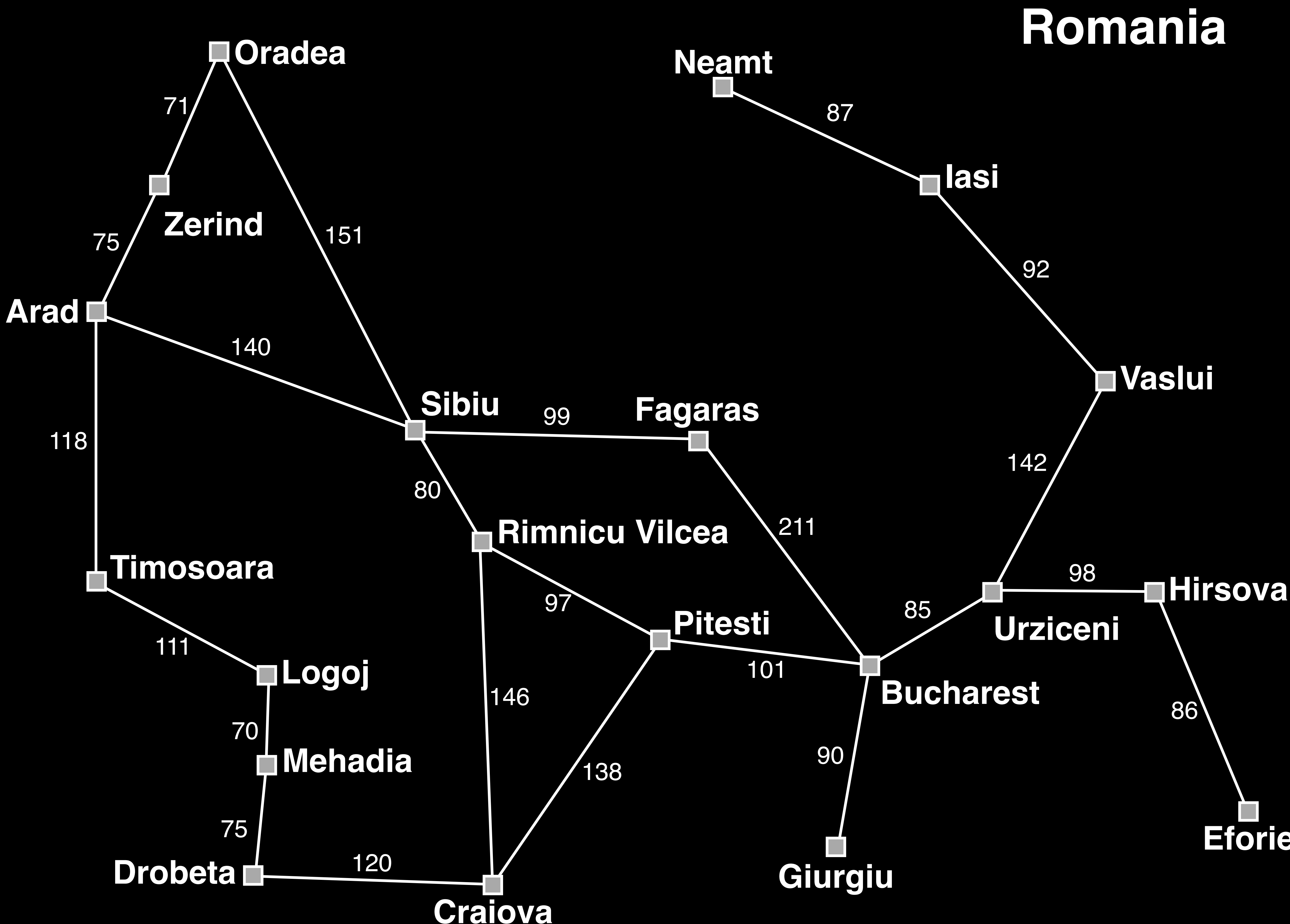
Heuristic function

$$h(n)$$

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

Frontier:
(Heap)





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

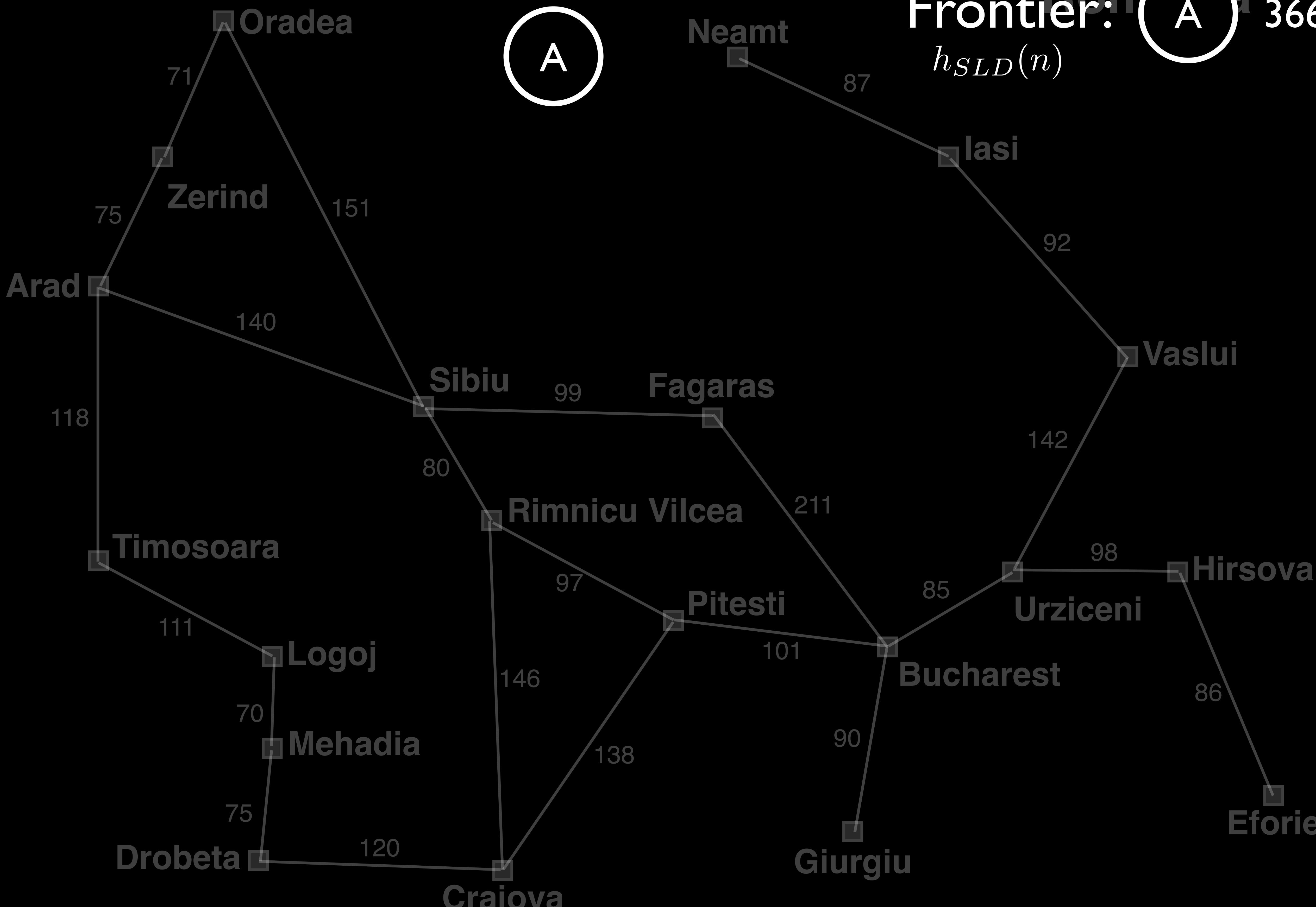
# $h_{SLD}$
## (straight-line distance to Bucharest)

| | |
|---|---|
| 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

$h_{SLD}(n)$

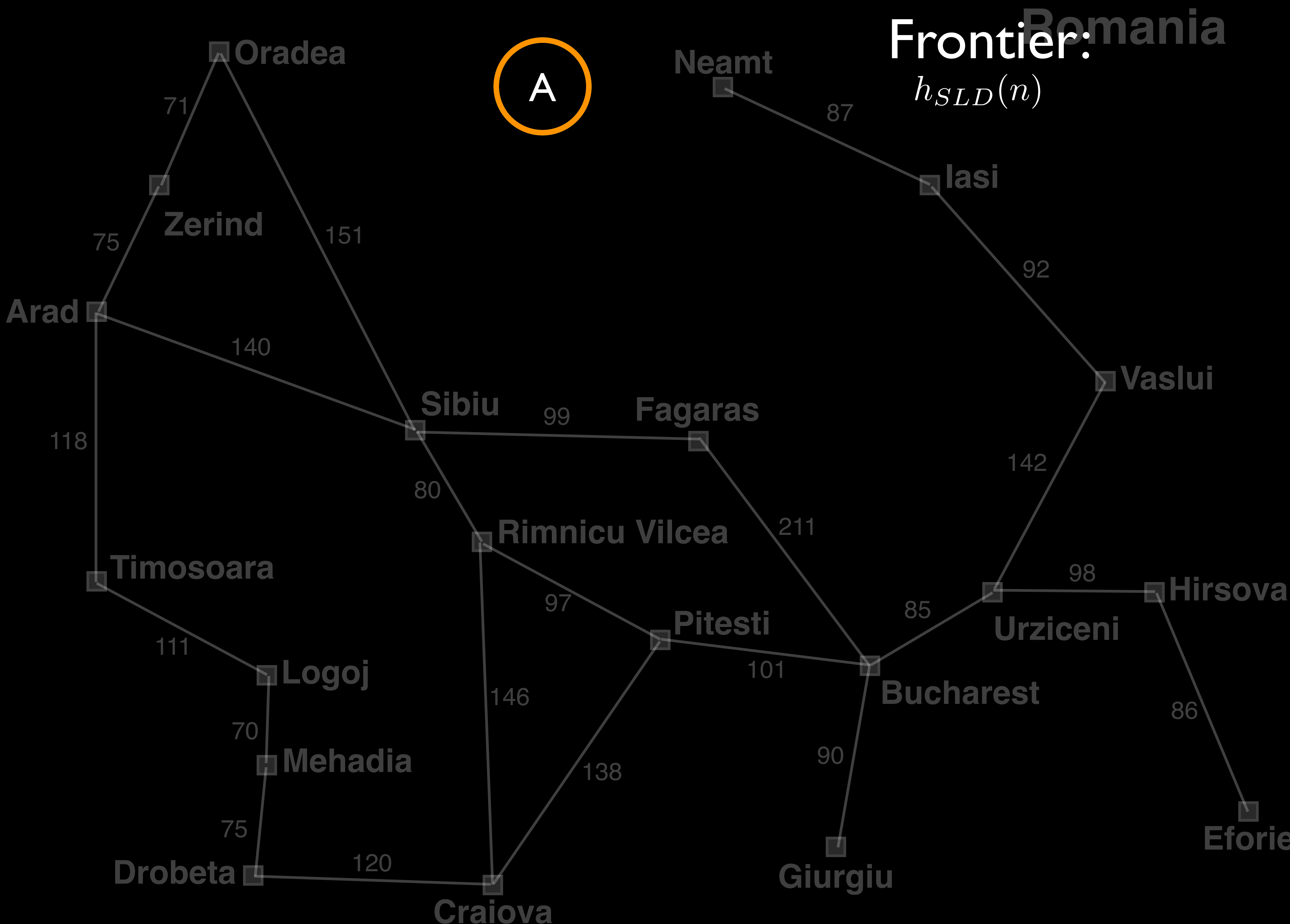


Frontier: A 366

$h_{SLD}(n)$

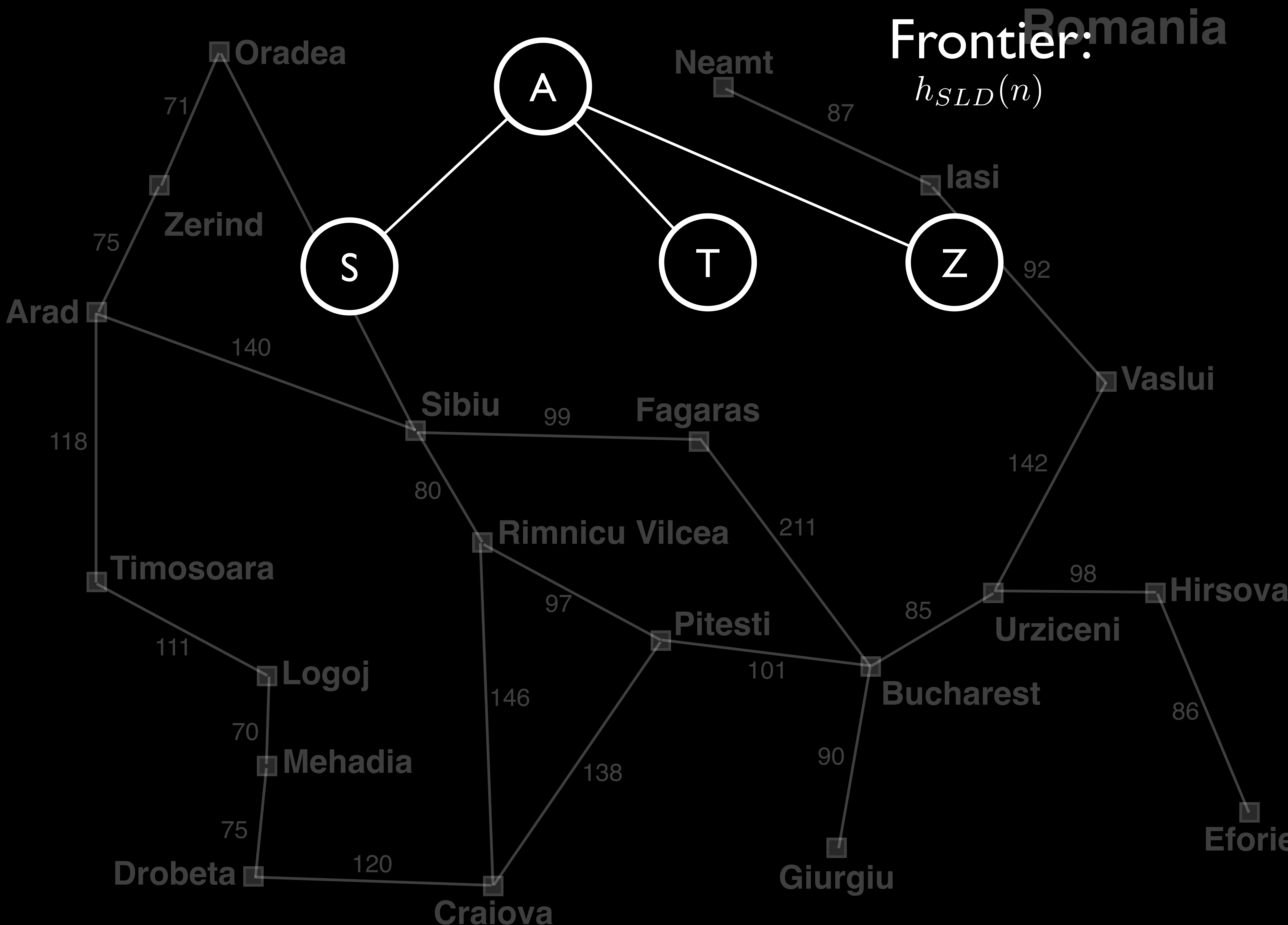
Frontier:

$h_{SLD}(n)$



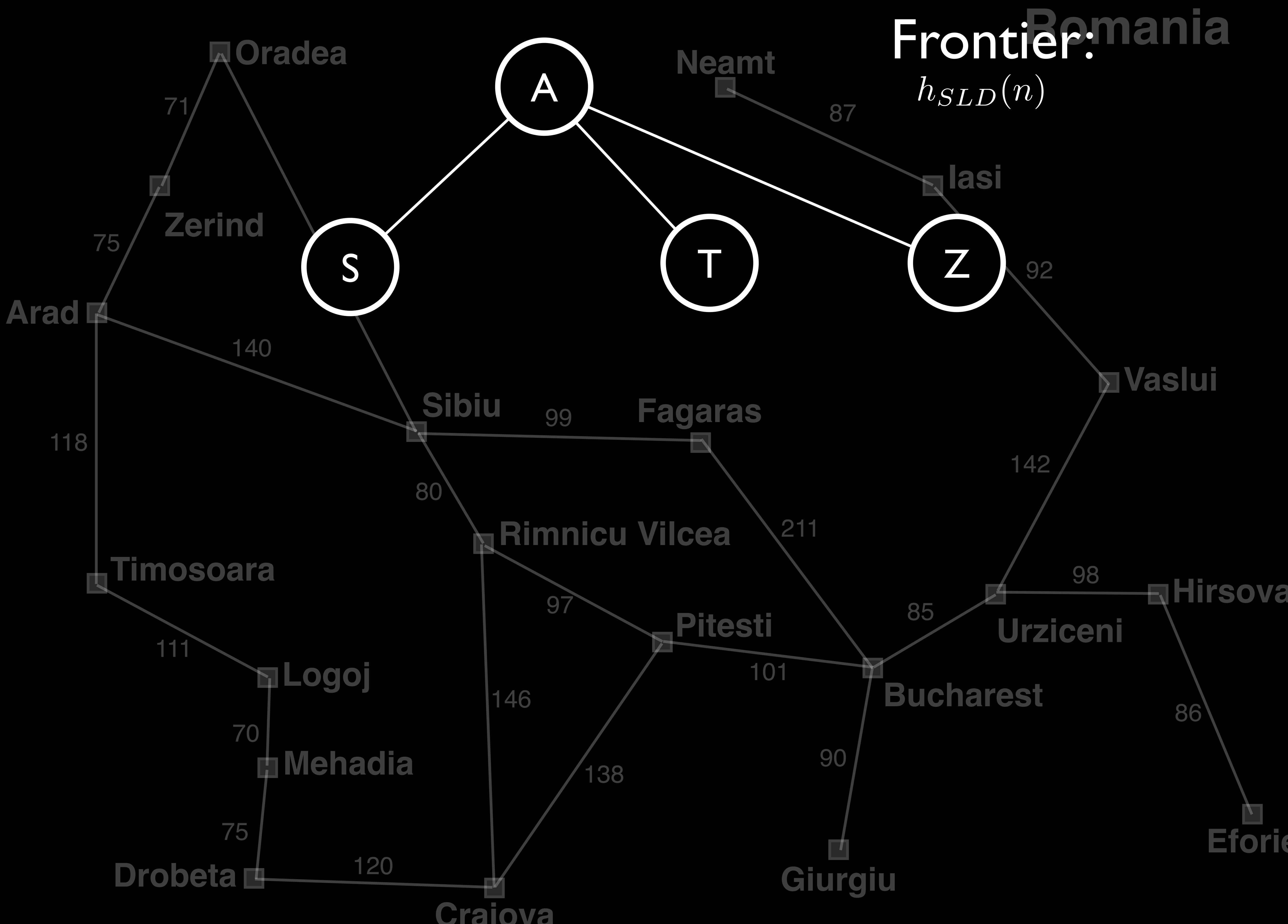
Frontier:

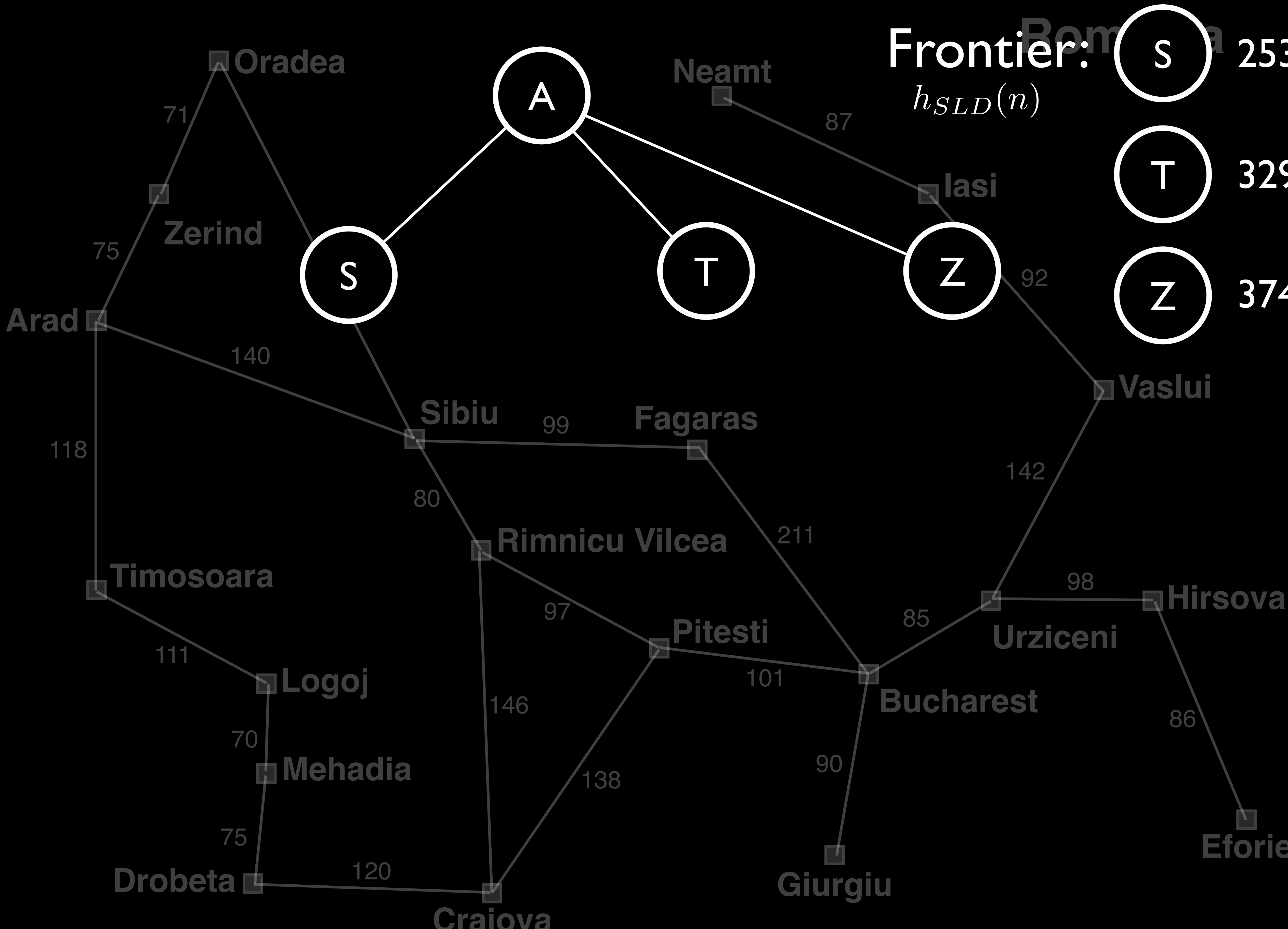
$h_{SLD}(n)$



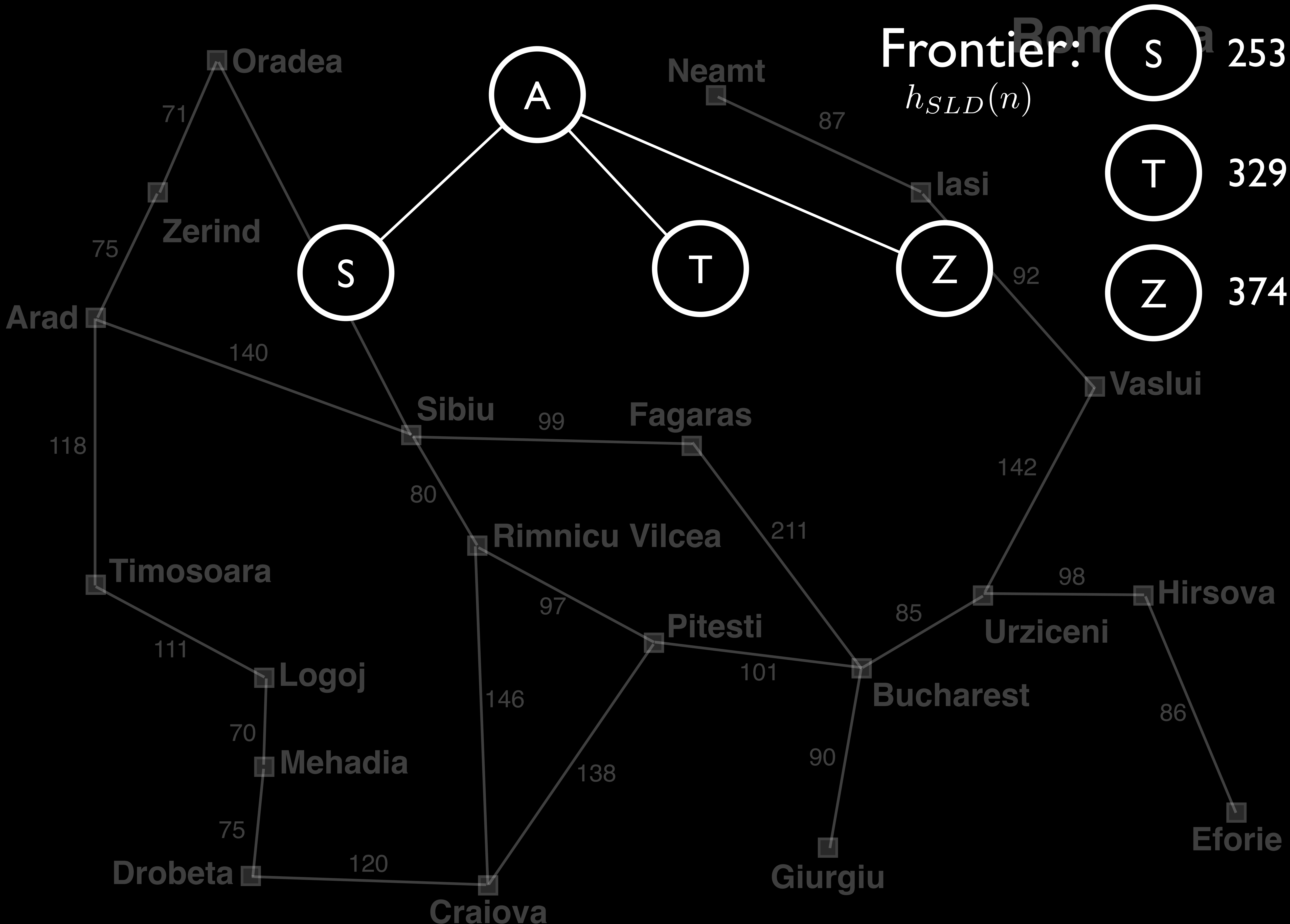
Frontier:

$h_{SLD}(n)$

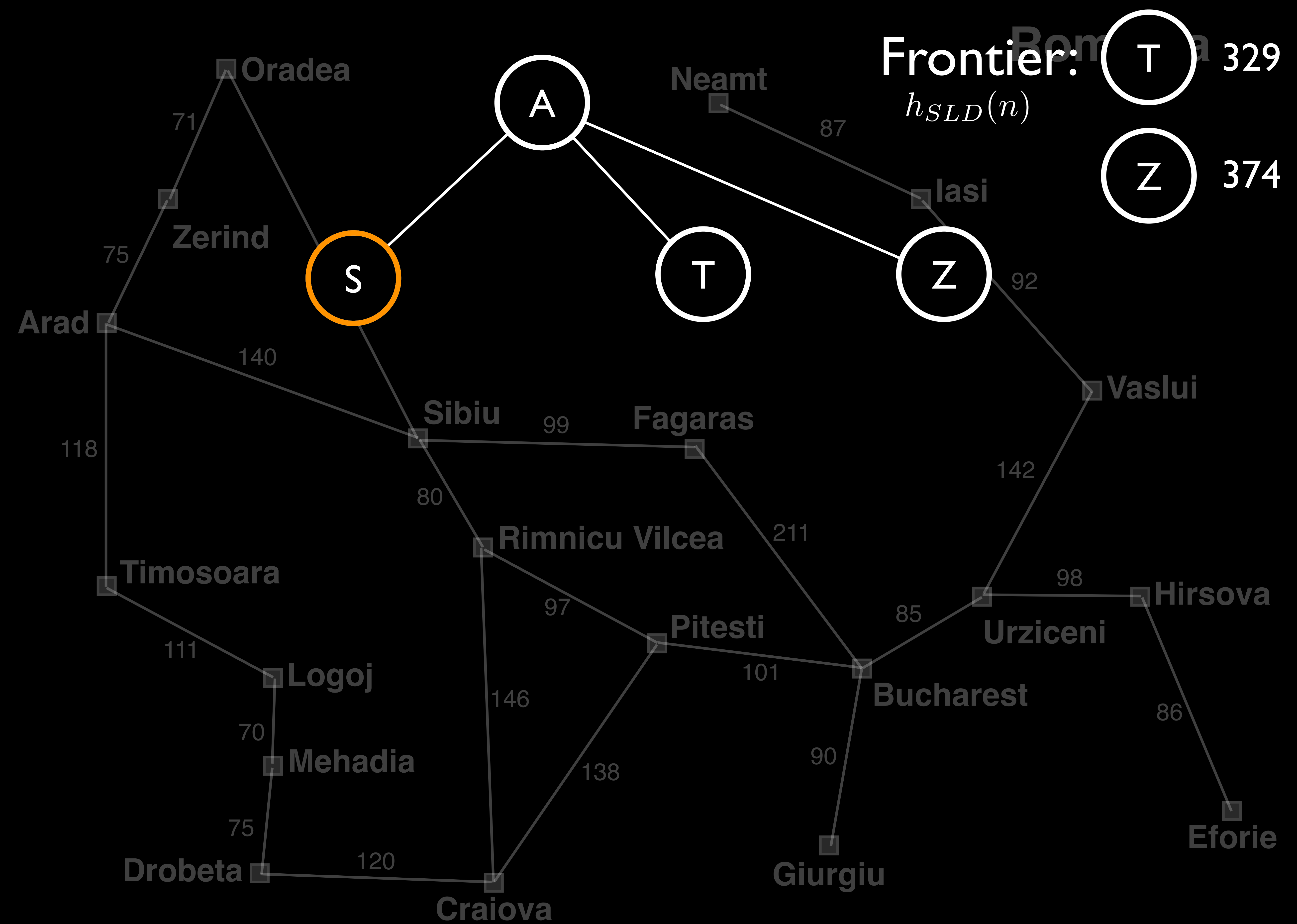


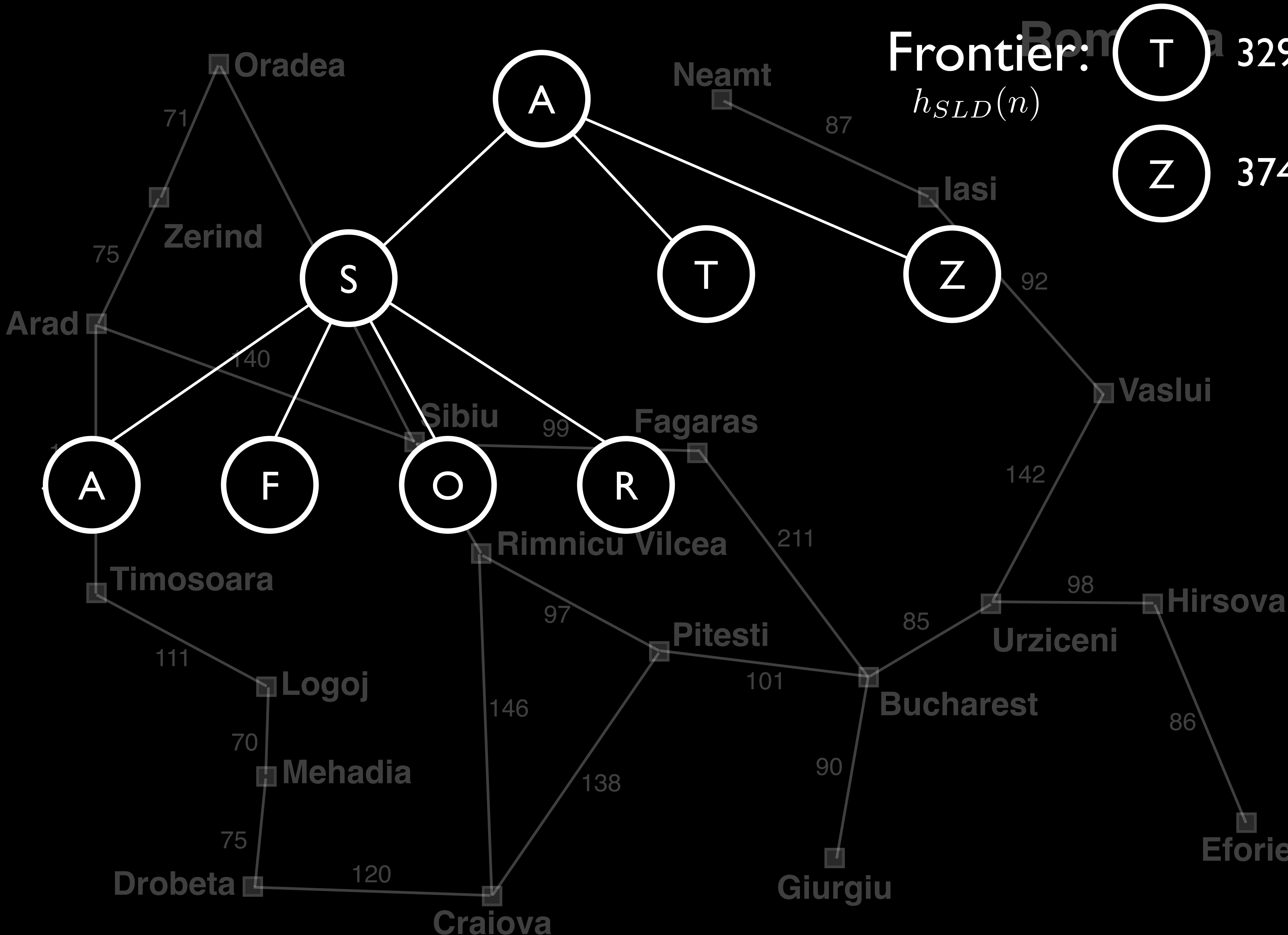


Frontier:
$h_{SLD}(n)$
S 253
T 329
Z 374
A
S
T
Z
Oradea
Neamt
Zerind
Iasi
Arad
Vaslui
Sibiu
Fagaras
Rimnicu Vilcea
Timosoara
Hirsova
Urziceni
Pitesti
Logoj
Bucharest
Mehadia
Drobeta
Eforie
Giurgiu
Craiova
71
87
75
92
140
99
118
142
80
211
98
97
85
111
101
146
86
70
90
138
75
120

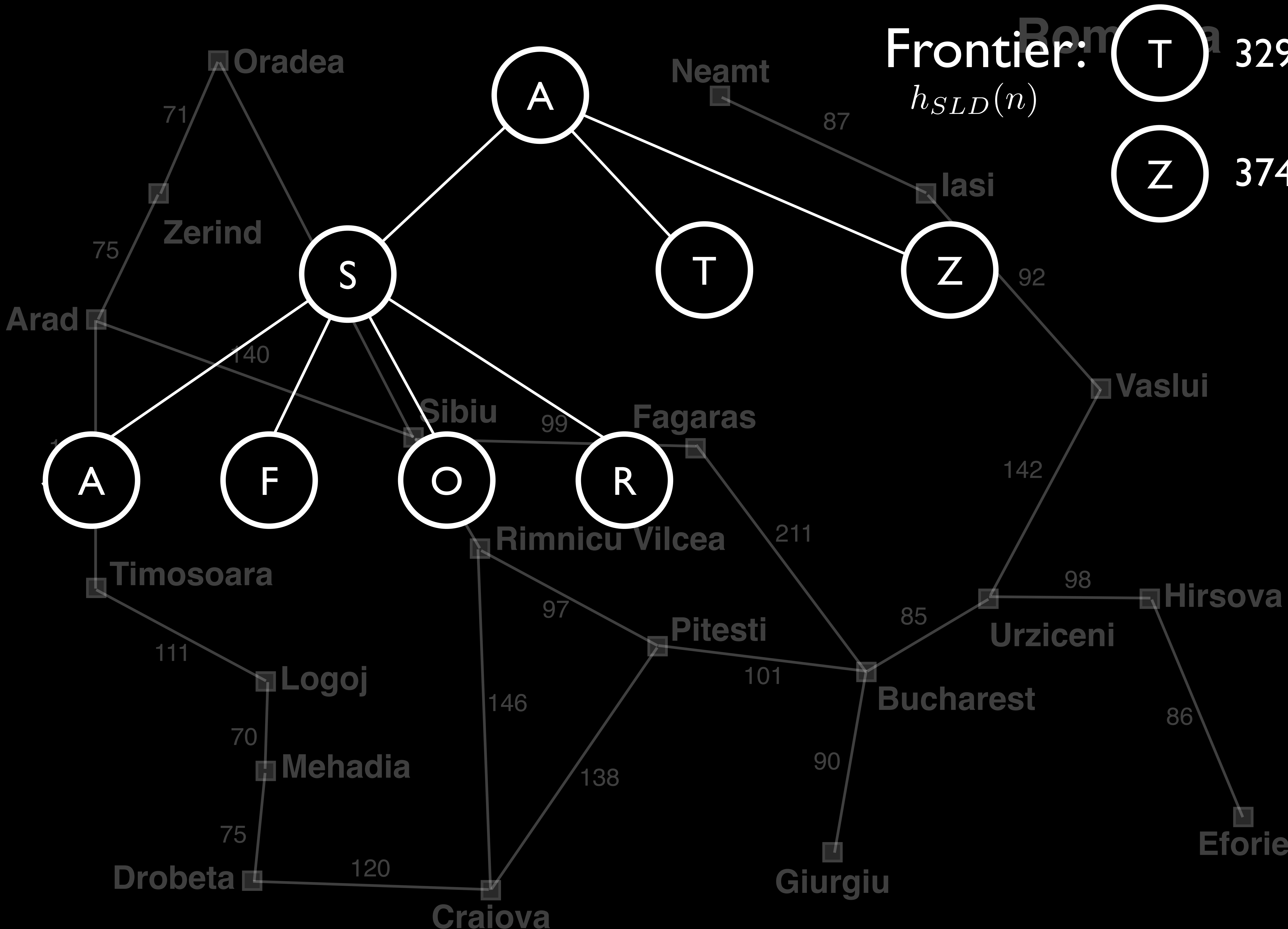


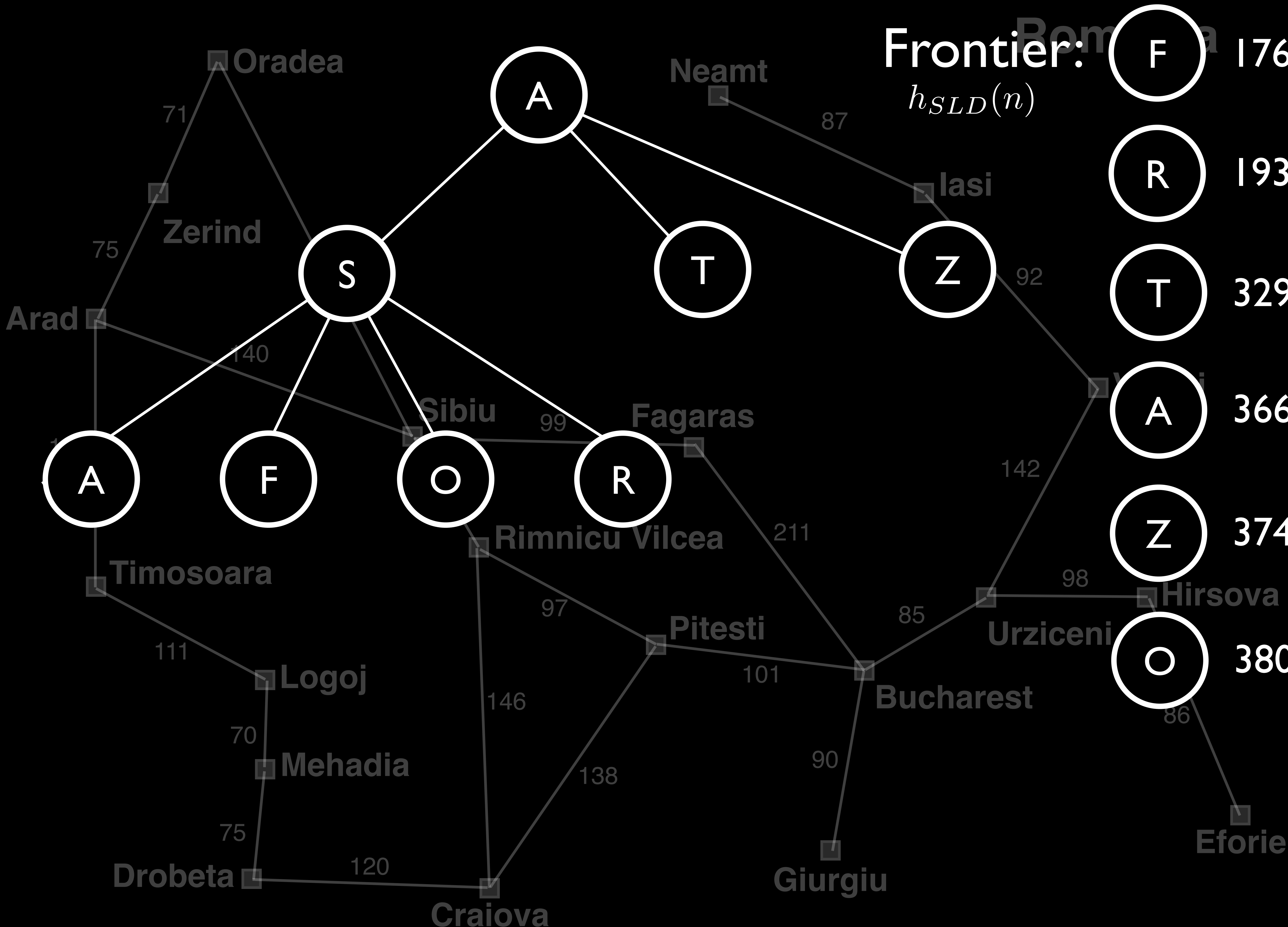
Frontier:
$h_{SLD}(n)$
S 253
T 329
Z 374
A
S
T
Z
Oradea
Neamt
Iasi
Zerind
Arad
Sibiu
Fagaras
Vaslui
Rimnicu Vilcea
Timosoara
Pitesti
Urziceni
Hirsova
Logoj
Mehadia
Bucharest
Drobeta
Craiova
Giurgiu
Eforie
71
75
87
92
140
99
118
80
142
211
97
98
85
111
101
146
70
86
90
138
75
120


Frontier:
T 329
Z 374
$h_{SLD}(n)$
A
S
T
Z
Oradea
Neamt
Iasi
Zerind
Arad
Sibiu
Fagaras
Vaslui
Rimnicu Vilcea
Timosoara
Pitesti
Urziceni
Hirsova
Logoj
Bucharest
Mehadia
Eforie
Drobeta
Giurgiu
Craiova
71
75
87
92
140
99
118
80
142
211
98
97
85
111
101
146
86
70
90
138
75
120

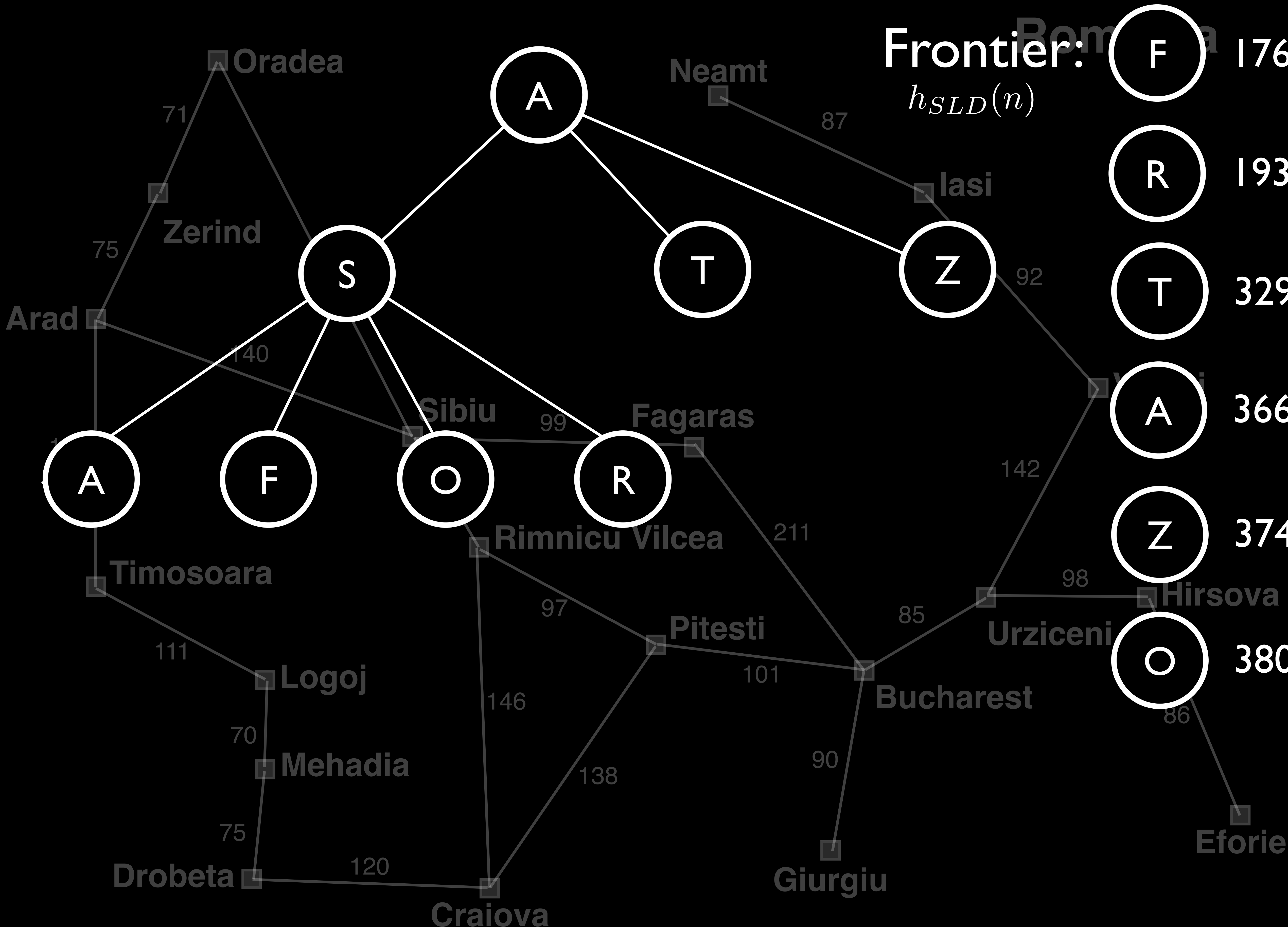


Frontier:
$h_{SLD}(n)$
T
329
Z
374
A
S
T
Z
A
F
O
R
Romania
Oradea
Neamt
71
87
Iasi
Zerind
75
92
Arad
140
Vaslui
Sibiu
99
Fagaras
142
Rimnicu Vilcea
211
Timosoara
98
97
Hirsova
Pitesti
85
Urziceni
111
Logoj
101
Bucharest
146
86
70
Mehadia
90
138
75
Eforie
120
Drobeta
Giurgiu
Craiova

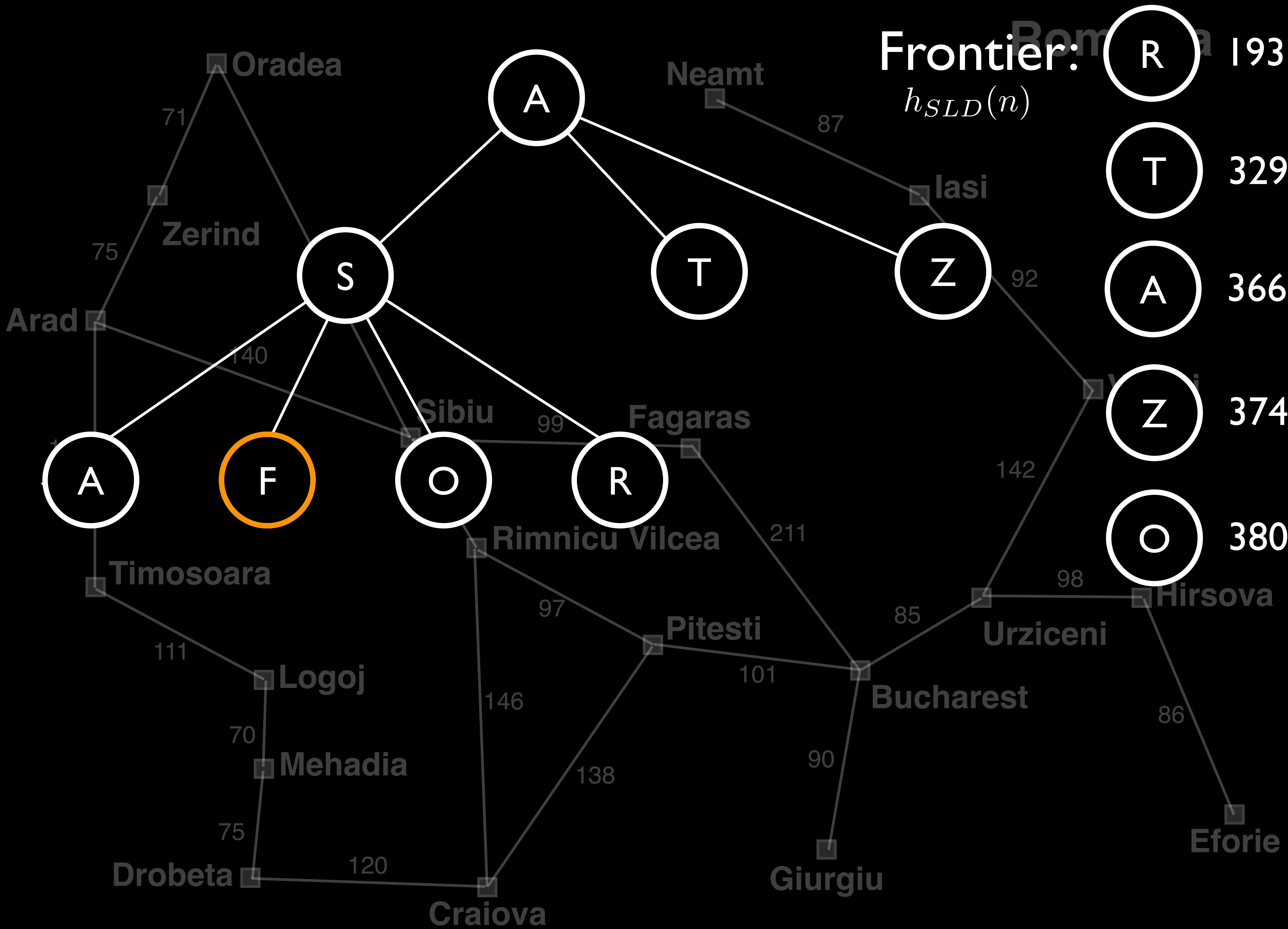

Frontier:
$h_{SLD}(n)$
T 329
Z 374
A
S
T
Z
A
F
O
R
Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Vaslui
Rimnicu Vilcea
Timosoara
Hirsova
Urziceni
Pitesti
Logoj
Bucharest
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
75
87
92
140
99
142
211
98
97
85
111
101
146
86
70
90
138
75
120



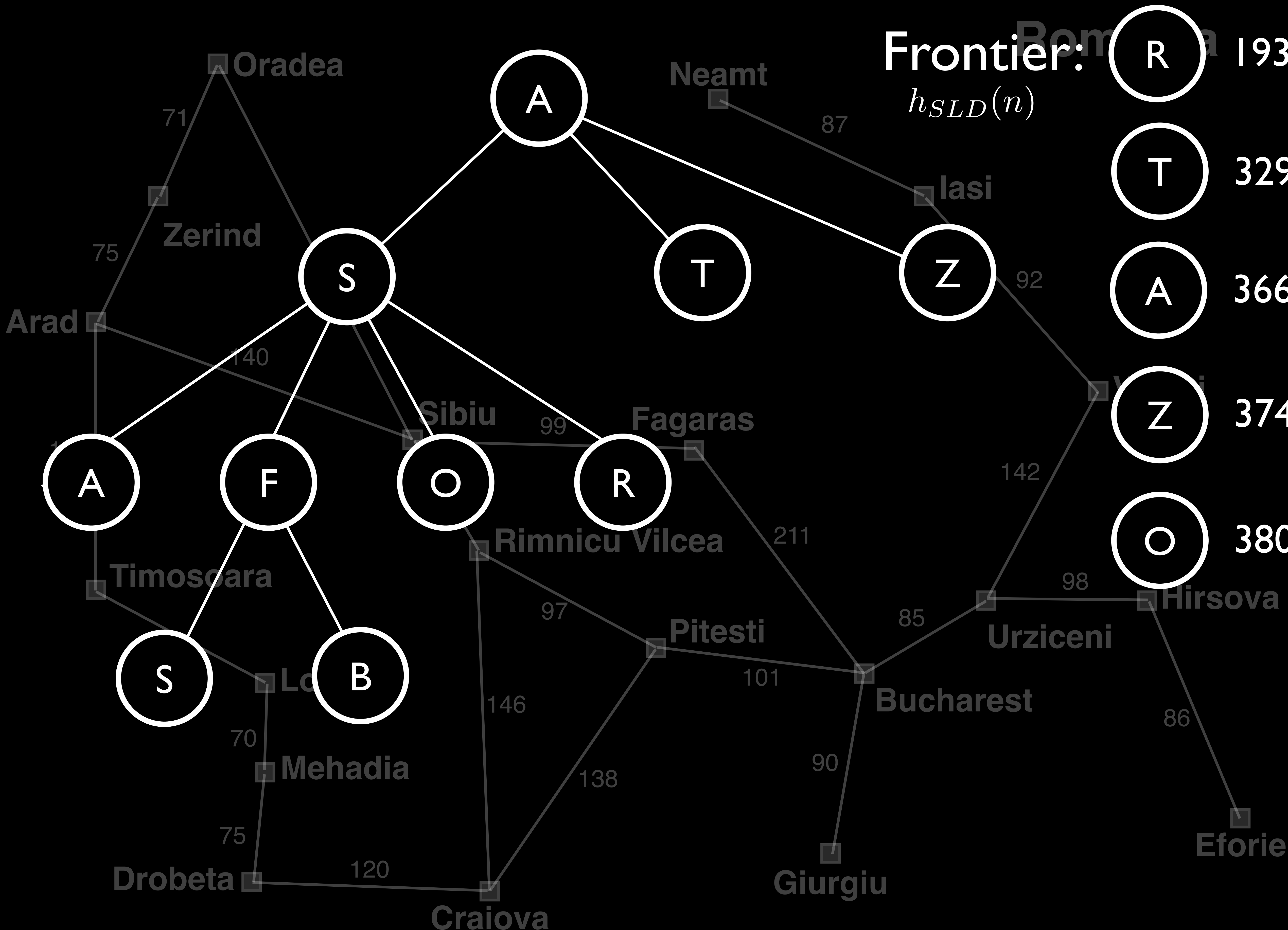
Frontier:
$h_{SLD}(n)$
F 176
R 193
T 329
A 366
Z 374
O 380
A
S
T
Z
A
F
O
R
Oradea
Neamt
Iasi
Zerind
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Timosoara
Pitesti
Urziceni
Hirsova
Bucharest
Logoj
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
87
75
92
140
99
142
211
98
97
85
111
101
146
86
70
90
138
75
120



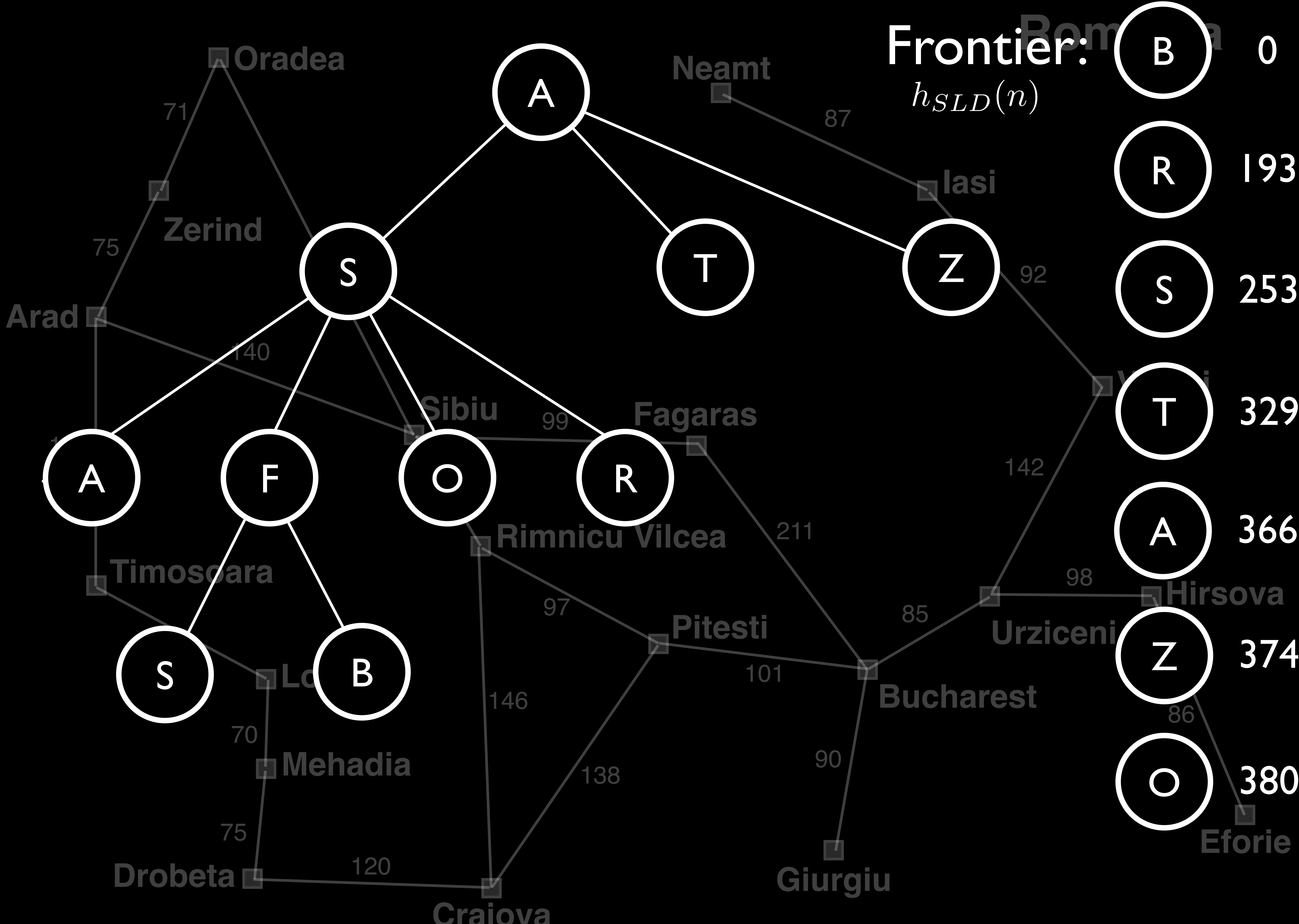
Frontier:
$h_{SLD}(n)$
F 176
R 193
T 329
A 366
Z 374
O 380
A
S
T
Z
A
F
O
R
Oradea
Neamt
71
87
Iasi
Zerind
75
92
Arad
140
Sibiu
99
Fagaras
142
211
Rimnicu Vilcea
Timosoara
98
Hirsova
97
85
Pitesti
Urziceni
111
Logoj
101
Bucharest
146
86
70
Mehadia
90
138
75
Eforie
120
Drobeta
Giurgiu
Craiova



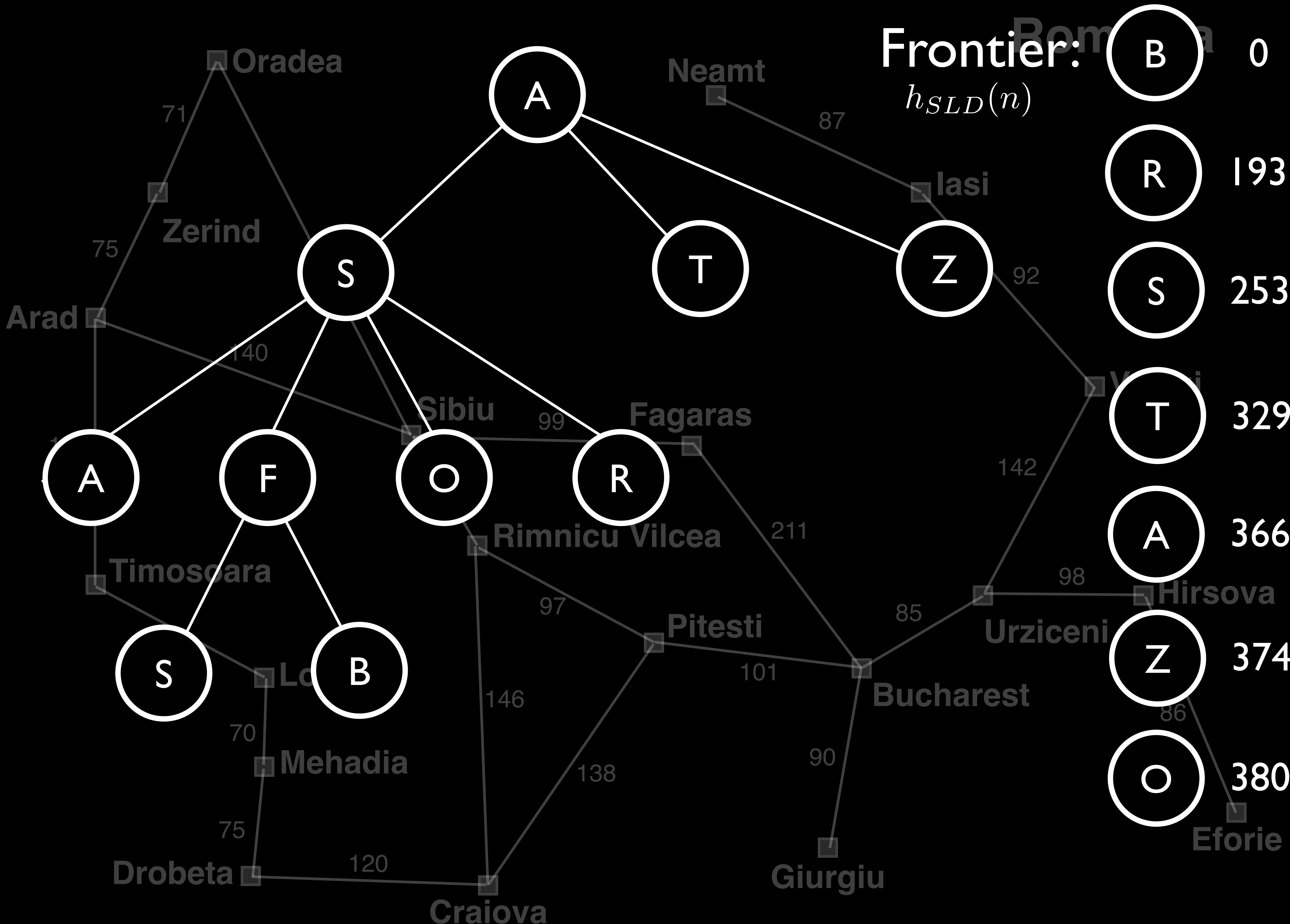
Frontier:
$h_{SLD}(n)$
R 193
T 329
A 366
Z 374
O 380
A
S
T
Z
A
F
O
R
Oradea
Neamt
Iasi
Zerind
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Timosoara
Pitesti
Urziceni
Hirsova
Lugoj
Bucharest
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
87
75
92
140
99
142
211
97
98
85
111
101
146
86
70
90
138
75
120

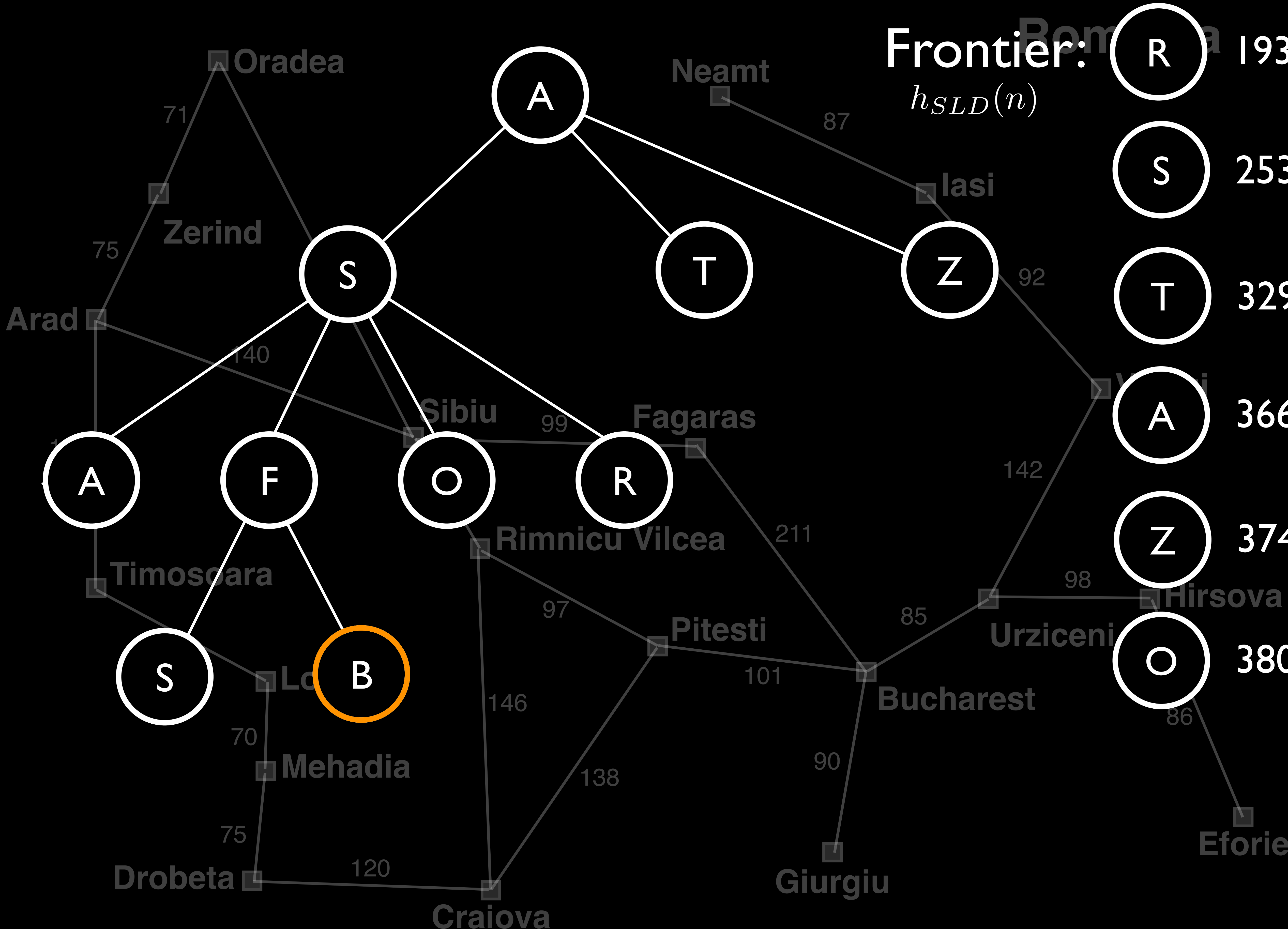


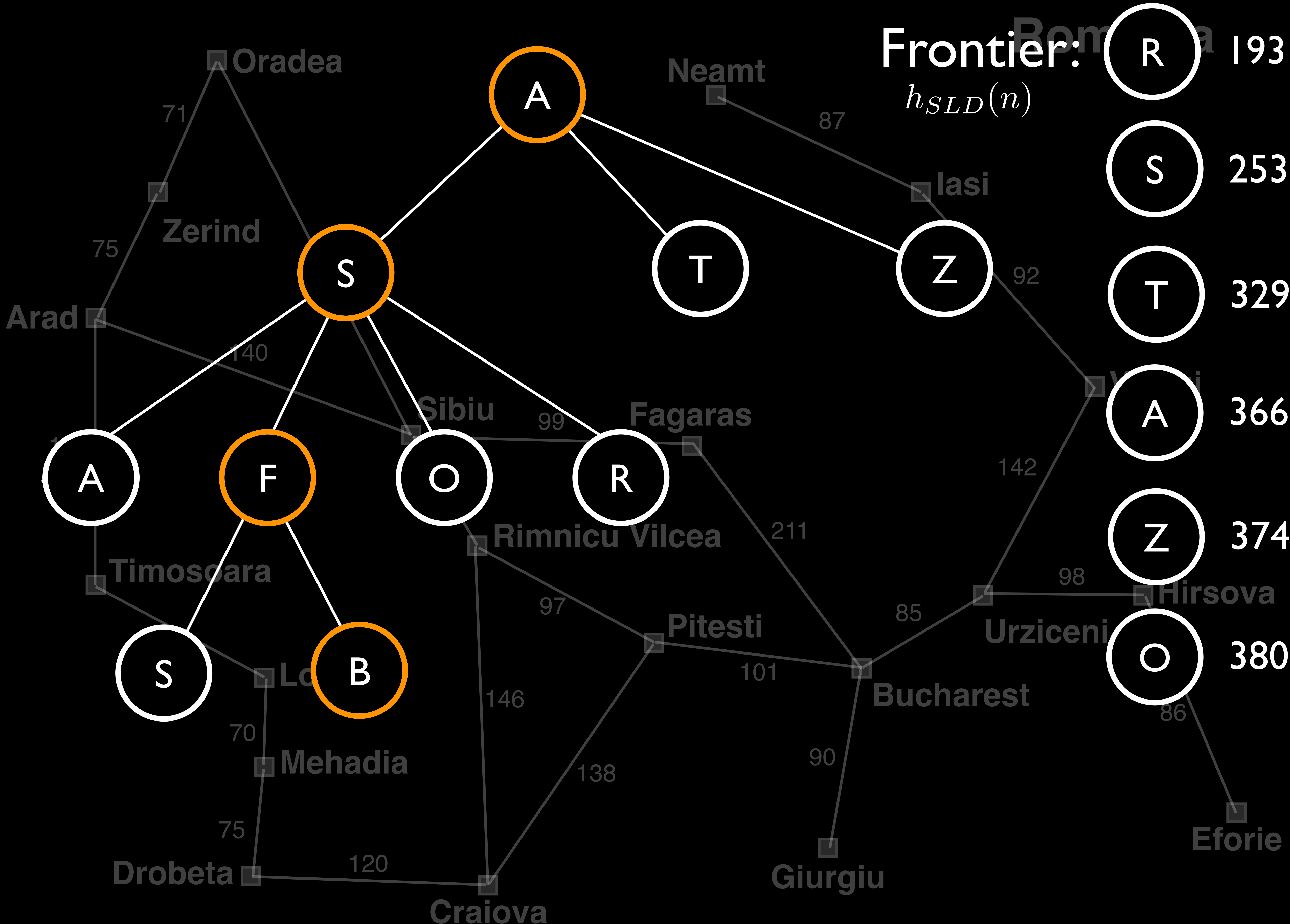
Frontier:
$h_{SLD}(n)$
R 193
T 329
A 366
Z 374
O 380
A
S
T
Z
A
F
O
R
S
B
Oradea
Neamt
71
87
Iasi
Zerind
75
92
Arad
140
Sibiu
99
Fagaras
142
Rimnicu Vilcea
211
Timosoara
98
Hirsova
97
Pitesti
85
Urziceni
101
Bucharest
146
70
Mehadia
86
90
138
75
Drobeta
120
Giurgiu
Eforie
Craiova



Frontier:
$h_{SLD}(n)$
B 0
R 193
S 253
T 329
A 366
Z 374
O 380
A
S T Z
A F O R
S B
Oradea
Neamt
Iasi
Zerind
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Pitesti
Urziceni
Hirsova
Bucharest
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
75
87
92
140
99
142
211
97
98
85
101
146
86
70
90
138
75
120

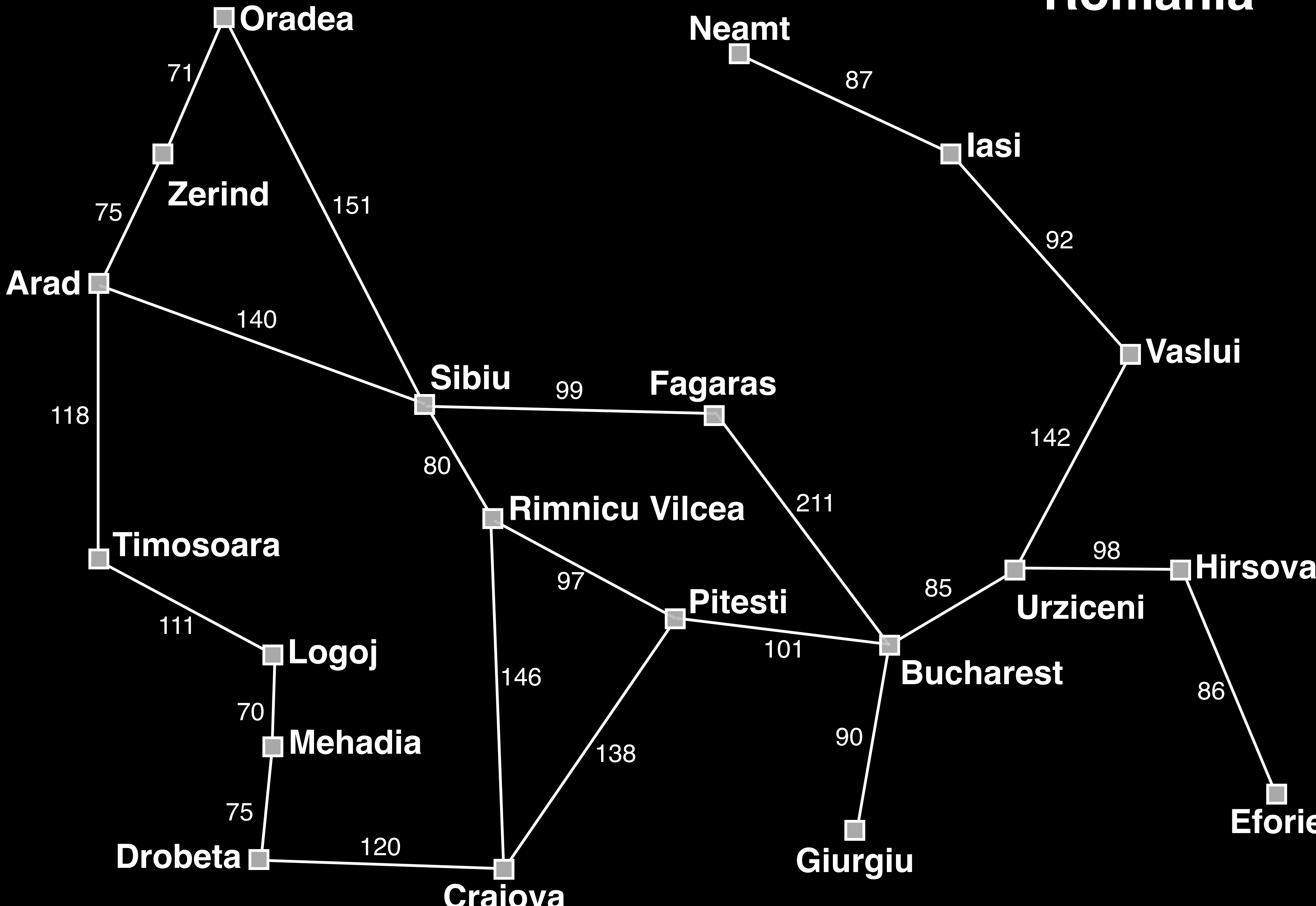

Frontier:
$h_{SLD}(n)$
B 0
R 193
S 253
T 329
A 366
Z 374
O 380
A
S
T
Z
A
F
O
R
S
B
Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Timosoara
Pitesti
Hirsova
Urziceni
Bucharest
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
75
87
92
140
99
142
211
98
97
85
101
146
70
86
90
138
75
120


Frontier:
$h_{SLD}(n)$
R 193
S 253
T 329
A 366
Z 374
O 380
A
S
T
Z
A
F
O
R
S
B
Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Timosoara
Pitesti
Urziceni
Hirsova
Bucharest
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
87
75
92
140
99
142
211
98
97
85
101
146
70
90
138
86
75
120

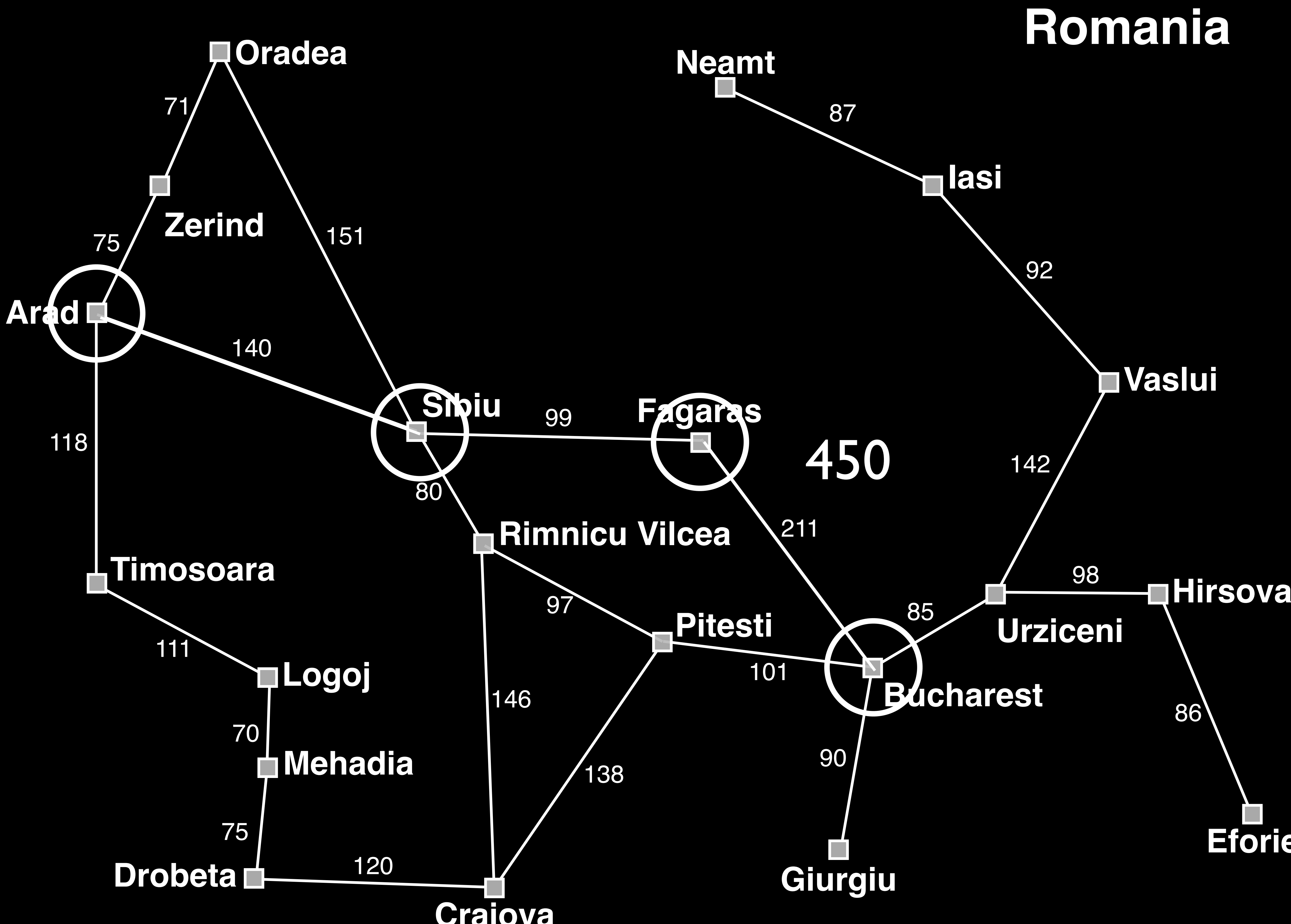


Frontier:
$h_{SLD}(n)$
R 193
S 253
T 329
A 366
Z 374
O 380
A
S
T
Z
A
F
O
R
S
B
Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Timisoara
Pitesti
Urziceni
Hirsova
Bucharest
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
87
75
92
140
99
142
211
98
97
85
101
146
86
70
90
138
75
120

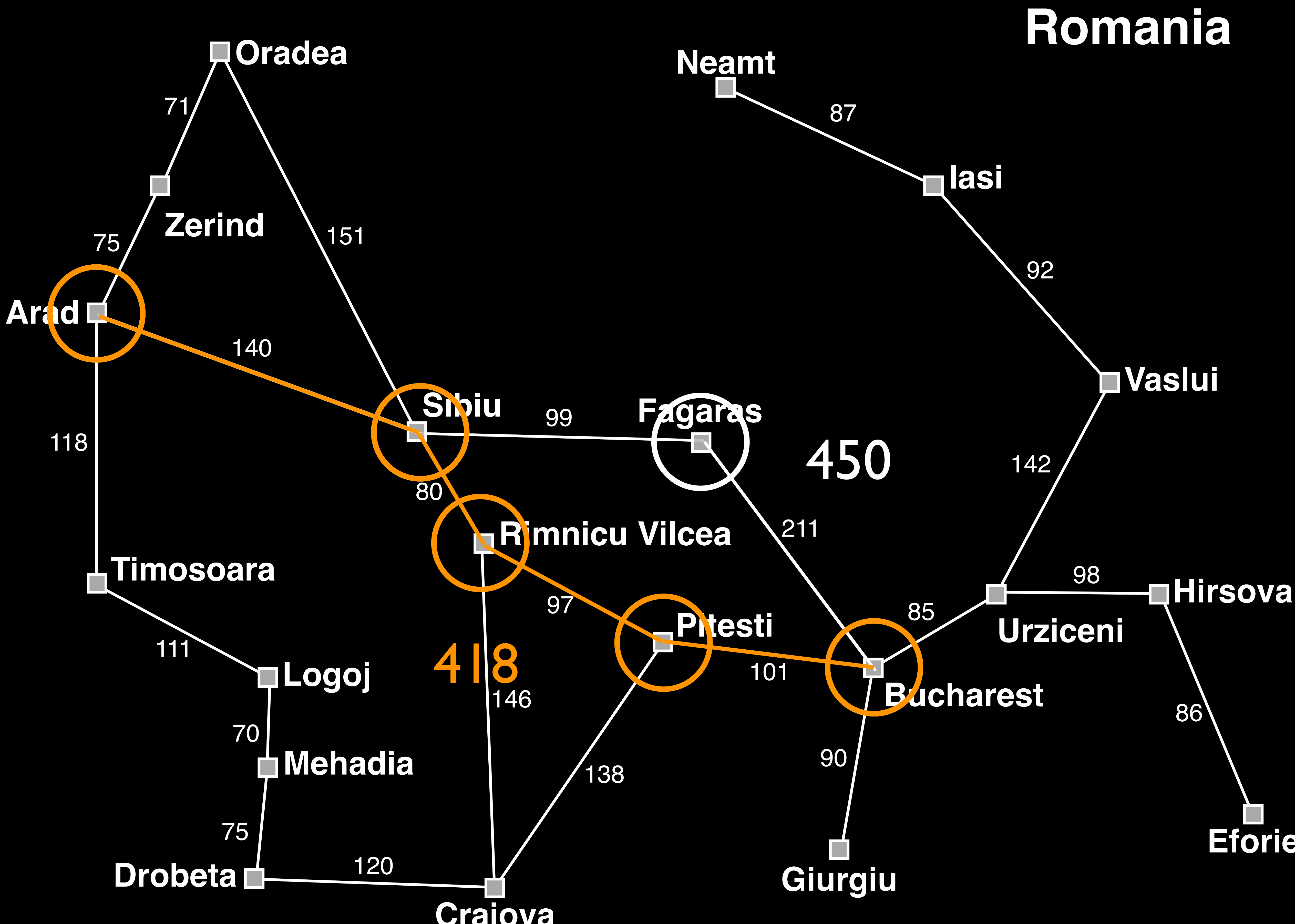
# Romania



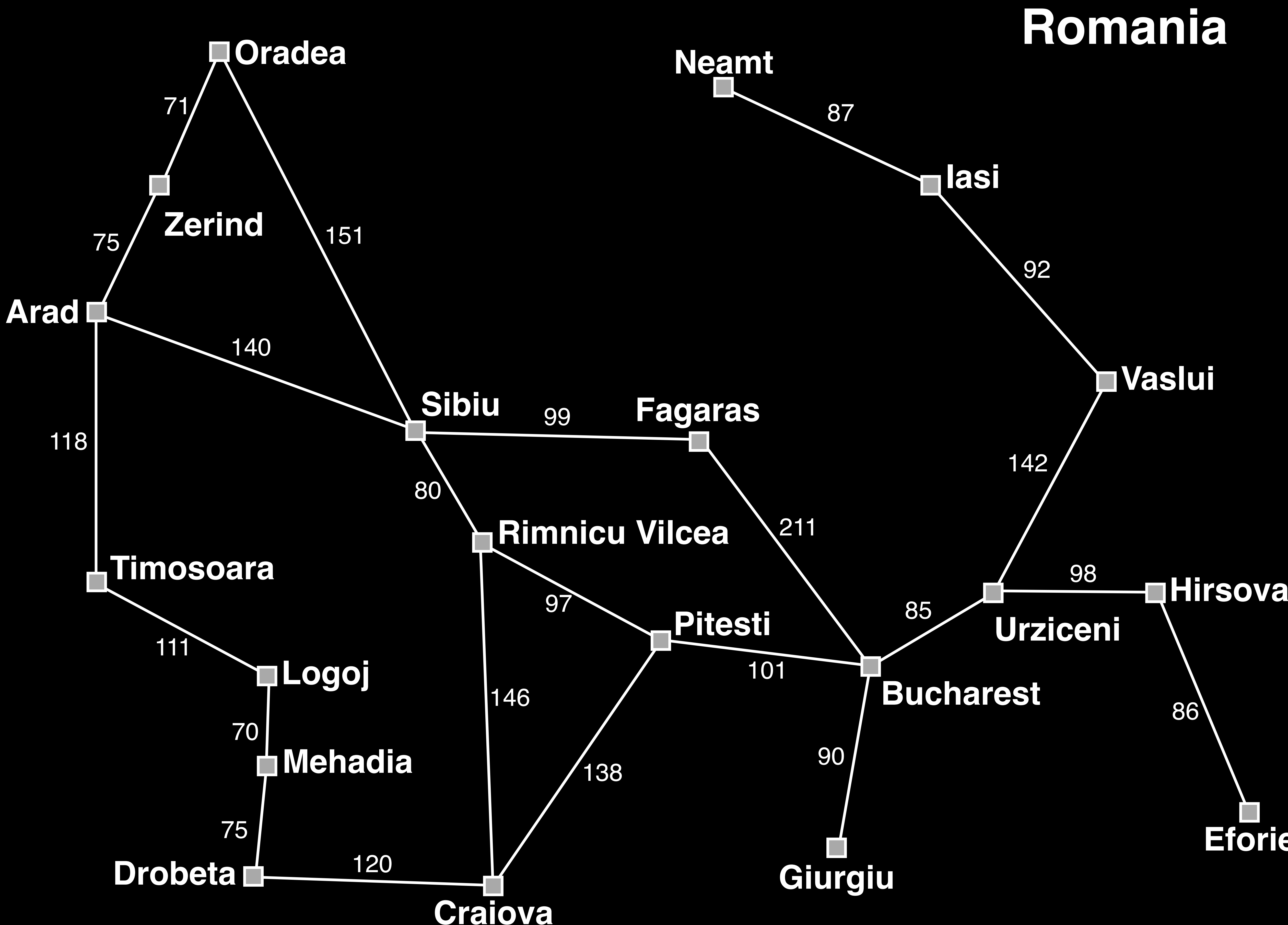
Oradea
Neamt
71
87
Iasi
Zerind
75
151
92
Arad
140
Vaslui
Sibiu
99
Fagaras
118
142
80
Rimnicu Vilcea
211
Timosoara
98
Hirsova
97
85
Urziceni
Pitesti
111
Logoj
101
Bucharest
146
86
70
Mehadia
90
138
75
Eforie
Drobeta
120
Giurgiu
Craiova

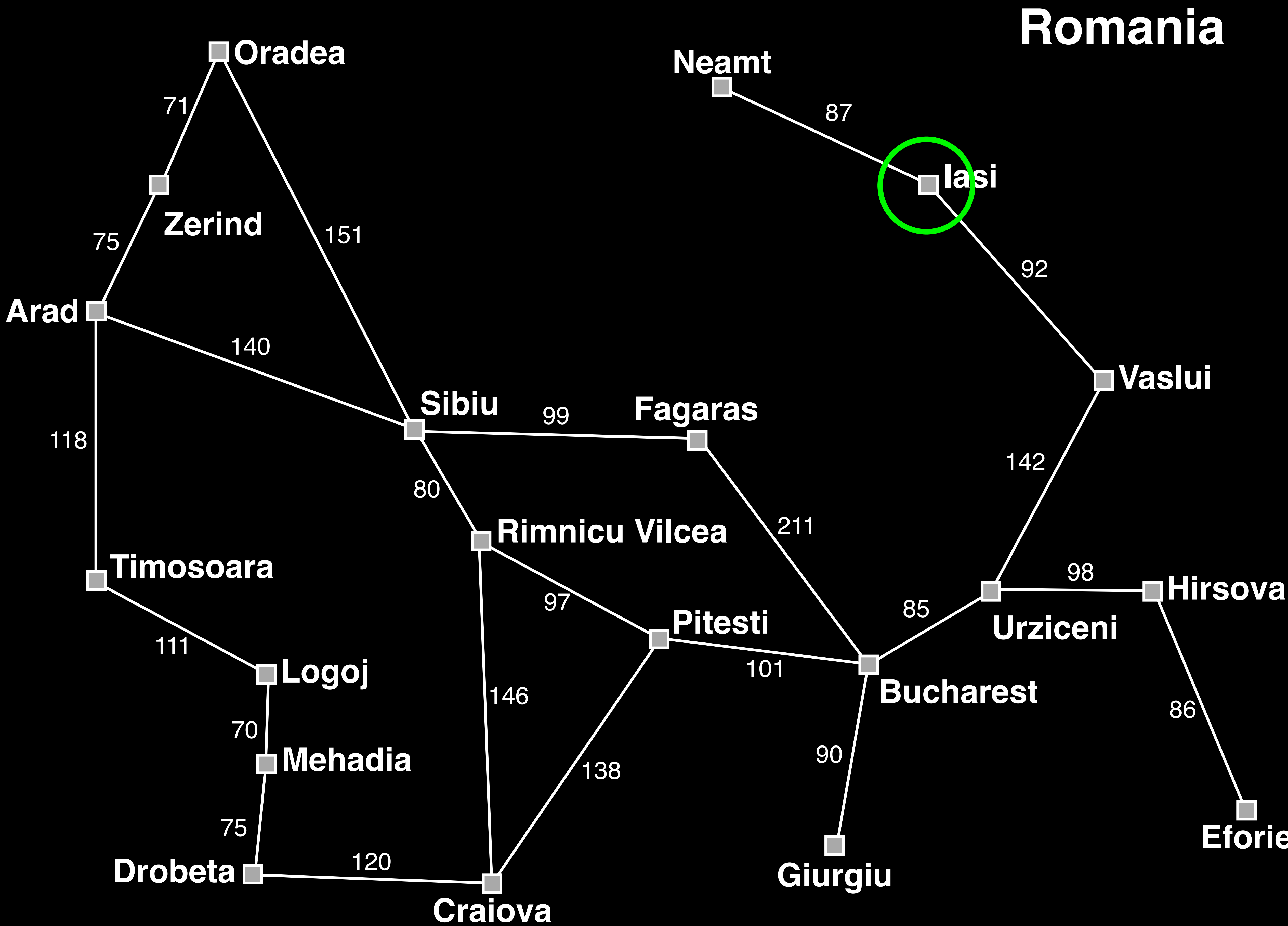


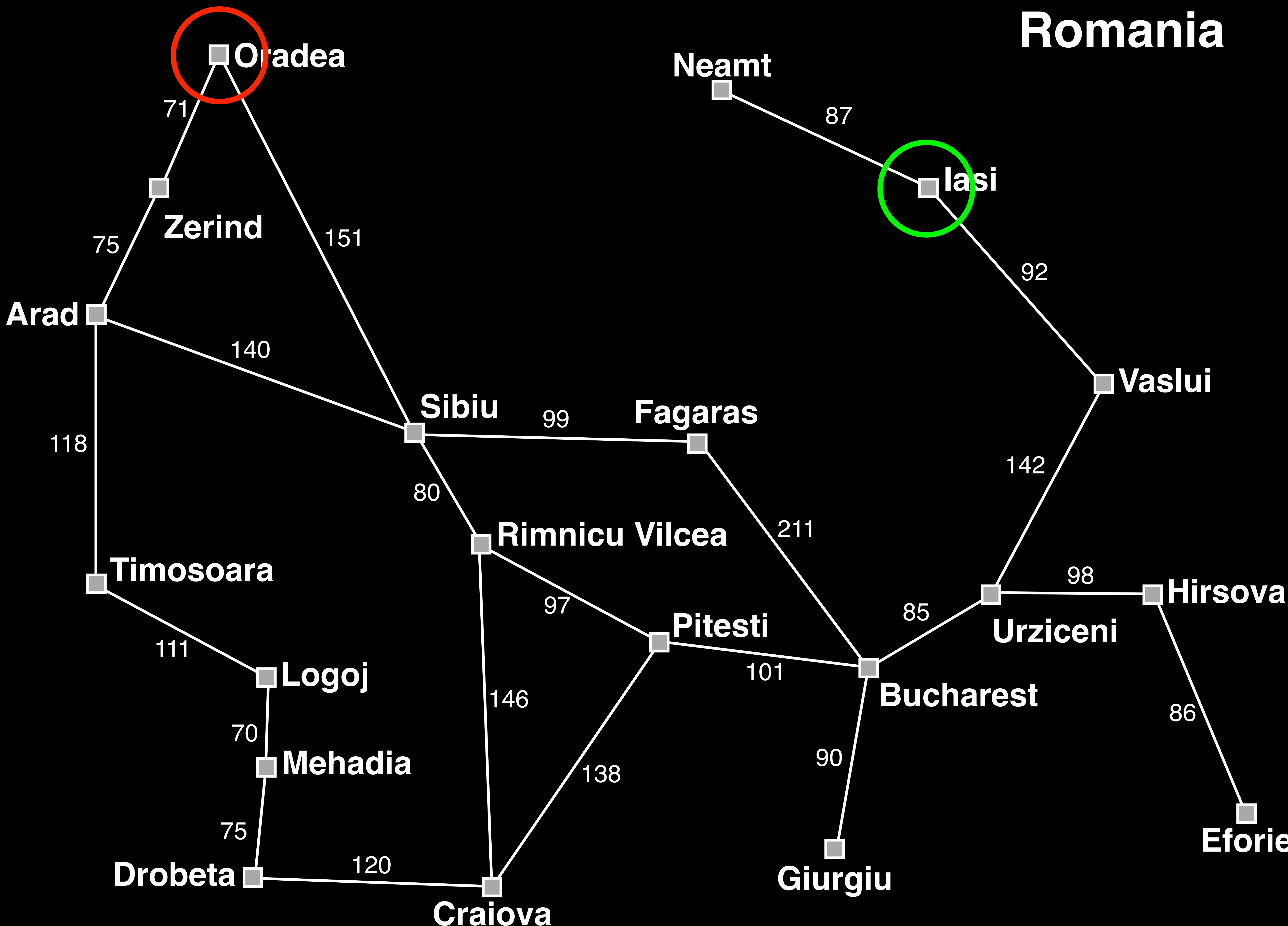
Romania
Oradea
Neamt
71
87
Zerind
151
Iasi
75
92
Arad
140
Vaslui
Sibiu
99
Fagaras
118
450
142
80
Rimnicu Vilcea
211
Timosoara
98
97
Pitesti
85
Hirsova
Urziceni
111
Logoj
101
Bucharest
146
86
70
Mehadia
90
138
75
Eforie
Drobeta
120
Giurgiu
Craiova



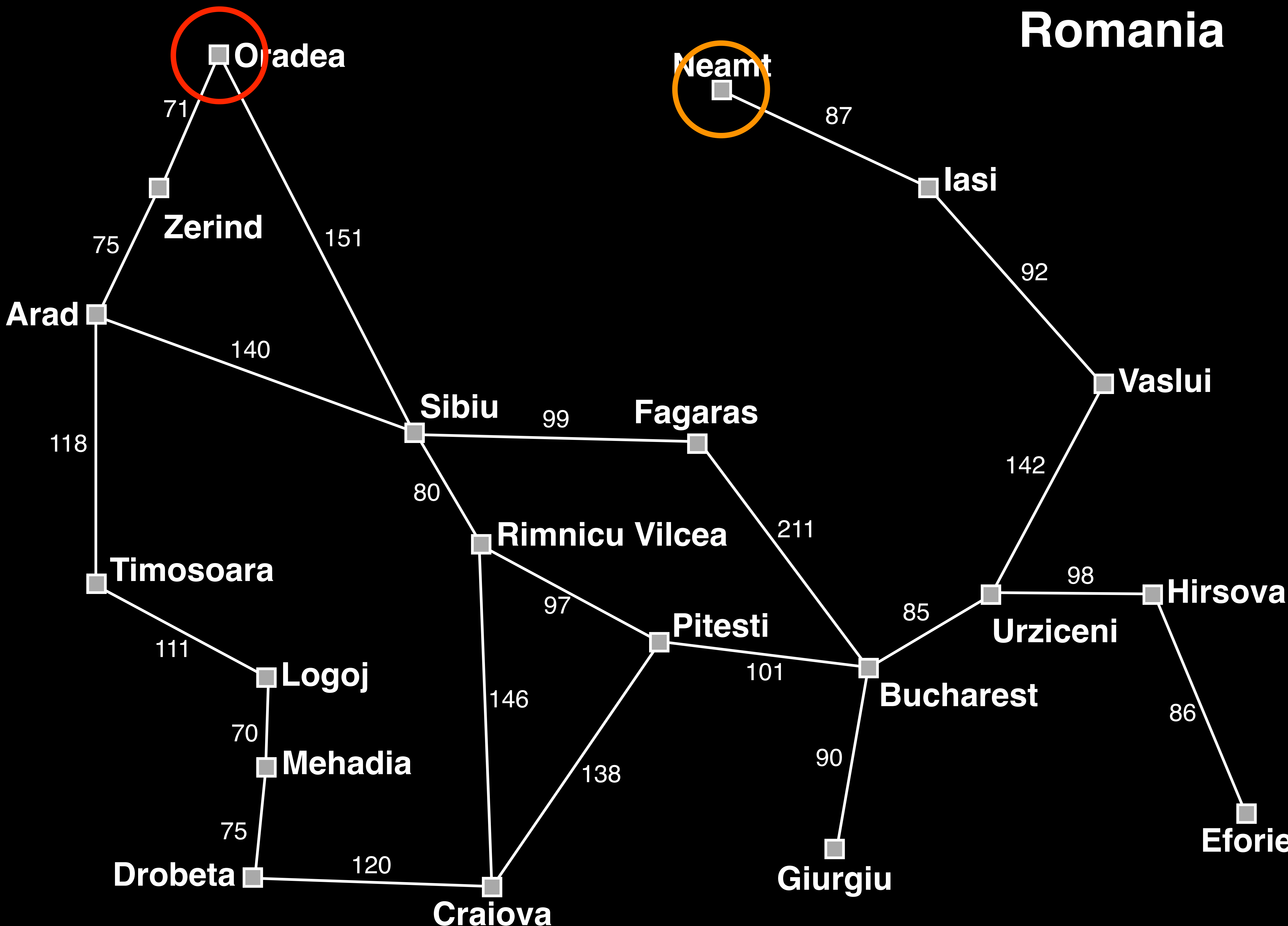
Romania
Oradea
Neamt
71
87
Zerind
Iasi
75
151
92
Arad
140
Vaslui
Sibiu
99
Fagaras
118
450
142
80
Rimnicu Vilcea
211
Timosoara
98
Hirsova
97
Pitesti
85
111
Urziceni
Logoj
418
101
146
Bucharest
86
70
Mehadia
90
138
75
Eforie
Drobeta
120
Giurgiu
Craiova

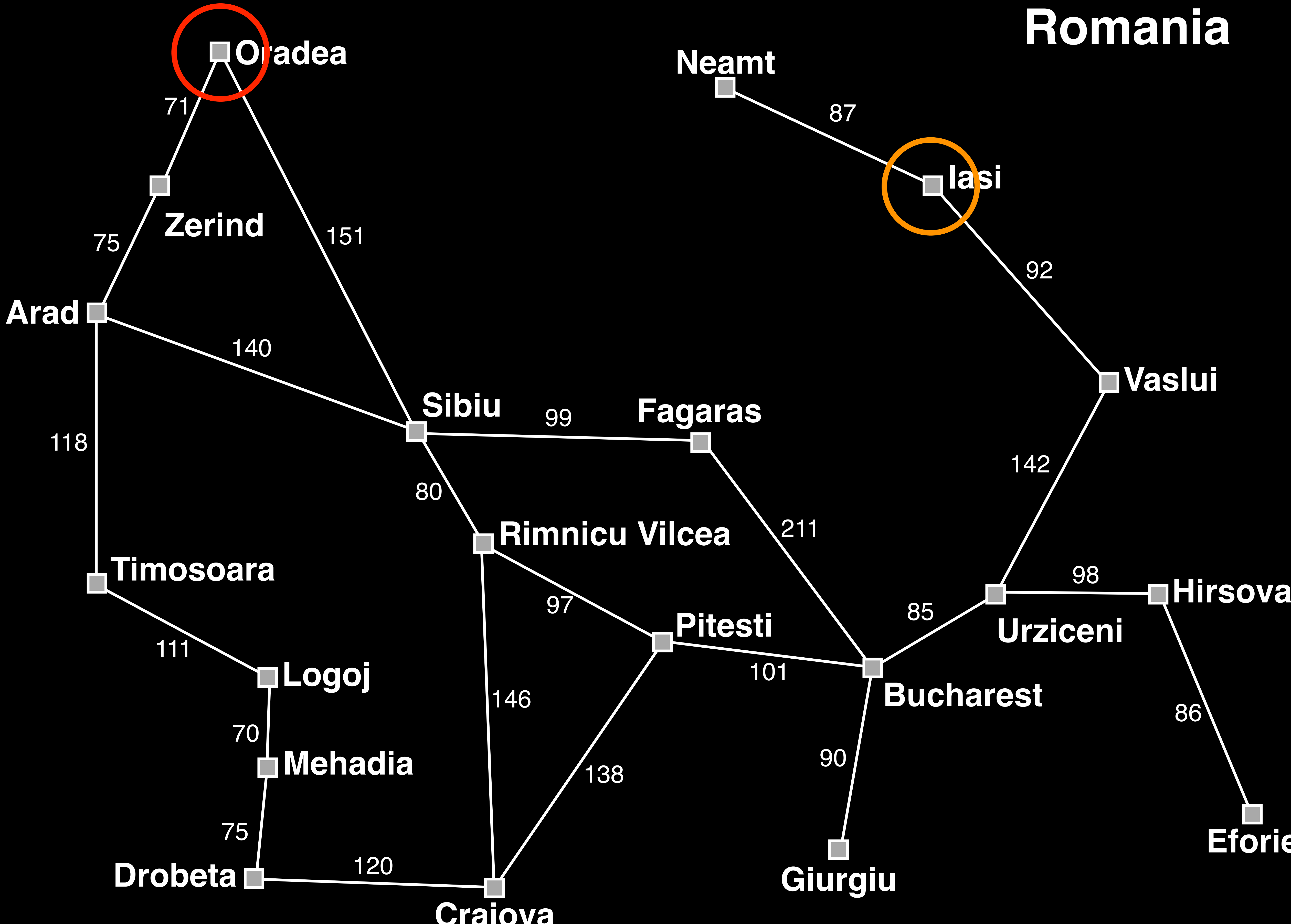

Romania
Oradea
Neamt
71
87
Zerind
Iasi
75
151
92
Arad
140
Vaslui
Sibiu
99
Fagaras
118
142
80
Rimnicu Vilcea
211
Timosoara
98
97
Hirsova
85
Pitesti
Urziceni
111
Logoj
101
Bucharest
146
70
86
Mehadia
90
138
75
Eforie
120
Drobeta
Giurgiu
Craiova


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



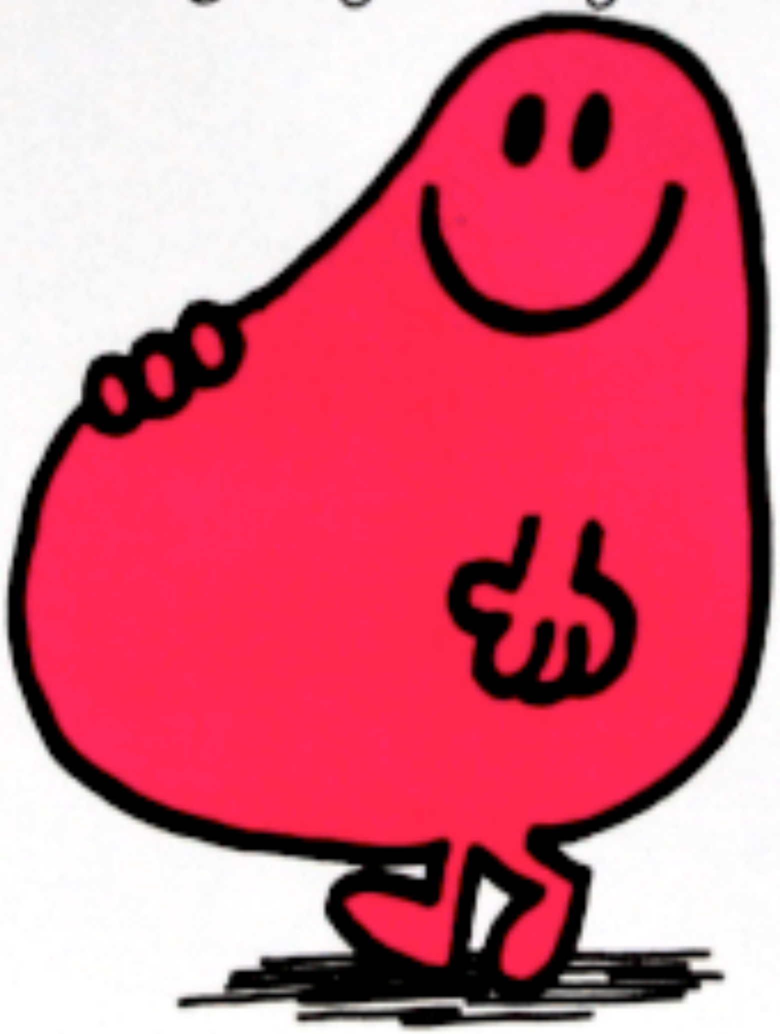
Romania
Oradea
Neamt
87
71
Iasi
Zerind
151
75
92
Arad
140
Vaslui
Sibiu
99
Fagaras
118
142
80
Rimnicu Vilcea
211
Timosoara
98
Hirsova
97
Pitesti
85
Urziceni
111
Logoj
101
Bucharest
146
86
70
Mehadia
90
138
75
Eforie
Drobeta
120
Giurgiu
Craiova


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


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

# MR. GREEDY

by Roger Hargreaves



# Greedy Best-First Search

# Greedy Best-First Search

Completeness

# Greedy Best-First Search

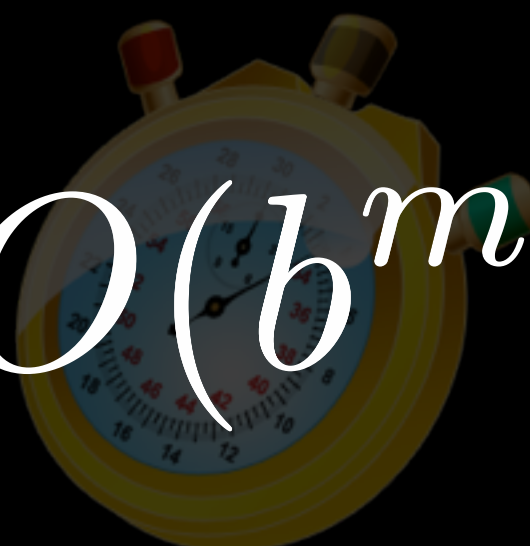

Completeness


Optimality

# Greedy Best-First Search

Completeness

Optimality



$$O(b^m)$$

Time Complexity

# Greedy Best-First Search

Completeness

Optimality

$$O(b^m)$$

Time Complexity

$$O(b^m)$$

Space Complexity

Evaluation function

$$f(n)$$

Evaluation function

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

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

Evaluation function

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

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

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

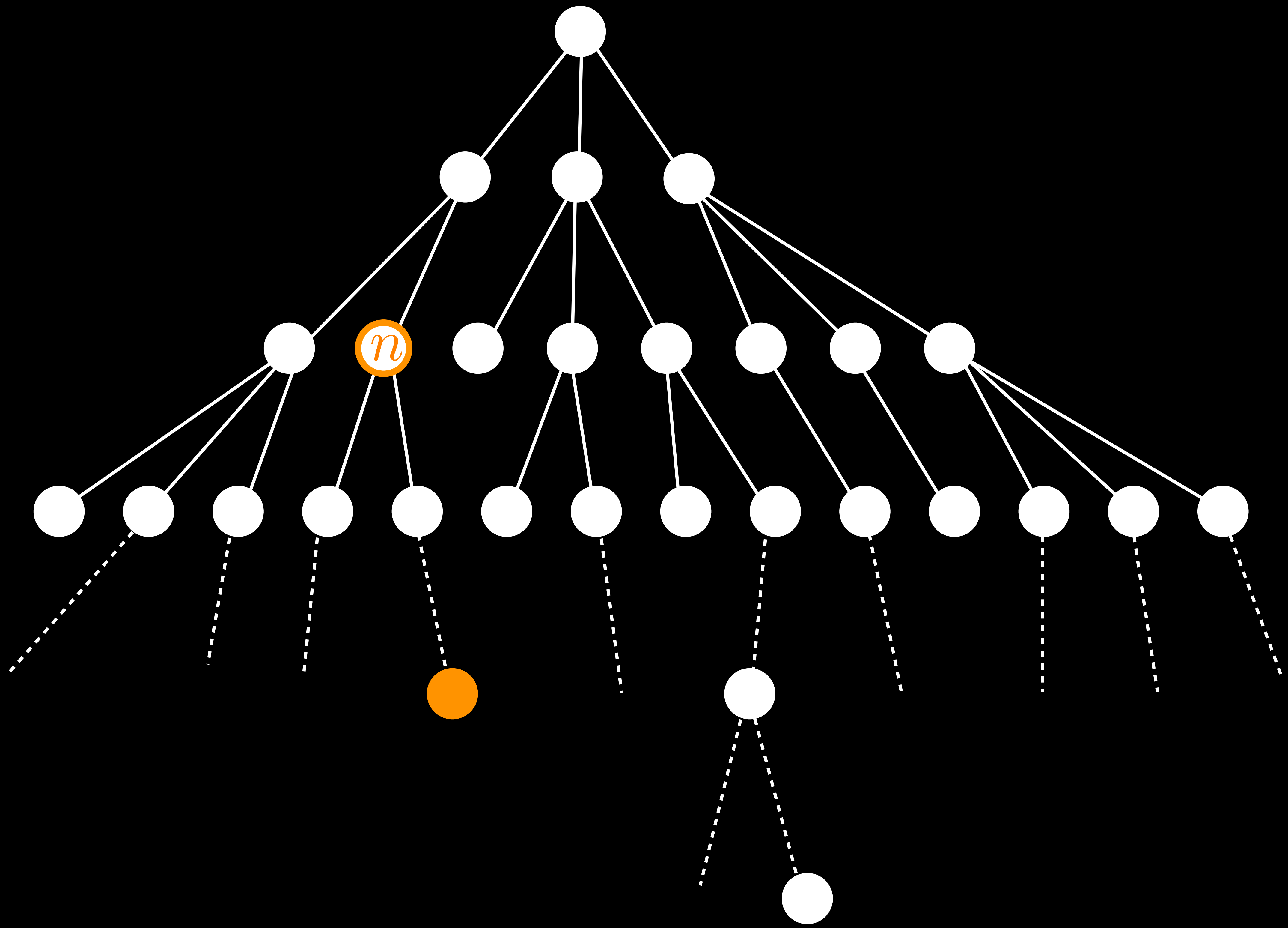
# A*

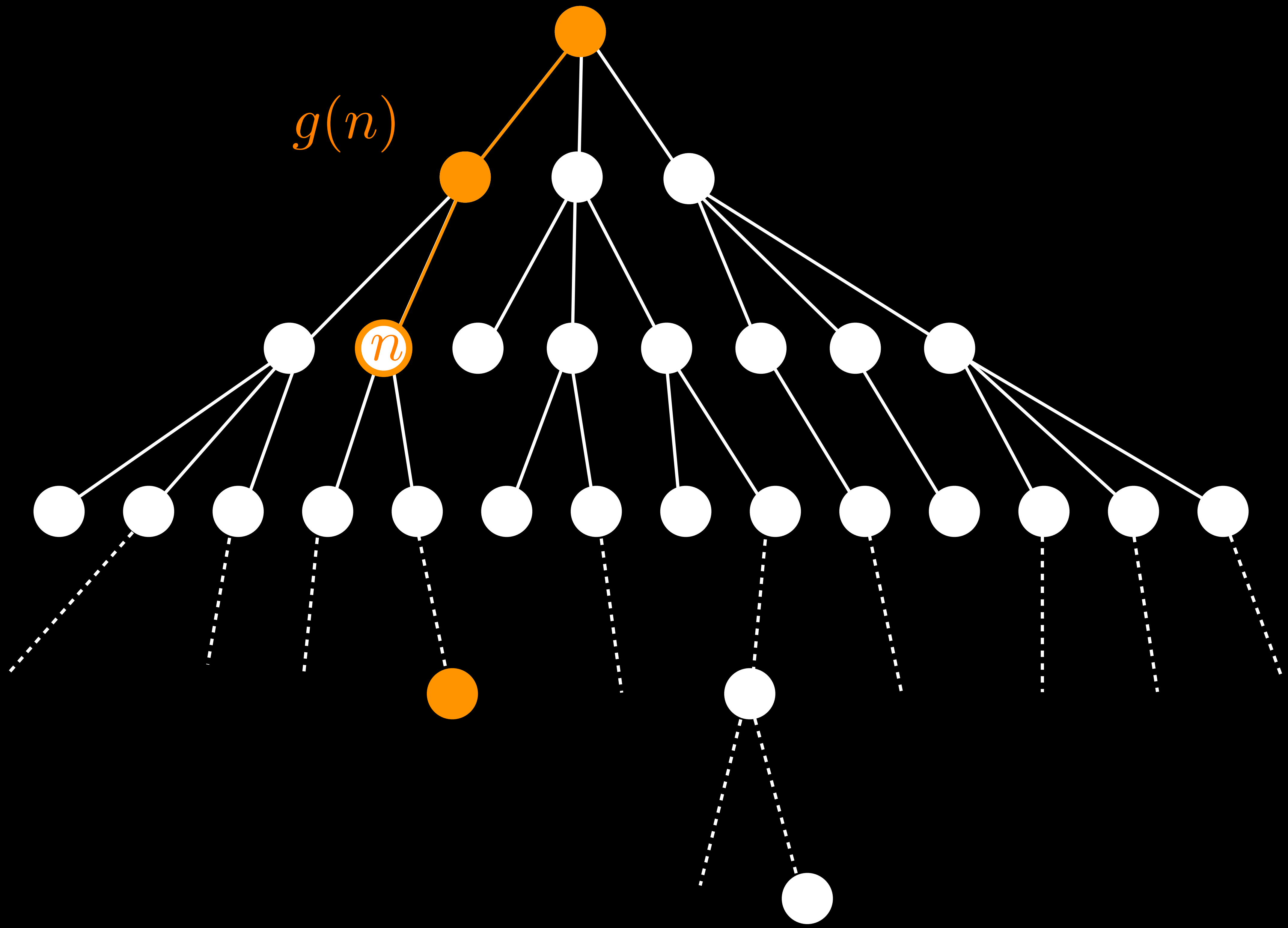
Evaluation function

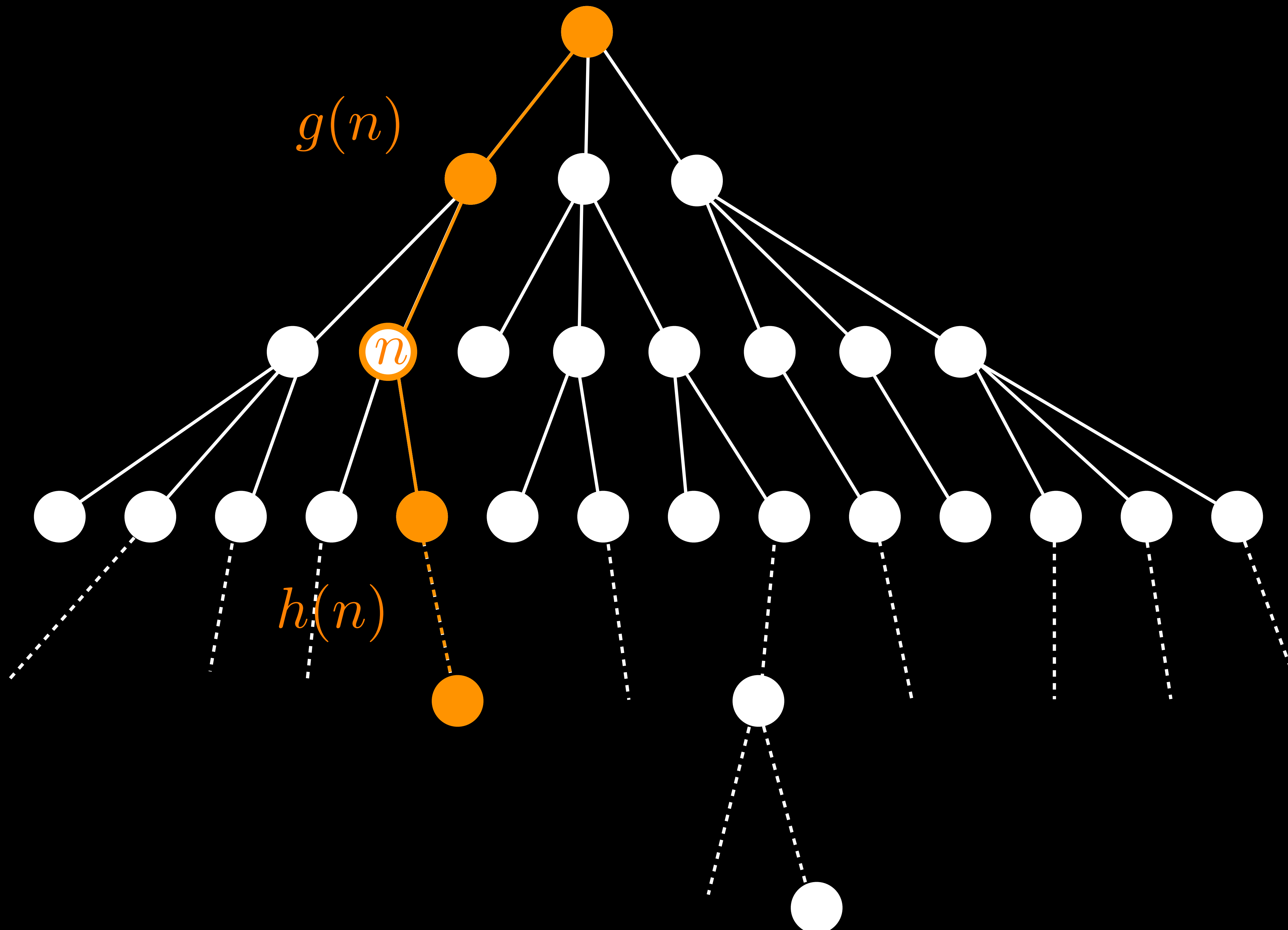
$$f(n) = g(n) + h(n)$$

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

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


n


g(n)
n


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

Evaluation function

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

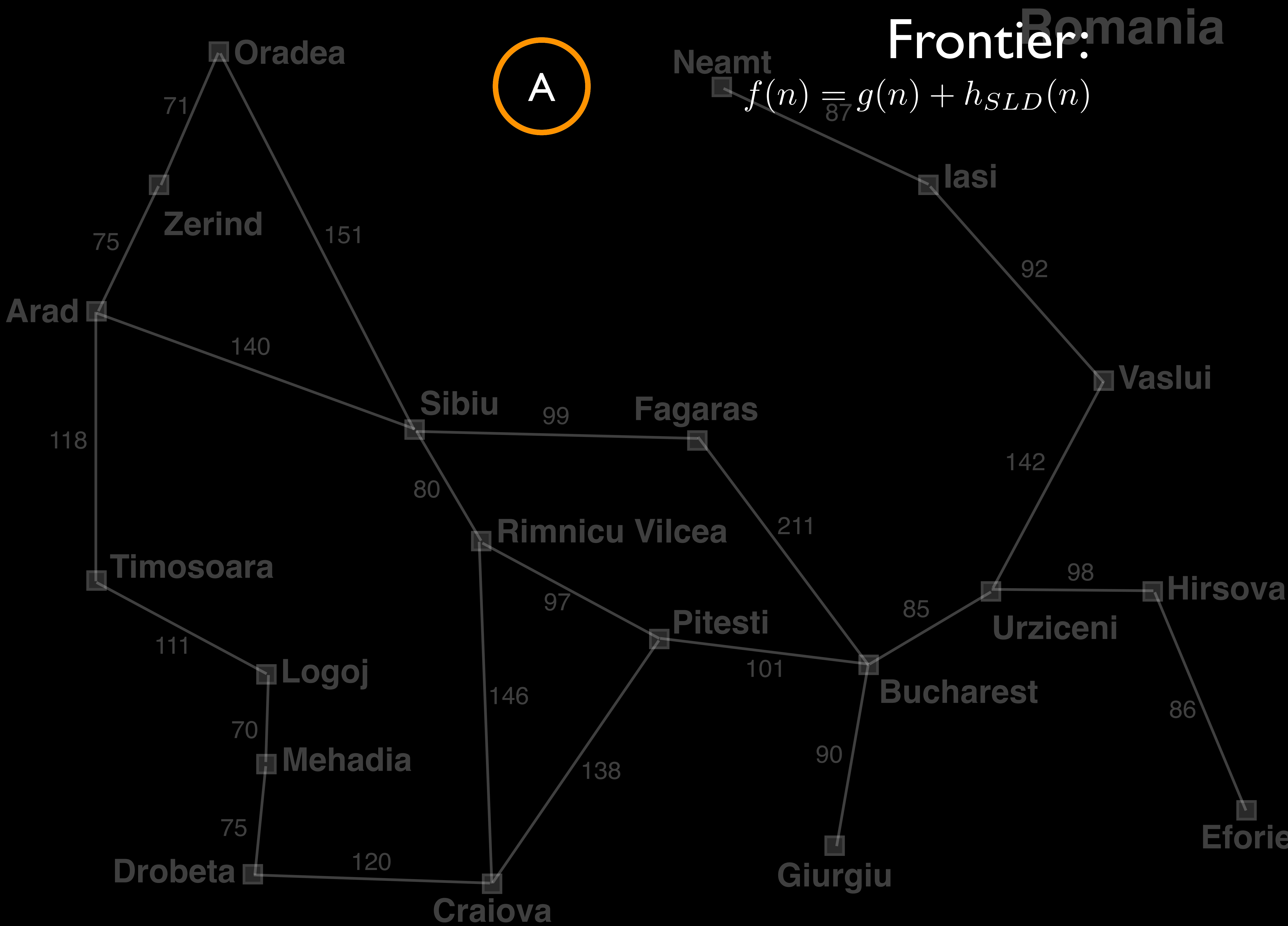
$=$ Estimated cost of cheapest solution through *n*

Frontier: A 0+366 =366

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

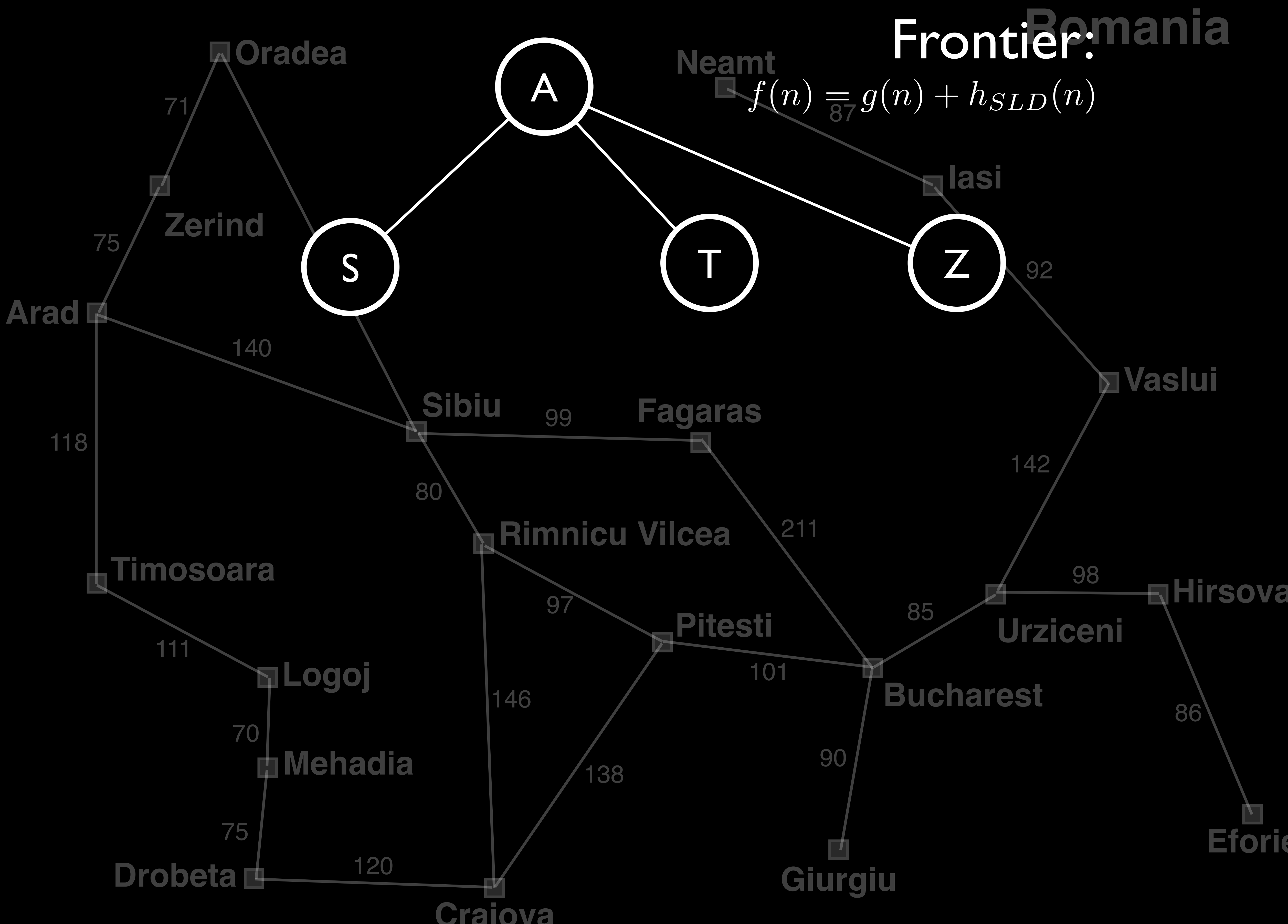
## Frontier:

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



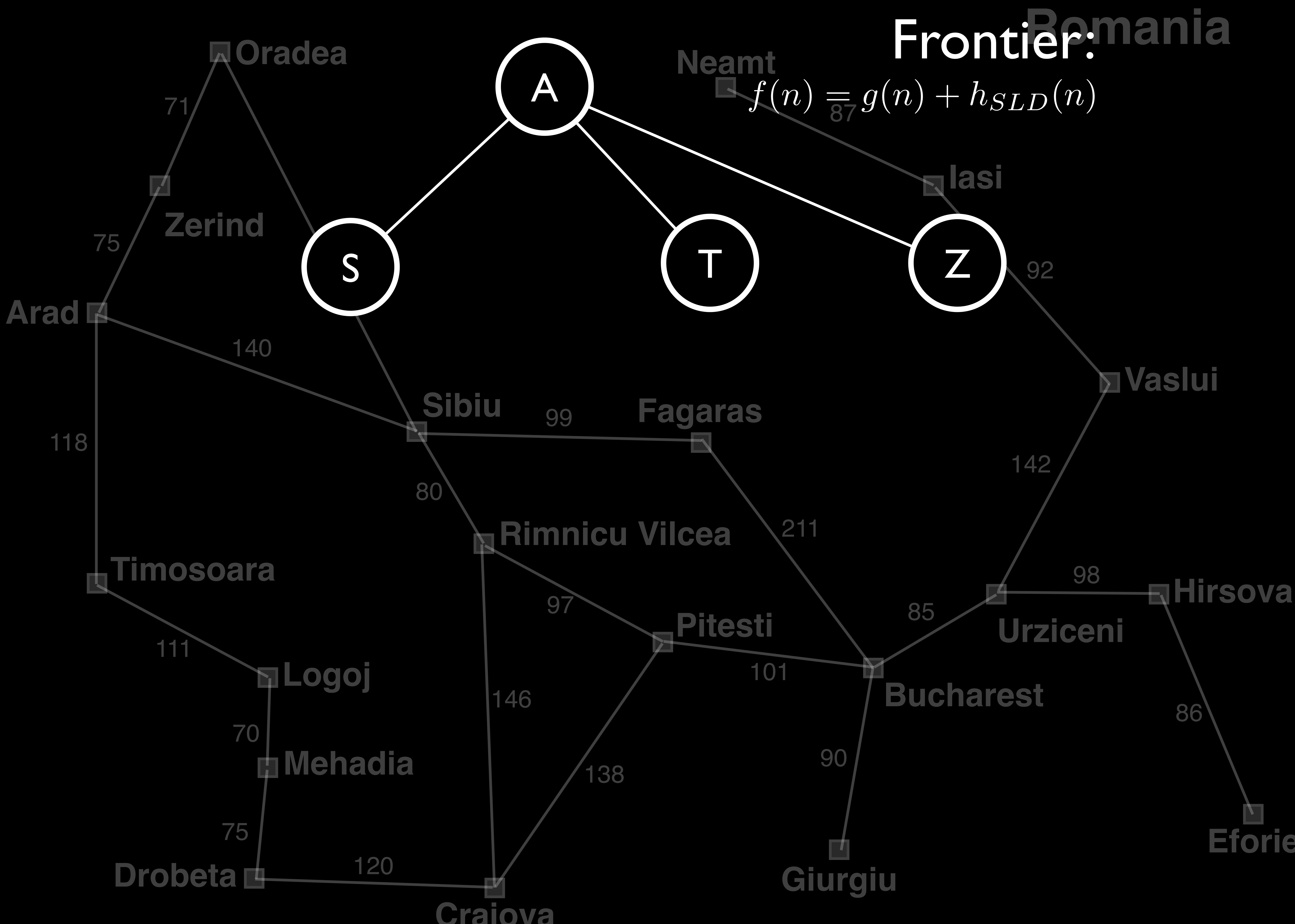
Frontier:

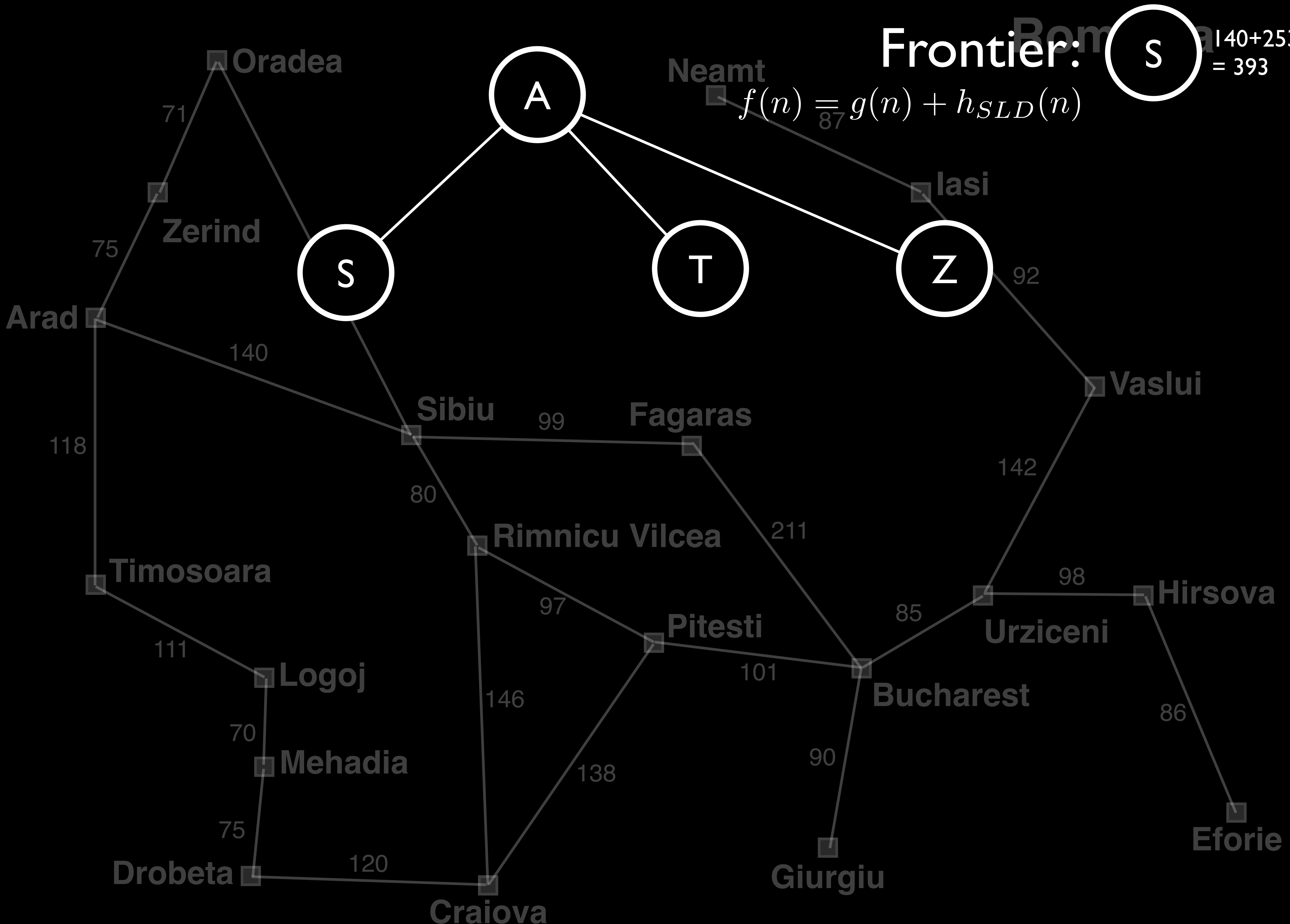
$$f(n) = g(n) + h_{SLD}(n)$$

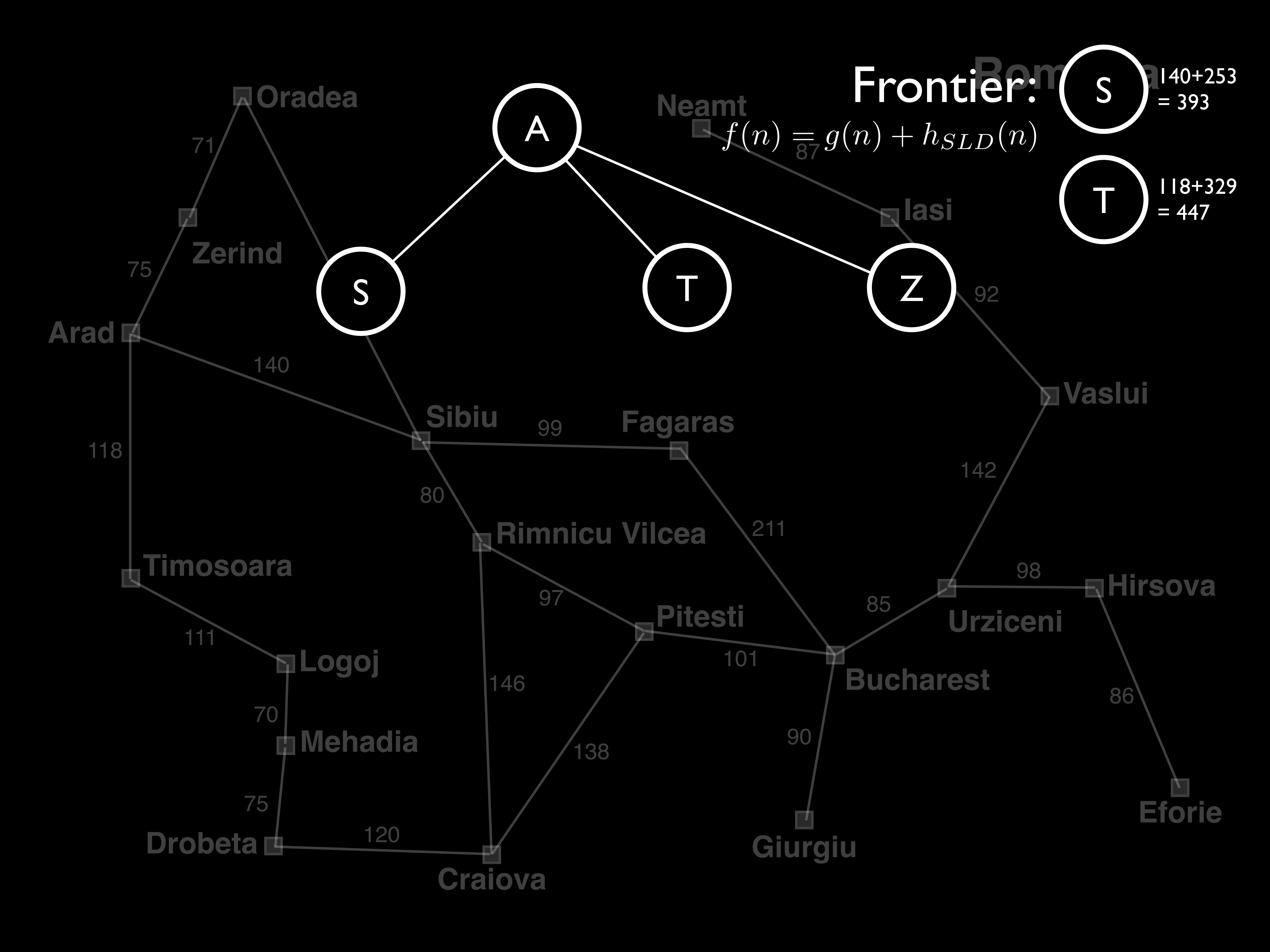


# Frontier:

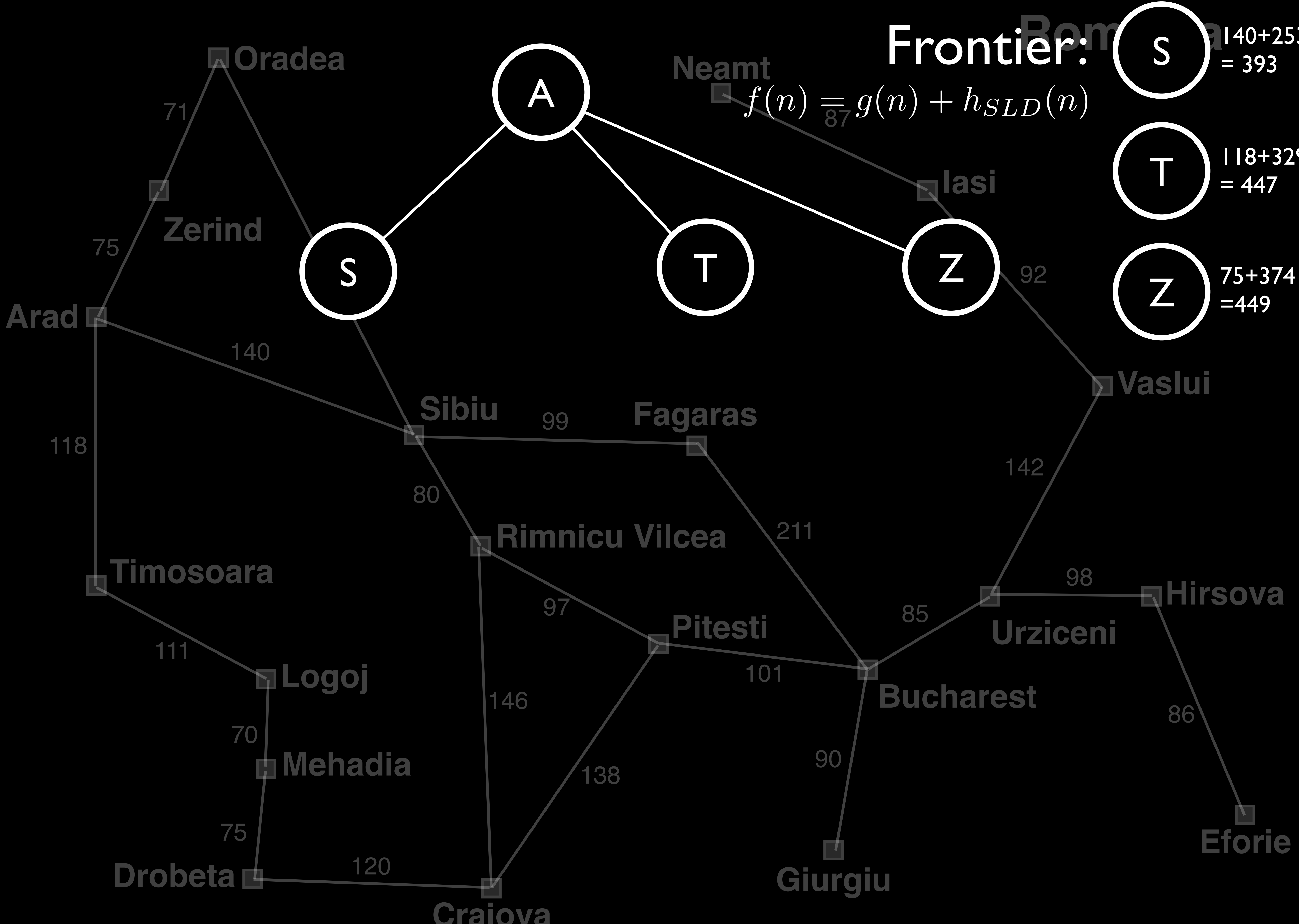
$$f(n) = g(n) + h_{SLD}(n)$$




Frontier:
S
140+253
= 393
$f(n) = g(n) + h_{SLD}(n)$
Romania
A
S
T
Z
Oradea
Neamt
71
87
Iasi
Zerind
75
92
Arad
140
Vaslui
Sibiu
99
Fagaras
118
142
80
Rimnicu Vilcea
211
Timosoara
98
Hirsova
97
85
Pitesti
Urziceni
111
Logoj
101
Bucharest
146
86
70
Mehadia
90
138
Eforie
75
120
Drobeta
Giurgiu
Craiova

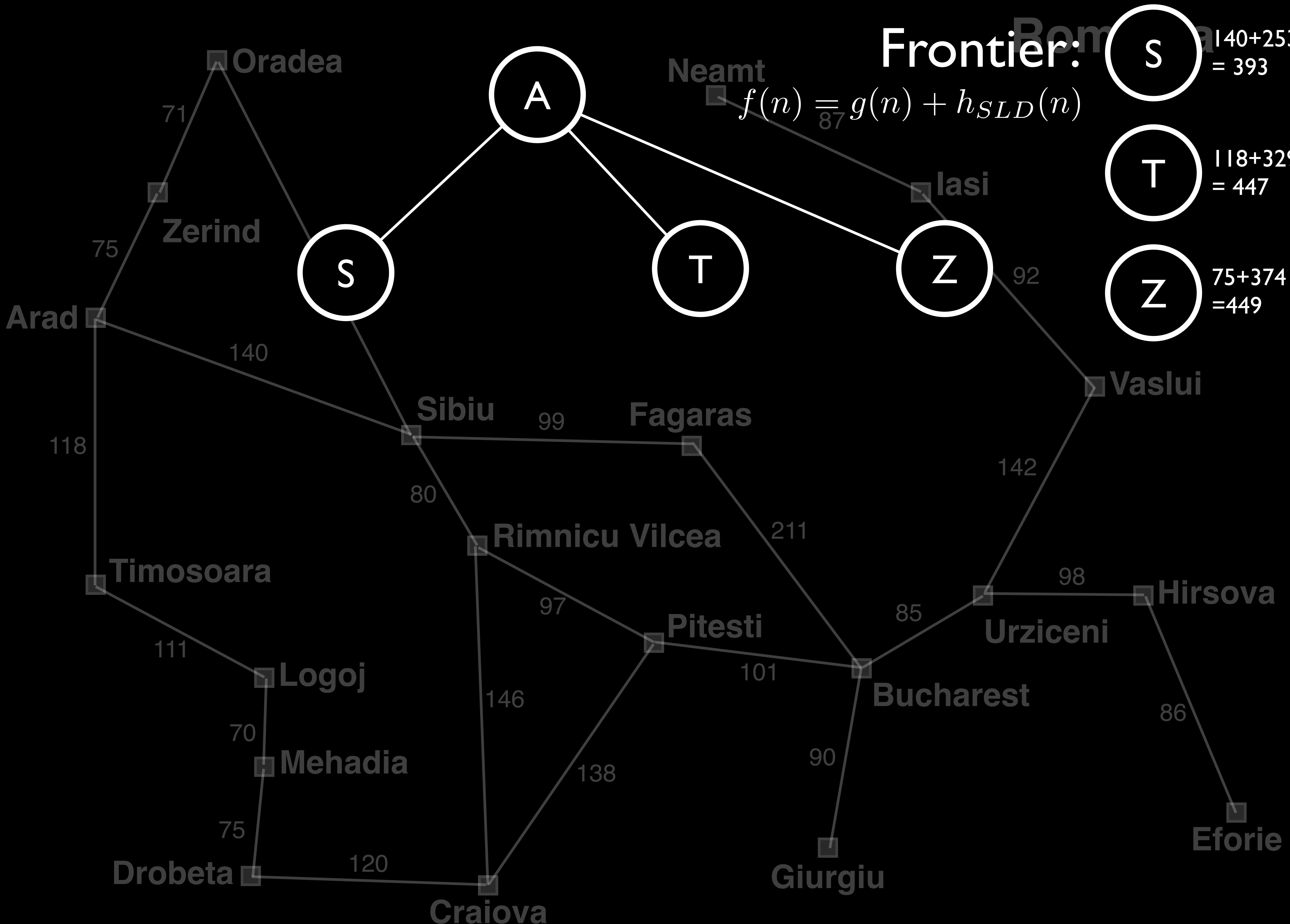


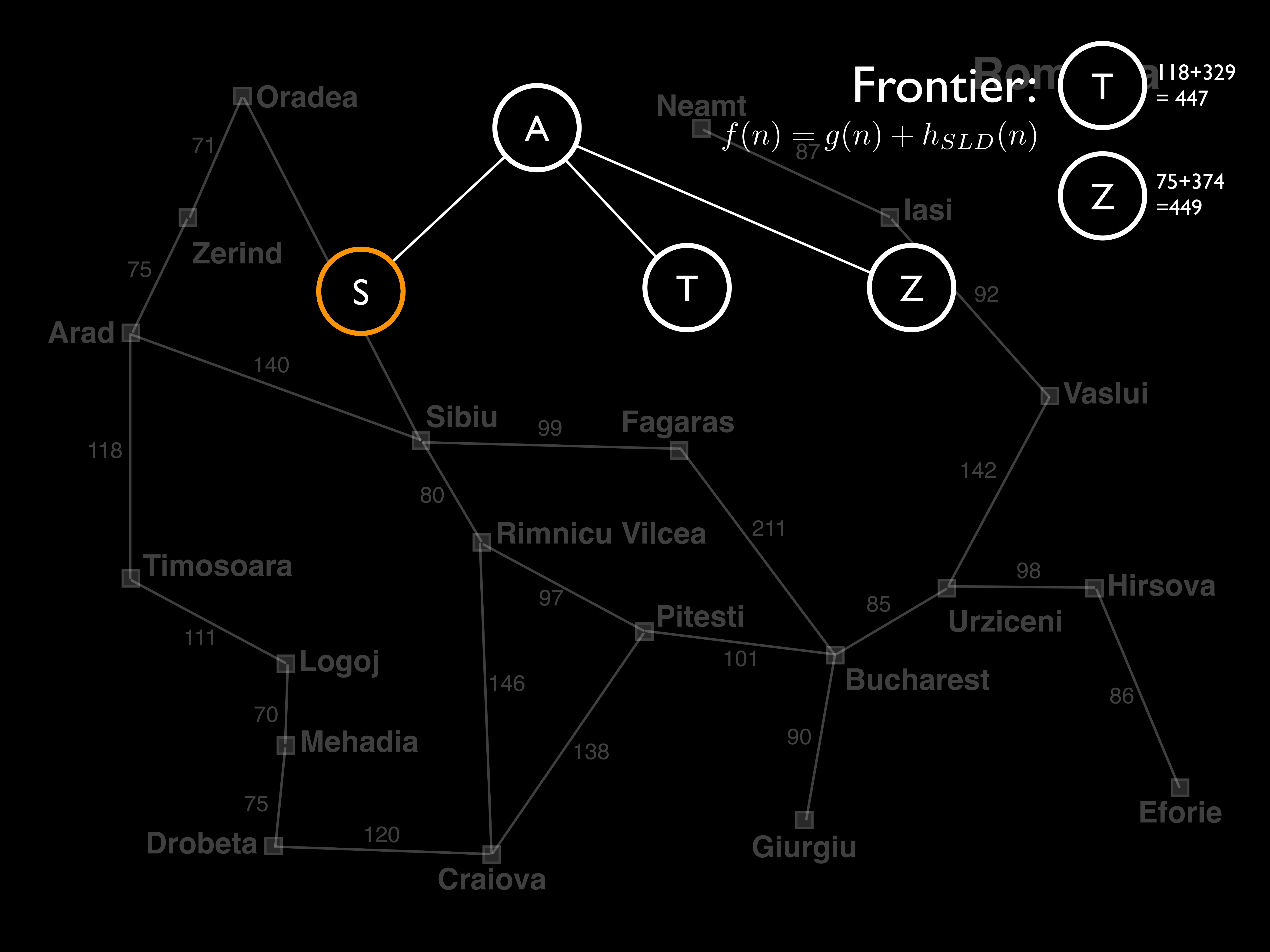
Frontier:
$f(n) = g(n) + h_{SLD}(n)$
S
140+253
= 393
T
118+329
= 447
A
S
T
Z
Oradea
Neamt
Iasi
Zerind
Arad
Sibiu
Fagaras
Vaslui
Rimnicu Vilcea
Timosoara
Urziceni
Hirsova
Pitesti
Logoj
Bucharest
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
75
87
92
140
99
118
80
142
211
98
97
85
111
101
146
70
86
90
138
75
120


Frontier:
$f(n) = g(n) + h_{SLD}(n)$
S 140+253 = 393
T 118+329 = 447
Z 75+374 =449
A
S
T
Z
Romania
Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Vaslui
Rimnicu Vilcea
Timosoara
Pitesti
Urziceni
Hirsova
Logoj
Mehadia
Bucharest
Drobeta
Craiova
Giurgiu
Eforie
71
75
87
92
140
99
118
80
142
211
97
98
85
111
101
146
86
70
90
138
75
120

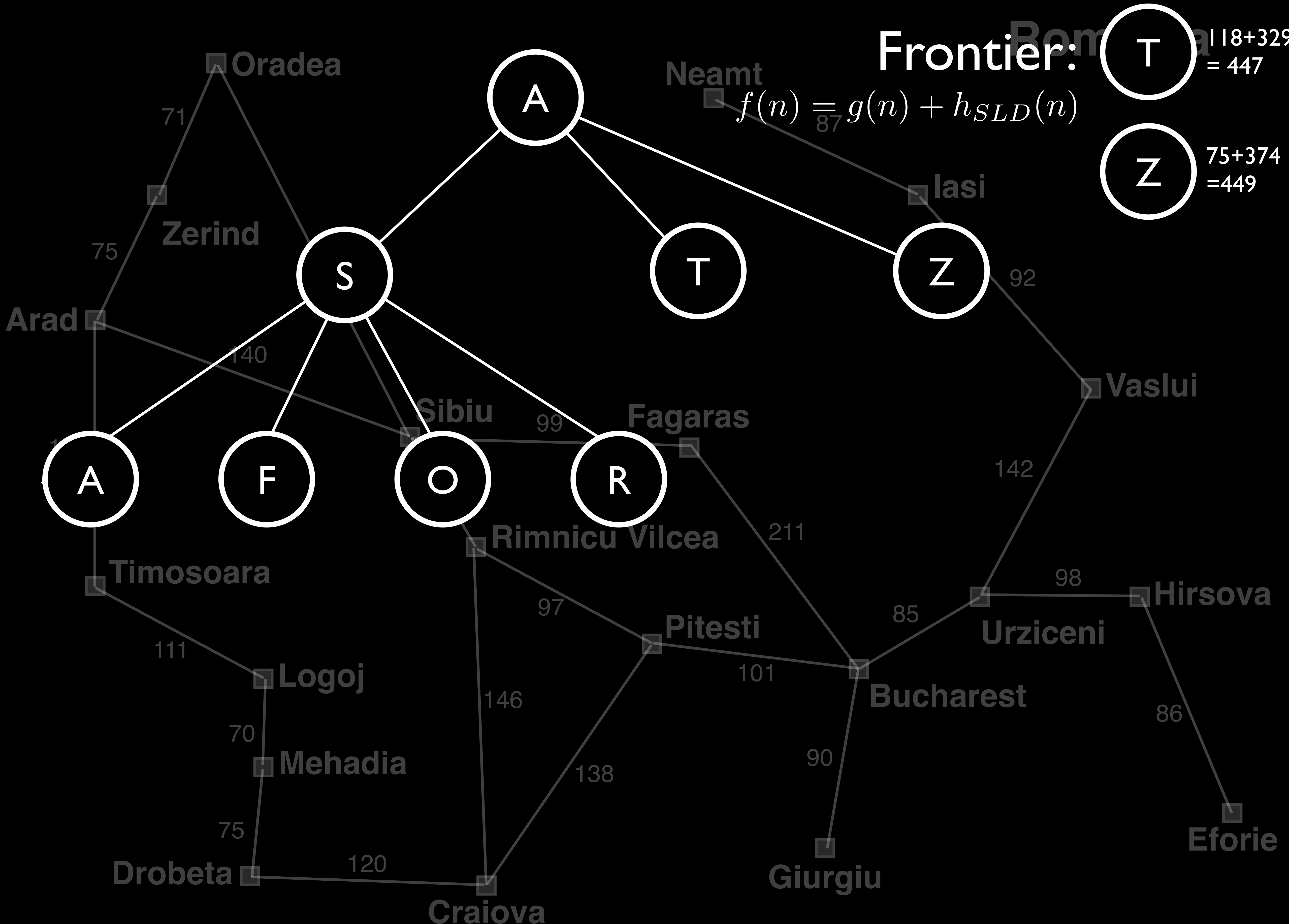
## Frontier:

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



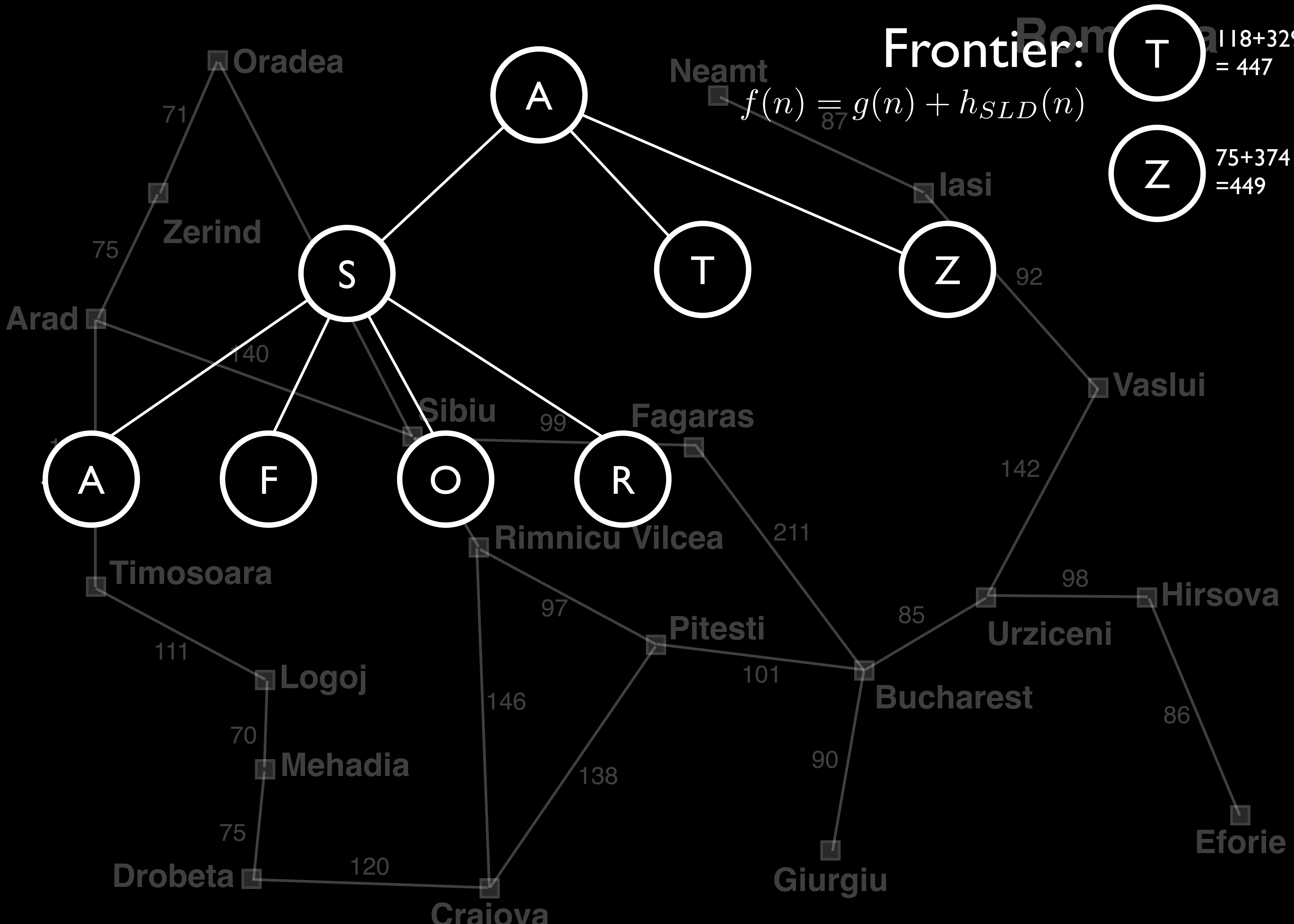


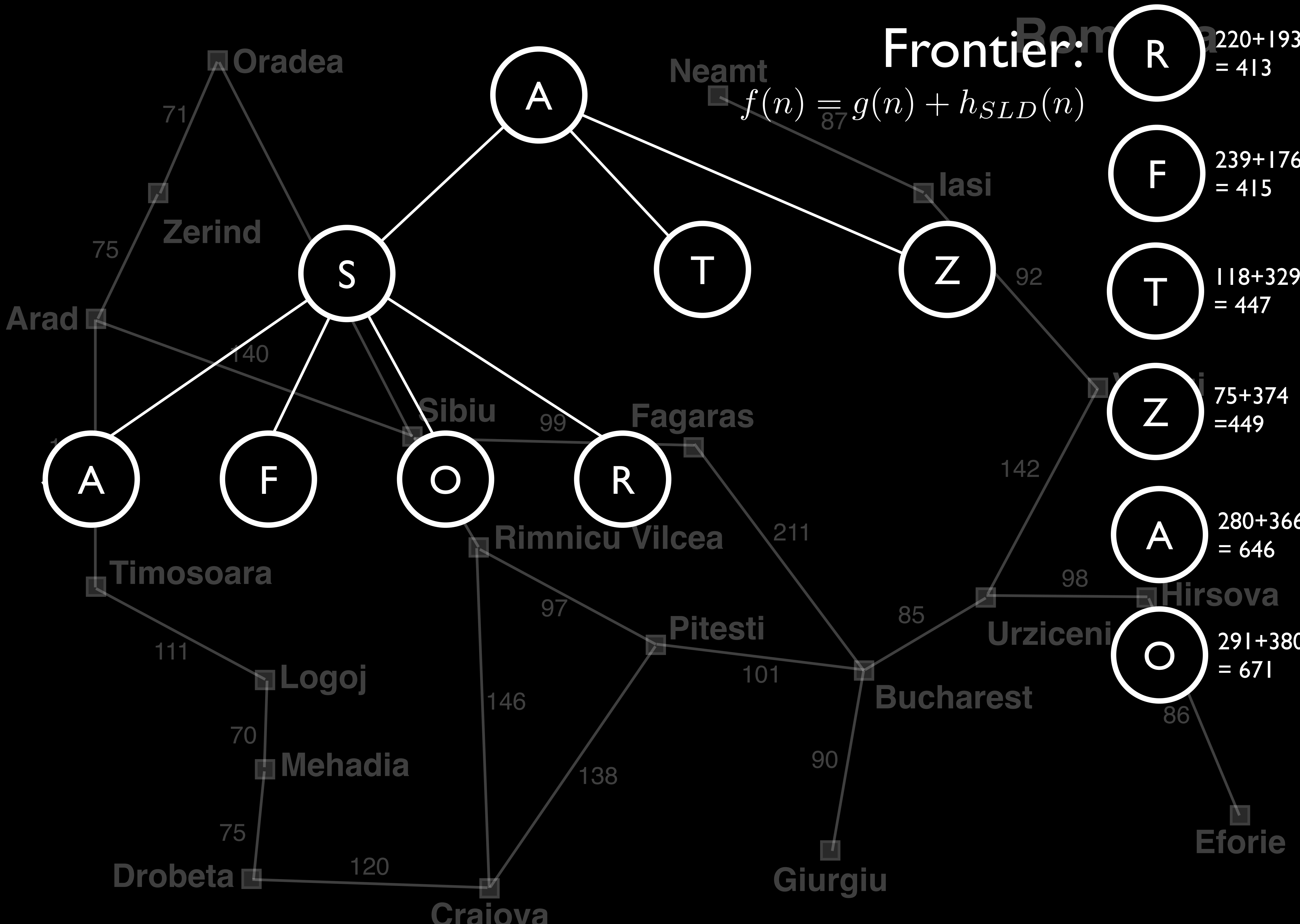
Frontier:
$f(n) = g(n) + h_{SLD}(n)$
T
118+329
= 447
Z
75+374
=449
A
S
T
Z
Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Vaslui
Rimnicu Vilcea
Timosoara
Pitesti
Hirsova
Urziceni
Logoj
Bucharest
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
87
75
92
140
99
118
142
80
211
98
97
85
111
101
146
86
70
90
138
75
120


Frontier:
T
118+329
= 447
Z
75+374
=449
$f(n) = g(n) + h_{SLD}(n)$
A
S
T
Z
A
F
O
R
Oradea
Neamt
Iasi
Zerind
Arad
Sibiu
Fagaras
Vaslui
Rimnicu Vilcea
Timosoara
Urziceni
Hirsova
Pitesti
Logoj
Bucharest
Mehadia
Eforie
Drobeta
Craiova
Giurgiu
71
87
75
92
140
99
142
211
98
97
85
111
101
146
86
70
90
138
75
120

Frontier:

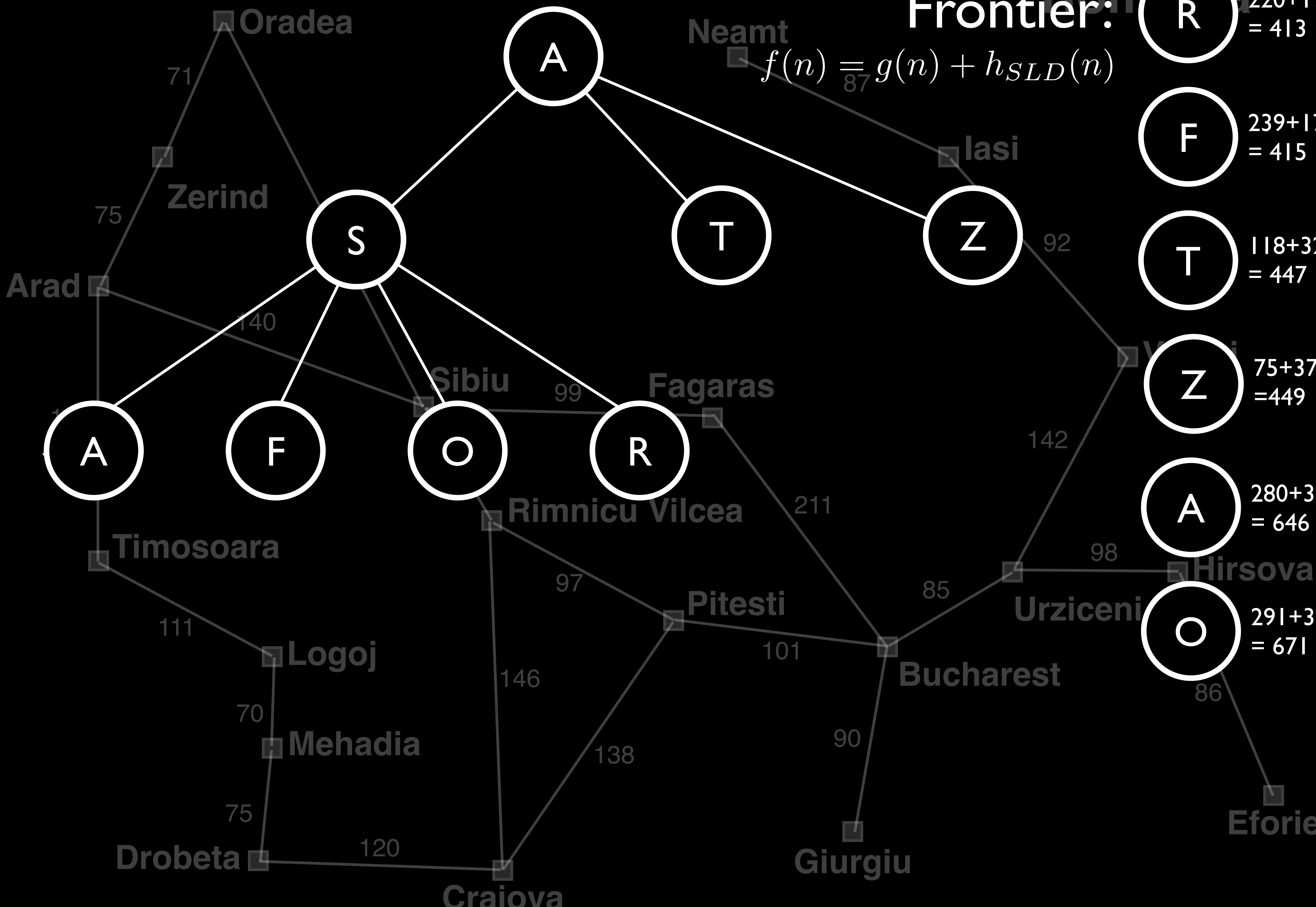
$$f(n) = g(n) + h_{SLD}(n)$$





Frontier:
$f(n) = g(n) + h_{SLD}(n)$
R 220+193 = 413
F 239+176 = 415
T 118+329 = 447
Z 75+374 =449
A 280+366 = 646
O 291+380 = 671
A
S
T
Z
A
F
O
R
Oradea
Neamt
Iasi
Zerind
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Timosoara
Logoj
Mehadia
Drobeta
Craiova
Pitesti
Bucharest
Giurgiu
Urziceni
Hirsova
Eforie
71
75
87
92
140
99
142
211
97
98
85
111
101
146
70
86
90
138
75
120

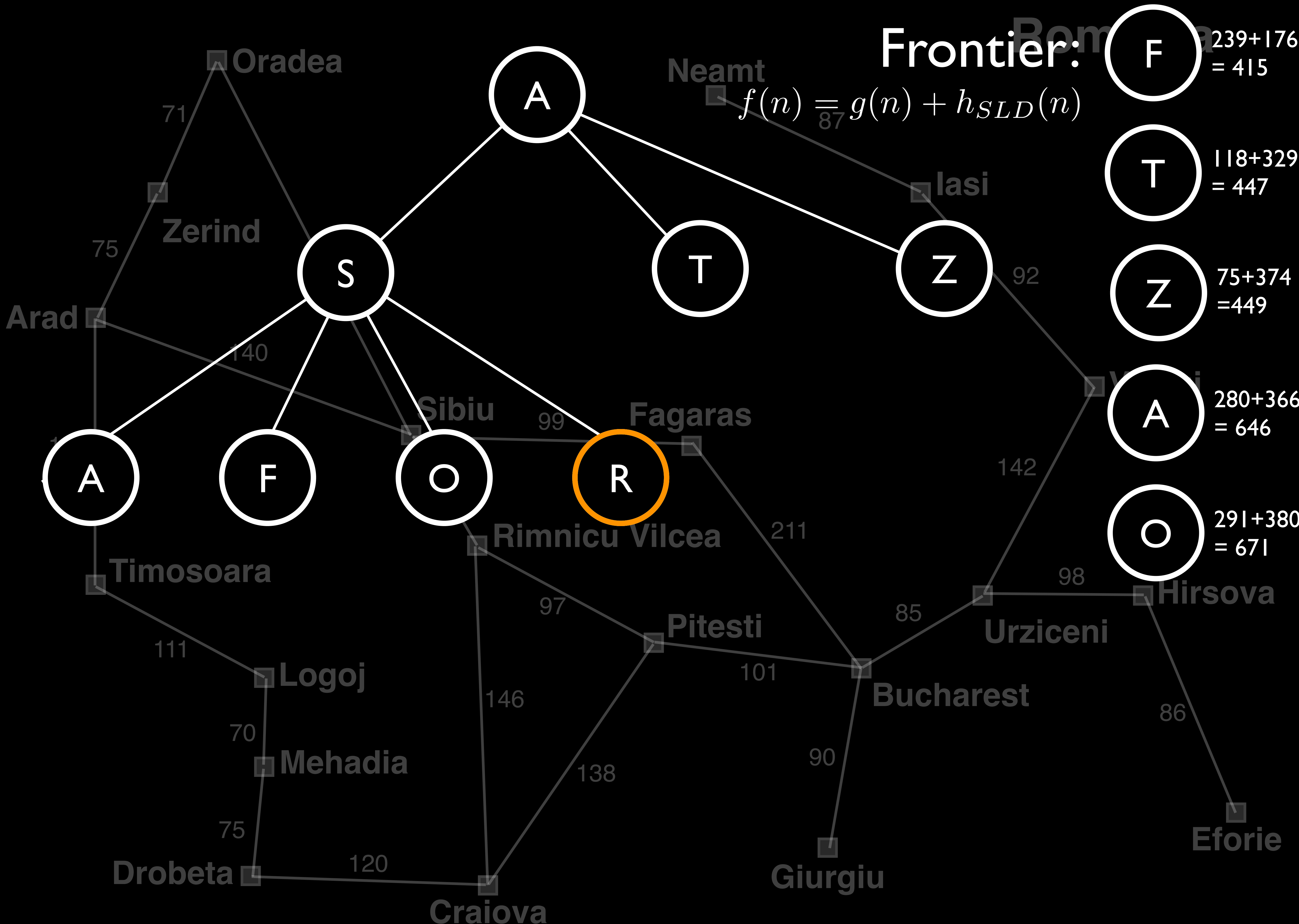
Frontier: $f(n) = g(n) + h_{SLD}(n)$

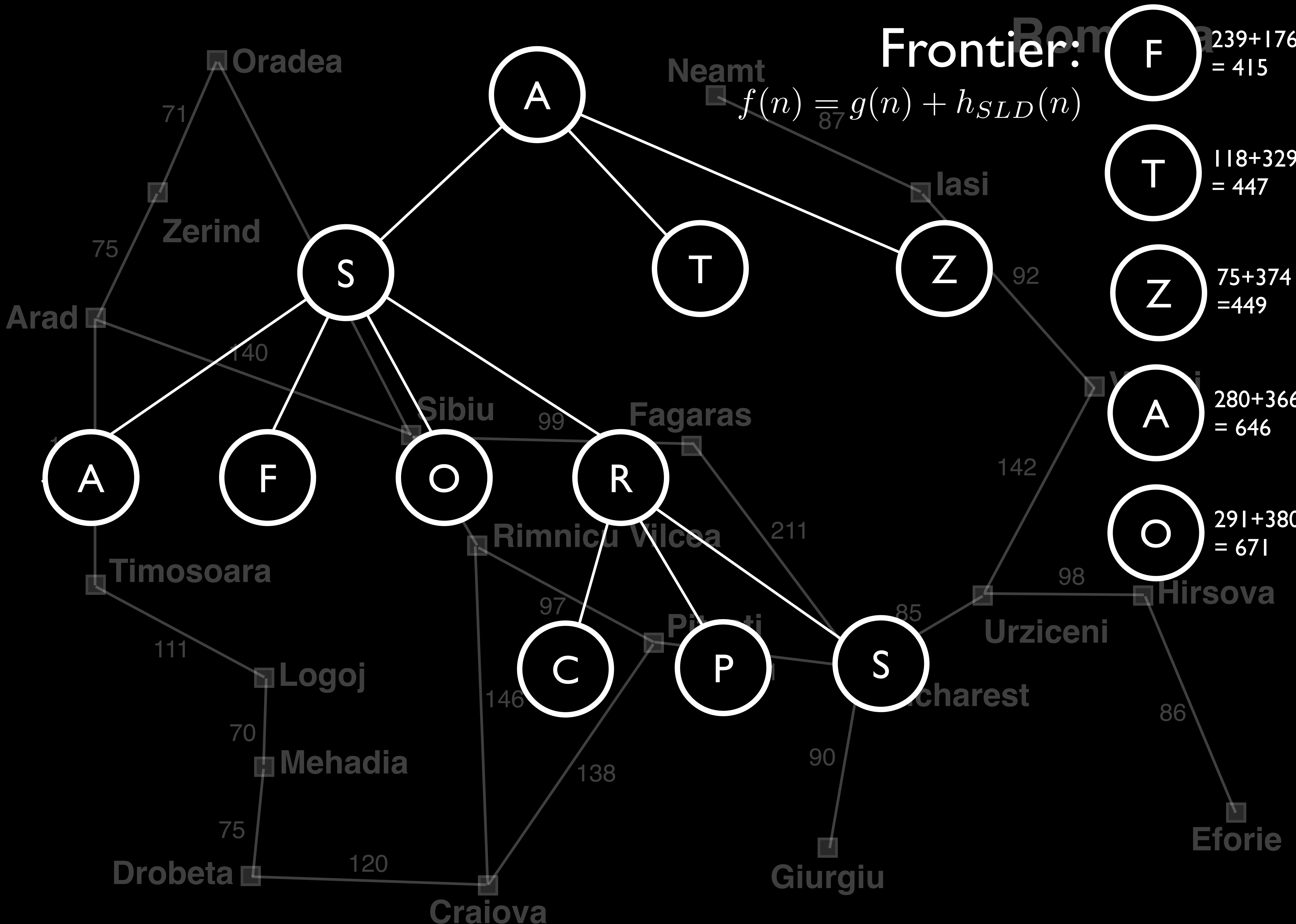



- R: 220+193 = 413
- F: 239+176 = 415
- T: 118+329 = 447
- Z: 75+374 =449
- A: 280+366 = 646
- O: 291+380 = 671

## Frontier:

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

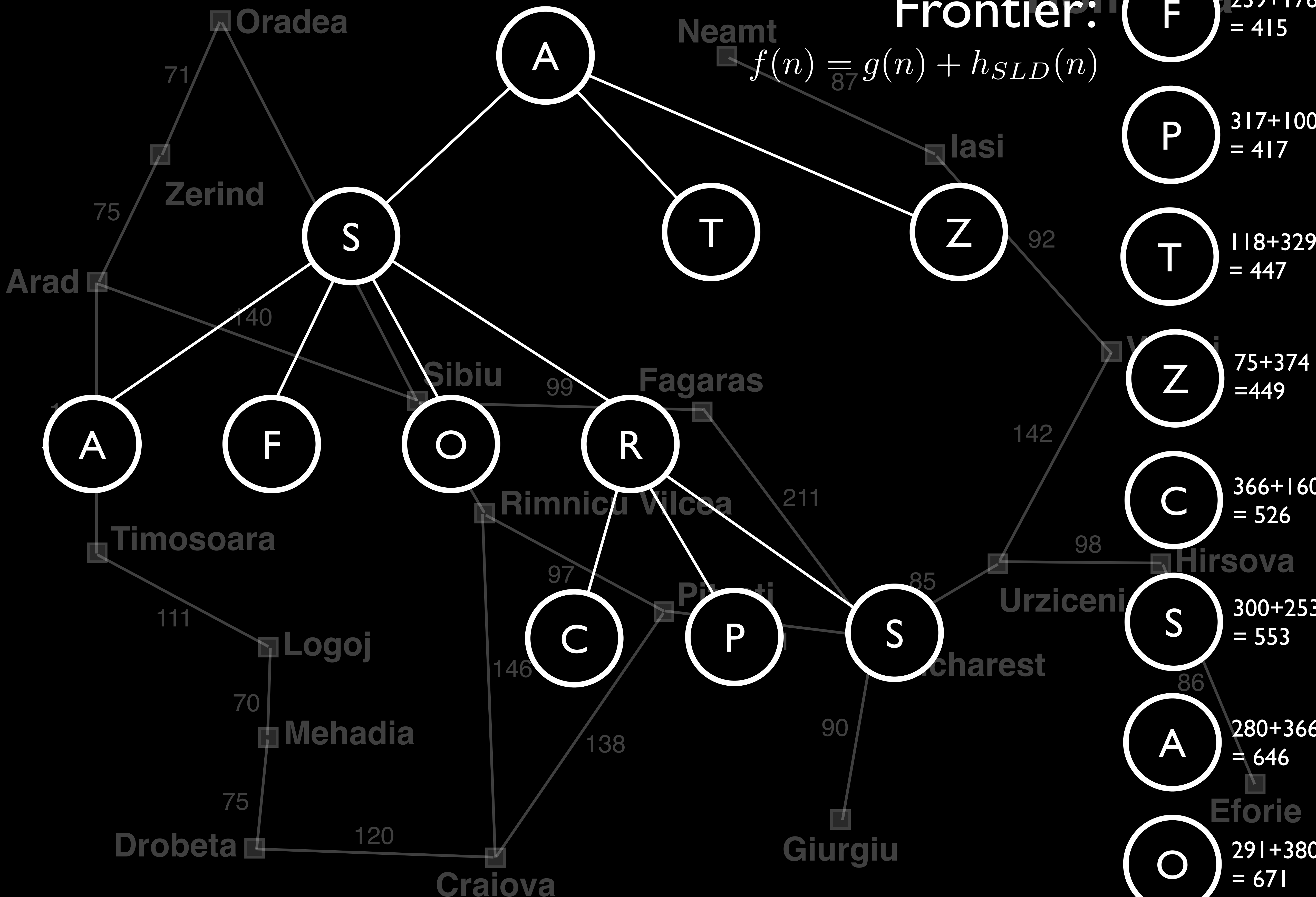


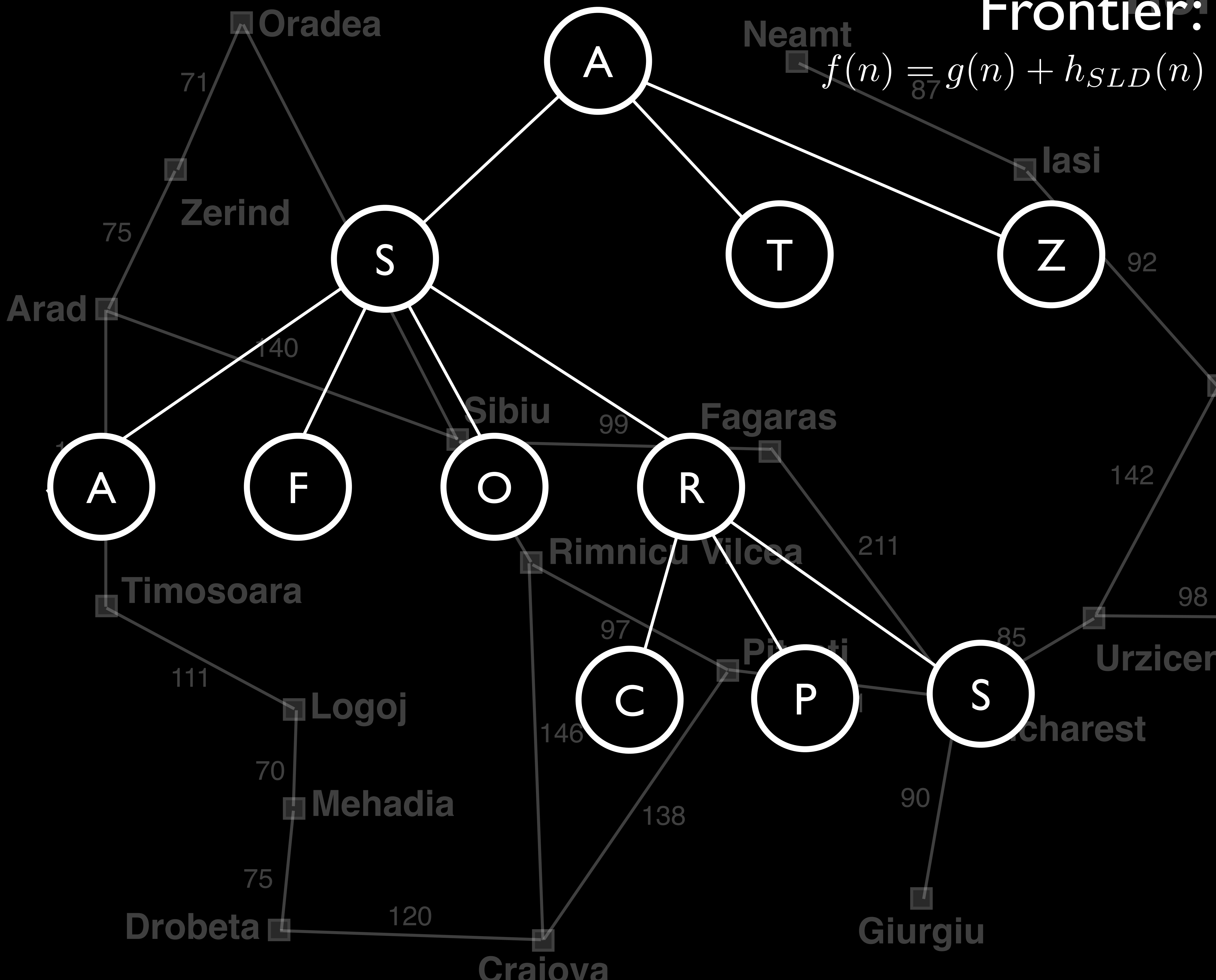


Frontier:
$f(n) = g(n) + h_{SLD}(n)$
F 239+176 = 415
T 118+329 = 447
Z 75+374 =449
A 280+366 = 646
O 291+380 = 671
A
S
T
Z
A
F
O
R
C
P
S
Oradea
Zerind
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Timosoara
Logoj
Mehadia
Drobeta
Craiova
Giurgiu
Urziceni
Hirsova
Eforie
Neamt
Iasi
71
75
140
99
97
211
111
70
75
120
146
138
90
85
98
86
142
92
87

## Frontier:

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





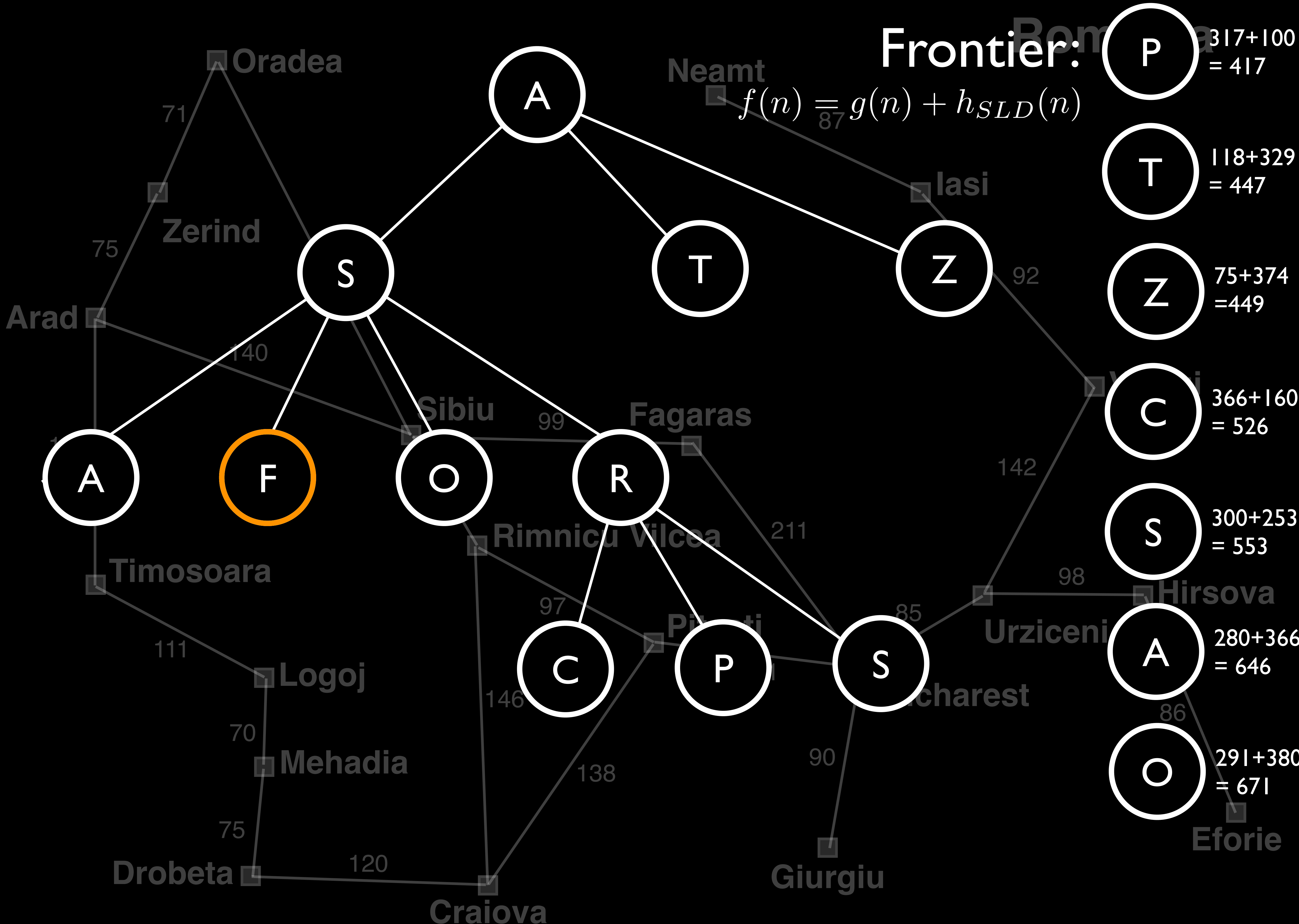

# Frontier:

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

- F: 239+176 = 415
- P: 317+100 = 417
- T: 118+329 = 447
- Z: 75+374 =449
- C: 366+160 = 526
- S: 300+253 = 553
- A: 280+366 = 646
- O: 291+380 = 671

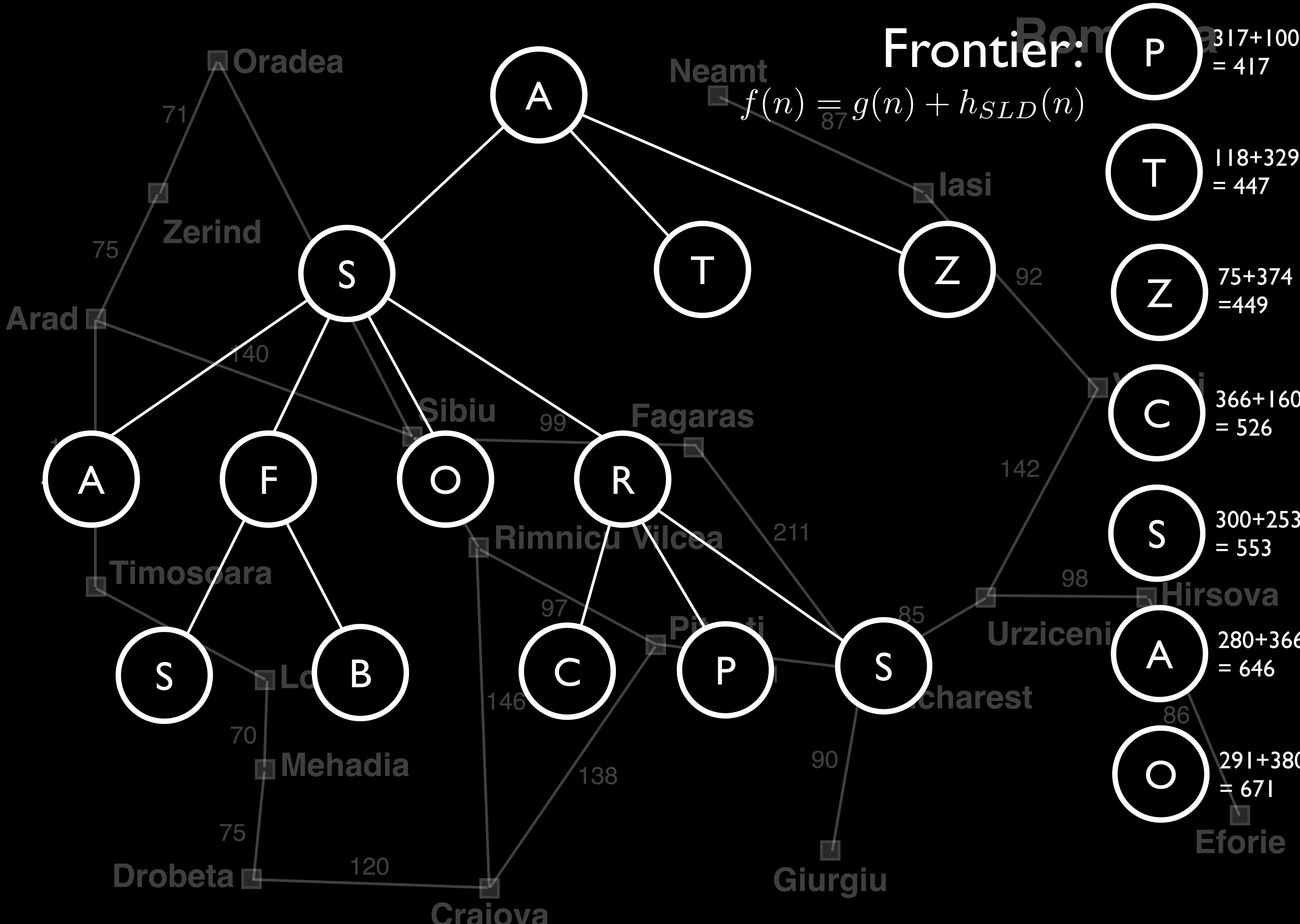
# Frontier:

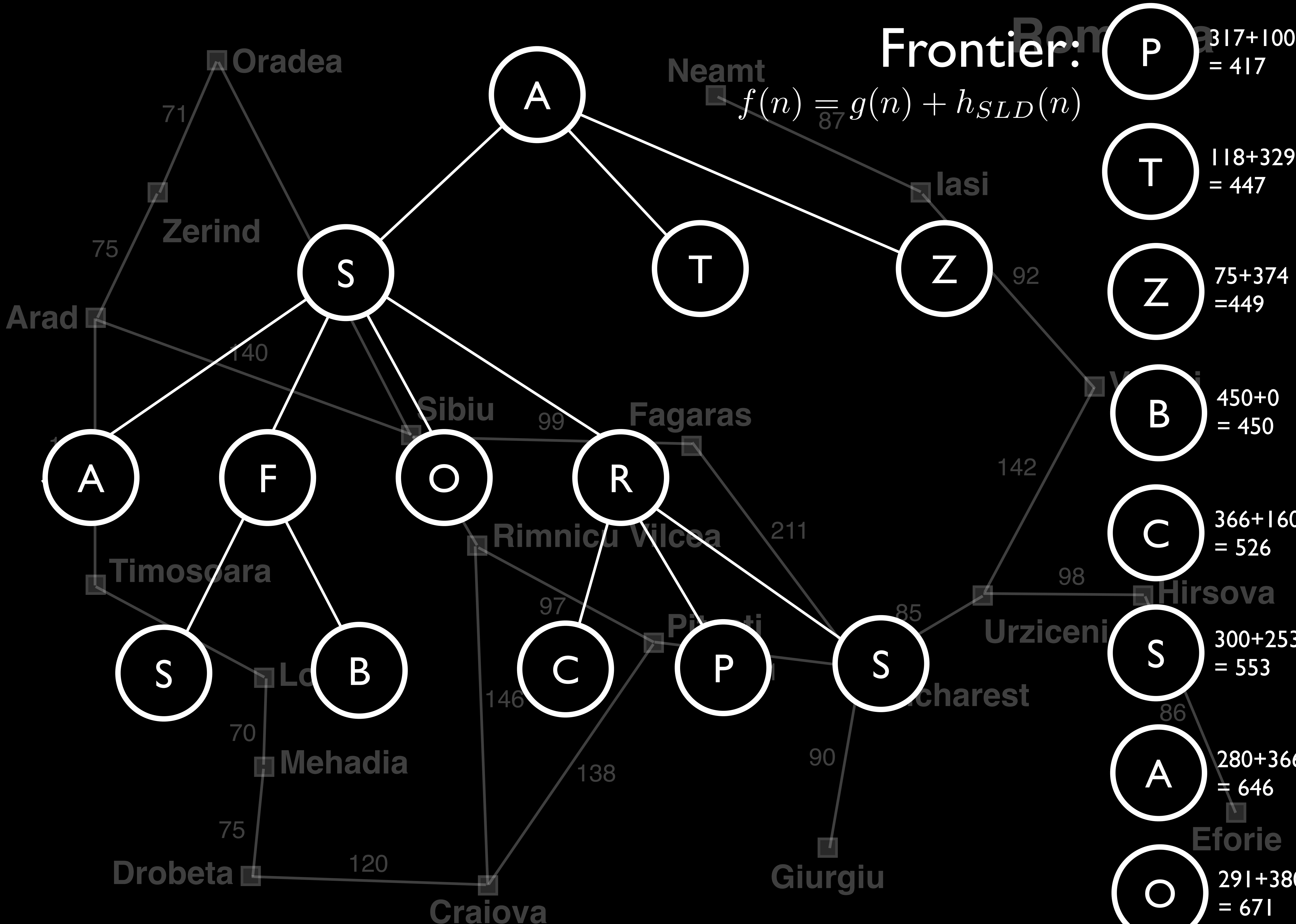
$$f(n) = g(n) + h_{SLD}(n)$$



# Frontier:

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

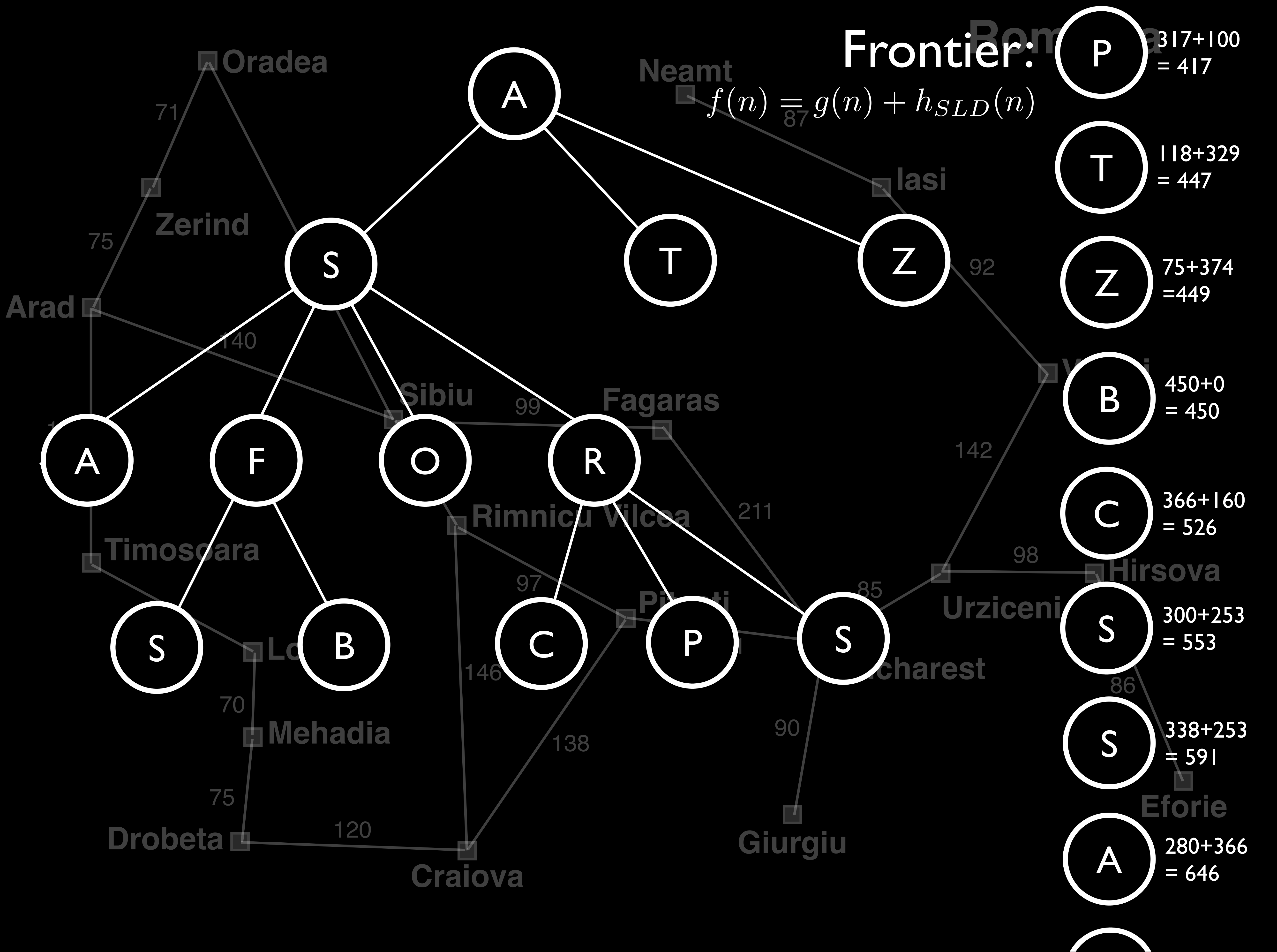




Frontier:
$f(n) = g(n) + h_{SLD}(n)$
A
S
T
Z
A
F
O
R
S
B
C
P
S
P
317+100
= 417
T
118+329
= 447
Z
75+374
=449
B
450+0
= 450
C
366+160
= 526
S
300+253
= 553
A
280+366
= 646
O
291+380
= 671
Oradea
Neamt
Iasi
Zerind
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Urziceni
Hirsova
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
75
87
92
140
99
142
211
97
98
85
146
70
90
138
86
75
120

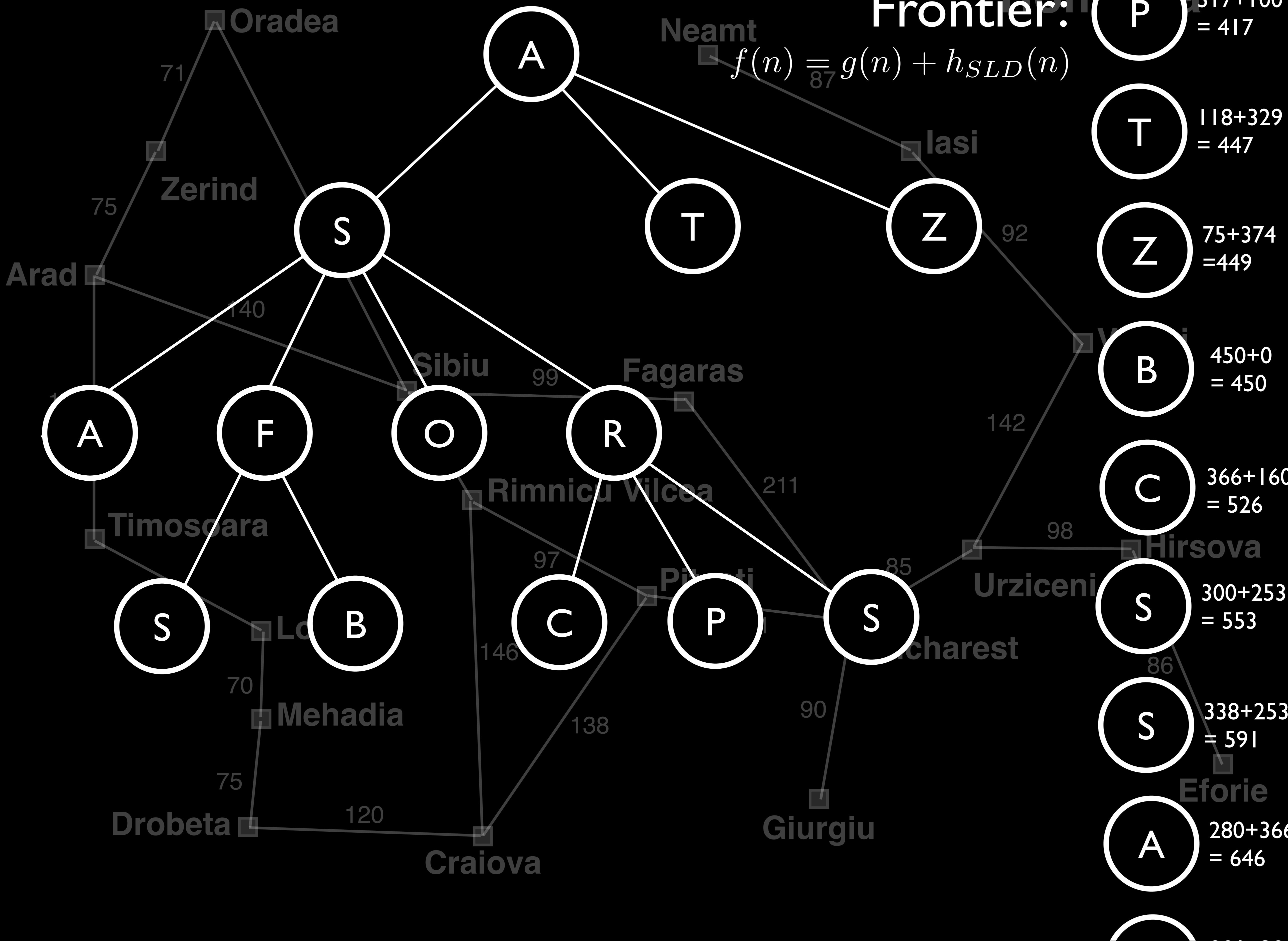
## Frontier:

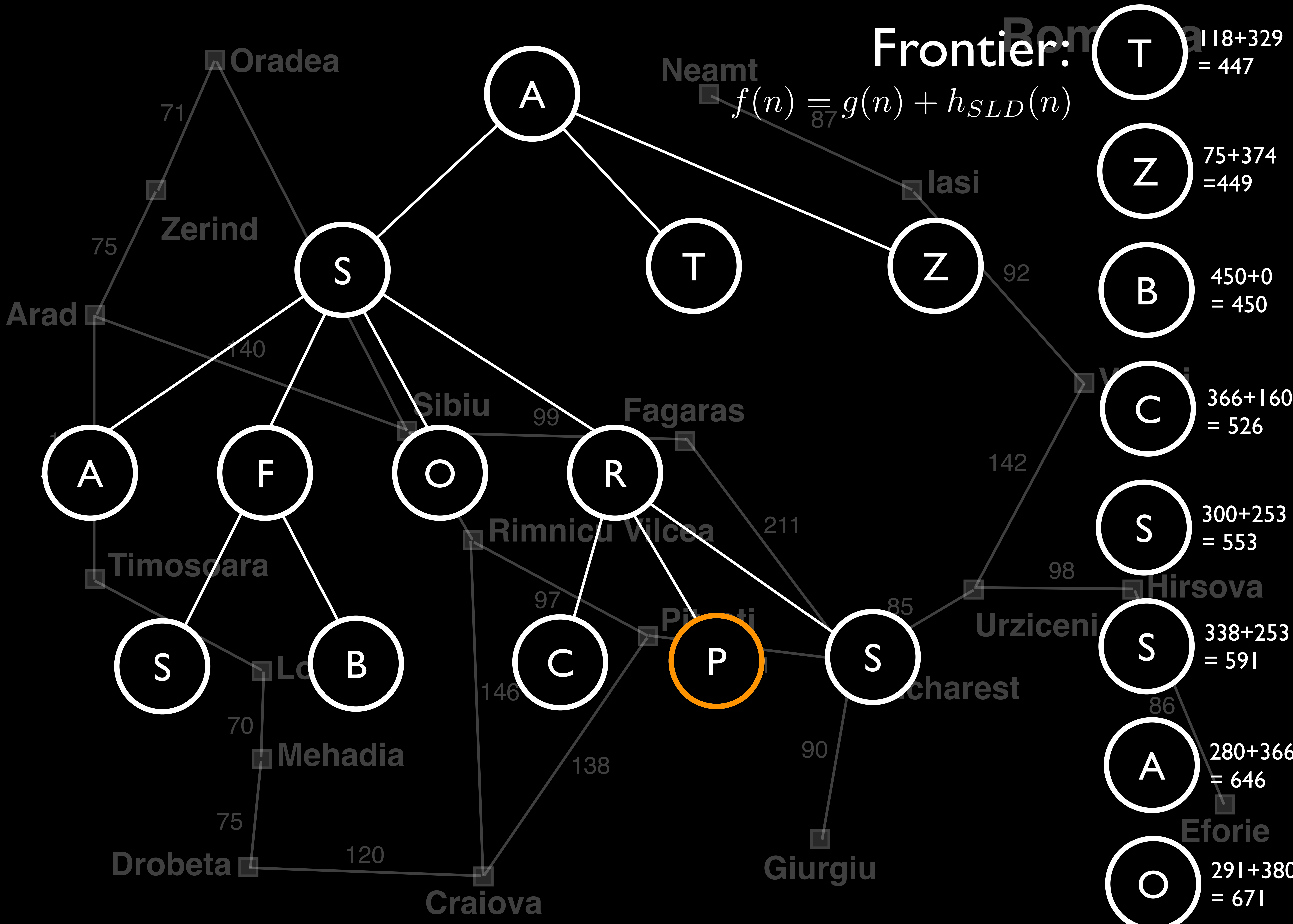
$$f(n) = g(n) + h_{SLD}(n)$$



# Frontier:

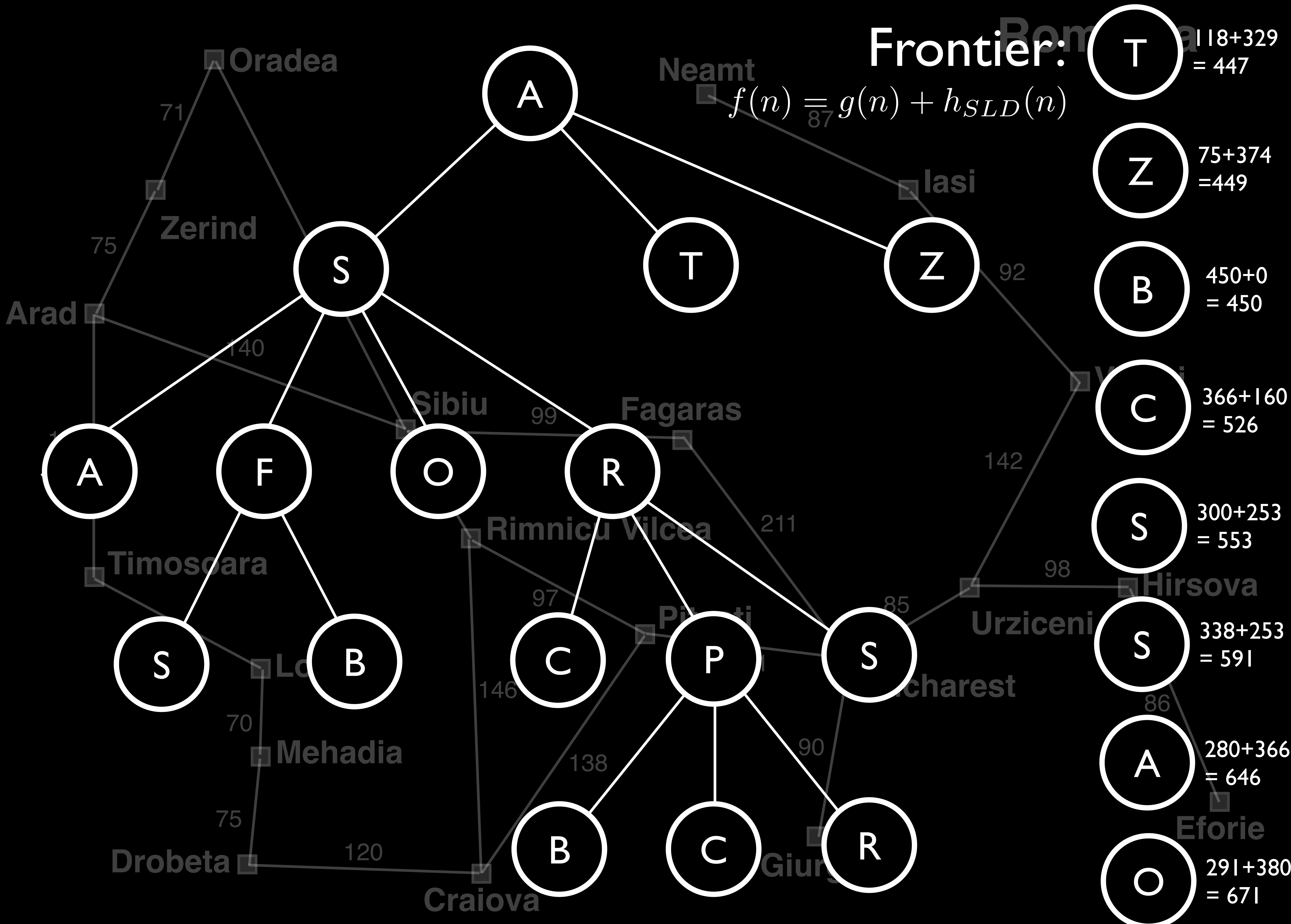
$$f(n) = g(n) + h_{SLD}(n)$$




Frontier:
$f(n) = g(n) + h_{SLD}(n)$
T 118+329 = 447
Z 75+374 =449
B 450+0 = 450
C 366+160 = 526
S 300+253 = 553
S 338+253 = 591
A 280+366 = 646
O 291+380 = 671
A
S
T
Z
A
F
O
R
S
B
C
P
S
Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Timisoara
Urziceni
Hirsova
Mehadia
Drobeta
Craiova
Giurgiu
Eforie
71
75
87
92
140
99
142
211
97
98
85
146
86
70
90
138
75
120

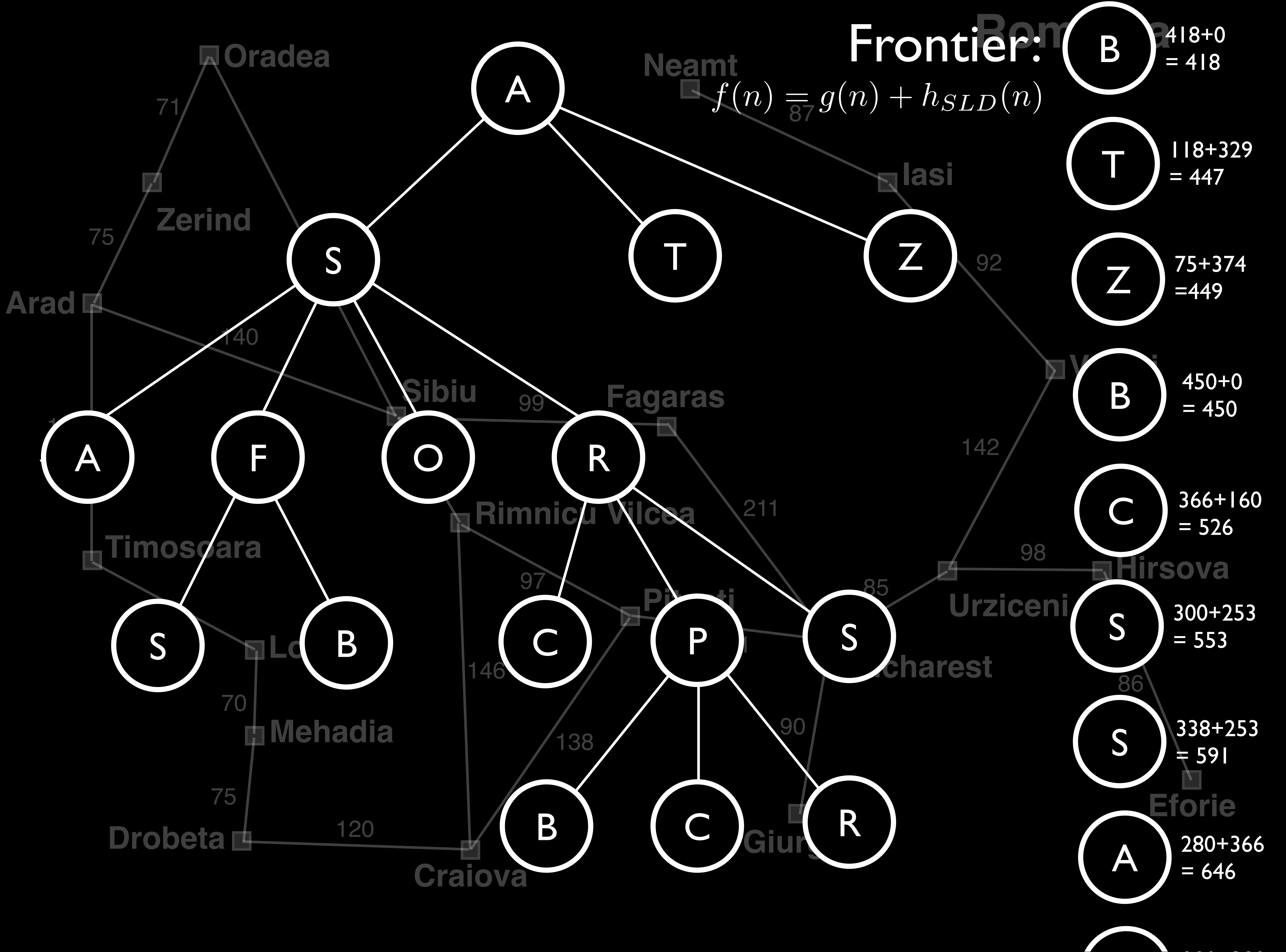
## Frontier:

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



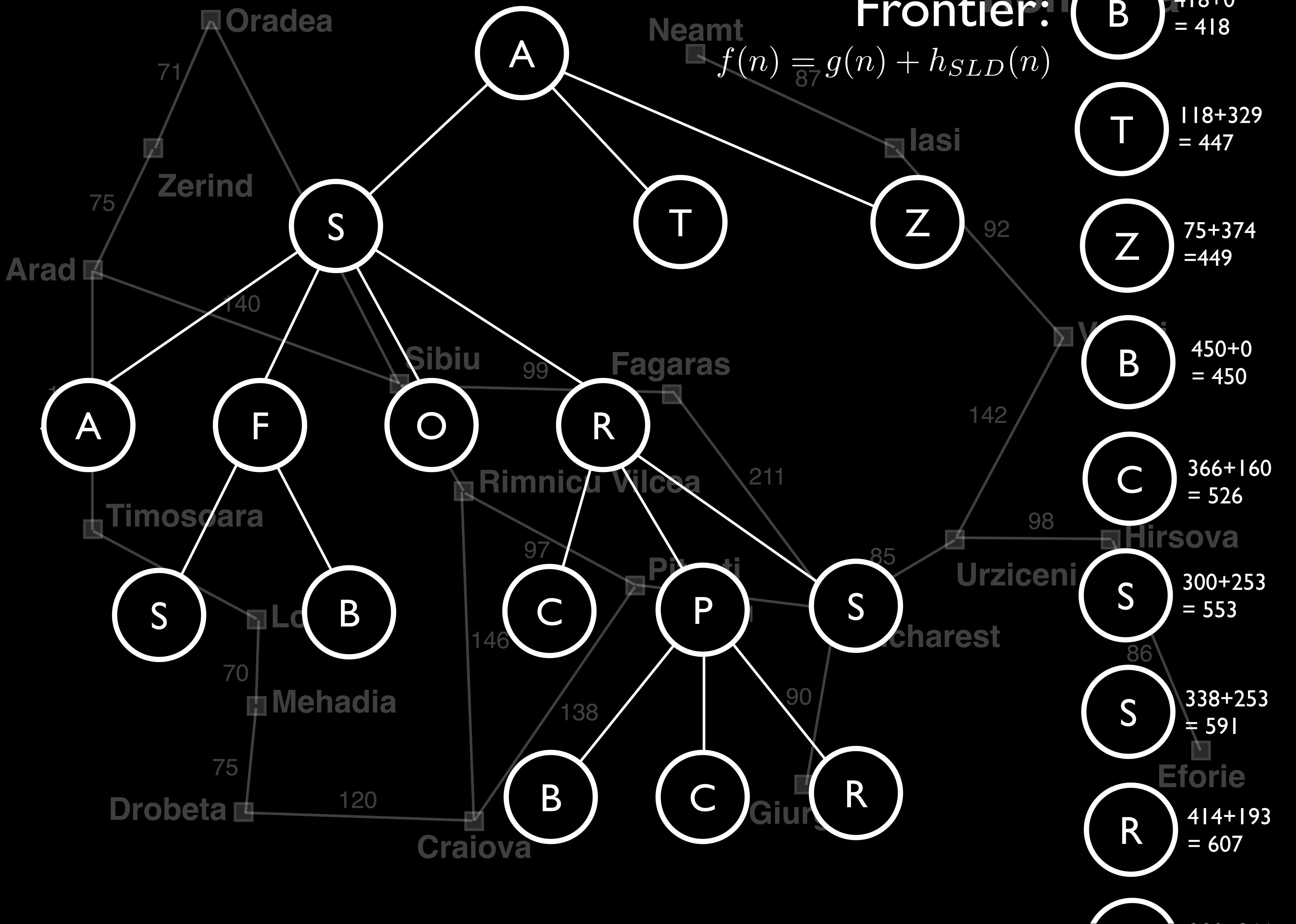
# Frontier:

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



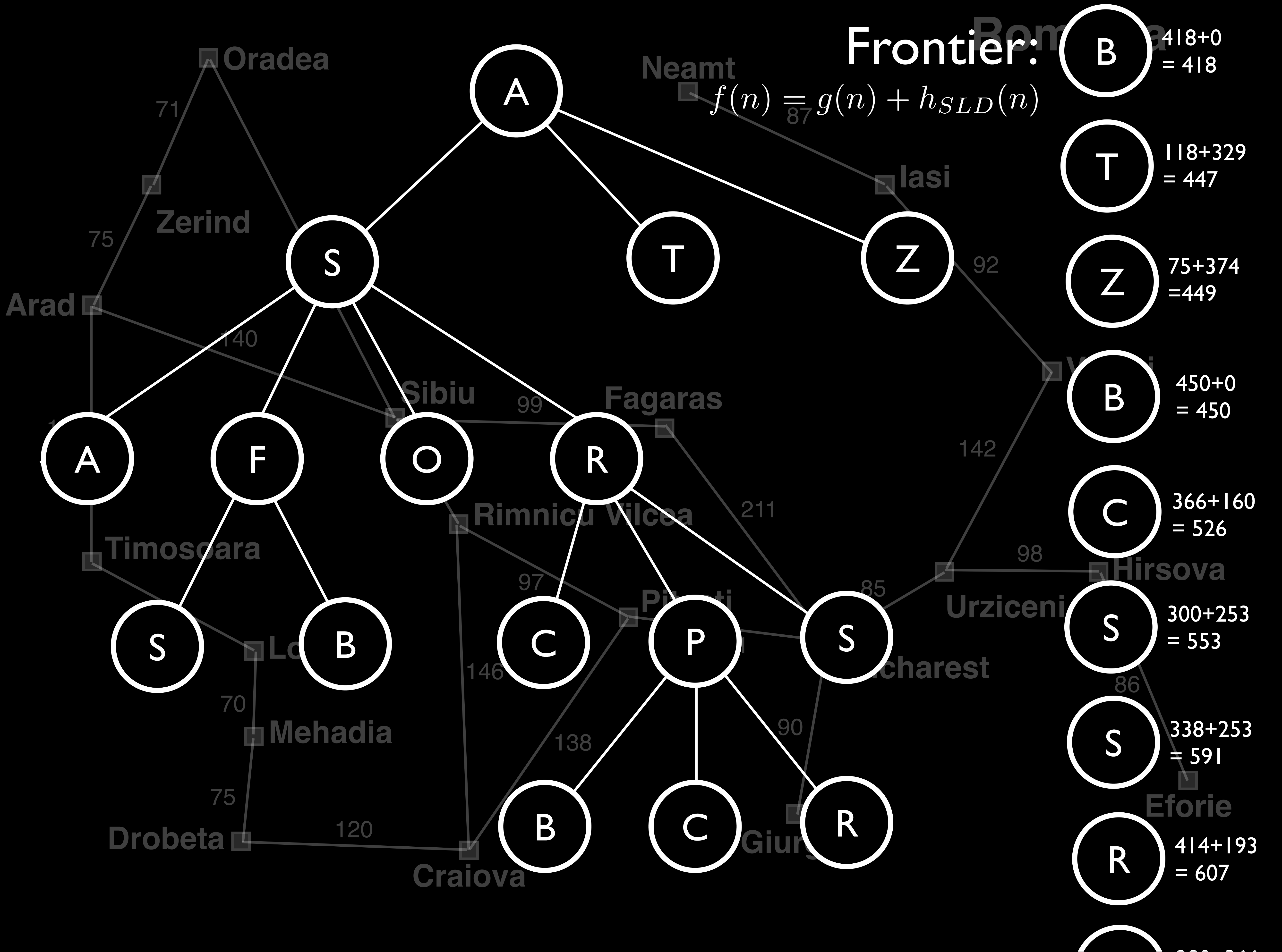
## Frontier:

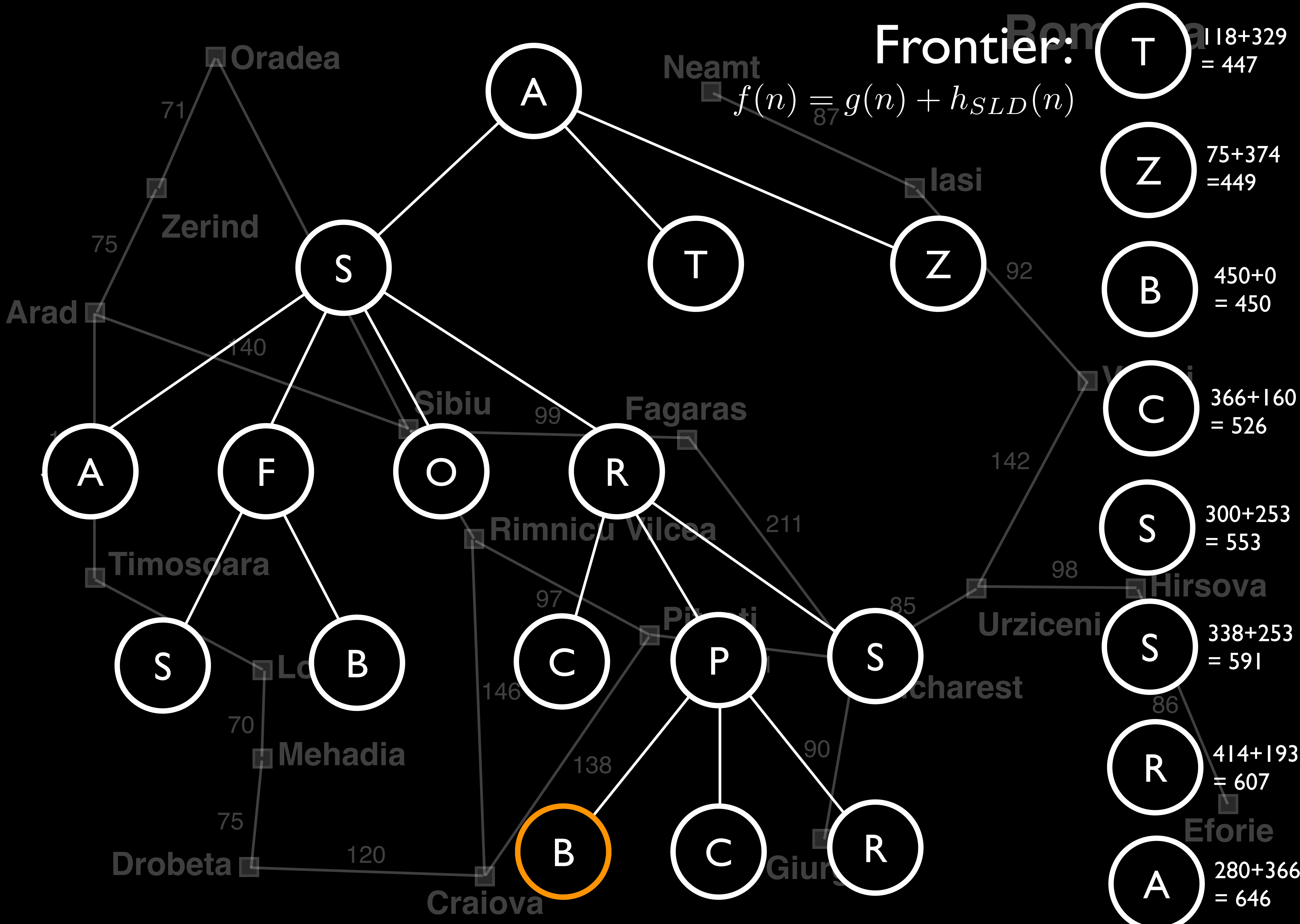
$$f(n) = g(n) + h_{SLD}(n)$$

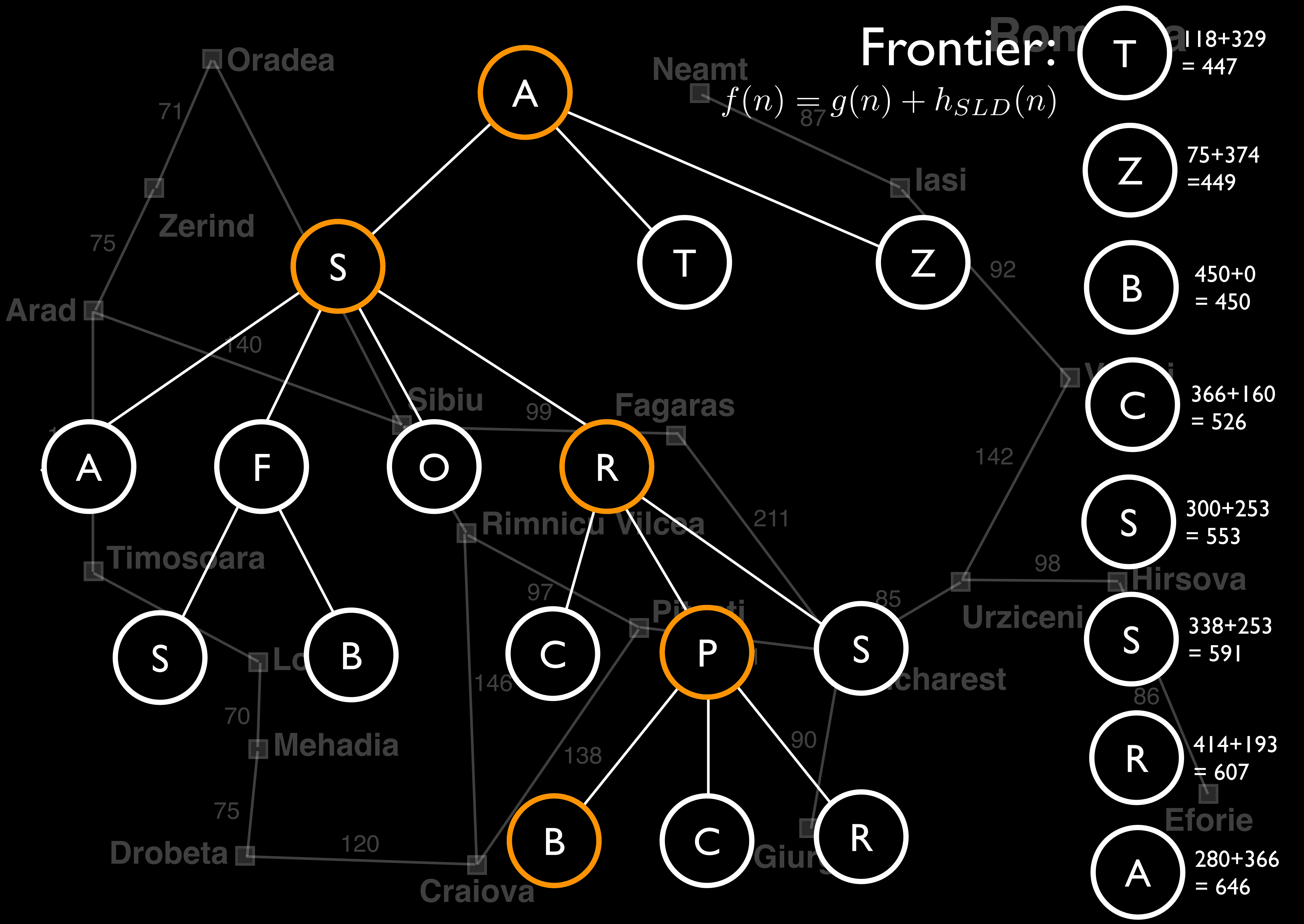


Frontier:

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




Frontier:
$f(n) = g(n) + h_{SLD}(n)$
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 414+193 = 607
A 280+366 = 646
A
S
T
Z
A
F
O
R
S
B
C
P
S
B
C
R
Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Timosoara
Hirsova
Urziceni
Mehadia
Drobeta
Craiova
Eforie
71
75
87
92
140
99
142
211
98
97
85
146
70
138
90
86
75
120


Frontier:
$f(n) = g(n) + h_{SLD}(n)$
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 414+193 = 607
A 280+366 = 646
Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Rimnicu Vilcea
Timisoara
Urziceni
Hirsova
Mehadia
Drobeta
Craiova
Eforie
71
75
87
92
140
99
142
211
98
97
85
146
70
138
90
86
75
120

Evaluation function

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

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

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

Evaluation function

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

$=$ Estimated cost of cheapest solution through *n*

# A* Search

# A* Search

Completeness

Optimality

# A* Search

If $h(n)$
is admissible

Completeness

Optimality

# Admissible Heuristic

Never overestimates the true cost of a solution

# Admissible Heuristic

Never overestimates the true cost
of a solution

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

# A* Search

If $h(n)$ is admissible

Completeness

Optimality

Time Complexity

Space Complexity

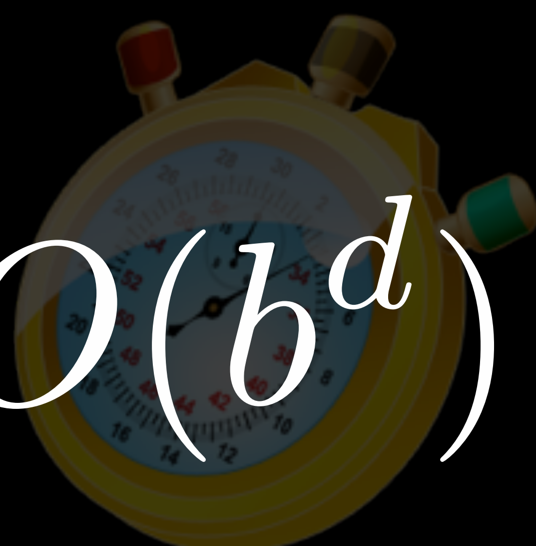
# A* Search

Completeness

If $h(n)$

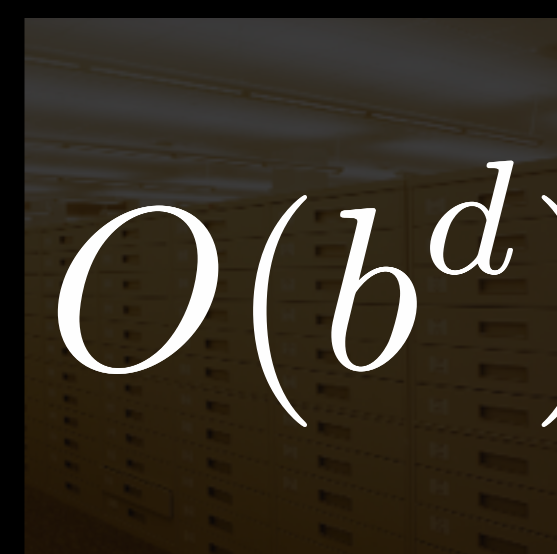
is admissible

Optimality



$O(b^d)$

Time Complexity



$O(b^d)$

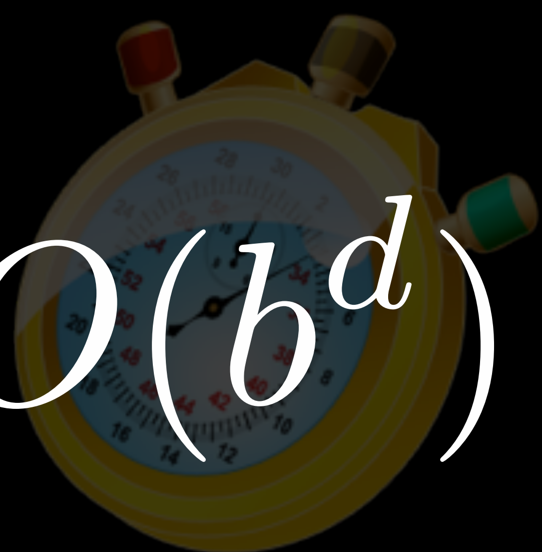
Space Complexity

# A* Search

If $h(n)$ is admissible

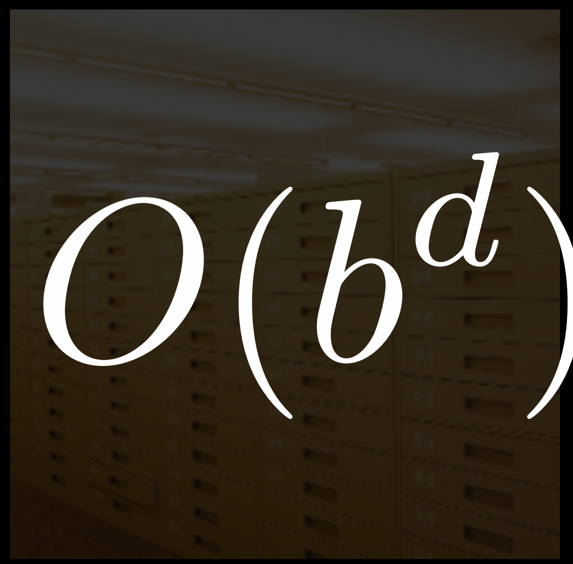
Completeness

Optimality



$O(b^d)$

?!?!?!



$O(b^d)$

Time Complexity

Space Complexity

# Time / Space Advantage of A*

# Time / Space Advantage of A*

- A* does much better than its worst case complexity when the heuristic is close to the true shortest cost to a goal

# Time / Space Advantage of A*

- A* does much better than its worst case complexity when the heuristic is close to the true shortest cost to a goal
- Where h* is the optimal (perfect) heuristic, A* runs in provably polynomial time when

$$\left|h(x) - h^*(x)\right| = O(\log h^*(x))$$

Now It's Time to Play...

# Heuristic Functions

Where do good heuristic functions come from?

# Heuristic Functions

Where do good heuristic functions come from?

Good question...
(See Section 3.6 for some ideas)

# Search Strategies

| | BFS | DFS | IDS | Greedy | A* |
|---|---|---|---|---|---|
| Complete? | ✓ | ✗ | ✓ | ✗ | ✓ |
| Optimal? | ✓ | ✗ | ✓ | ✗ | ✓ |
| Time | $O(b^d)$ | $O(b^m)$ | $O(b^d)$ | $O(b^m)$ | $O(b^{\epsilon d})$ |
| Space | $O(b^d)$ | $O(bm)$ | $O(bd)$ | $O(b^m)$ | $O(b^d)$ |

* If step costs are identical

† With an admissible heuristic

Thursday: Homework Assignment - Check Website

Next Tuesday: Quiz