CSC242: Introduction to Artificial Intelligence

Lecture 1.2

Please put away all electronic devices

Problem (Domain): $\langle S,\ A,\ \textsc{Actions},\ \textsc{Result},\ \textsc{Cost}\rangle$

$\textsc{Actions} : s \in S \rightarrow$
$\quad \{\ a \in A : a \text{ can be executed (is applicable) in } s\ \}$

$\textsc{Result} : s \in S\ ,\ a \in A \rightarrow$
$\quad s' \in S \text{ s.t. } s' \text{ is the result of performing } a \text{ in } s$

$\textsc{Cost}$: Assigns a cost to each path/step $c(s, a, s')$

Problem (Instance): $\langle I \in S,\ G \subseteq S\rangle$

Solution: $\langle a_1,\ a_2,\ \ldots,\ a_n\rangle \in A^n$ s.t.

$$\textsc{Result}(\cdots\ \textsc{Result}(\textsc{Result}(I,\ a_1),\ a_2)\ \cdots,\ a_n) \in G$$

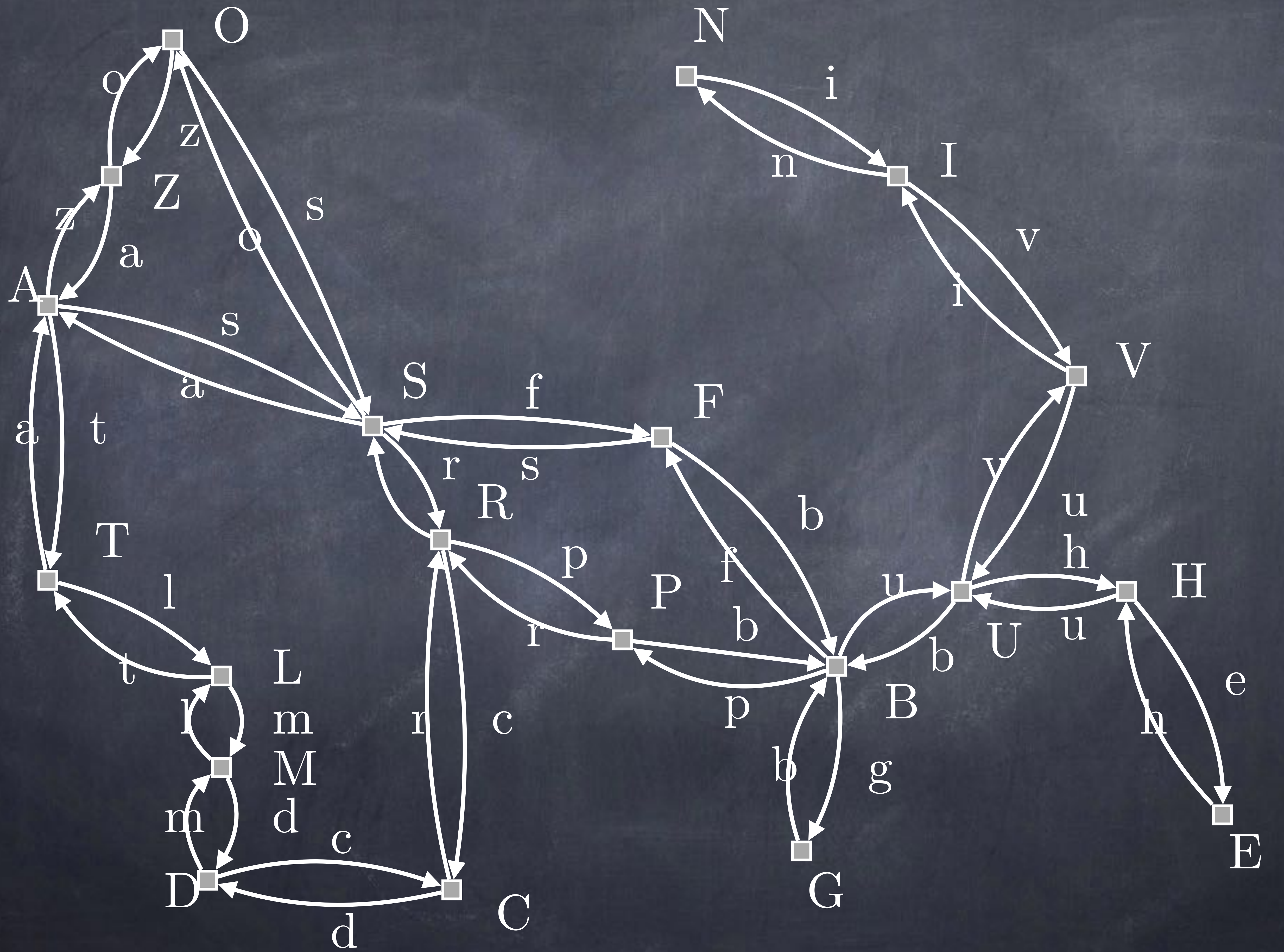
State Space

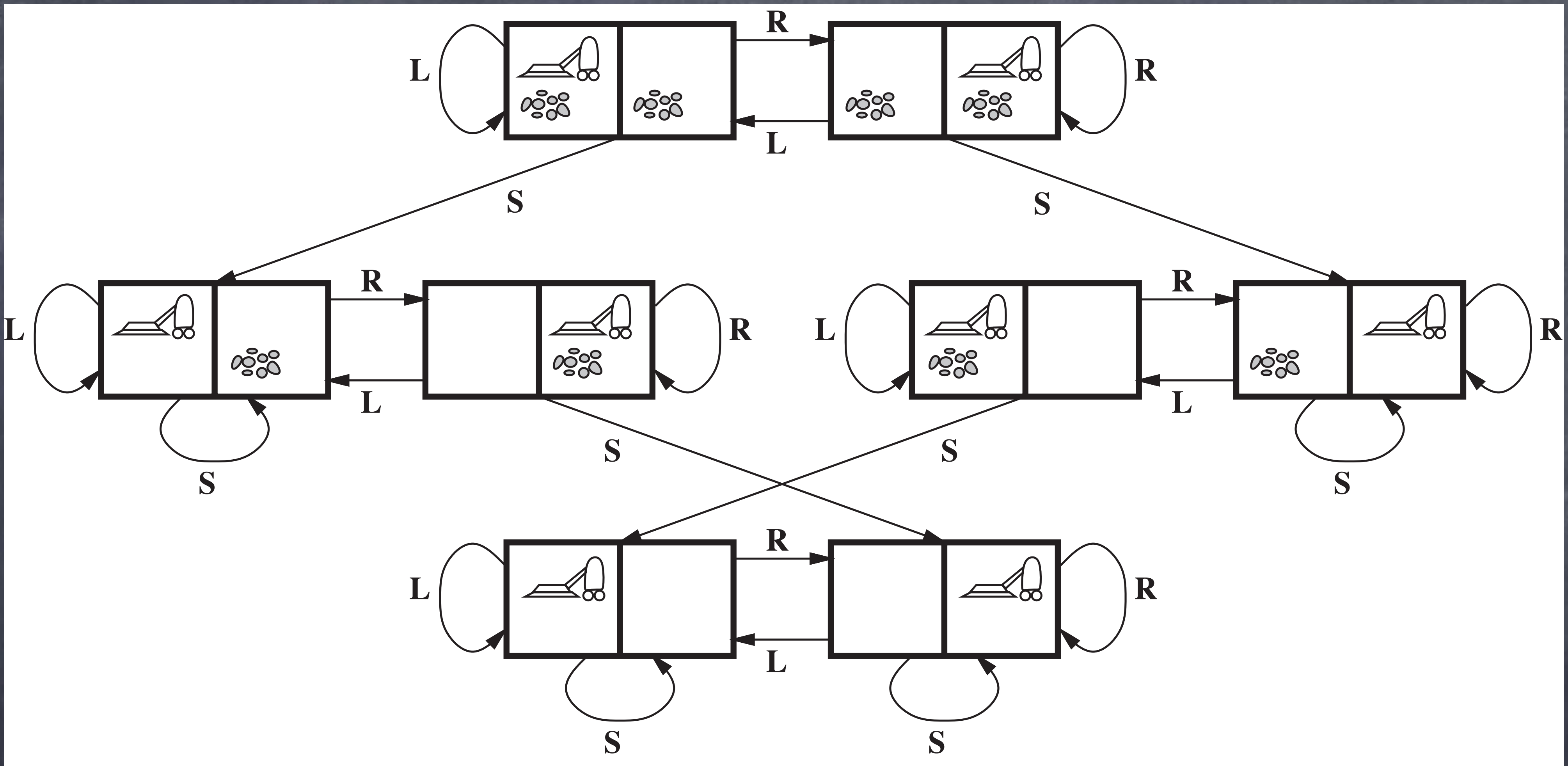
- The set of all states reachable from the initial state by some sequence of actions

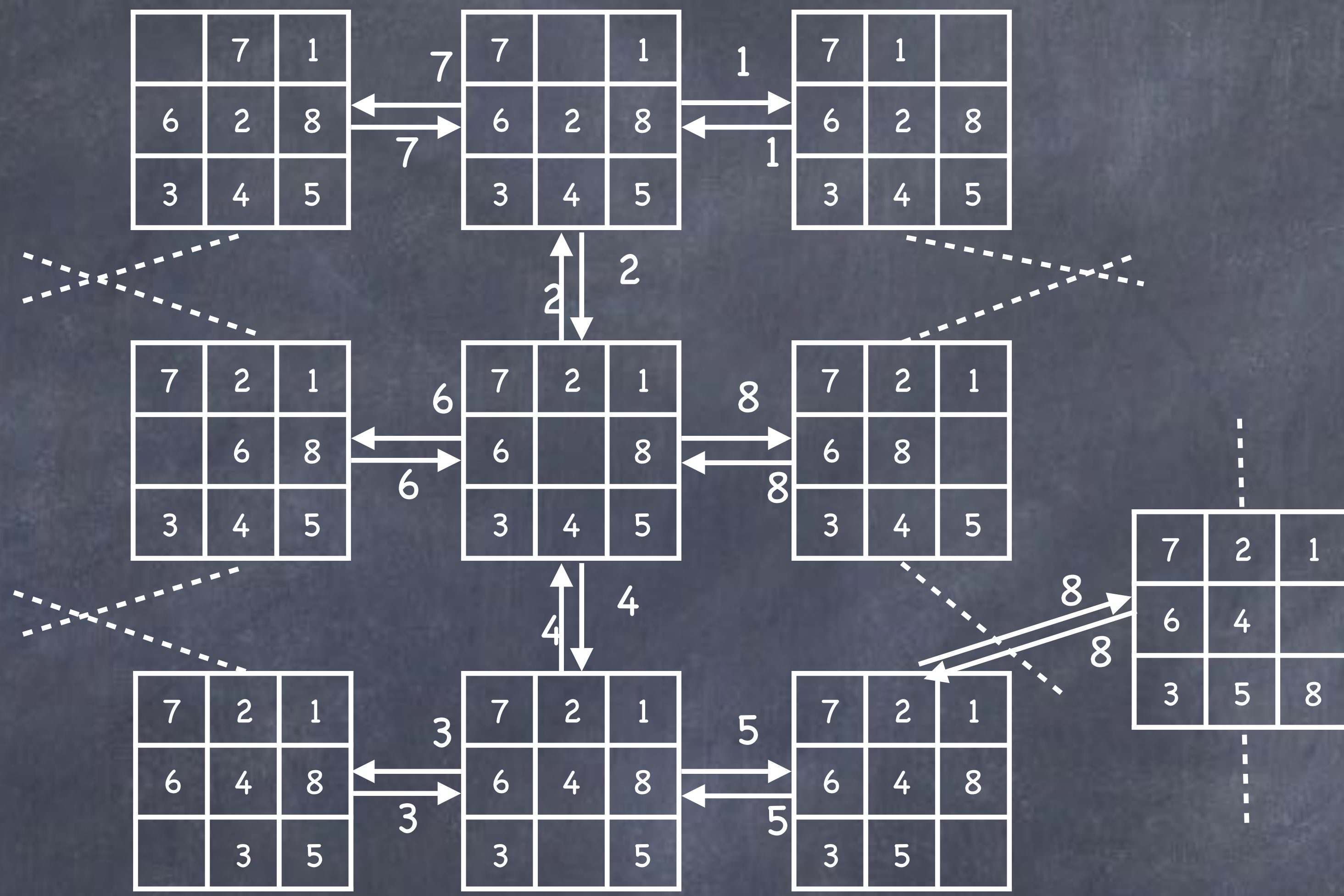
States

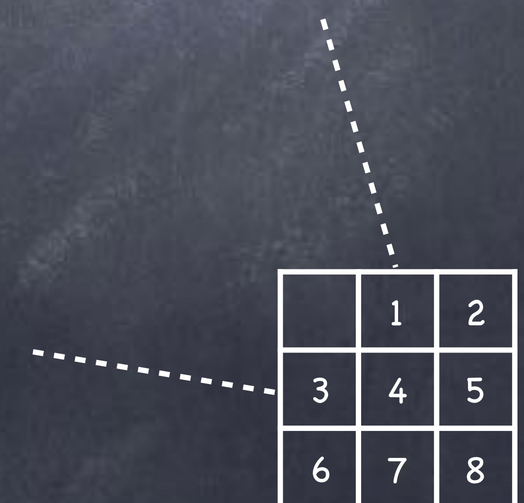
Actions → State Space

Transition Model


O
N
i
o
z
n
I
Z
z
s
a
o
v
A
s
i
V
a
S
f
F
a
t
r
s
y
R
b
u
T
p
h
l
f
u
P
H
b
r
b
U
u
t
L
e
B
p
l
m
r
c
h
M
b
g
m
d
c
E
D
C
G
d


R
L
R
L
S
S
R
L
R
L
S
R
L
R
L
S
S
S
S
R
L
R
L
S
S





8-puzzle: $9!/2 = 181{,}440$

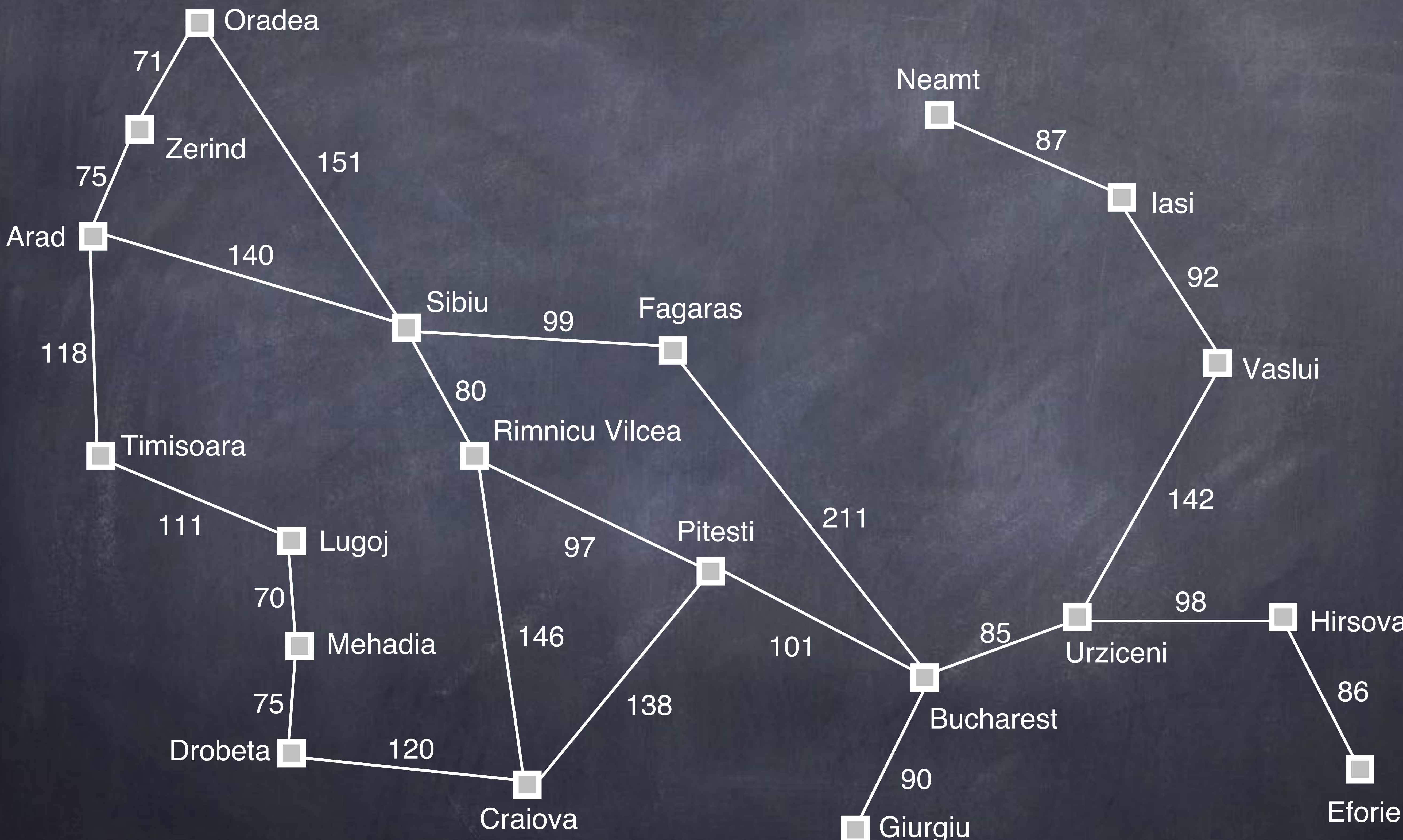
15-puzzle: $16!/2 = \tilde{}1.3$ trillion

24-puzzle: $25!/2 = \tilde{}10^{25}$

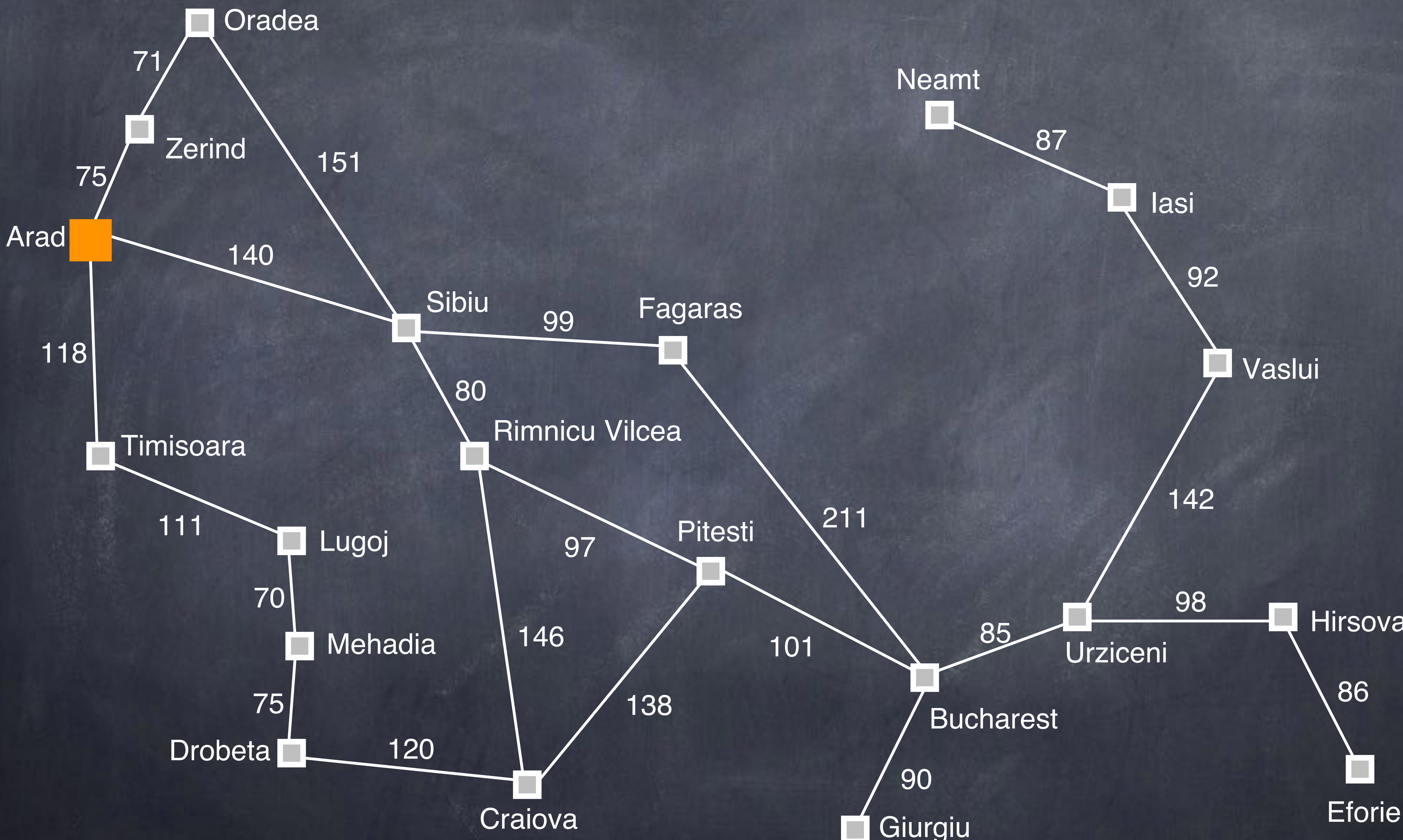
State-Based Model

- Model the world as a set of states
- Actions change the state of the world
- State Space
  - Defined by states, actions, and transition model
  - Small, large, or even infinite

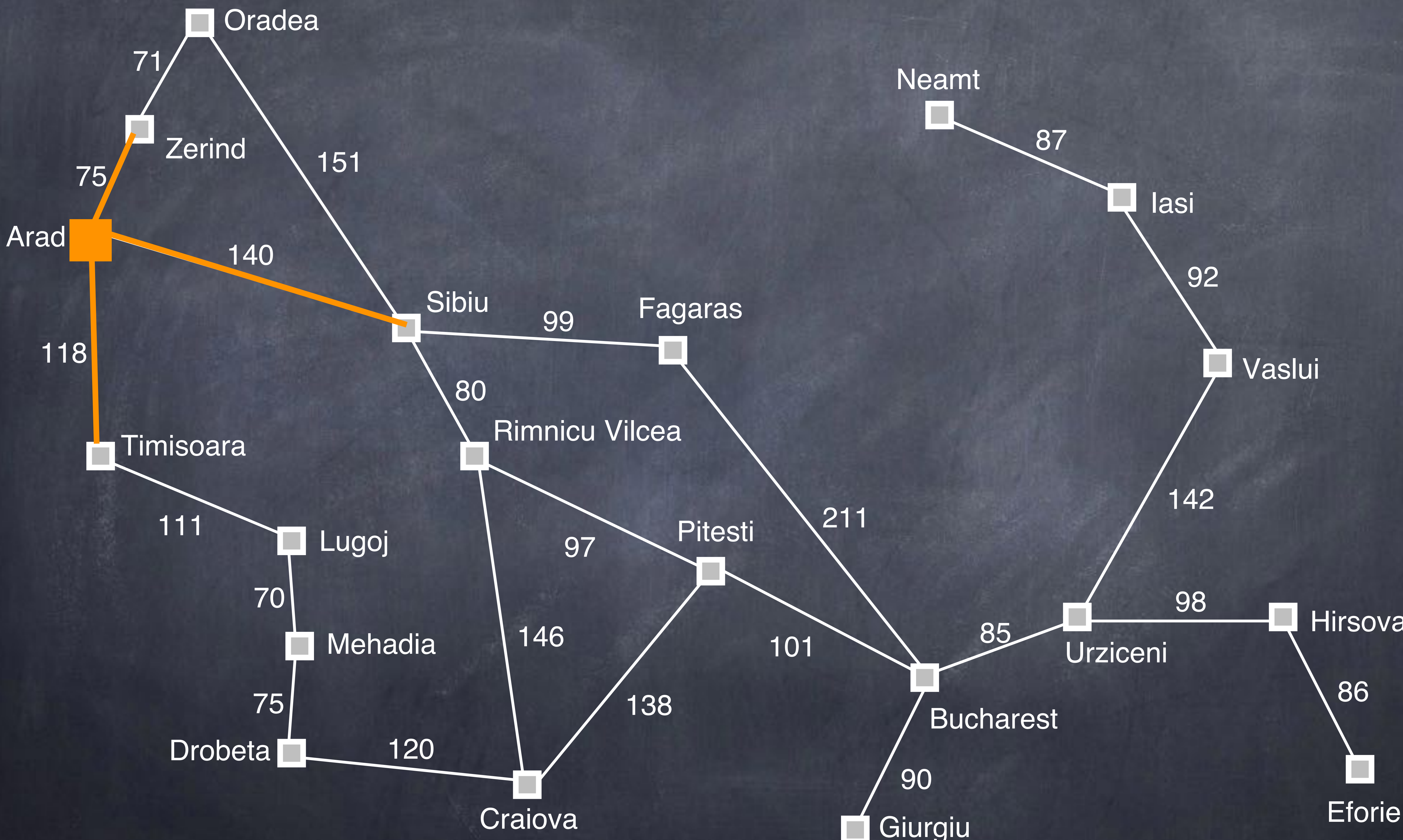
Abstraction!

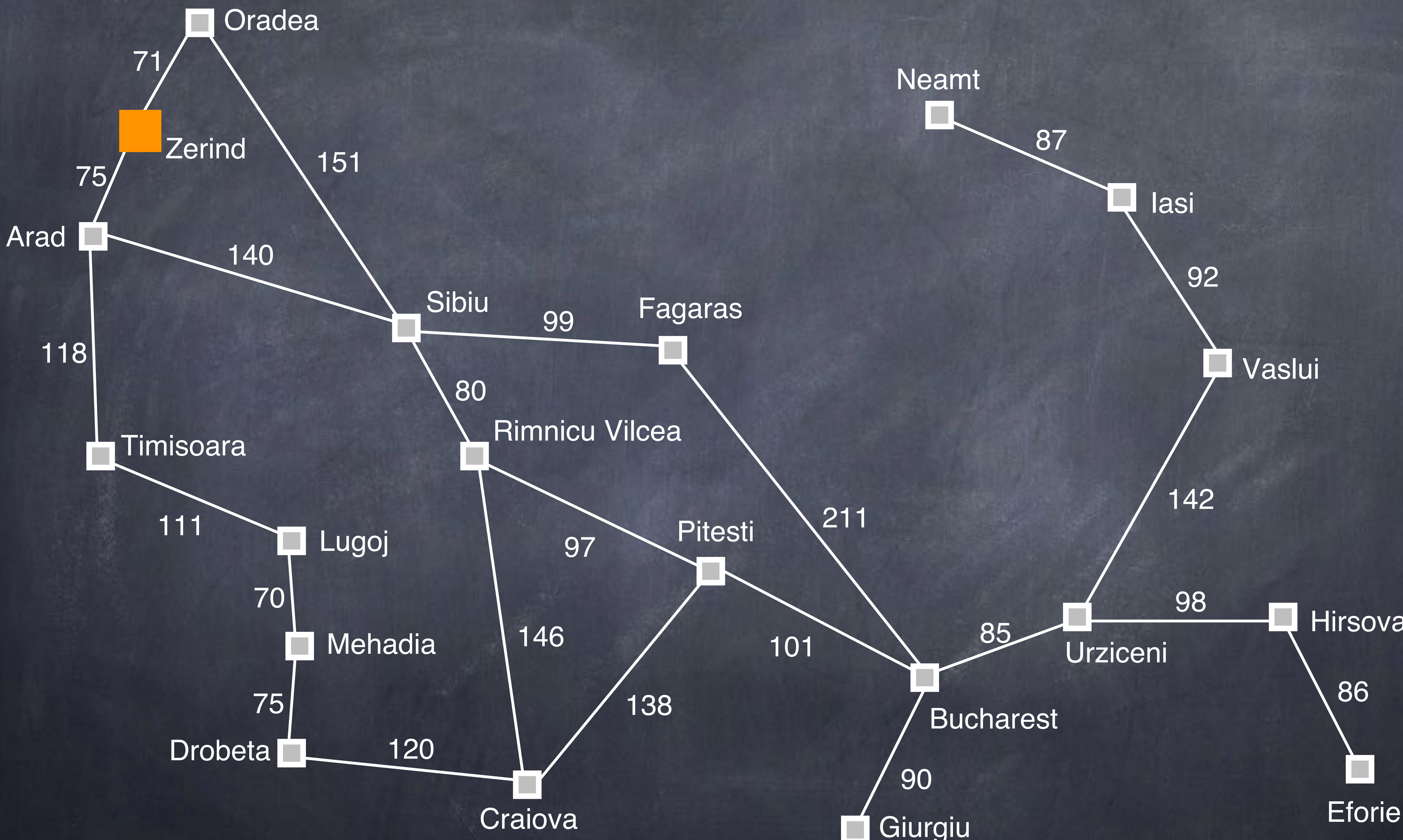


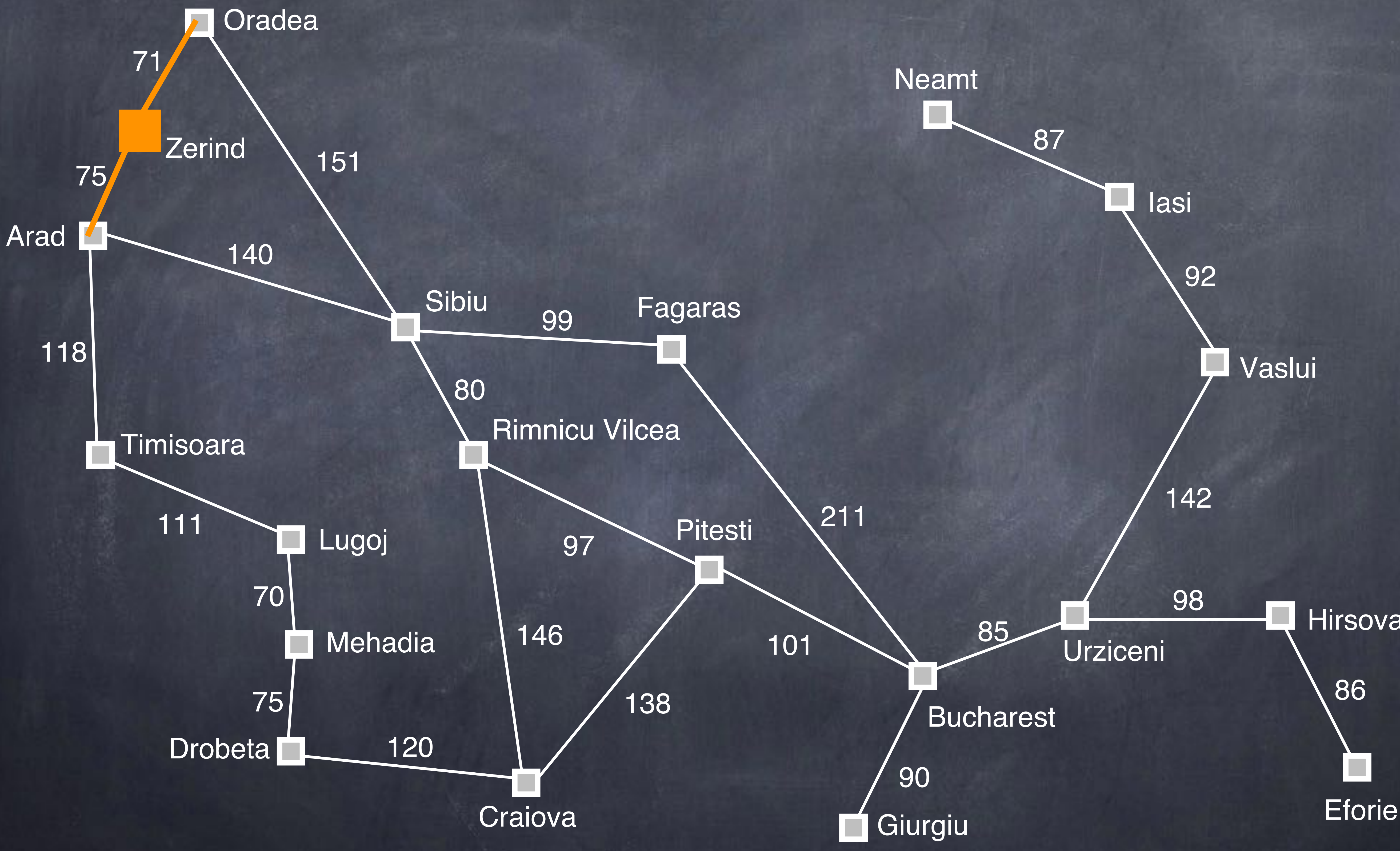
Oradea
71
Neamt
Zerind
151
87
75
Iasi
Arad
140
92
Sibiu
99
Fagaras
118
Vaslui
80
Rimnicu Vilcea
Timisoara
142
111
211
Lugoj
97
Pitesti
70
98
85
Hirsova
Mehadia
146
101
Urziceni
86
75
138
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Eforie



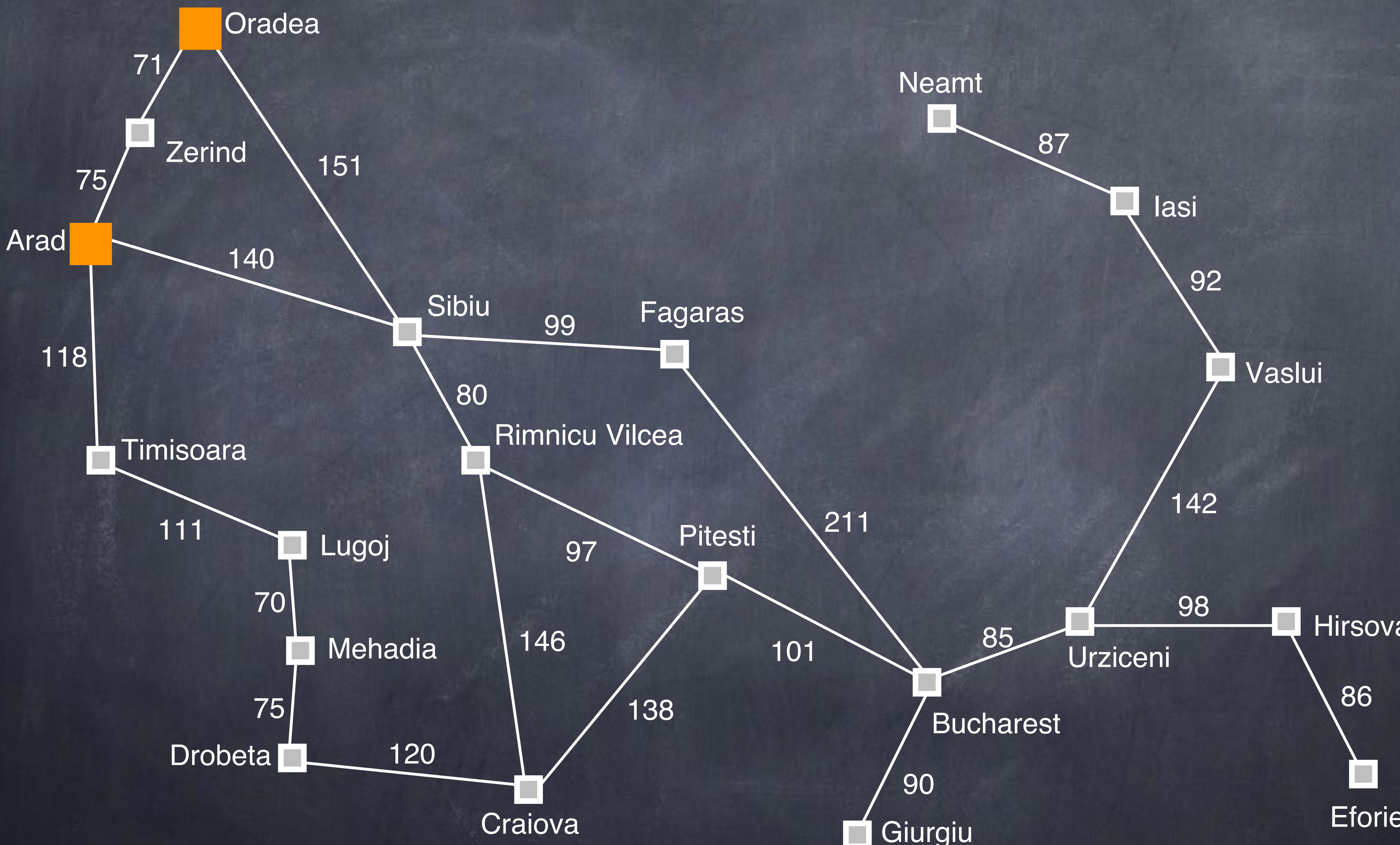
Oradea
71
Neamt
Zerind
151
87
75
Iasi
Arad
140
92
Sibiu
99
Fagaras
118
Vaslui
80
Rimnicu Vilcea
Timisoara
142
111
211
Lugoj
97
Pitesti
70
98
85
Hirsova
Mehadia
146
101
Urziceni
86
138
Bucharest
75
Drobeta
120
90
Craiova
Eforie
Giurgiu

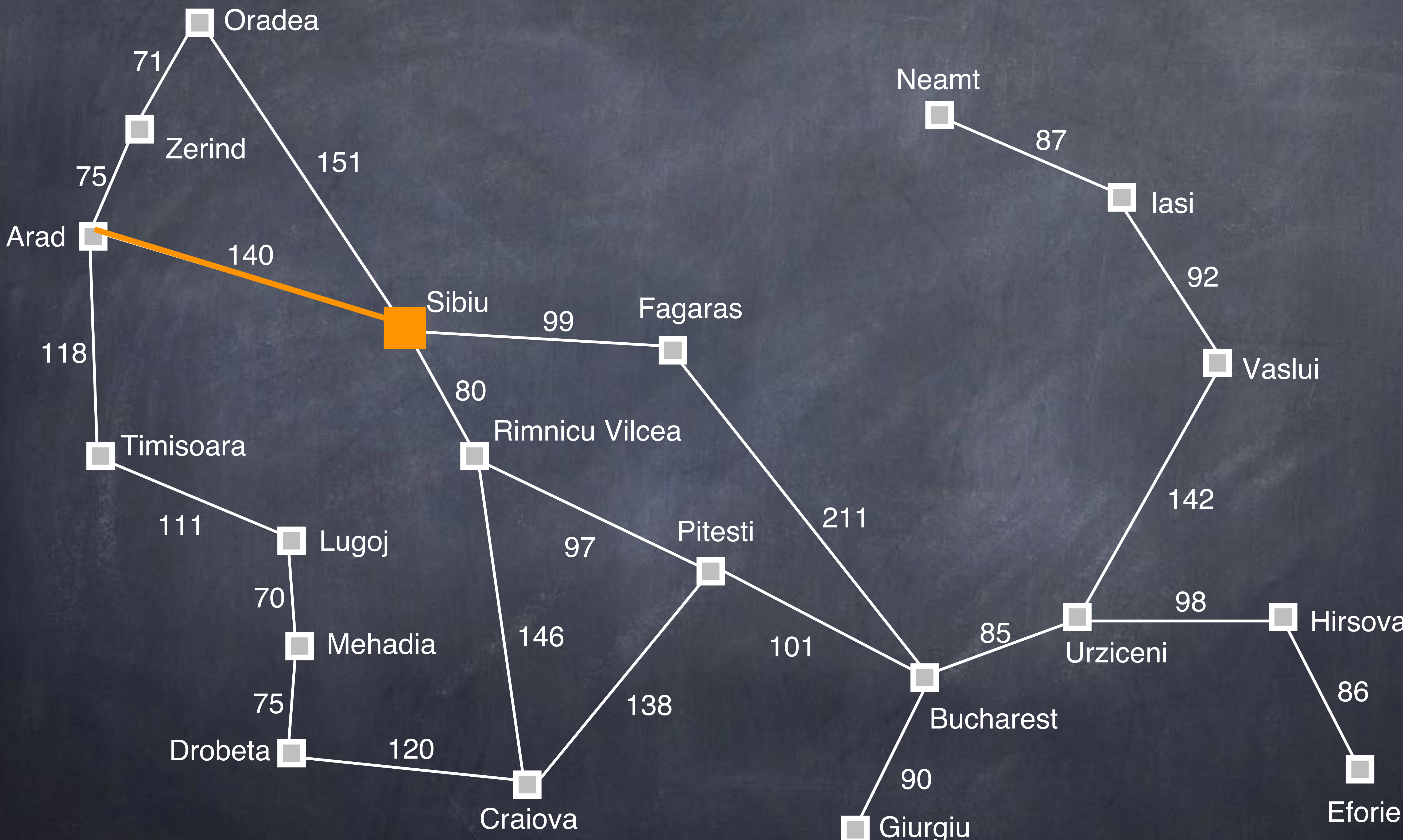

Oradea
71
Neamt
87
Zerind
151
Iasi
75
Arad
140
92
Sibiu
99
Fagaras
118
Vaslui
80
Rimnicu Vilcea
Timisoara
142
111
211
Lugoj
97
Pitesti
70
98
Hirsova
146
85
Mehadia
101
Urziceni
86
75
138
Bucharest
Drobeta
120
90
Craiova
Eforie
Giurgiu

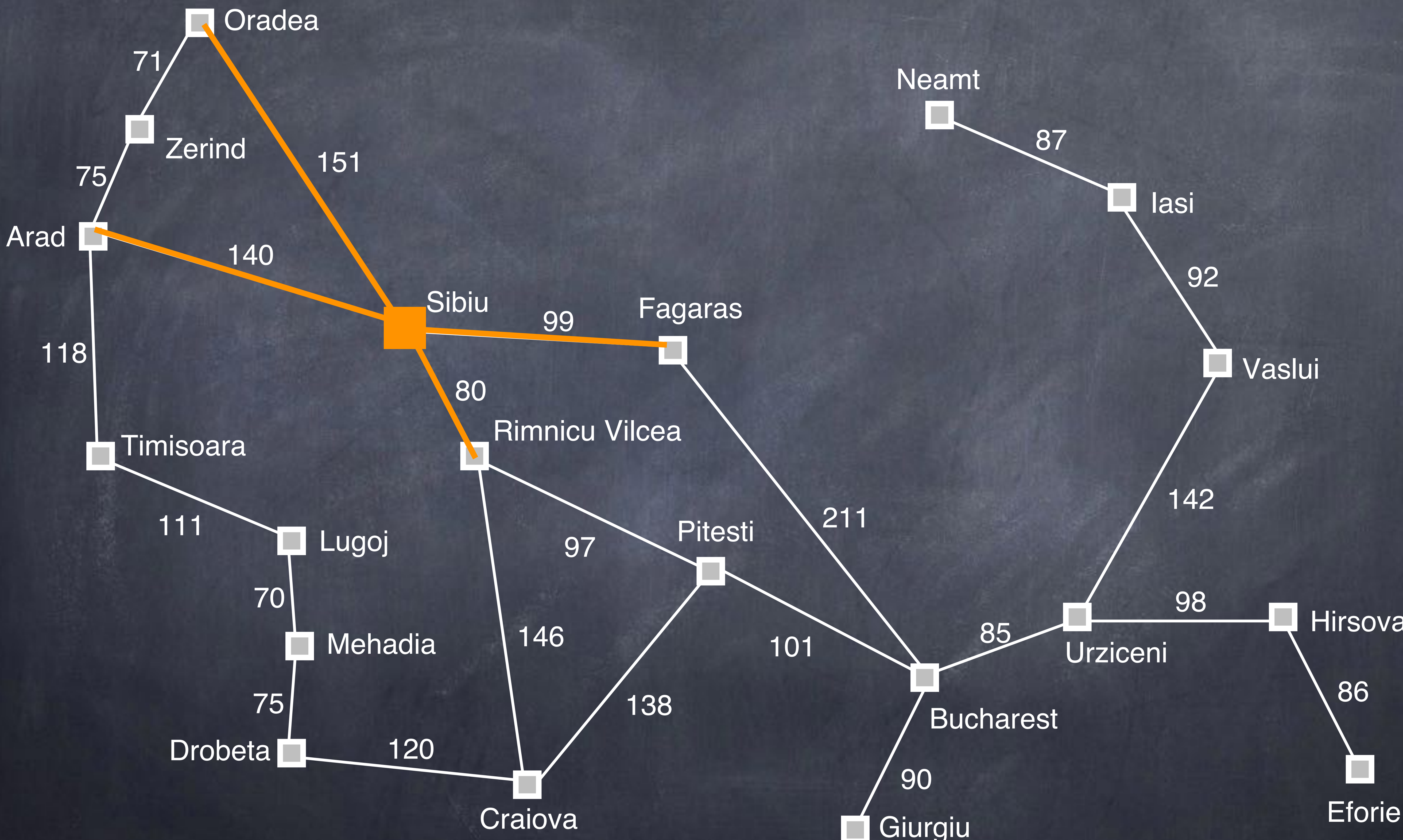

Oradea
71
Zerind
151
75
Arad
140
Neamt
87
Iasi
92
Sibiu
99
Fagaras
Vaslui
118
80
Rimnicu Vilcea
Timisoara
142
111
211
Lugoj
97
Pitesti
70
98
Mehadia
146
101
85
Urziceni
Hirsova
75
138
Bucharest
86
Drobeta
120
90
Craiova
Giurgiu
Eforie



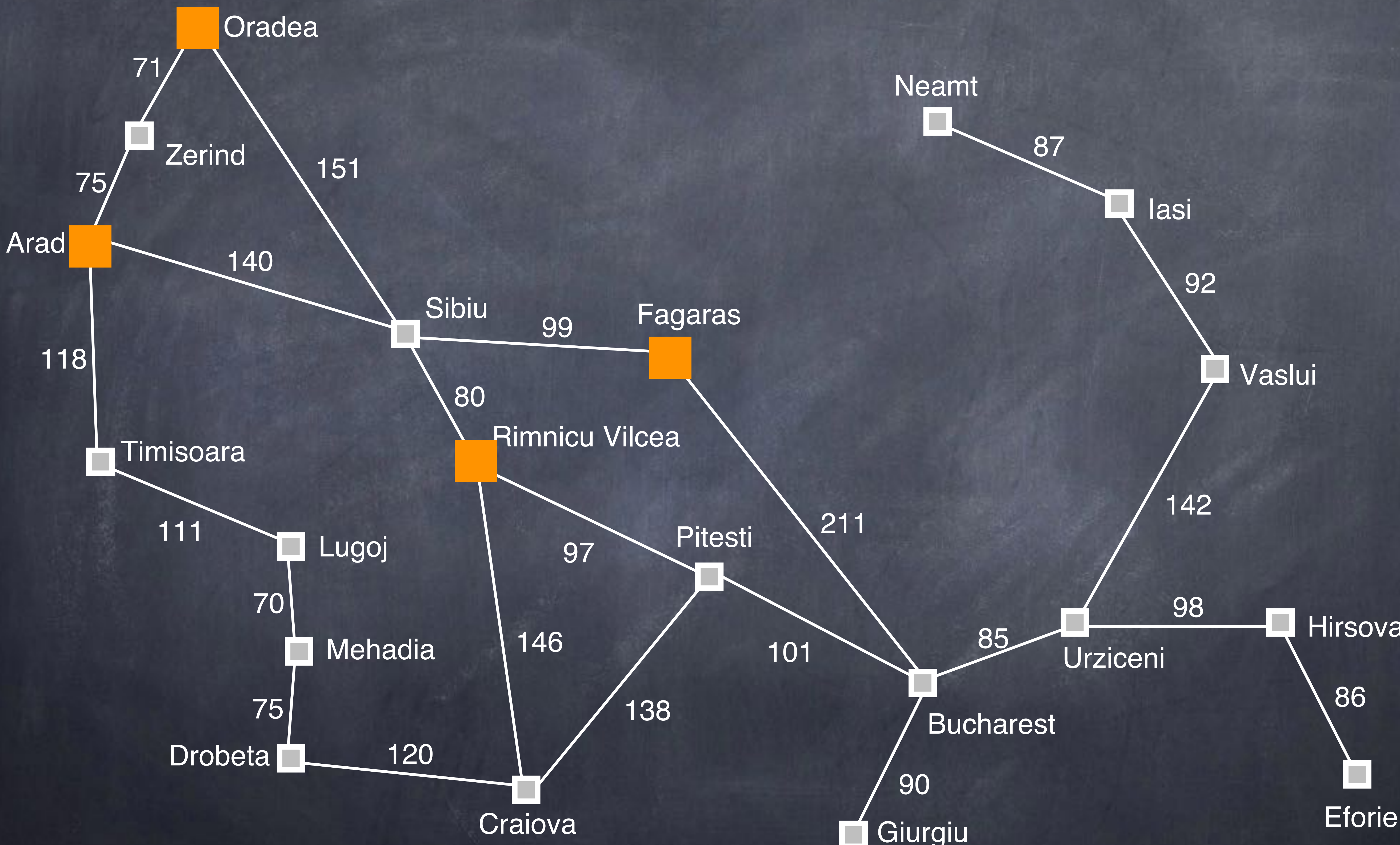
Oradea
71
Zerind
75
151
Arad
140
Neamt
87
Iasi
92
Sibiu
99
Fagaras
Vaslui
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
142
70
Mehadia
146
101
85
Urziceni
98
Hirsova
86
75
138
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Eforie

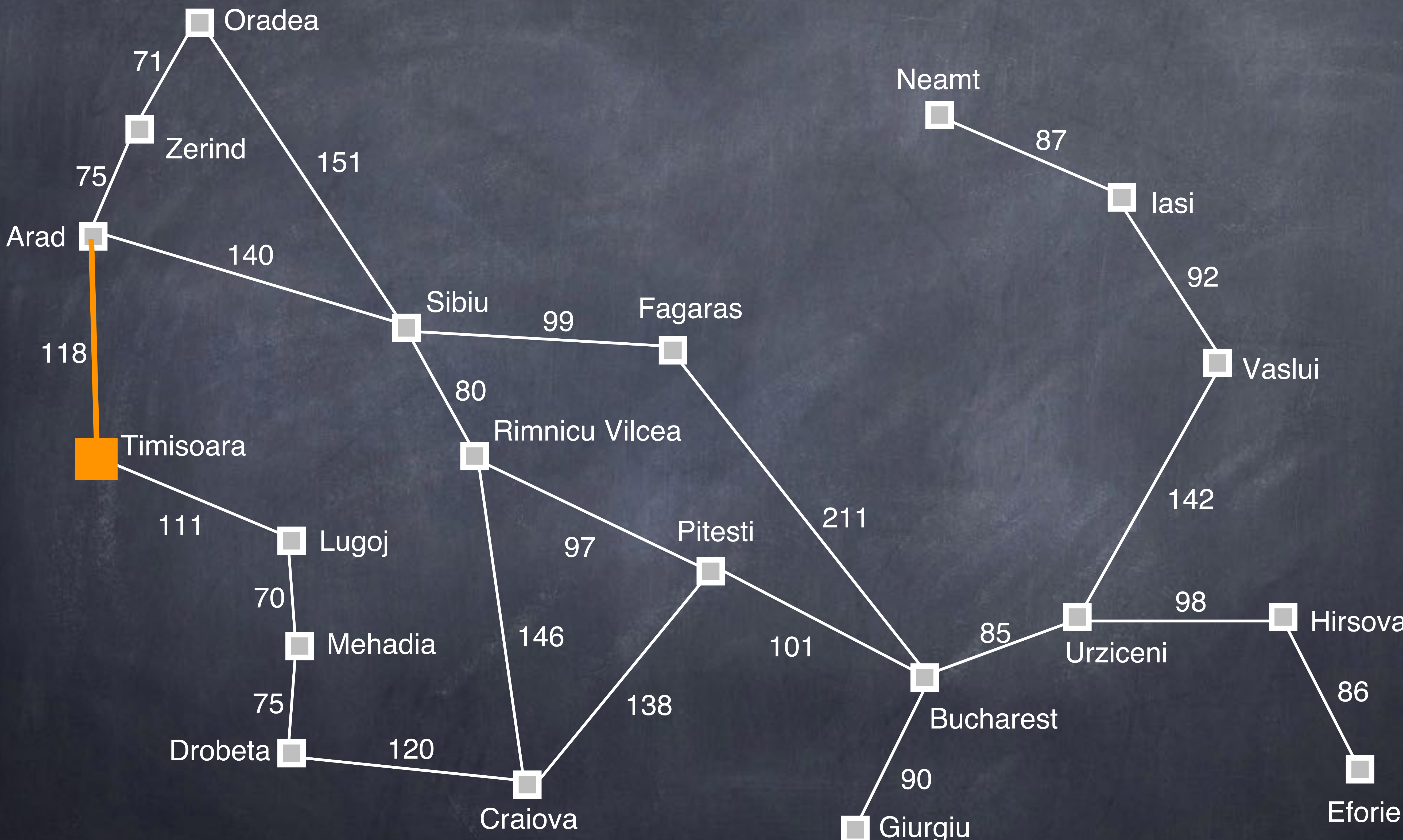

Oradea
71
151
Zerind
75
Arad
140
118
Sibiu
99
Fagaras
80
Rimnicu Vilcea
Timisoara
111
Lugoj
70
Mehadia
75
Drobeta
120
Craiova
146
97
Pitesti
138
101
211
Bucharest
90
Giurgiu
85
Urziceni
98
Hirsova
86
Eforie
142
Vaslui
92
Iasi
87
Neamt

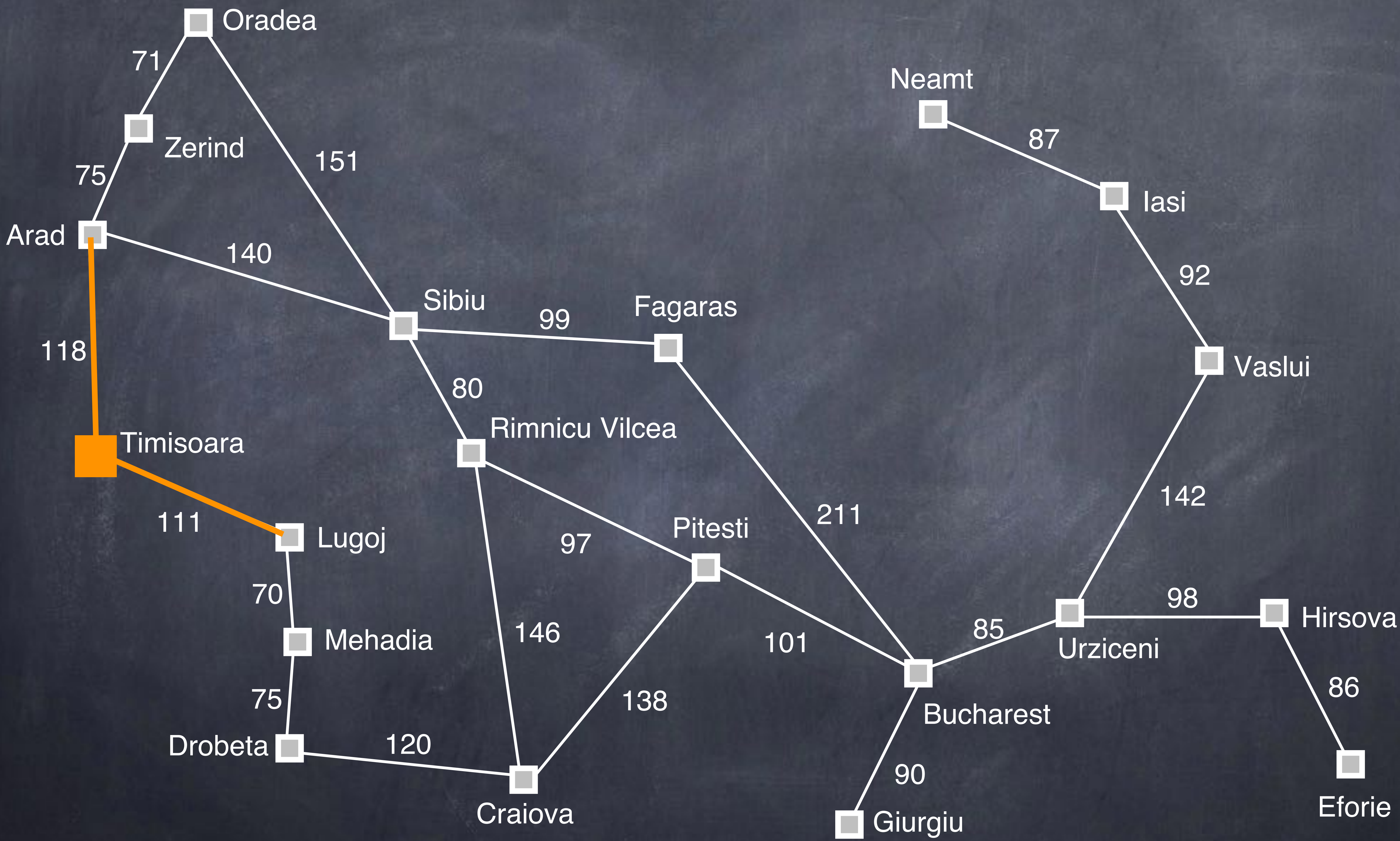

Oradea
71
Neamt
Zerind
151
87
75
Iasi
Arad
140
92
Sibiu
99
Fagaras
118
Vaslui
80
Rimnicu Vilcea
Timisoara
142
111
211
Lugoj
97
Pitesti
70
98
85
Hirsova
Mehadia
146
101
Urziceni
86
75
138
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Eforie



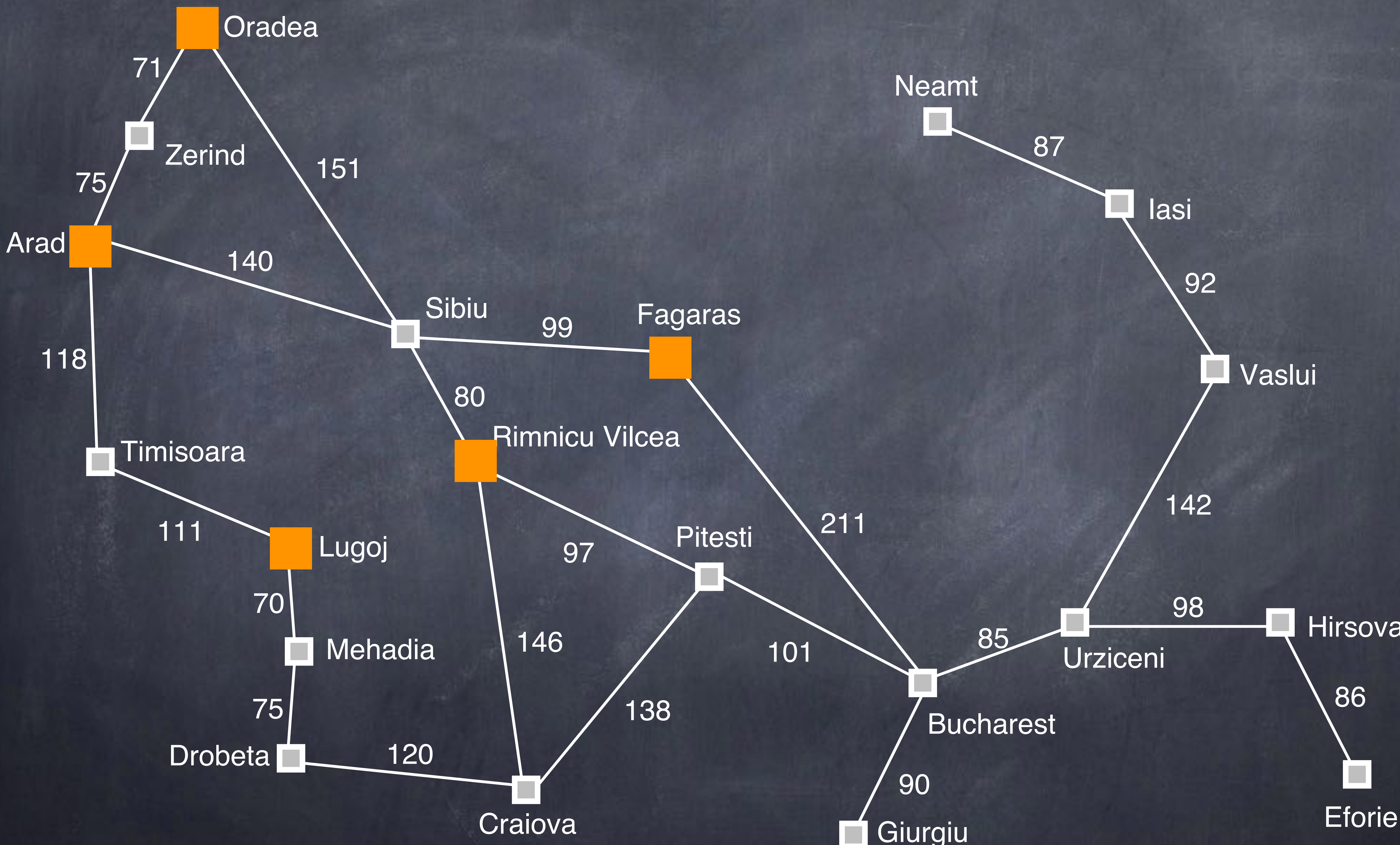
Oradea
71
Neamt
Zerind
151
87
75
Iasi
Arad
140
92
Sibiu
99
Fagaras
118
Vaslui
80
Rimnicu Vilcea
Timisoara
142
111
211
Lugoj
97
Pitesti
70
98
Mehadia
146
101
85
Hirsova
Urziceni
86
75
138
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Eforie

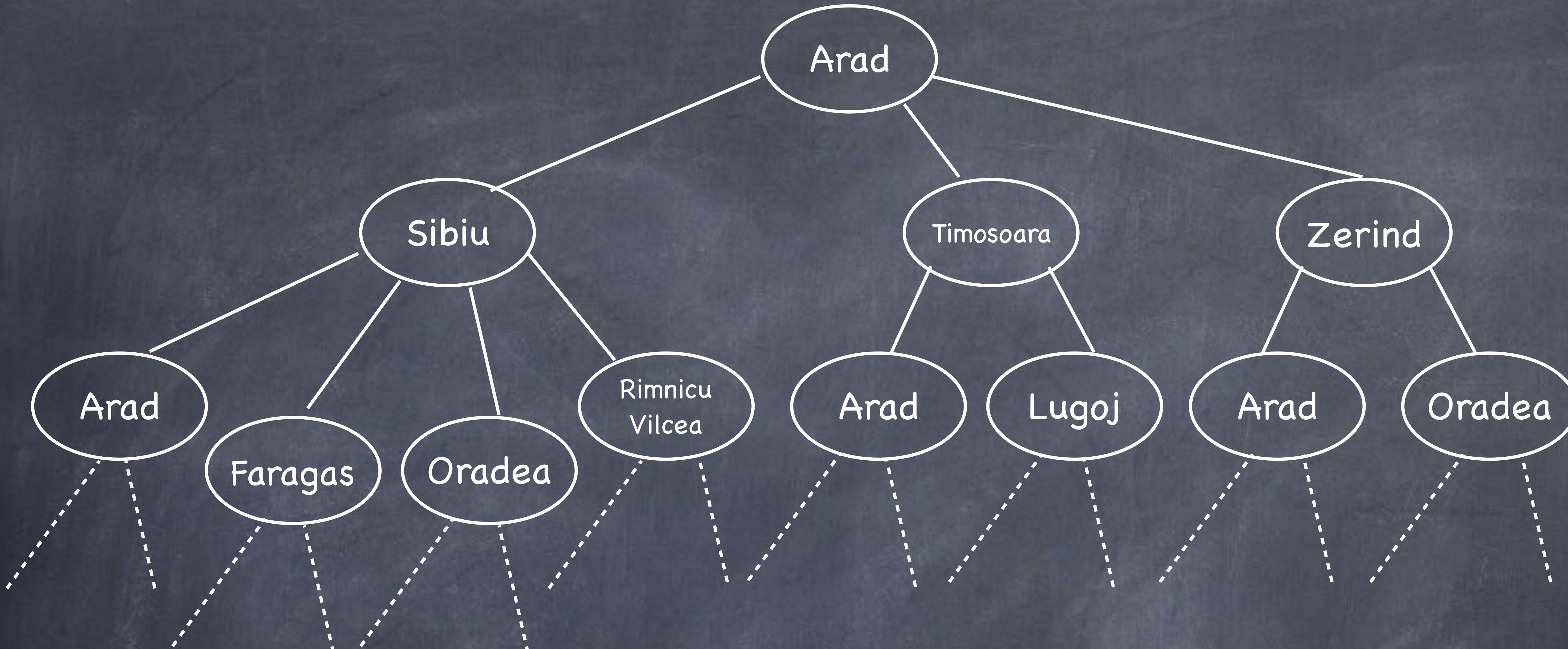

Oradea
71
Zerind
151
75
Arad
140
Neamt
87
Iasi
92
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Vaslui
Timisoara
111
Lugoj
97
Pitesti
211
142
70
Mehadia
146
101
85
Urziceni
98
Hirsova
75
Drobeta
120
138
Bucharest
86
Craiova
90
Giurgiu
Eforie

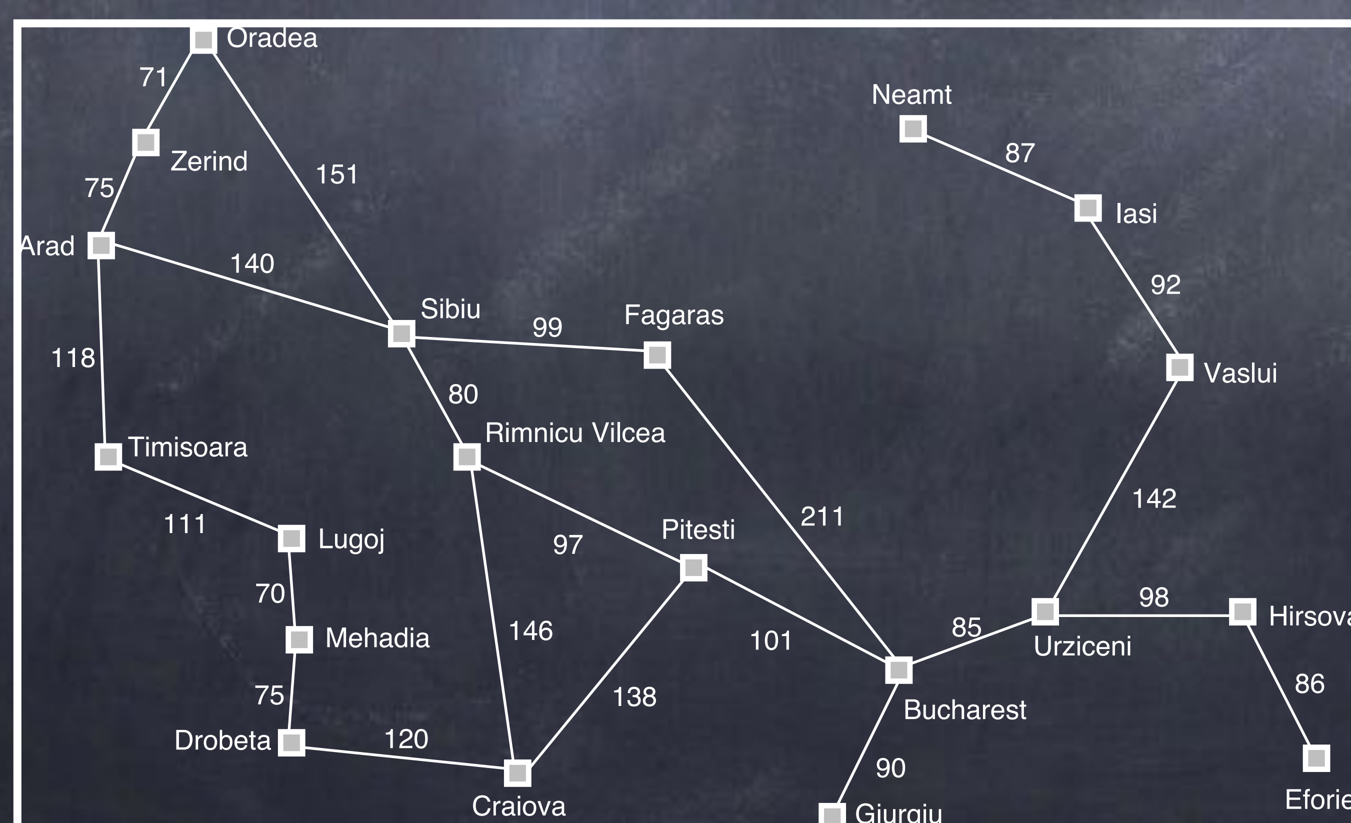

Oradea
71
Zerind
151
75
Arad
140
118
Timisoara
111
Lugoj
70
Mehadia
75
Drobeta
120
Sibiu
99
Fagaras
80
Rimnicu Vilcea
97
146
Craiova
Pitesti
138
101
211
Neamt
87
Iasi
92
Vaslui
142
Urziceni
85
98
Hirsova
86
Eforie
Bucharest
90
Giurgiu


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

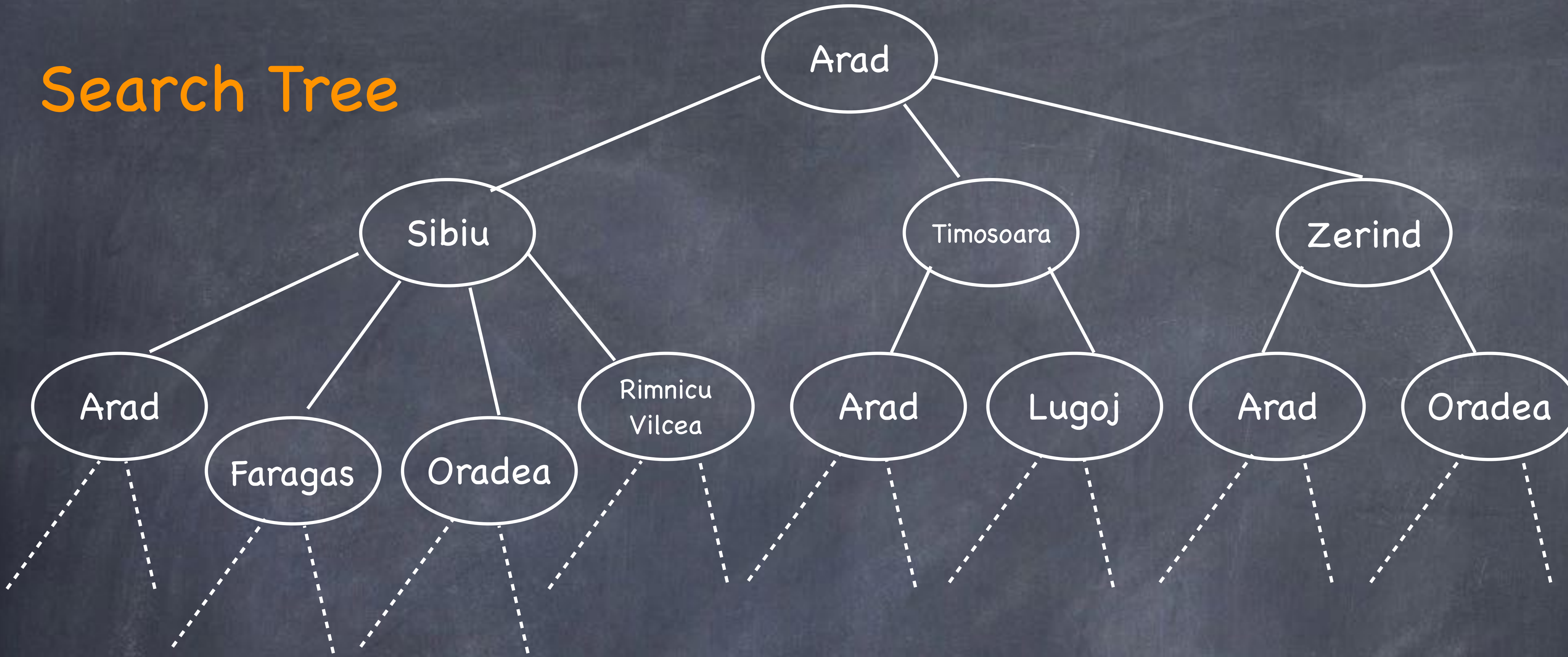

Oradea
71
Zerind
75
151
Arad
140
Neamt
87
Iasi
92
Sibiu
99
Fagaras
118
80
Vaslui
Rimnicu Vilcea
Timisoara
142
111
Lugoj
97
Pitesti
211
70
98
Hirsova
Mehadia
146
101
85
Urziceni
86
75
138
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Eforie

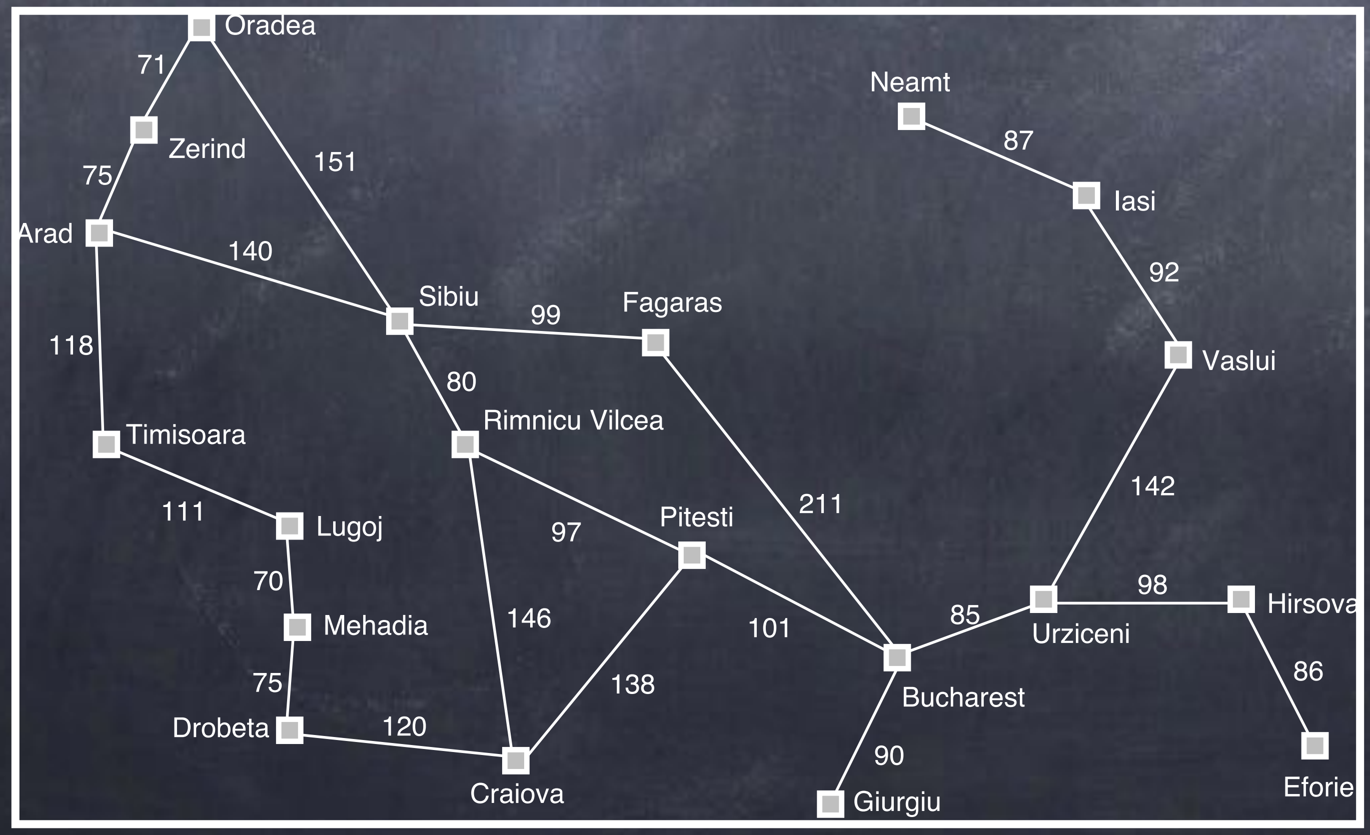

Oradea
71
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
98
Hirsova
85
Urziceni
Bucharest
86
90
Giurgiu
Eforie


Arad
Sibiu
Timosoara
Zerind
Arad
Faragas
Oradea
Rimnicu
Vilcea
Arad
Lugoj
Arad
Oradea

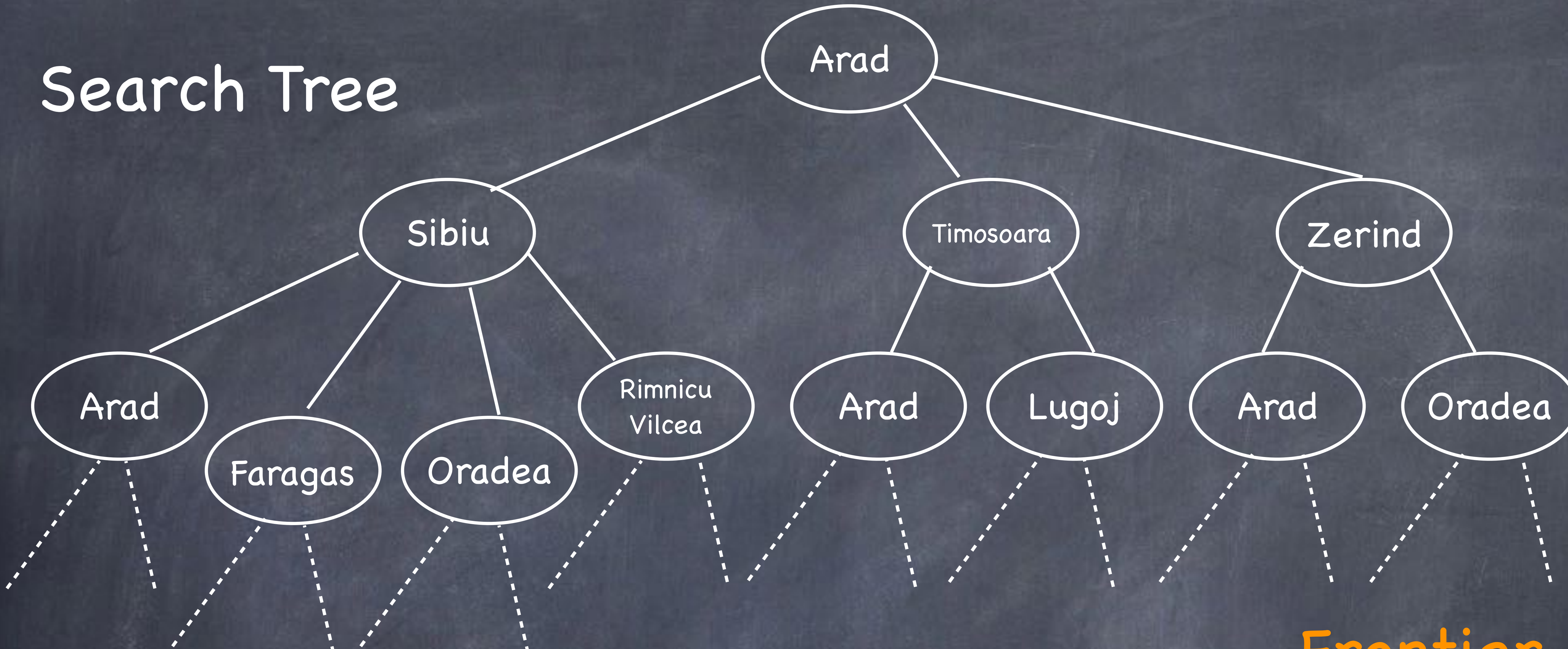

Oradea
71
Neamt
Zerind
151
87
75
Iasi
Arad
140
92
Sibiu
99
Fagaras
118
Vaslui
80
Rimnicu Vilcea
Timisoara
142
111
211
Lugoj
97
Pitesti
70
98
85
Hirsova
Mehadia
146
101
Urziceni
86
75
138
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Eforie

Search Tree

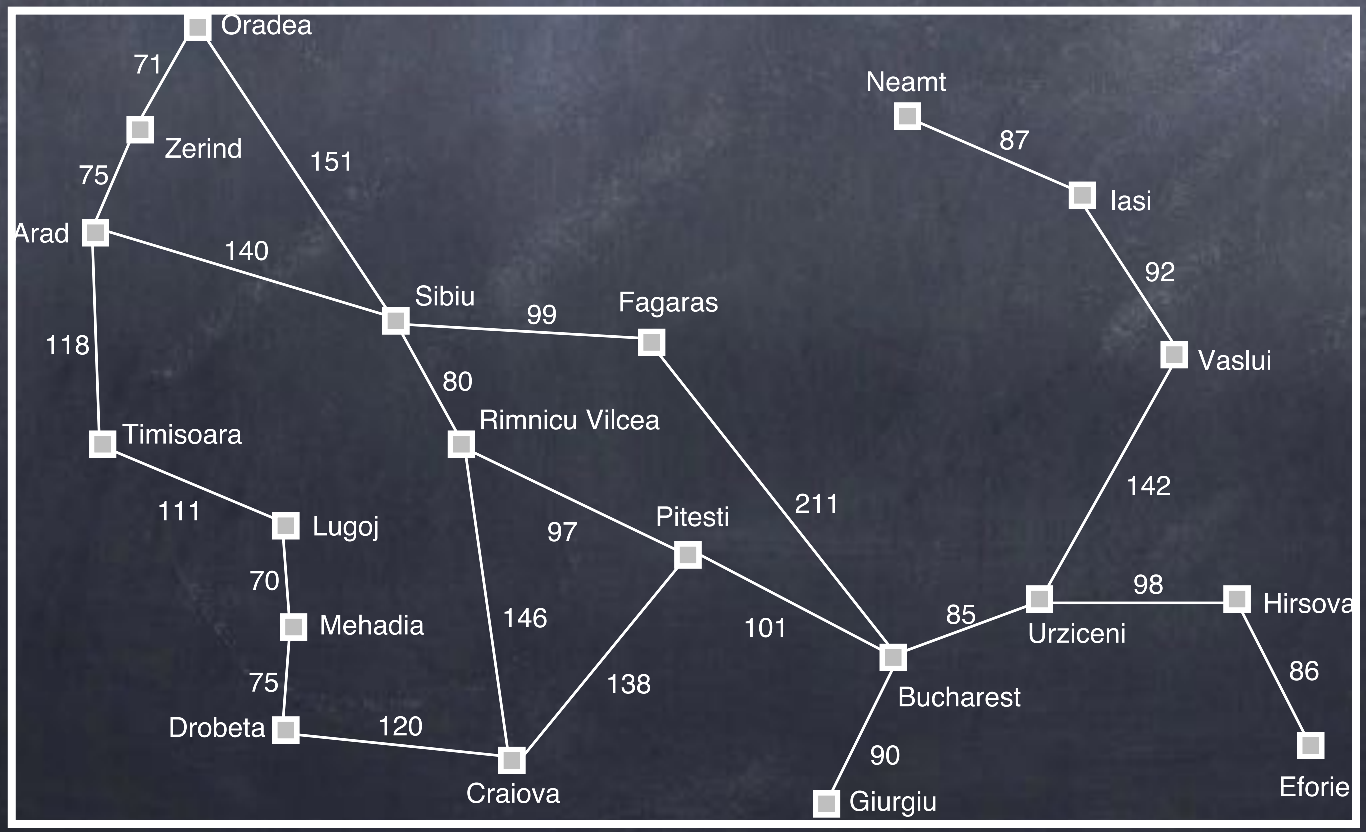

Arad
Sibiu
Timosoara
Zerind
Arad
Faragas
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea


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

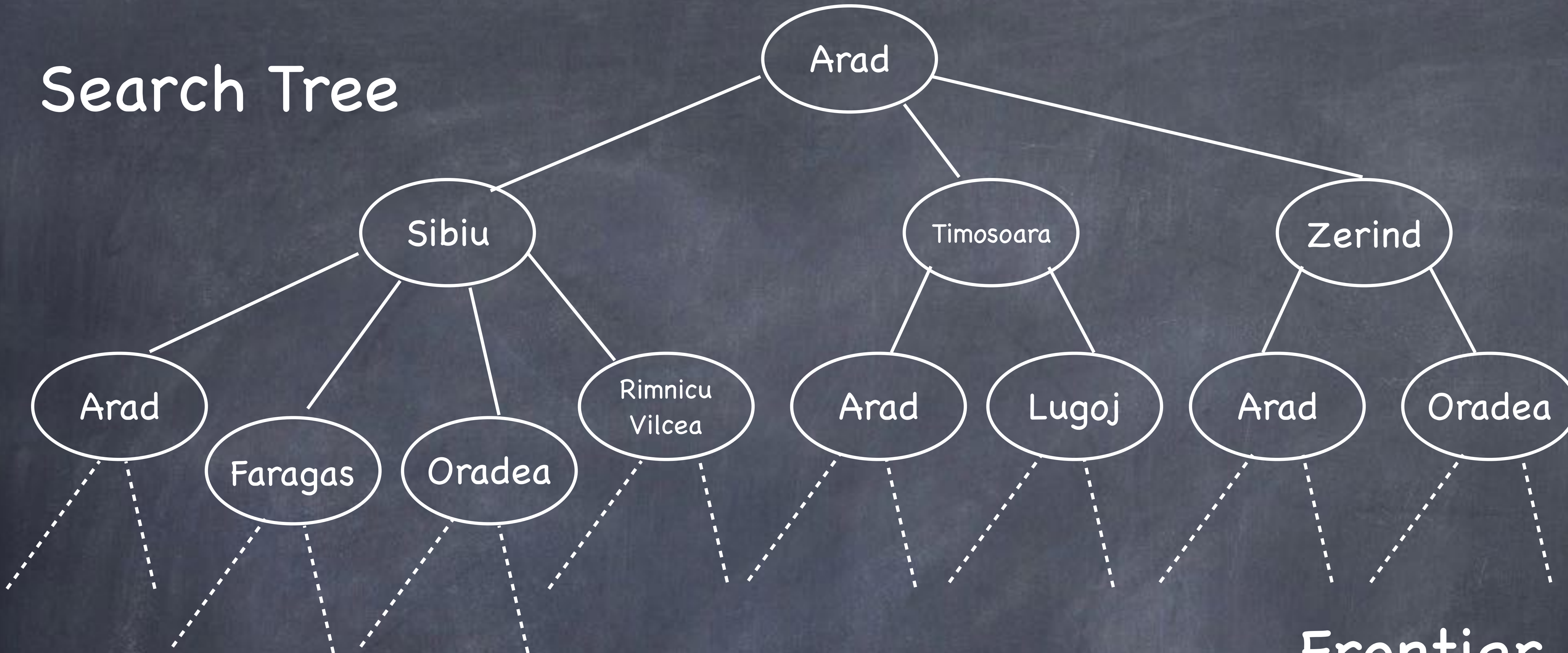
Search Tree



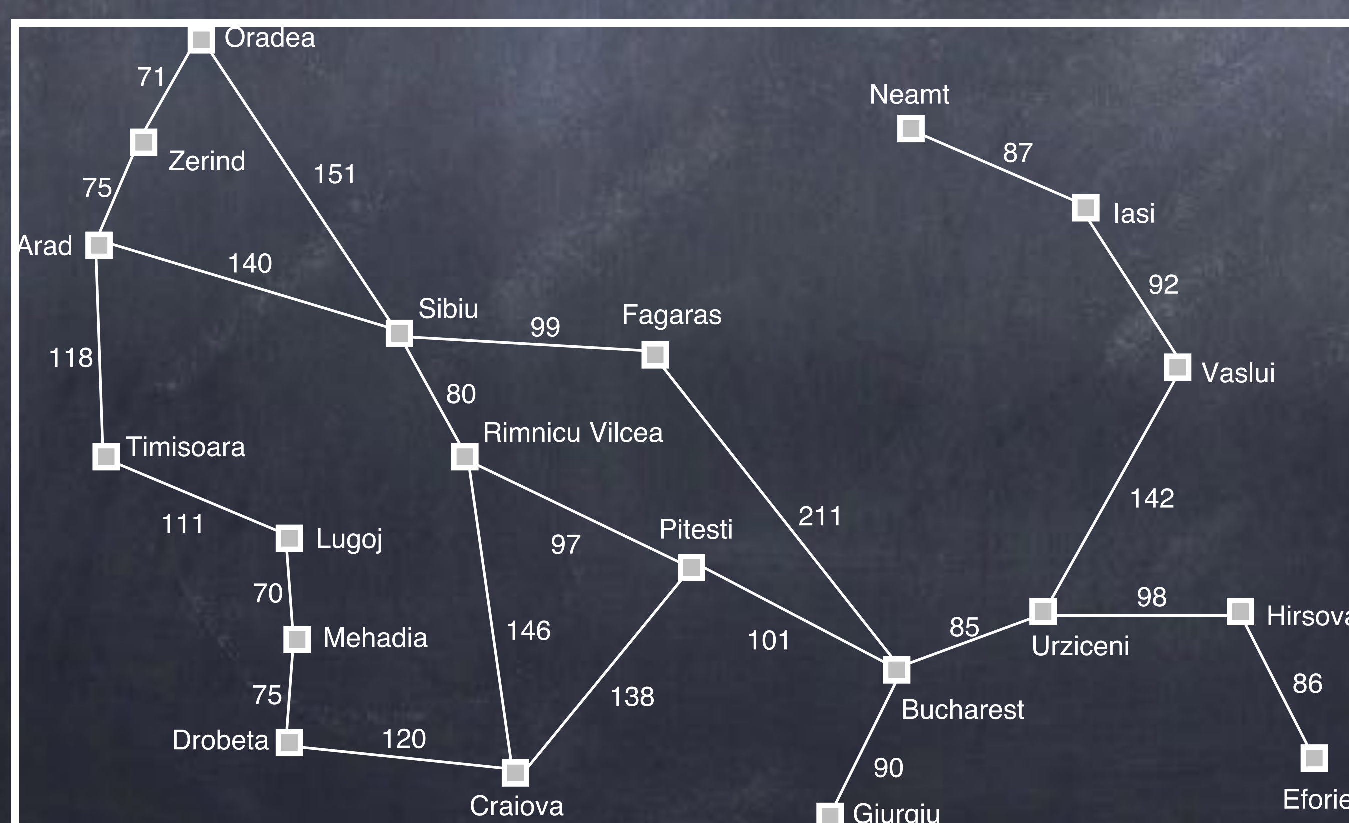
Frontier

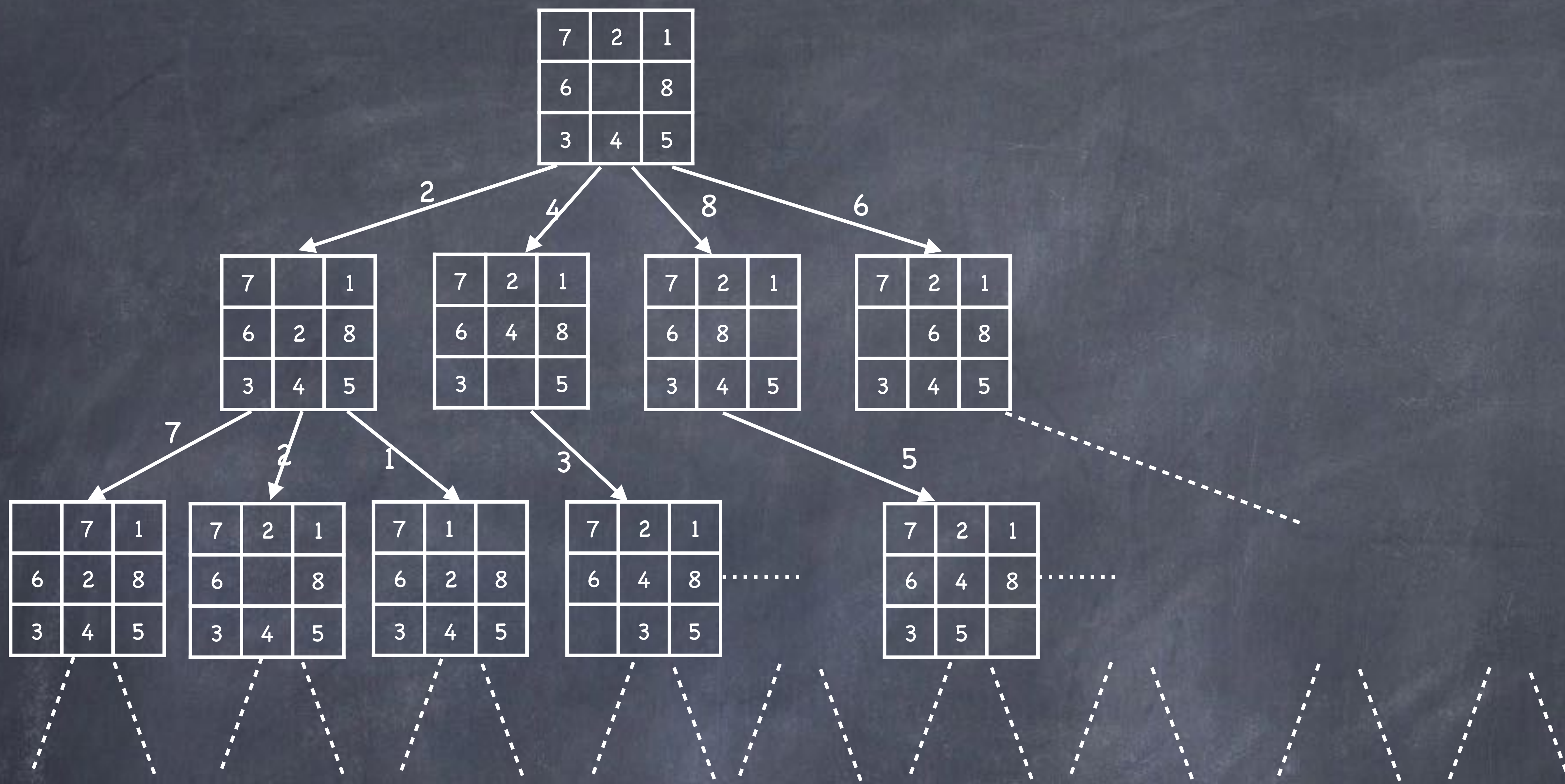


Search Tree



Frontier




7 2 1
6 8
3 4 5
2
4
8
6
7 1
6 2 8
3 4 5
7 2 1
6 4 8
3 5
7 2 1
6 8
3 4 5
7 2 1
6 8
3 4 5
7
2
1
3
5
7 1
6 2 8
3 4 5
7 2 1
6 8
3 4 5
7 1
6 2 8
3 4 5
7 2 1
6 4 8
3 5
7 2 1
6 4 8
3 5

State Space

- The set of all states reachable from the initial state by some sequence of actions

States

Actions → State Space

Transition Model

State-Space Search Tree

- The tree formed by starting with the initial state and using applicable actions to generate successor (child) states
- May be infinite (even for finite search spaces)
- Not stored explicitly

State-Space Search

- Start with initial state
- Generate successor states by applying applicable actions
- Until you find a goal state

Initialize the frontier to just I

While the frontier is not empty:

    Remove a state s from the frontier

    If $s \in G$:

        Return solution to s

    else:

        Add $successors(s)$ to the frontier

Universal Problem-Solving Procedure

Initialize the frontier to just I

While the frontier is not empty:

  Remove a state s from the frontier

  If $s \in G$:

    Return solution to s

  else:

    Add $successors(s)$ to the frontier

Coolest Program Ever

Initialize the frontier to just I

While the frontier is not empty:

 Remove a state s from the frontier

 If $s \in G$:

  Return solution to s

 else:

  Add $successors(s)$ to the frontier

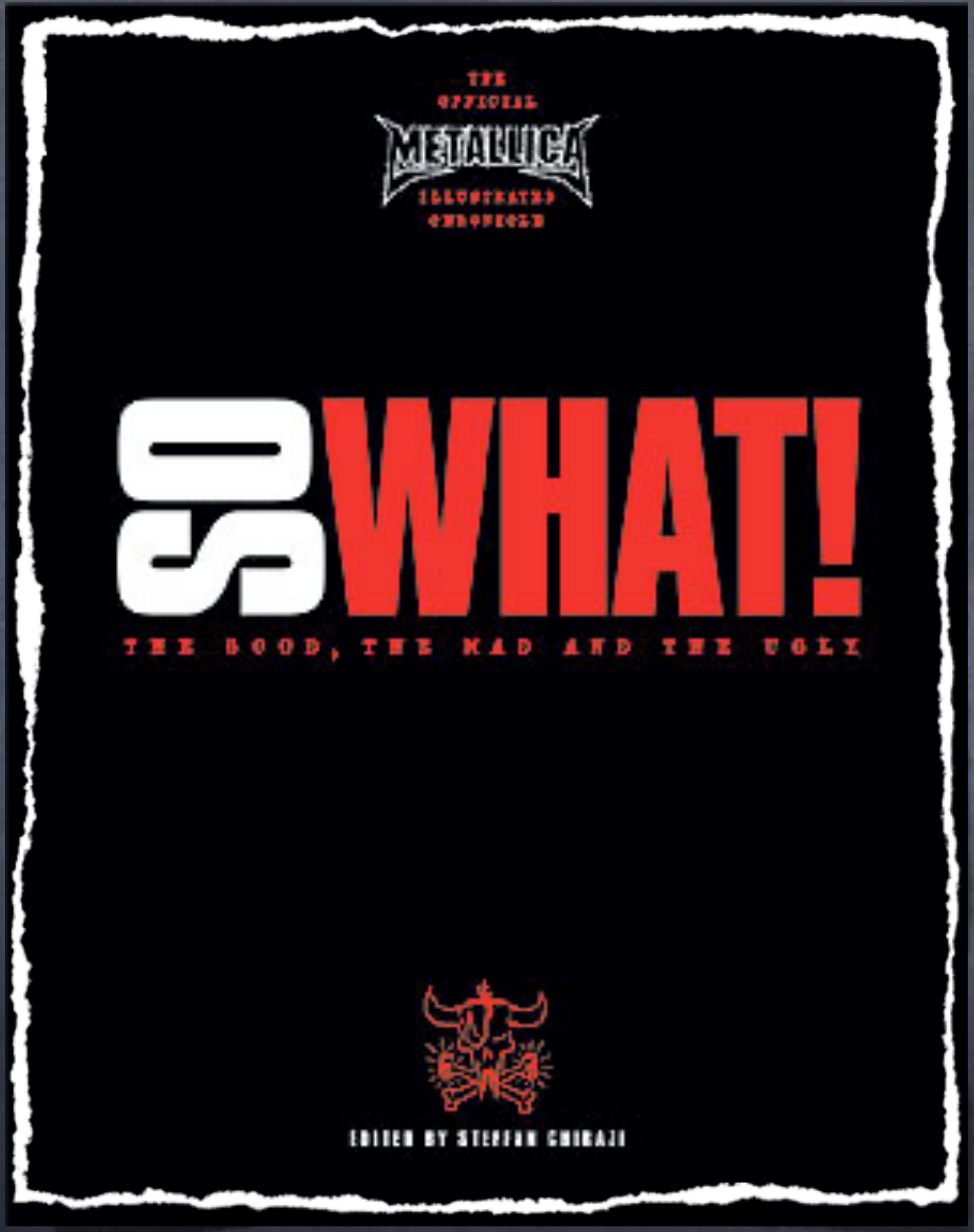
NO CODE REQUIRED!

AIMA Fig. 3.7

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

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

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

        frontier.add(n);

    }
  }
}
```

AIMA Fig. 3.7

```
Solution graphSearch(Problem p) {
  Set<Node> frontier = new Set<Node>(p.getInitialState());
  Set<Node> explored = new Set<Node>();
  while (true) {
    if (frontier.isEmpty()) {
      return null;
    }
    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);
      }
    }
  }
}
```


THE OFFICIAL
METALLICA
SO WHAT!
THE GOOD, THE MAD AND THE UGLY

State-Space Search

- Formal model of problems and solutions based on states and actions that transition between them
- General-purpose algorithm for solving any problem that can be represented using the model
- State-space search framework will allow us to explore and compare different problem-solving strategies

Problem Solving (By Computers)

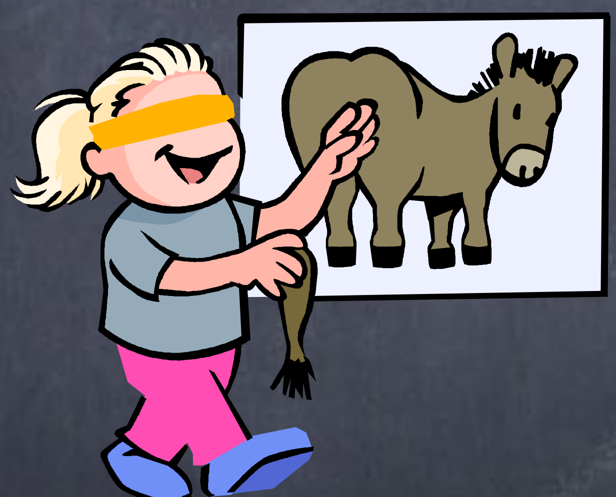
- If you have a better algorithm: use it
- If you don't: state space search
- For many (most? all?) interesting problems, we know there is no tractable (polynomial) algorithm

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 null;
    }
    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);
      }
    }
  }
}
```

Search Strategies

Uninformed



No additional information about states

Informed (Heuristic)

Can identify "promising" states

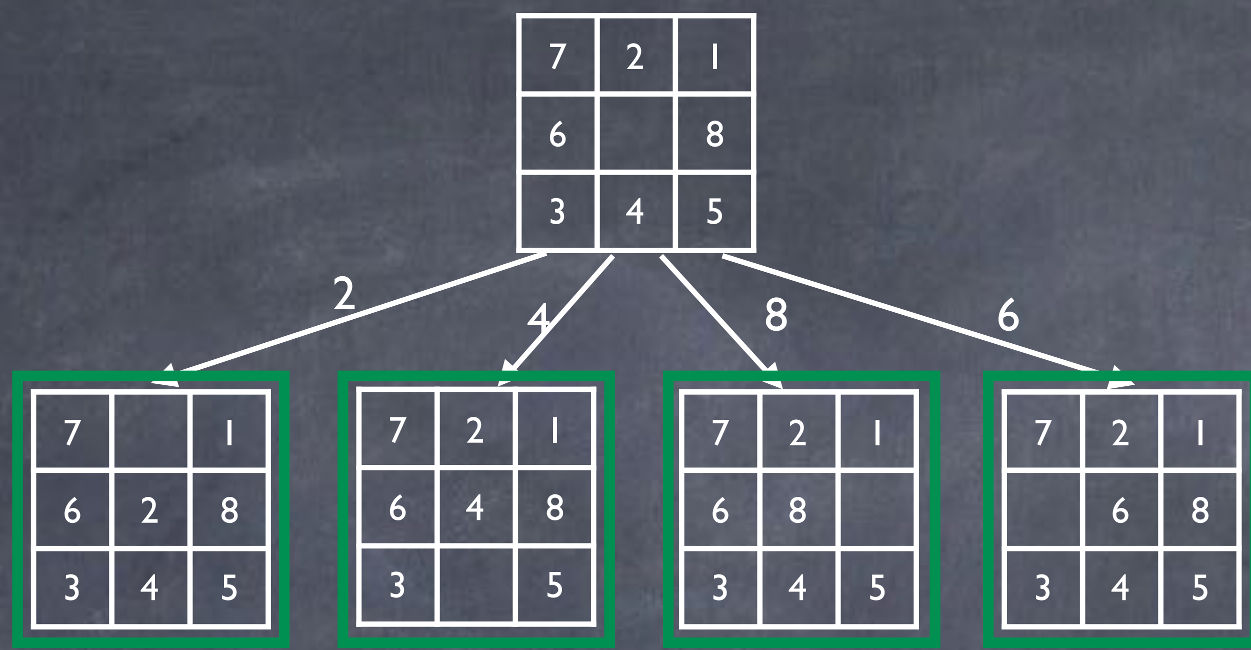
Uninformed Strategies

“all you can do is generate successors and distinguish a goal state from a non-goal state.” (p 81)

7	2	1
6		8
3	4	5

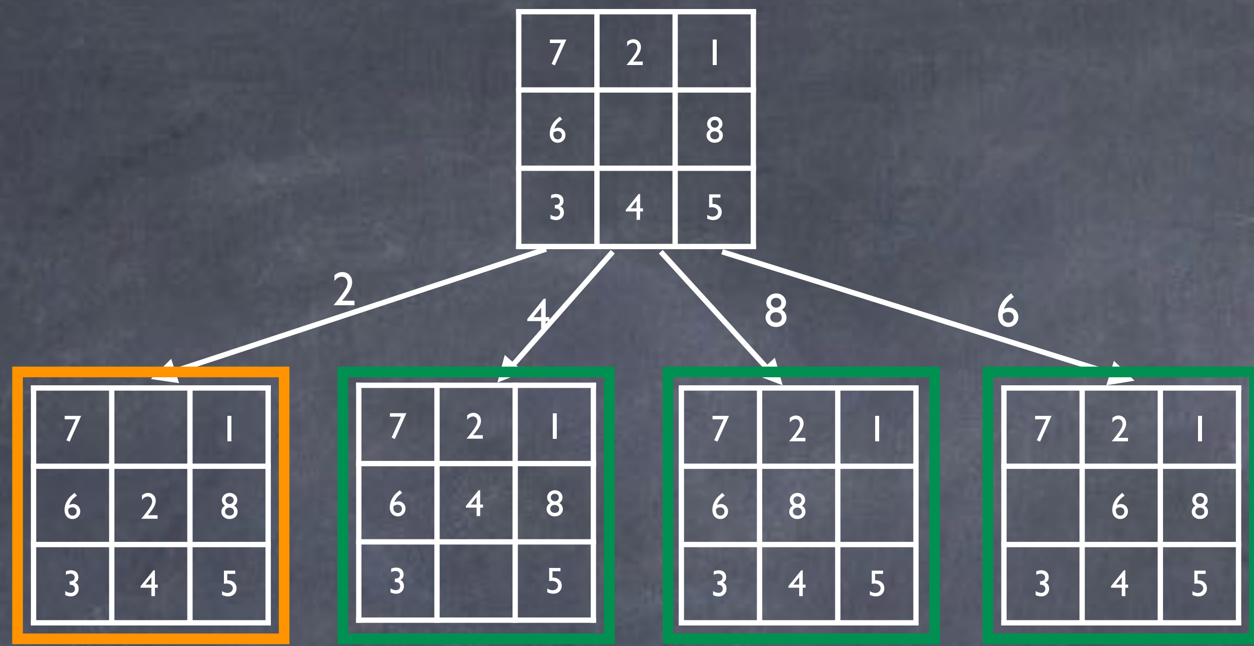
M^G:

0	1	2
3	4	5
6	7	8



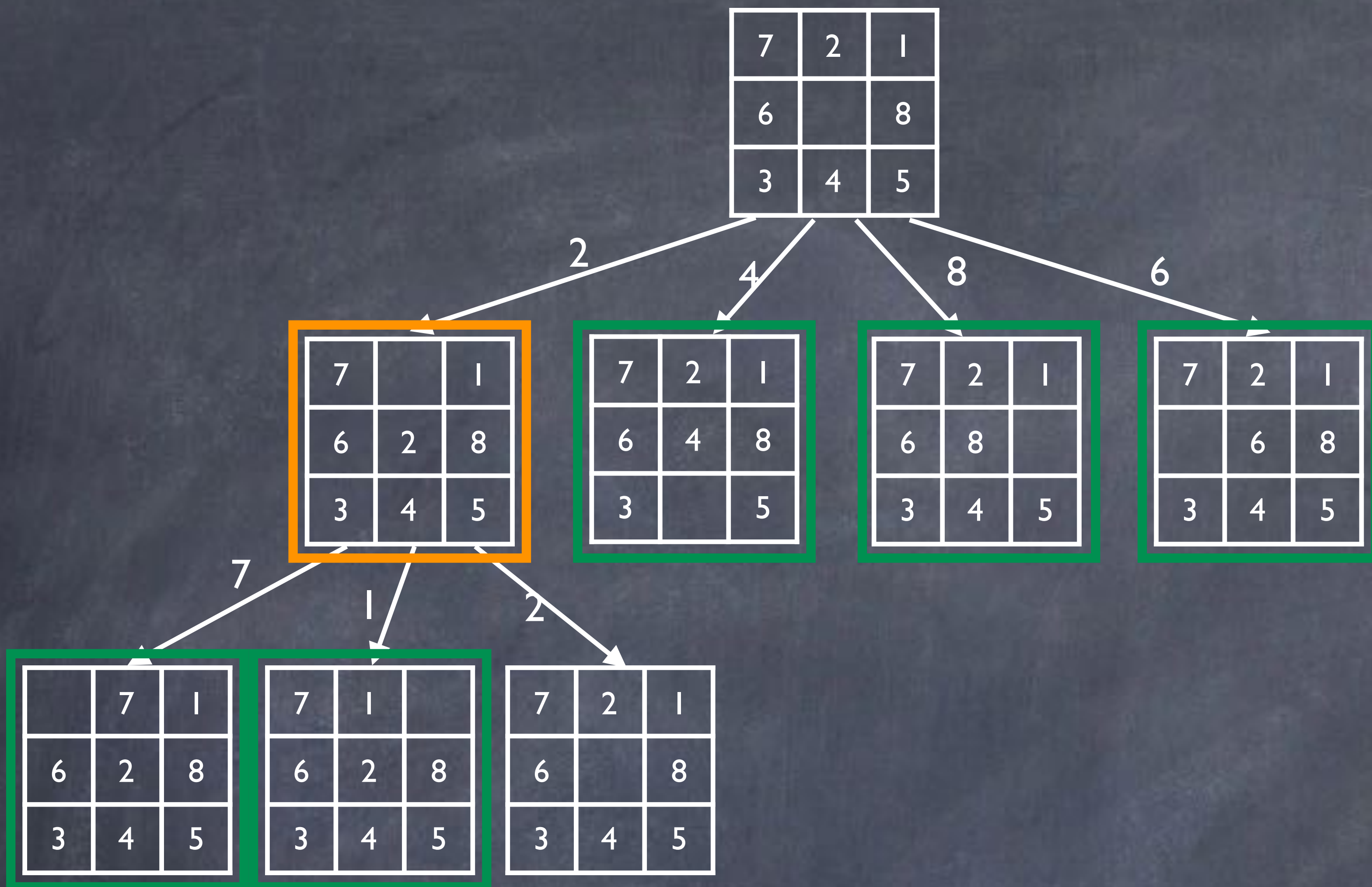
M^G:

0	1	2
3	4	5
6	7	8



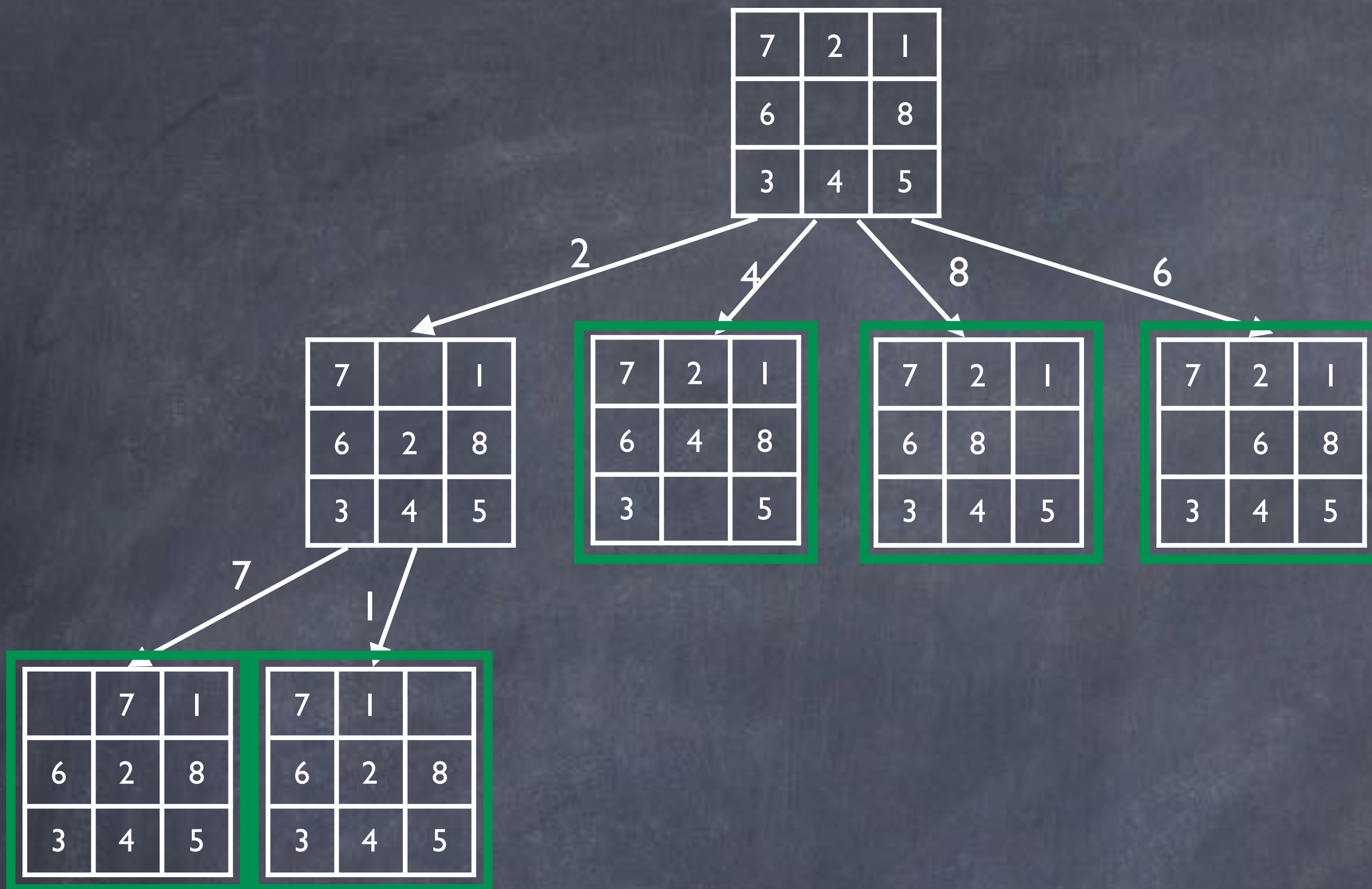
M^G:

0	1	2
3	4	5
6	7	8



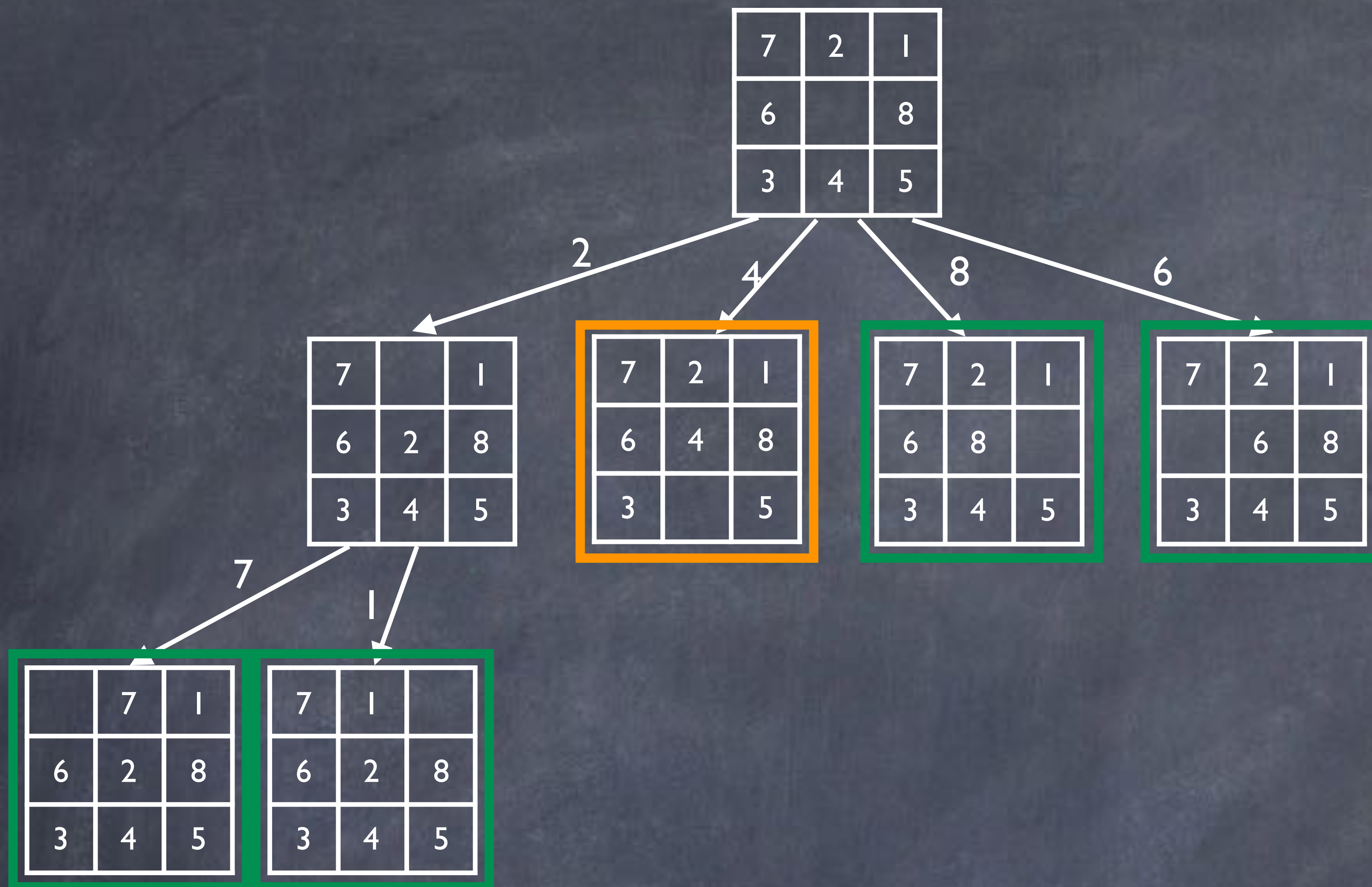
M^G:

0	1	2
3	4	5
6	7	8



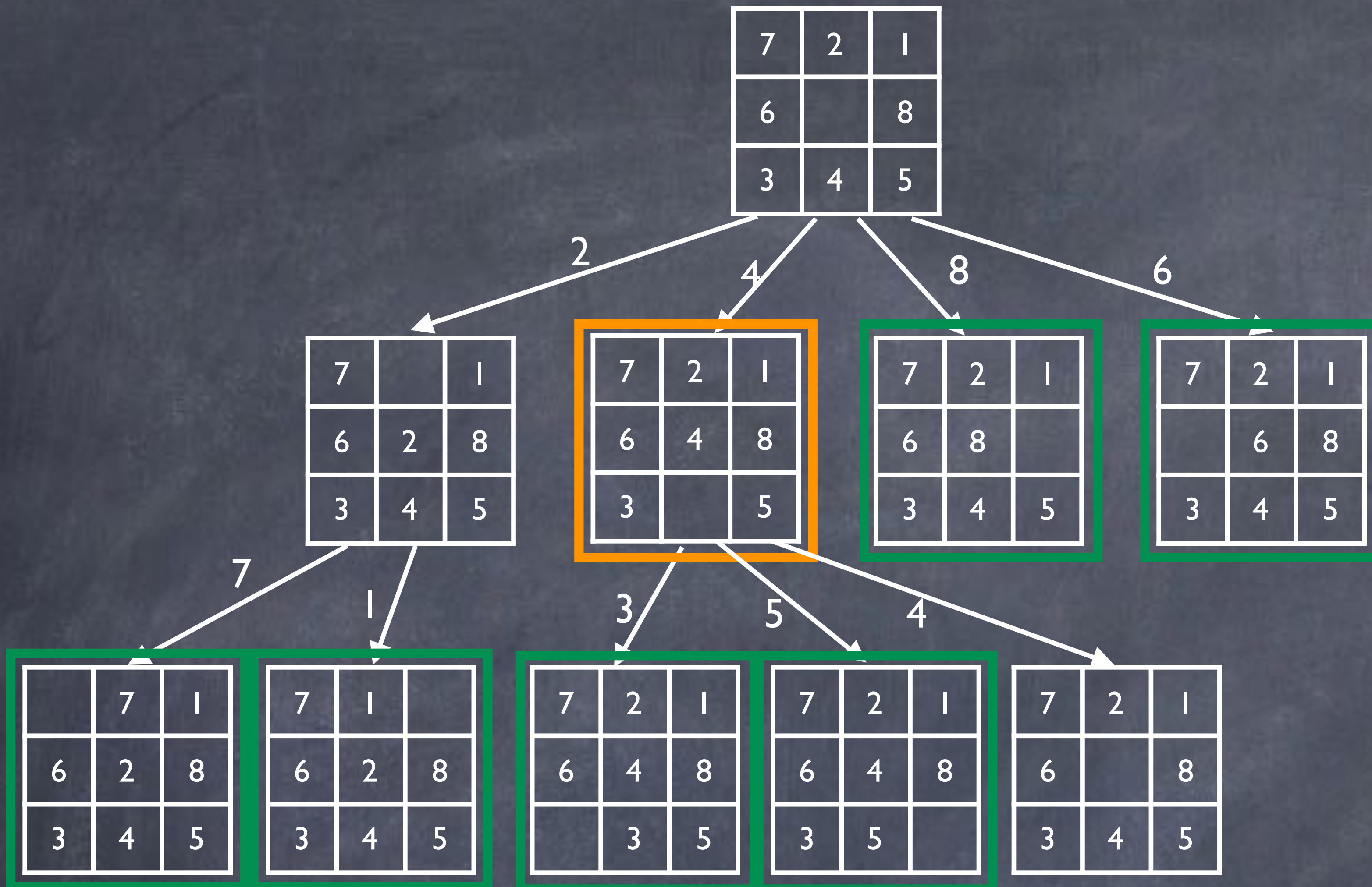
M^G:

0	1	2
3	4	5
6	7	8



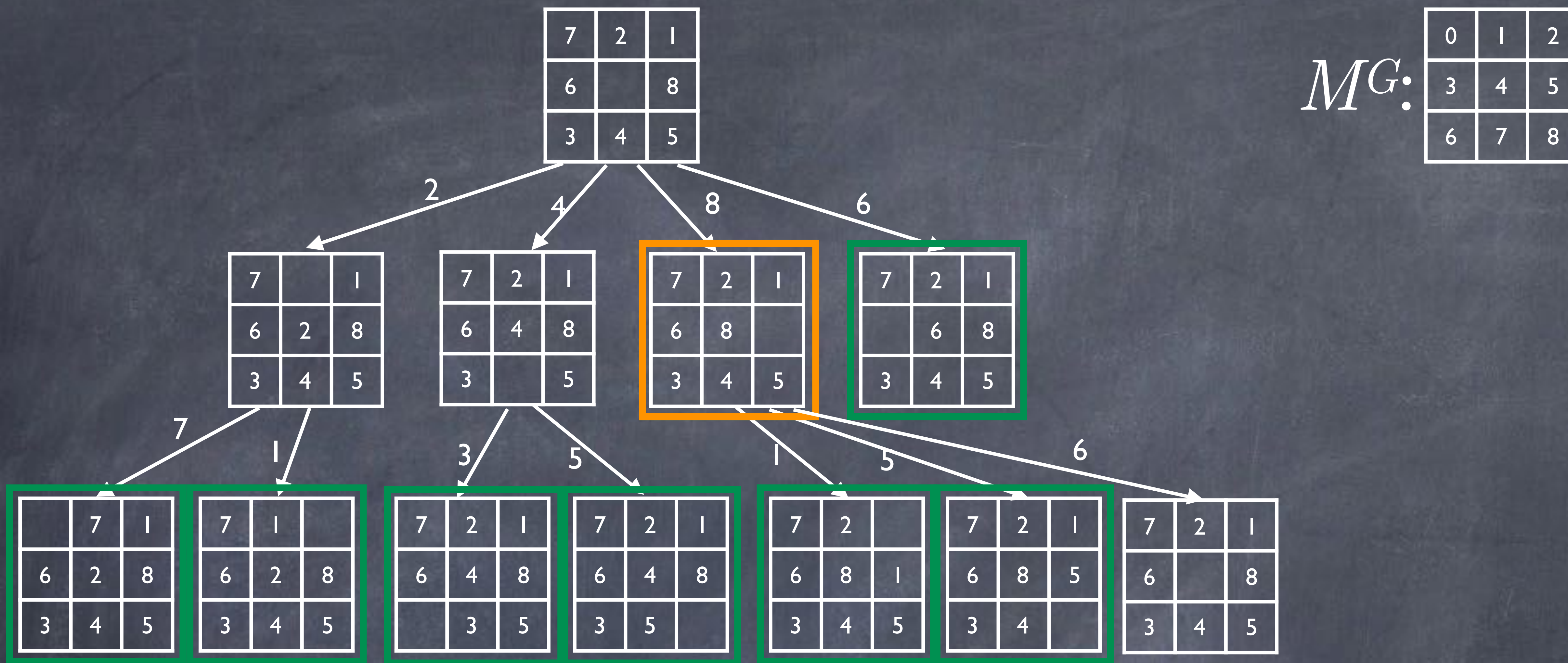
M^G:

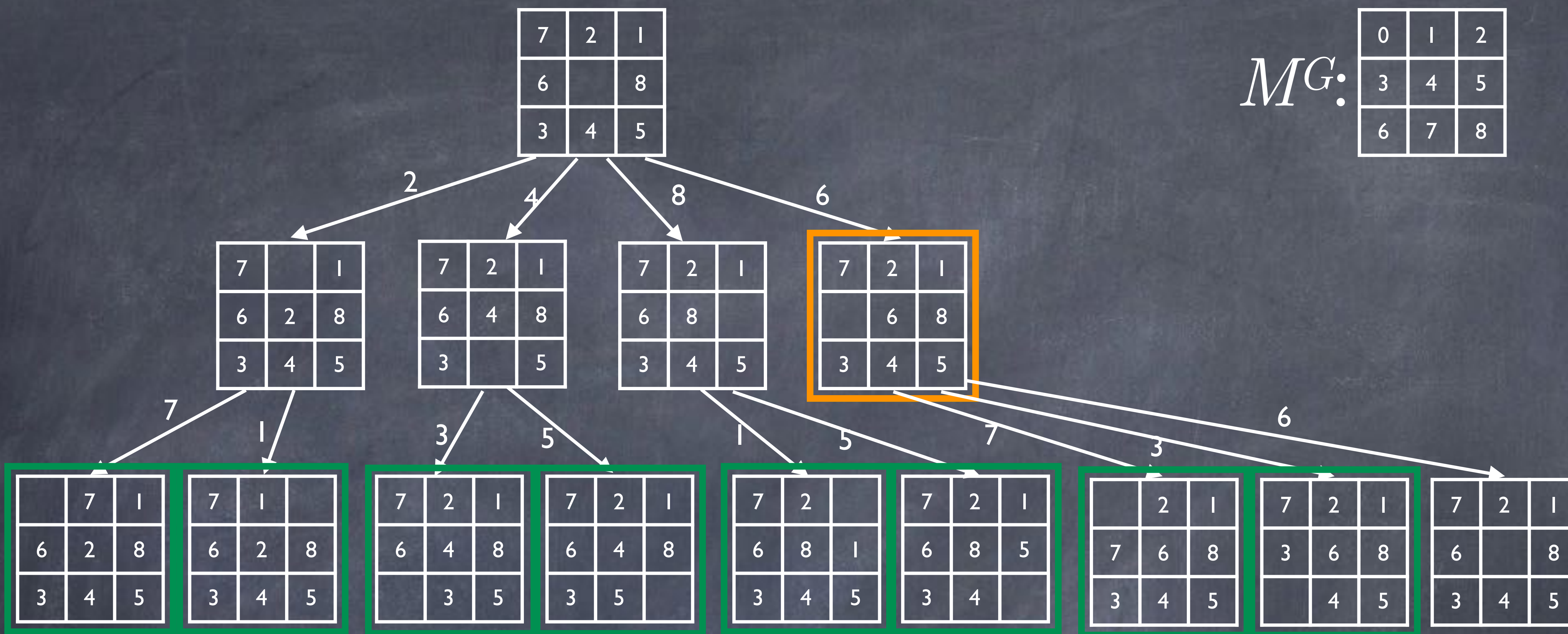
0	1	2
3	4	5
6	7	8



M^G:

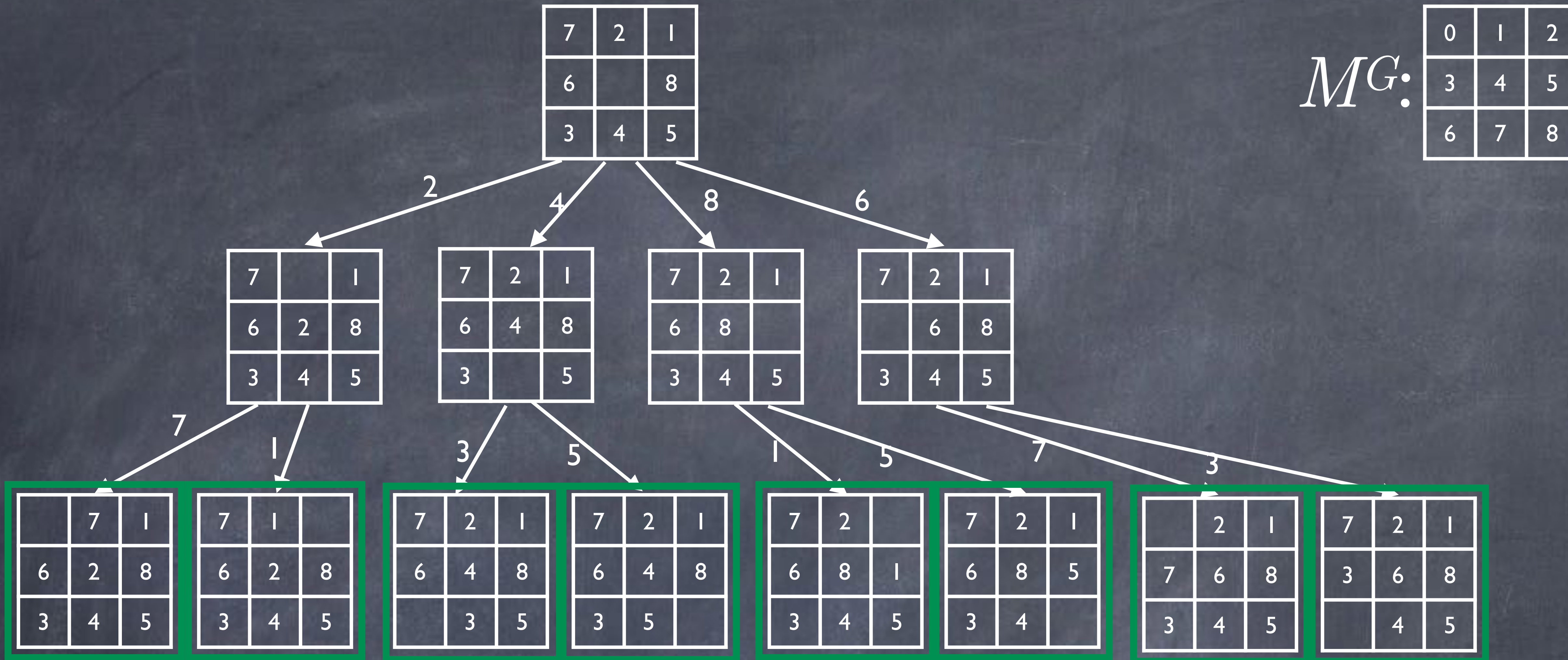
0	1	2
3	4	5
6	7	8


M^G:



M^G:

0	1	2
3	4	5
6	7	8


M^G:

Breadth-First Search

- Expand all the nodes in a level before expanding any of their children
- Expand the shallowest unexpanded node
- Use a FIFO queue for the frontier

Does It Work?

Completeness

Optimality

Time Complexity

Space Complexity

BFS Analysis

Completeness

Optimality

Time Complexity

Space Complexity

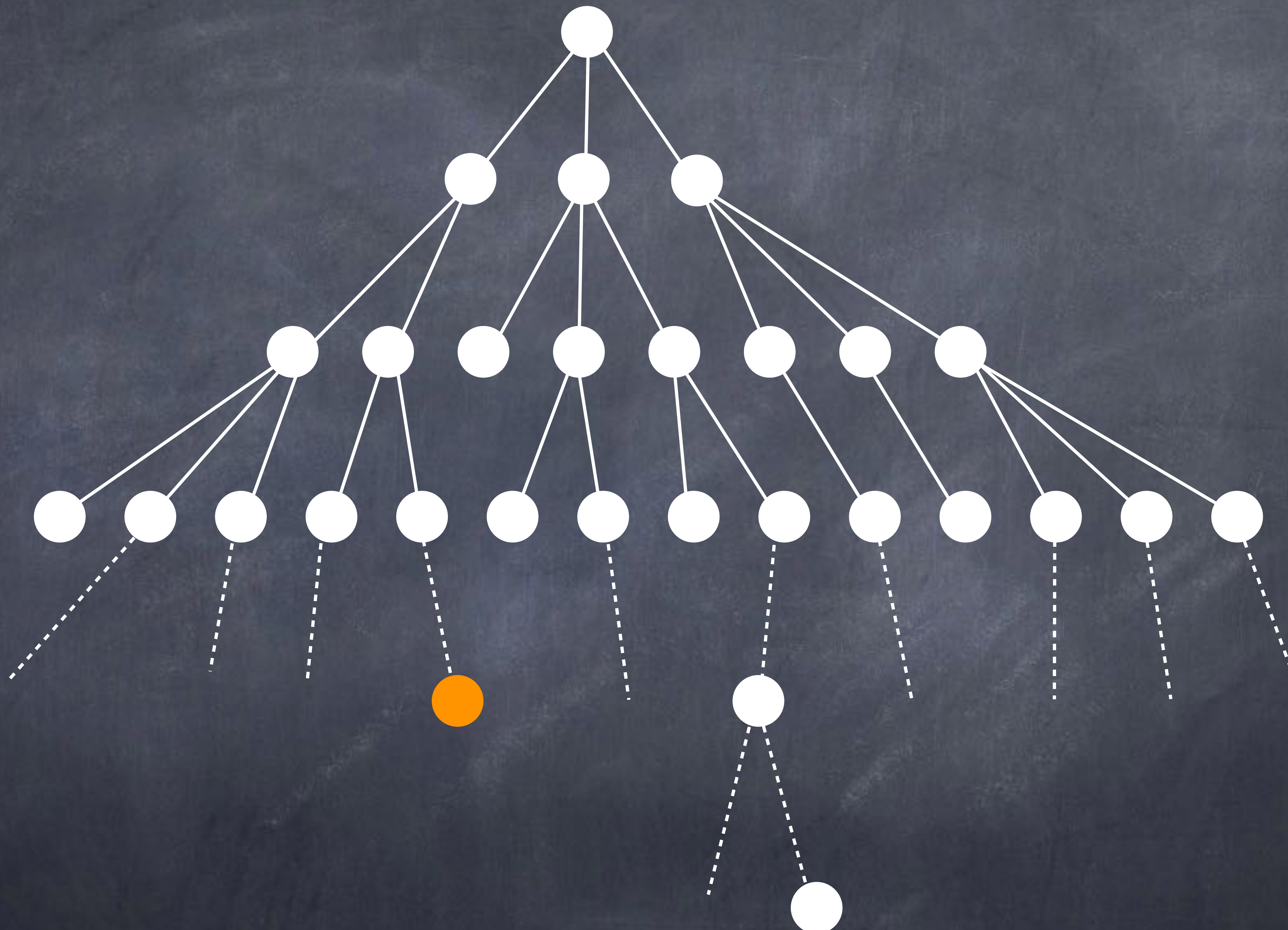
BFS Analysis

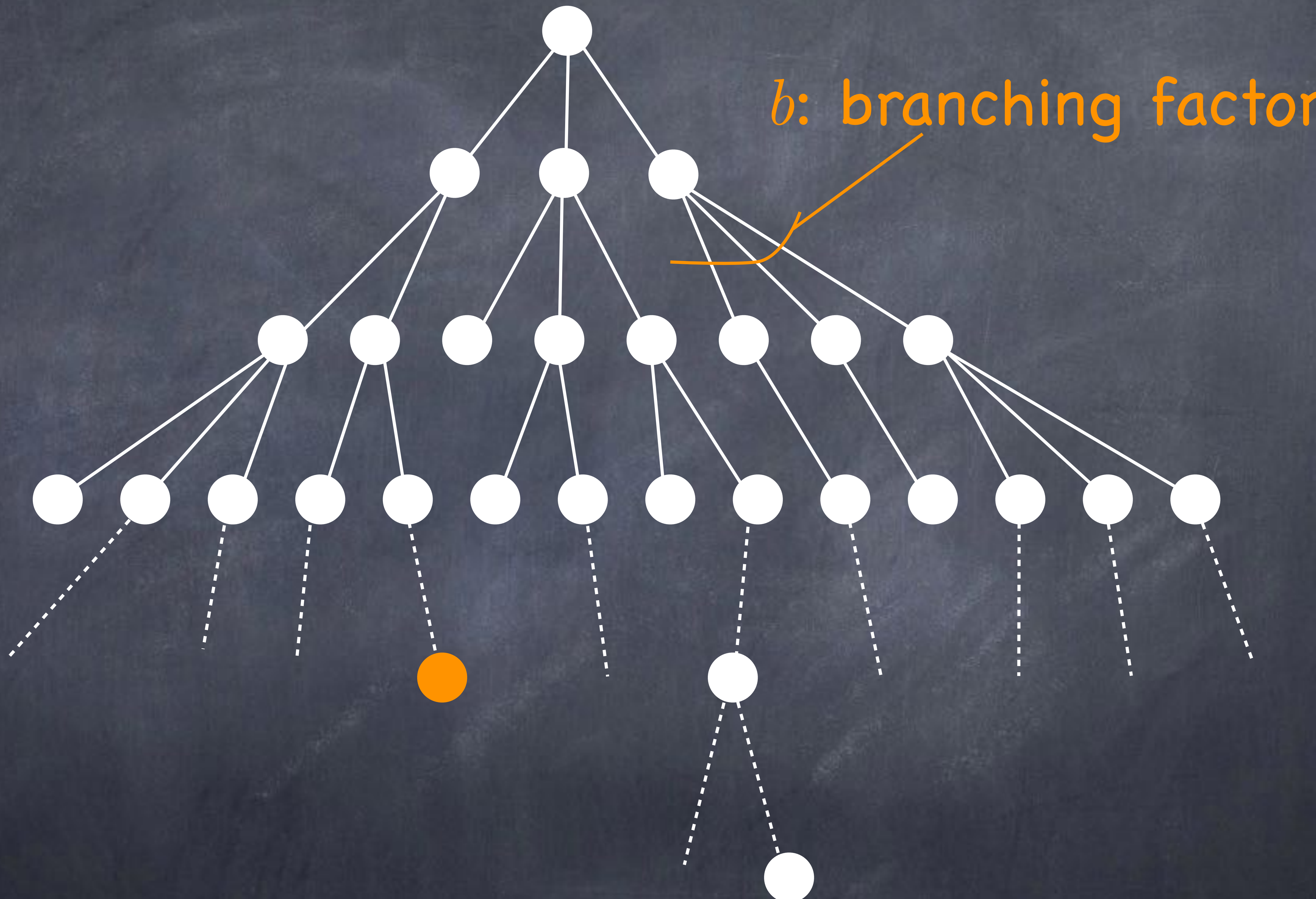
Completeness

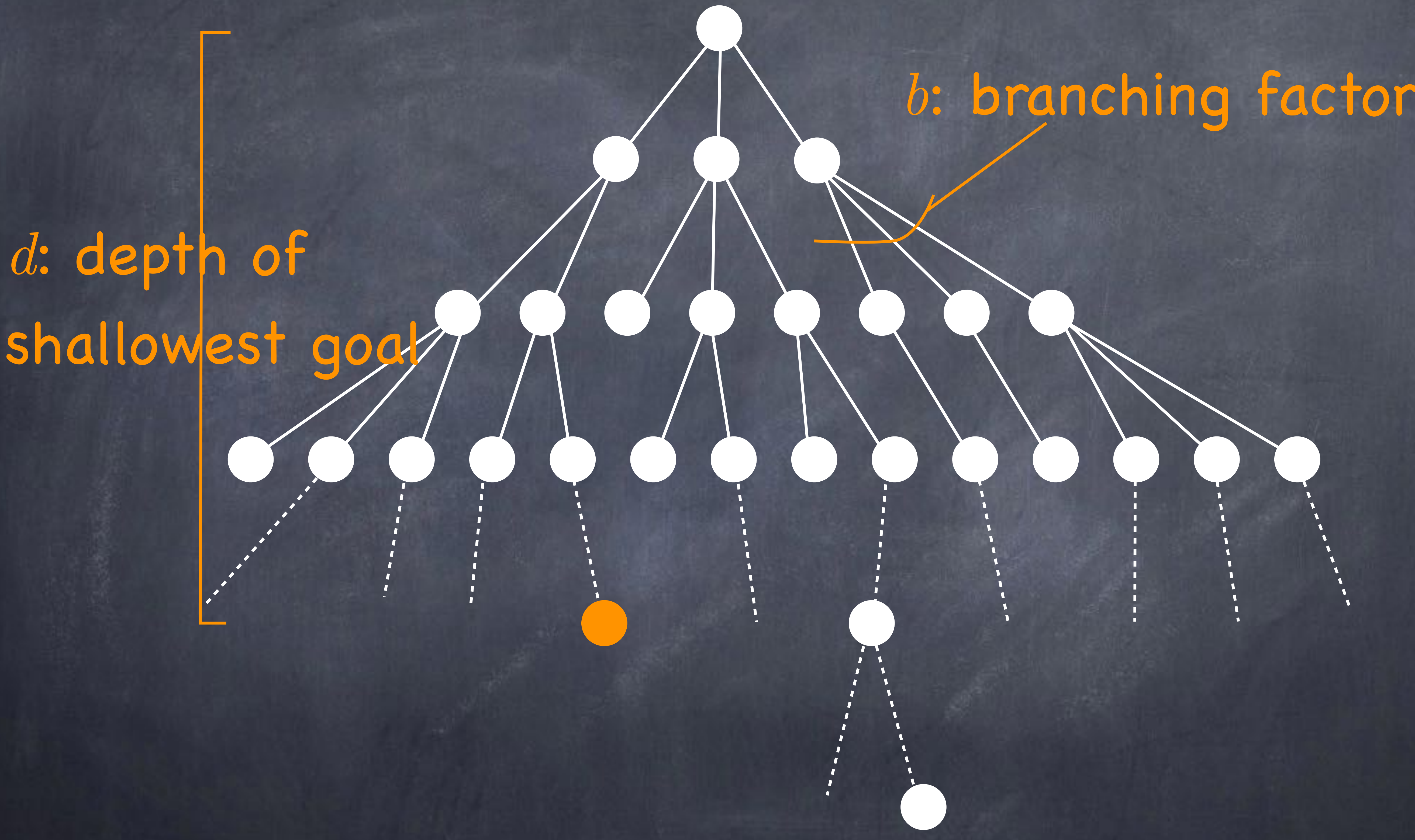
Optimality

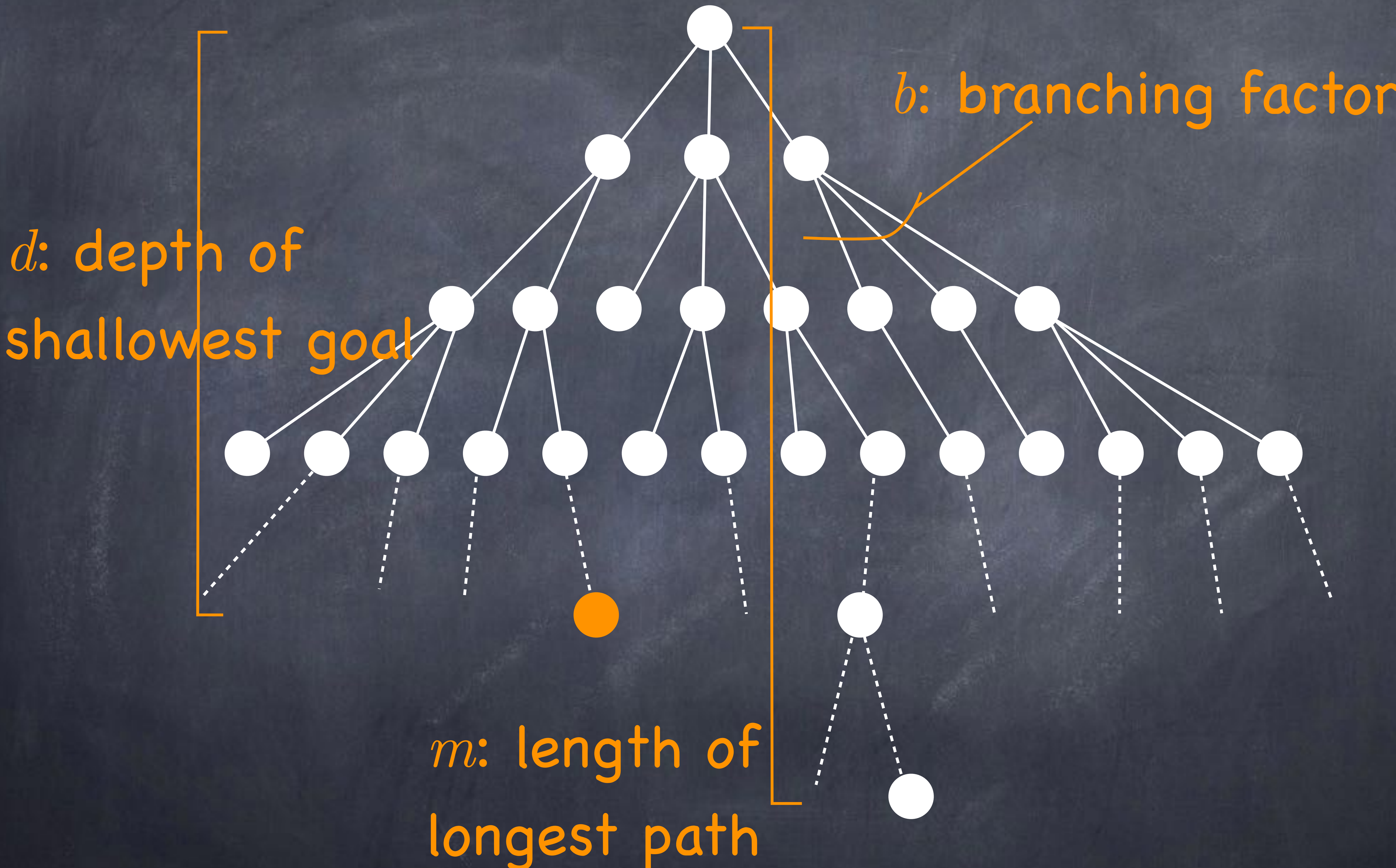
Time Complexity

Space Complexity



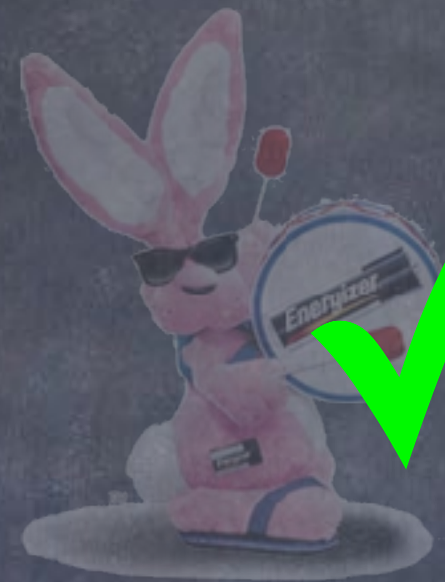

b: branching factor


b: branching factor
d: depth of shallowest goal



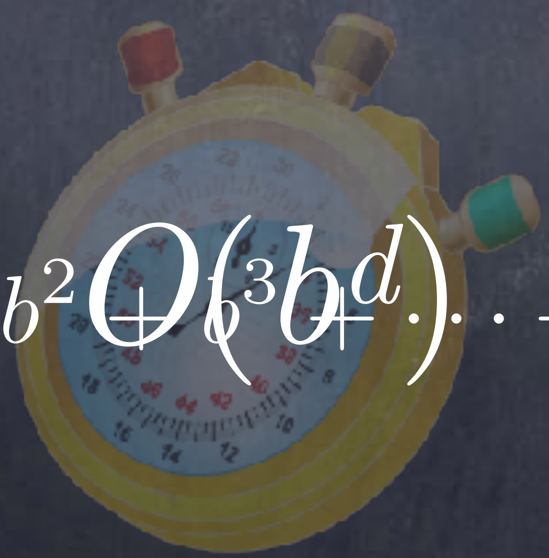
b: branching factor
d: depth of shallowest goal
m: length of longest path

BFS Analysis



Completeness

Optimality



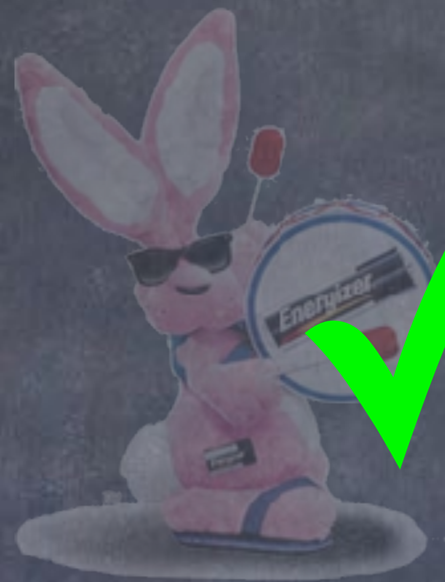
$$b + b^2 + b^3 + \cdots + b^d$$

$$O(b^d)$$

Time Complexity

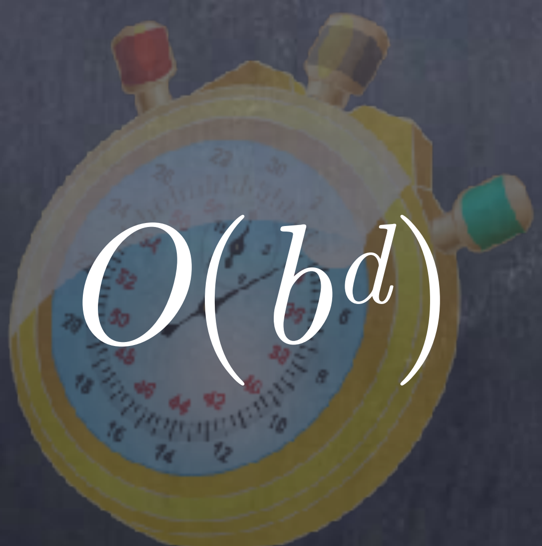
Space Complexity

BFS Analysis



Completeness

Optimality



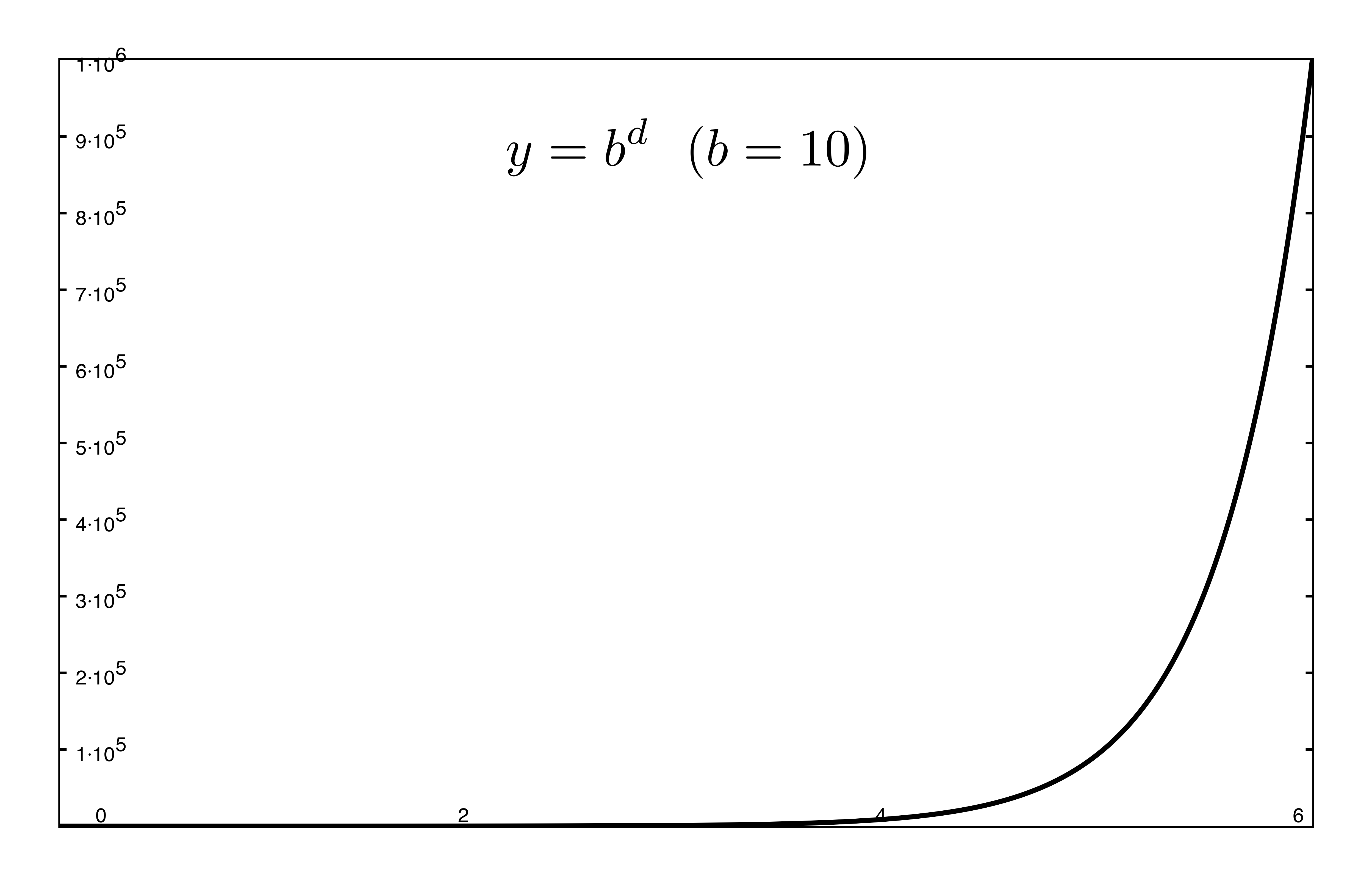
$O(b^d)$

Time Complexity

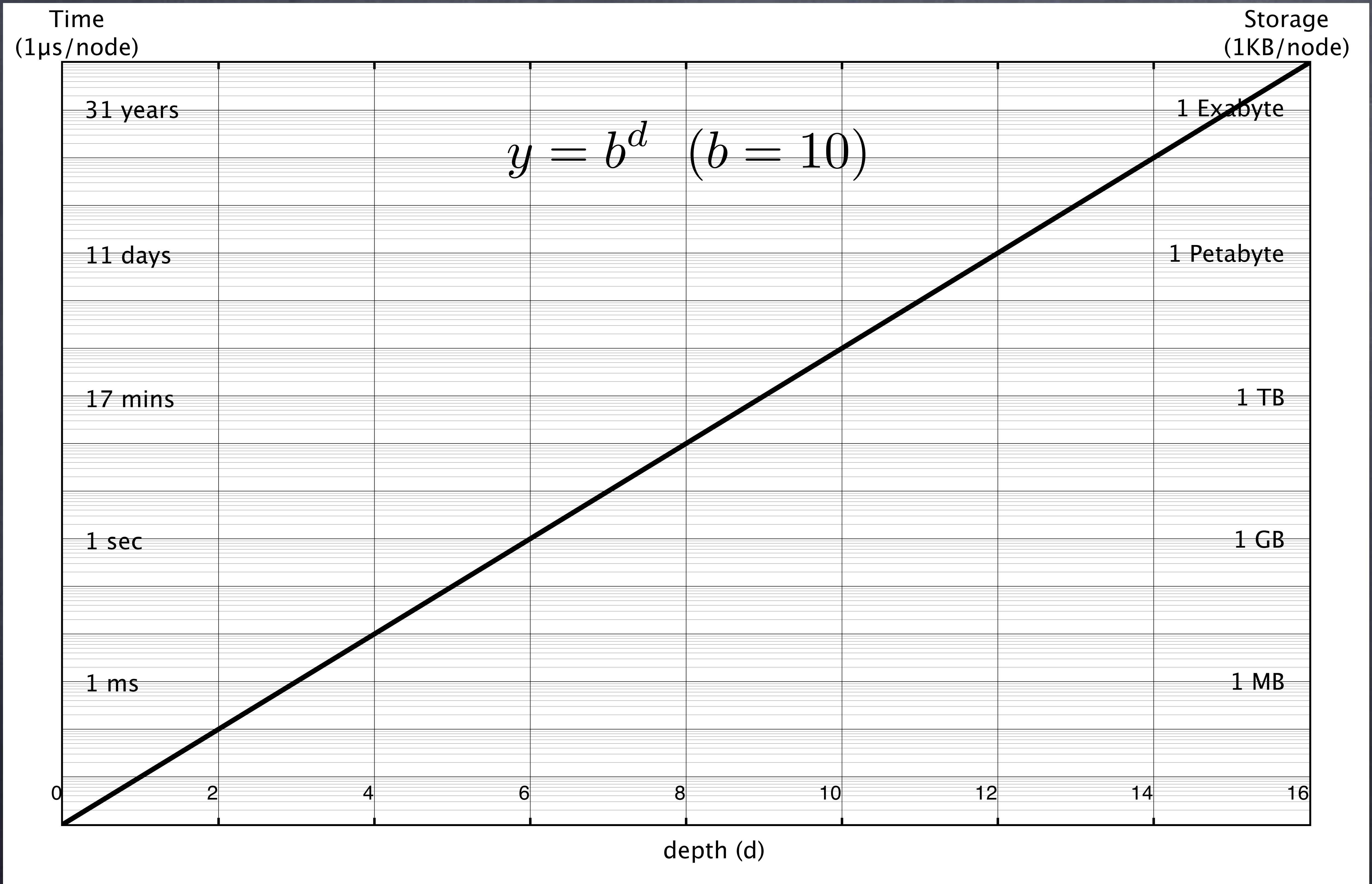
$O(b^d)$

$b^d + b^{d-1}$

Space Complexity



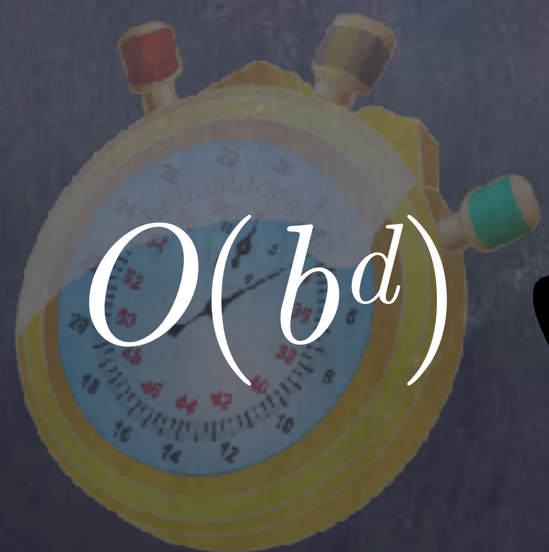
$y = b^d \quad (b = 10)$
$1\cdot10^6$
$9\cdot10^5$
$8\cdot10^5$
$7\cdot10^5$
$6\cdot10^5$
$5\cdot10^5$
$4\cdot10^5$
$3\cdot10^5$
$2\cdot10^5$
$1\cdot10^5$
0
2
4
6


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

BFS Analysis

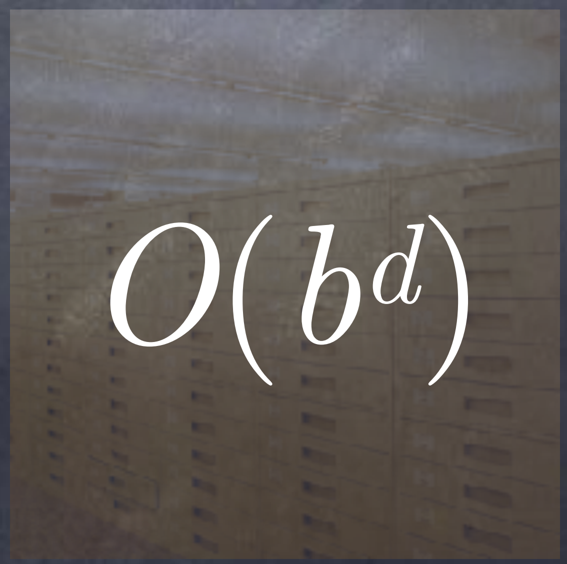
Completeness

Optimality



$O(b^d)$

Time Complexity



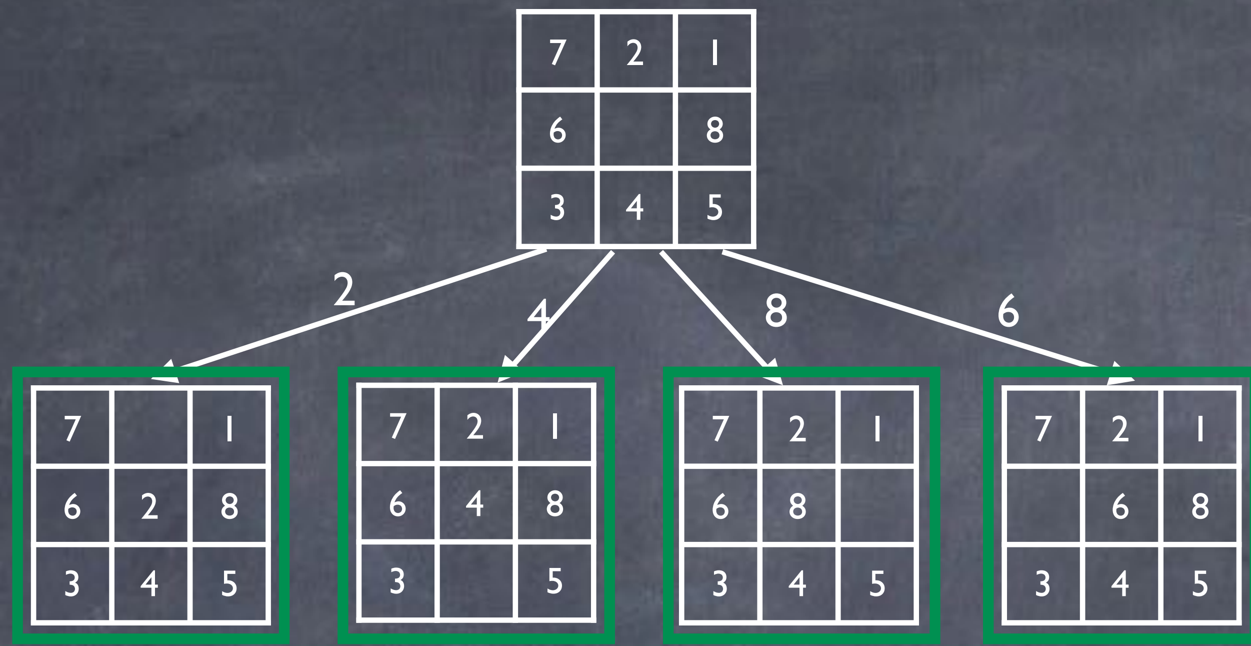
$O(b^d)$

Space Complexity

7	2	1
6		8
3	4	5

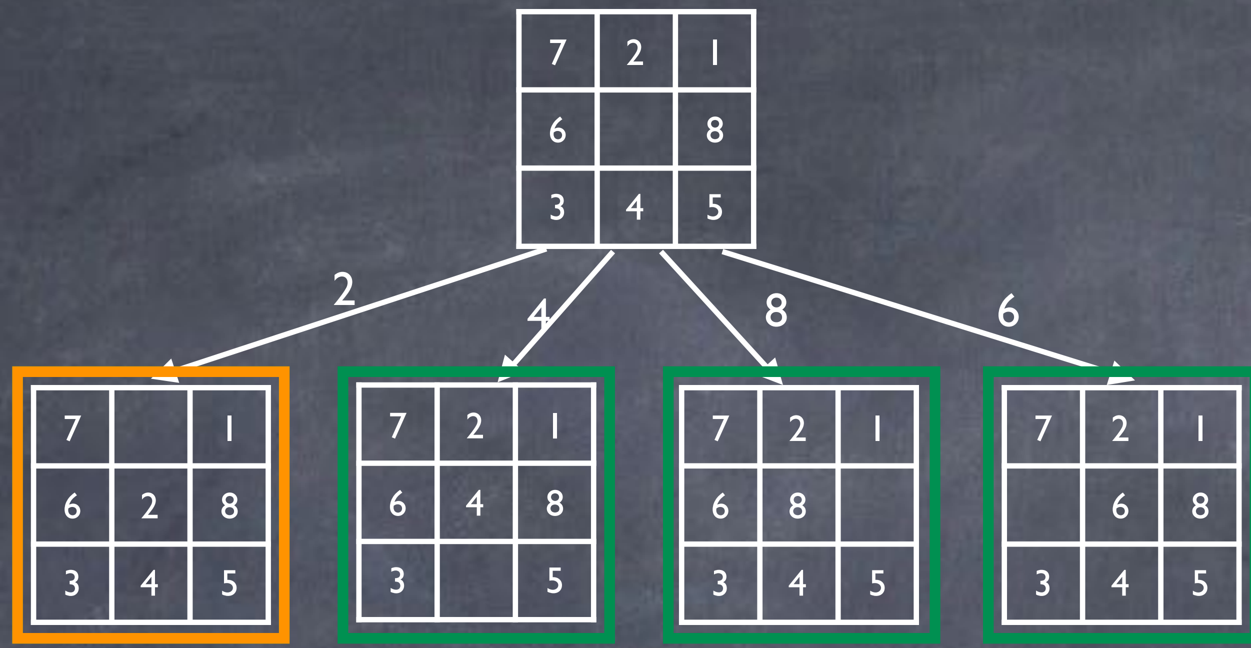
M^G:

1	2	3
4	5	6
7	8	



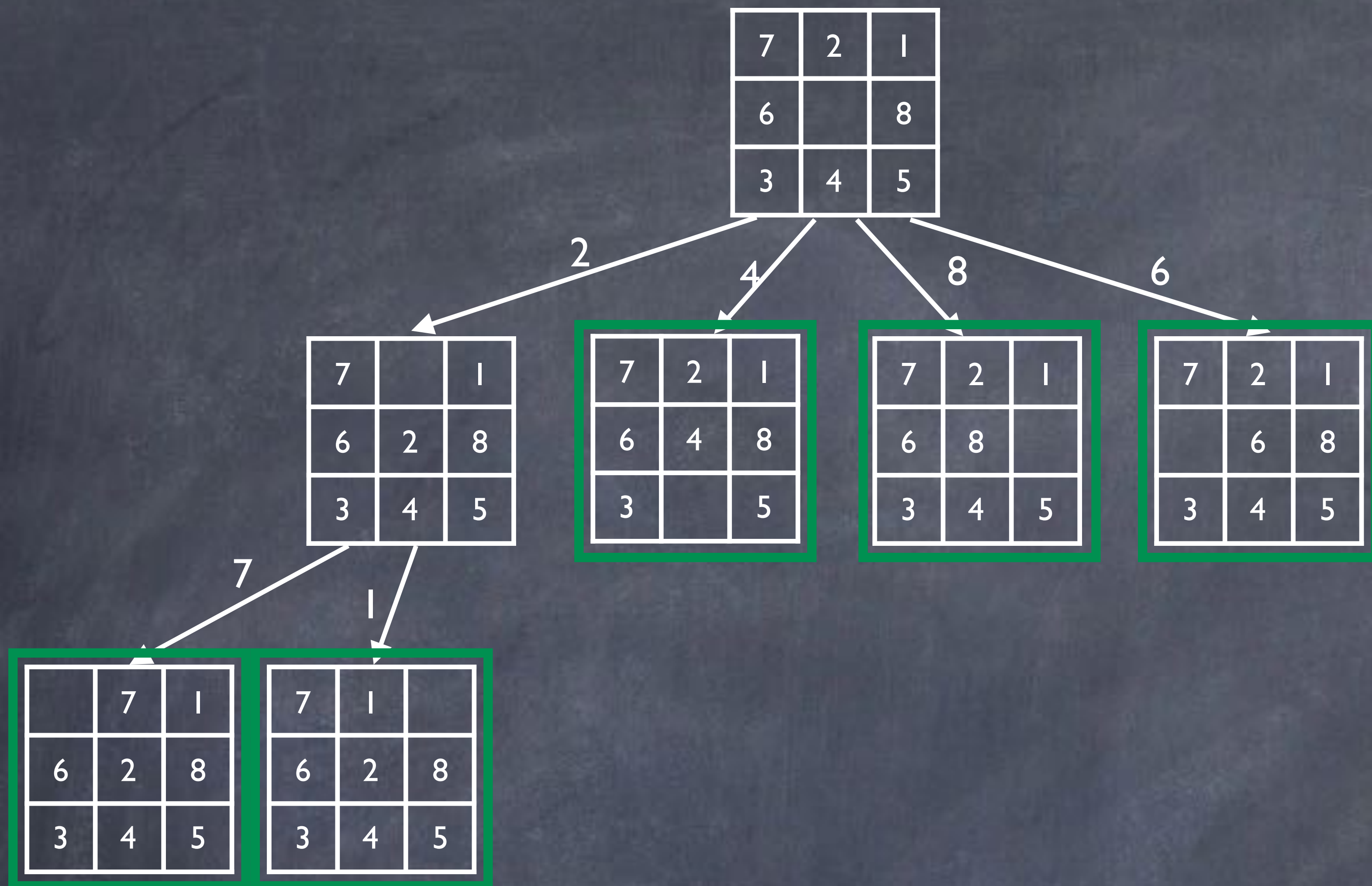
M^G:

0	1	2
3	4	5
6	7	8



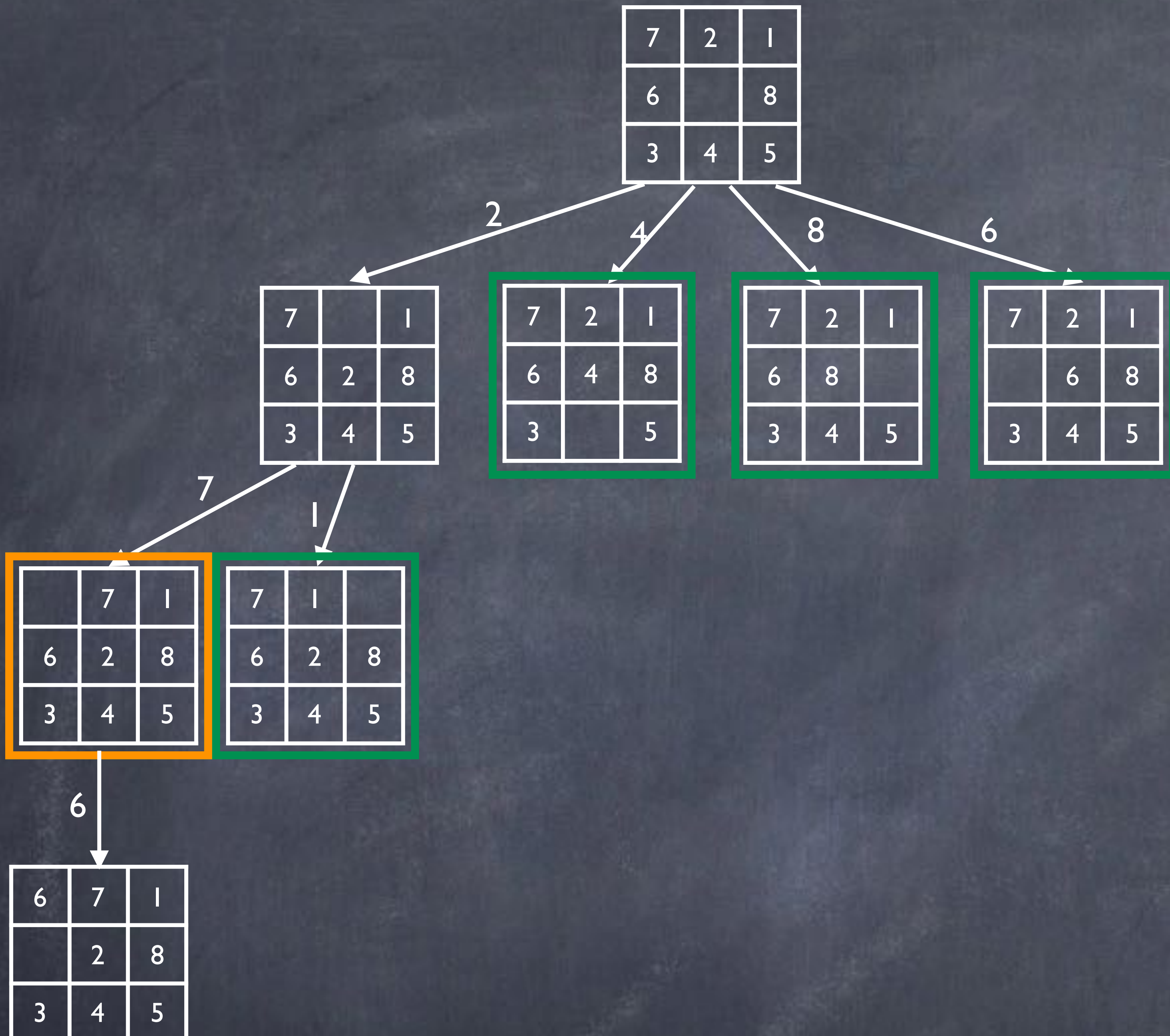
M^G:

0	1	2
3	4	5
6	7	8



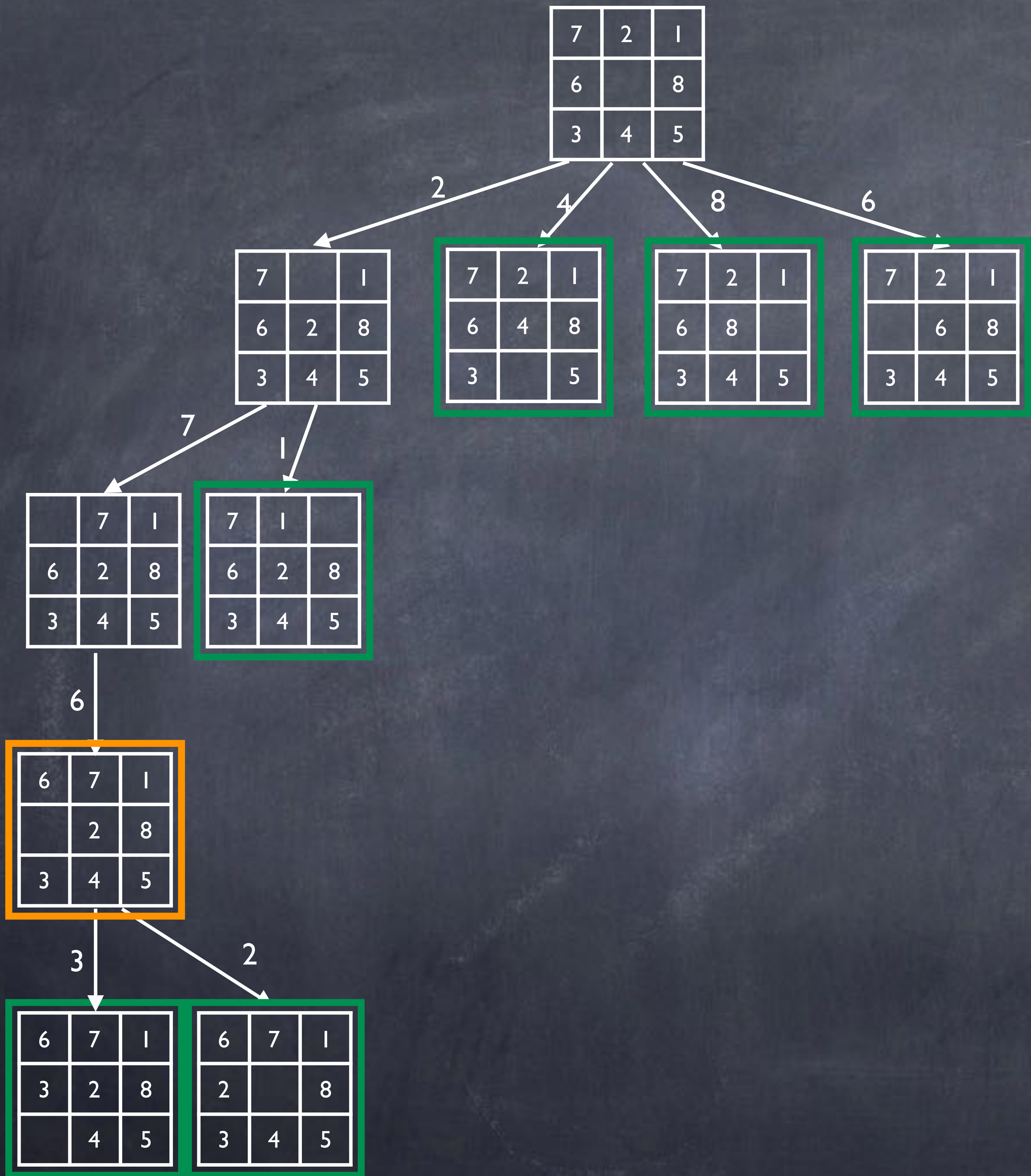
M^G:

0	1	2
3	4	5
6	7	8



M^G:

0	1	2
3	4	5
6	7	8



M^G:

0	1	2
3	4	5
6	7	8

Depth-First Search

- Expand a node's children before its siblings
- Expand the deepest unexpanded node
- Use a LIFO stack for the frontier

DFS Analysis

Graph-Search ✓

Tree-Search ✗

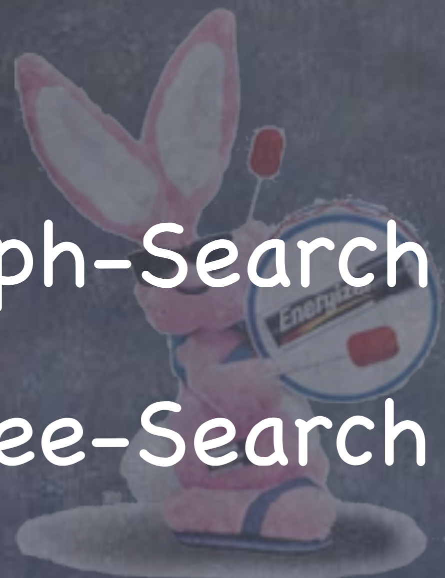
Completeness

Optimality

Time Complexity

Space Complexity

DFS Analysis



Graph-Search ✓

Tree-Search ✗

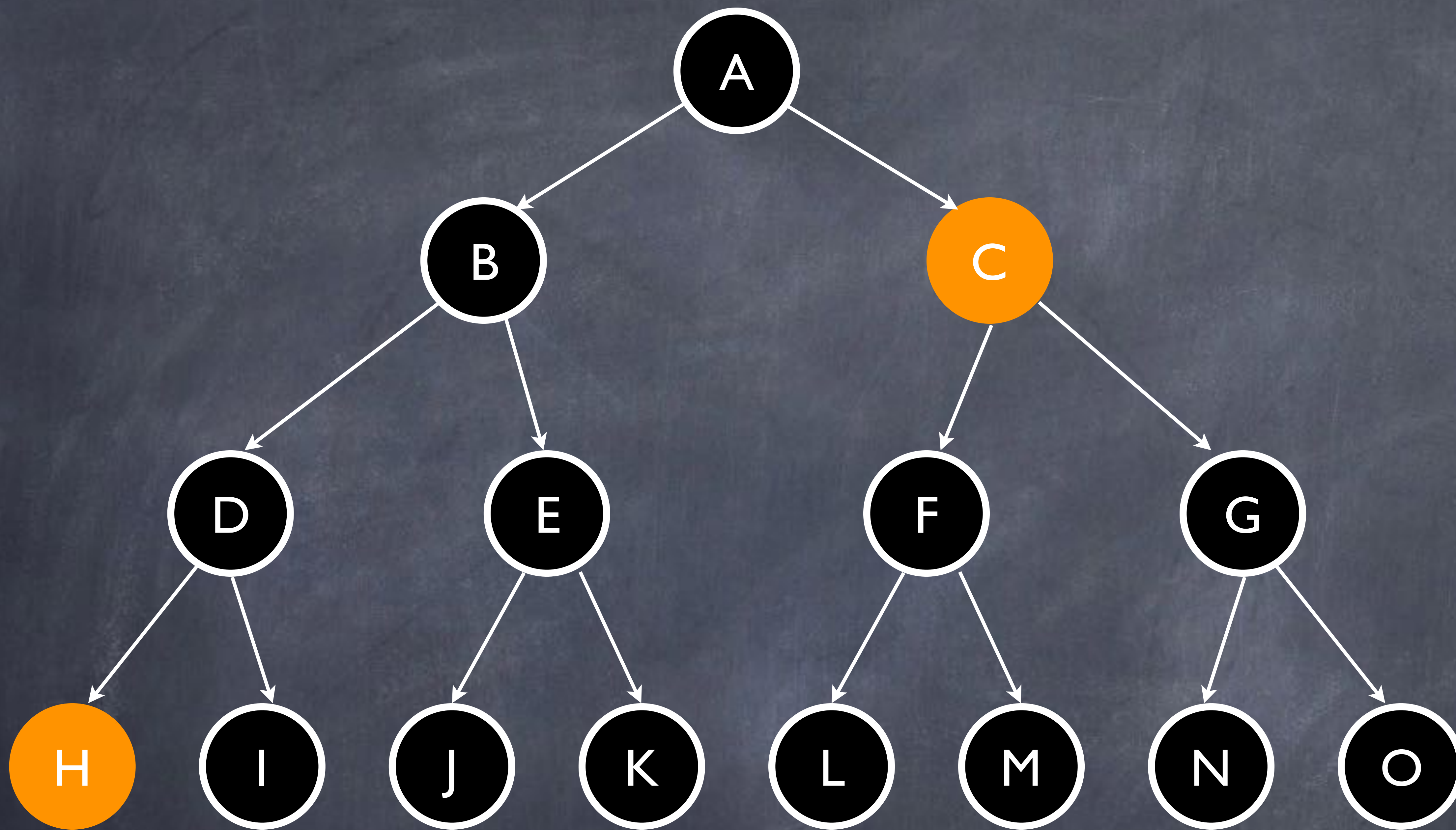
Completeness

✗

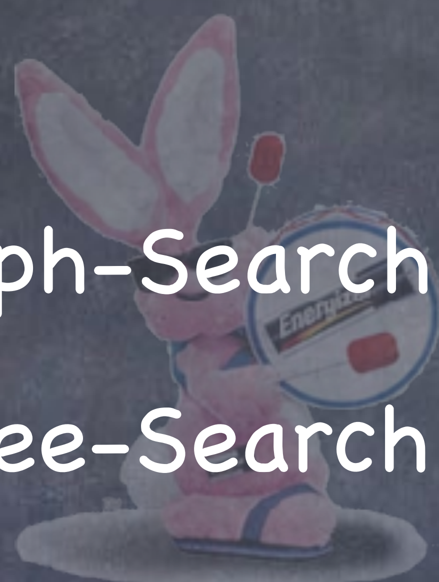
Optimality

Time Complexity

Space Complexity


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O

DFS Analysis



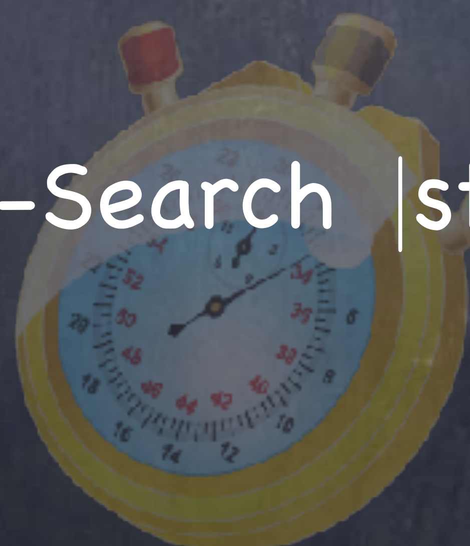
Graph-Search ✓

Tree-Search ✗

Completeness

✗

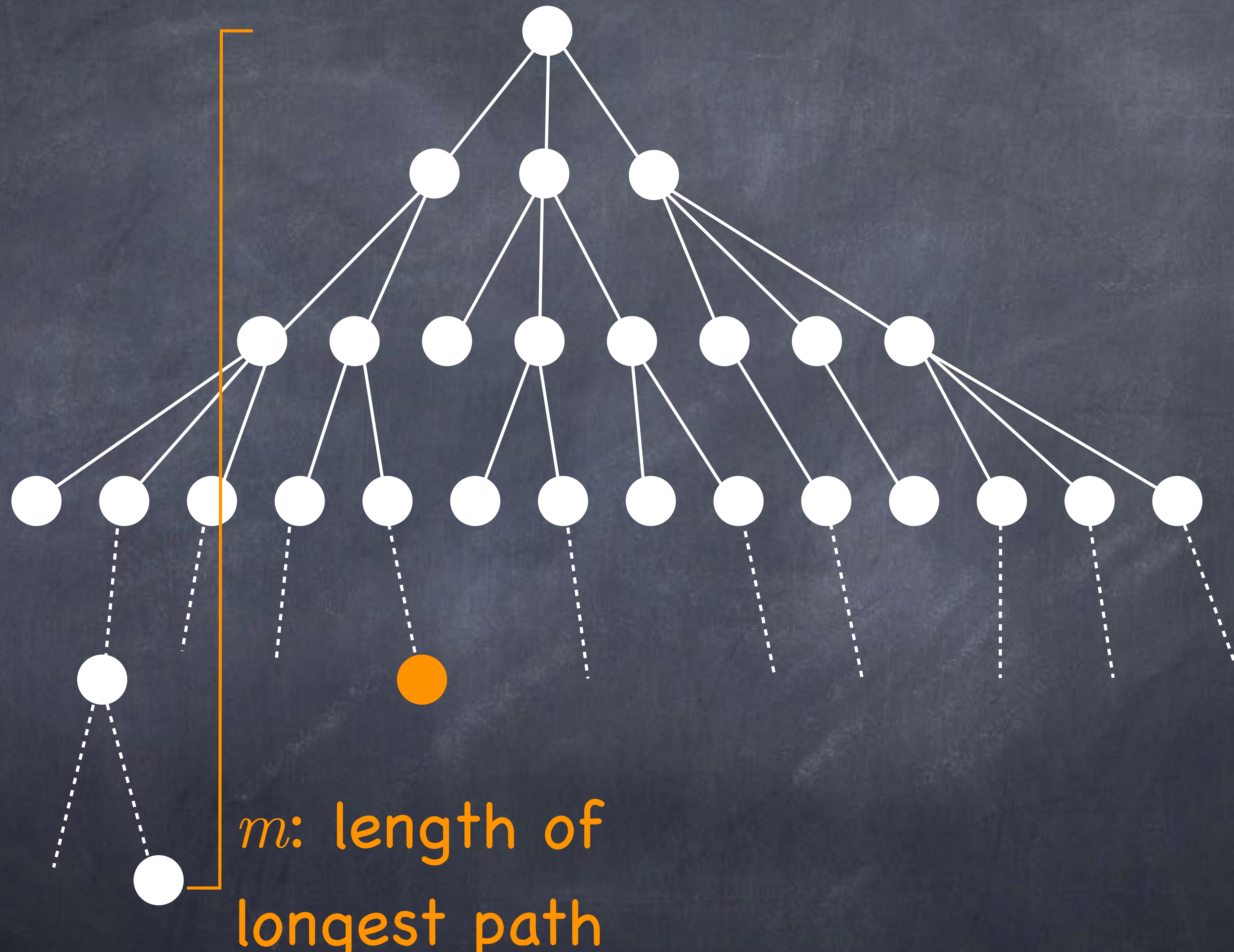
Optimality



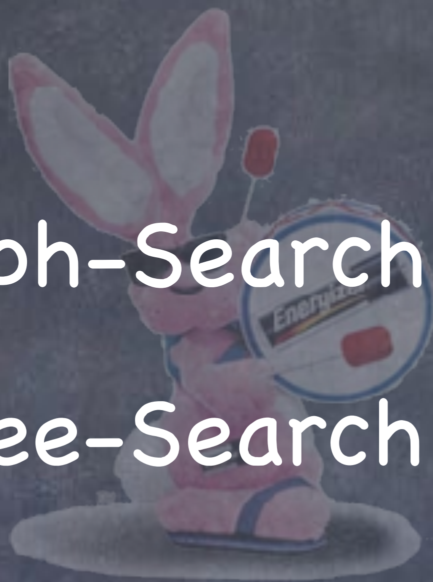
Graph-Search $|\text{state space}|$

Time Complexity

Space Complexity


m: length of longest path

DFS Analysis



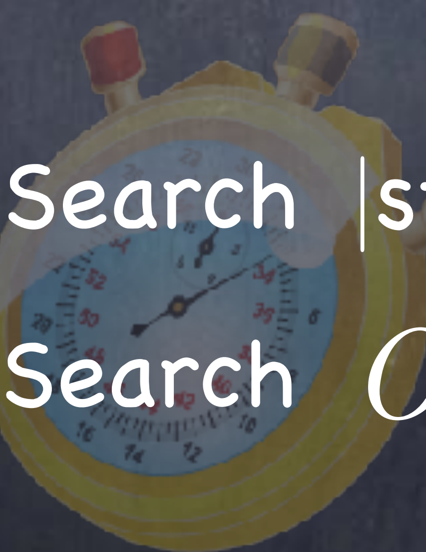
Graph-Search ✓

Tree-Search ✗

Completeness

✗

Optimality



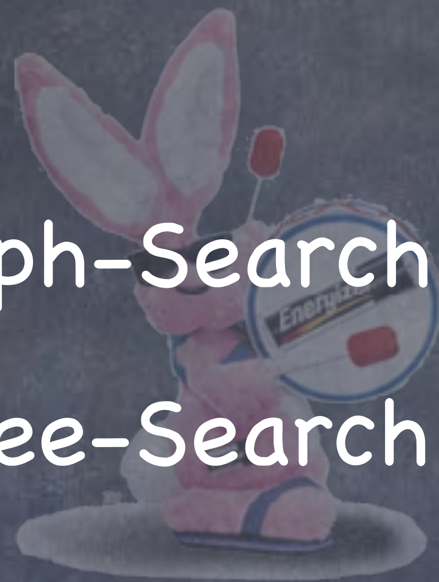
Graph-Search $|\text{state space}|$

Tree-Search $O(b^m)$

Time Complexity

Space Complexity

DFS Analysis

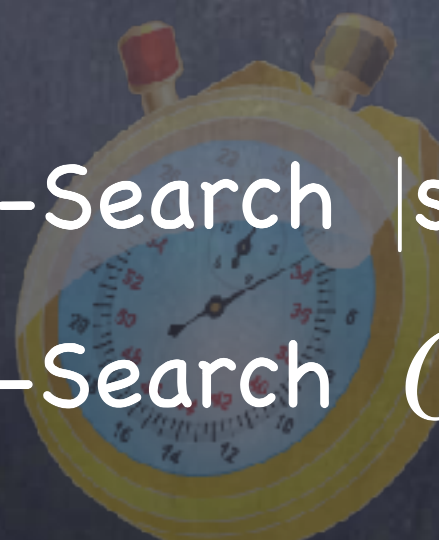


Graph-Search	✓
Tree-Search	✗

Completeness

✗

Optimality

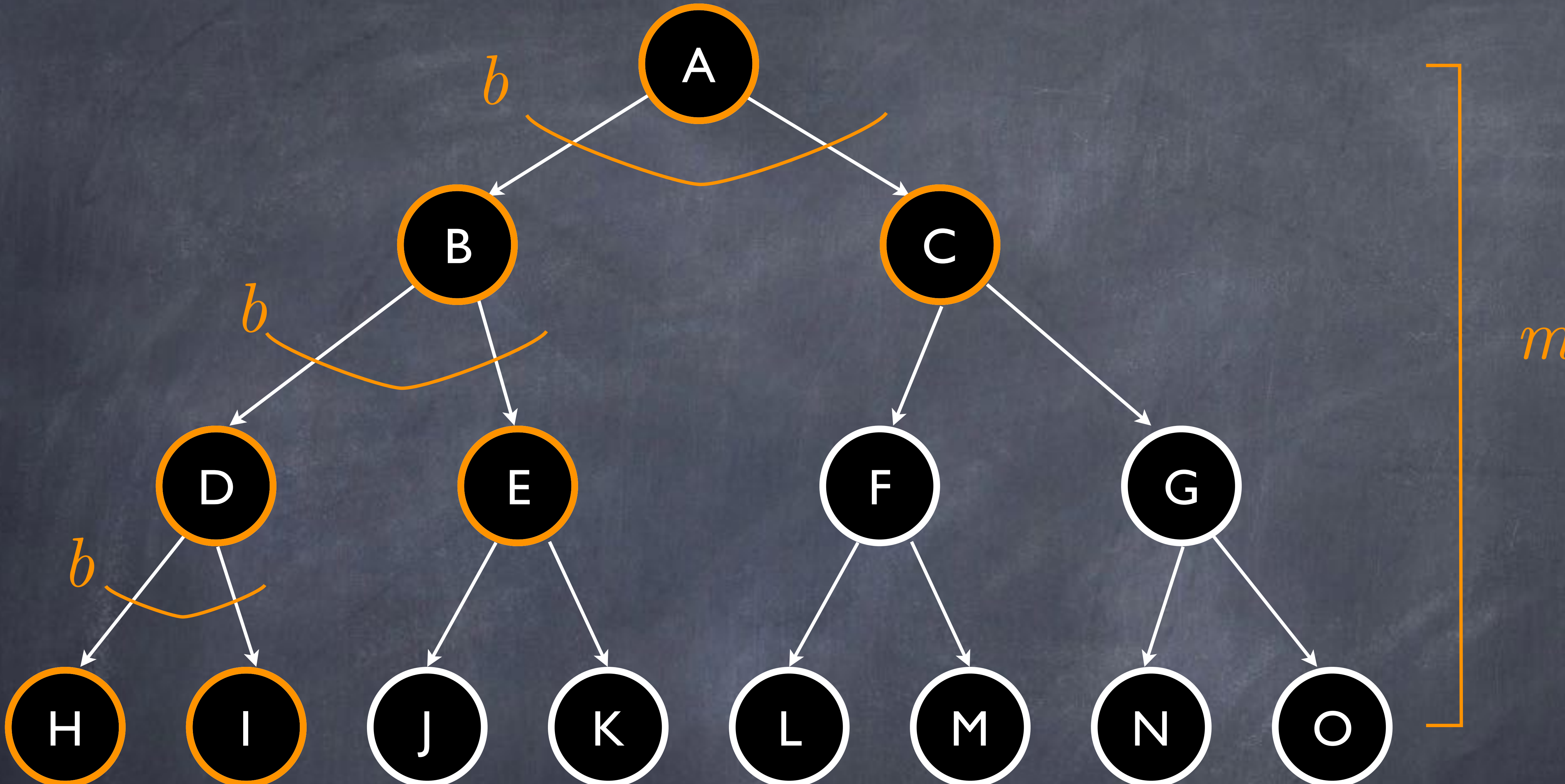


Graph-Search	$\lvert$state space$\rvert$
Tree-Search	$O(b^m)$

Time Complexity

Graph-Search	$O(b^m)$

Space Complexity


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
b
b
b
m

DFS Analysis

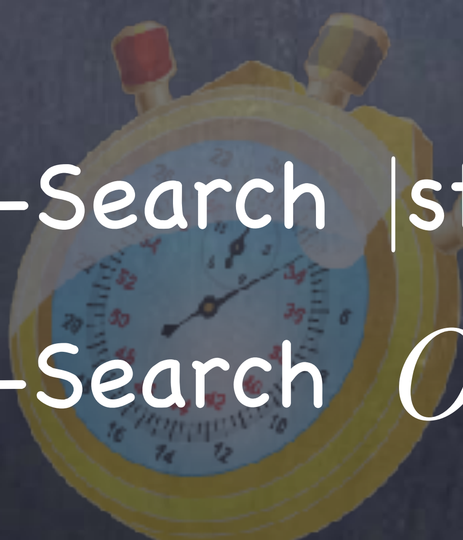
Graph-Search ✓

Tree-Search ✗

Completeness

✗

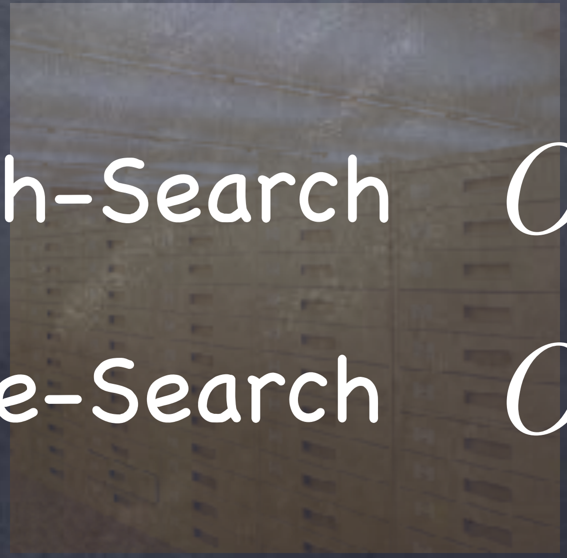
Optimality



Graph-Search $|\text{state space}|$

Tree-Search $O(b^m)$

Time Complexity



Graph-Search $O(b^m)$

Tree-Search $O(bm)$

Space Complexity

BFS vs. DFS

	BFS	DFS (graph)	DFS (tree)
Complete?	✓	✓	✗
Optimal?	✓*	✗	✗
Time	$O(b^d)$	$O(b^m)$	$O(b^m)$
Space	$O(b^d)$	$O(b^m)$	$O(bm)$

* If step costs are identical (see book)

AIMA

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

Depth-Limited Search

- DFS to some fixed depth limit k


d: depth of
shallowest goal
k

Depth-Limited Search

- DFS to some fixed depth limit k
- May not reach solution at depth d
  - Incomplete
- Good if you know or have a bound on d


d: depth of
shallowest goal
k

Iterative Deepening

For k = 1, 2, 3, ...

Do DFS to depth k

Until a goal is found.

Depth	Nodes expanded
1	b
2	$b + b^2$
3	$b + b^2 + b^3$
d	$b + b^2 + b^3 + \cdots b^d$

Depth	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 (graph)	DFS (tree)	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."

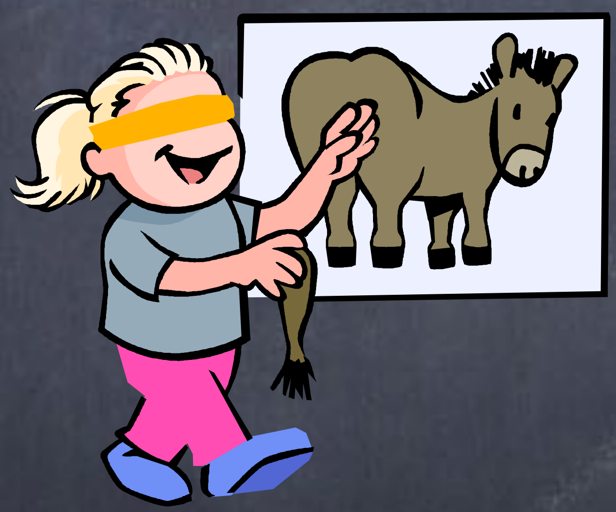
Uninformed Strategies

	BFS	DFS (graph)	DFS (tree)	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)

Search Strategies

Uninformed



No additional information about states

Informed (Heuristic)

Can identify "promising" states

Heuristic Strategies

Have “extra information” about states

heuristic |hyoōˈristik|

adjective

enabling a person to discover or learn something for themselves : *a "hands-on" or interactive heuristic approach to learning.*

- 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.'***

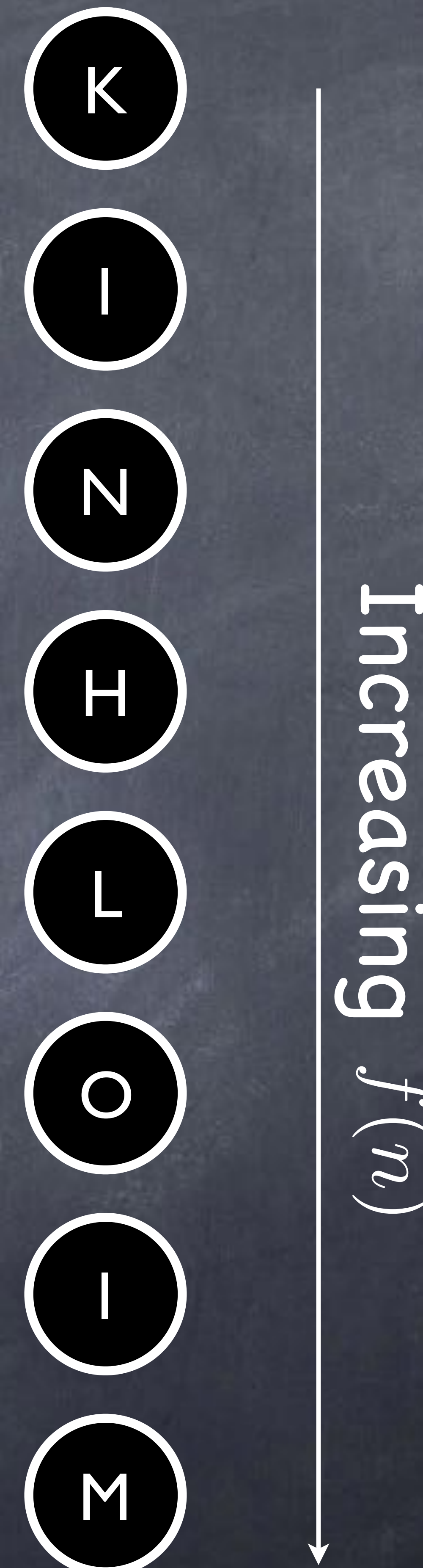
```
Solution graphSearch(Problem p) {
  Set<Node> frontier = new Set<Node>(p.getInitialState());
  Set<Node> explored = new Set<Node>();
  while (true) {
    if (frontier.isEmpty()) {
      return false;
    }
    Node node = frontier.selectOne();
    if (p.isGoalState(node.getState())) {
      return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

Evaluation Function

$$f(n)$$

Cost of cheapest path
from n to a goal node

Frontier:
(Heap)



Evaluation function

$$f(n)$$

Cost of cheapest path
from n to a goal node

Evaluation Function

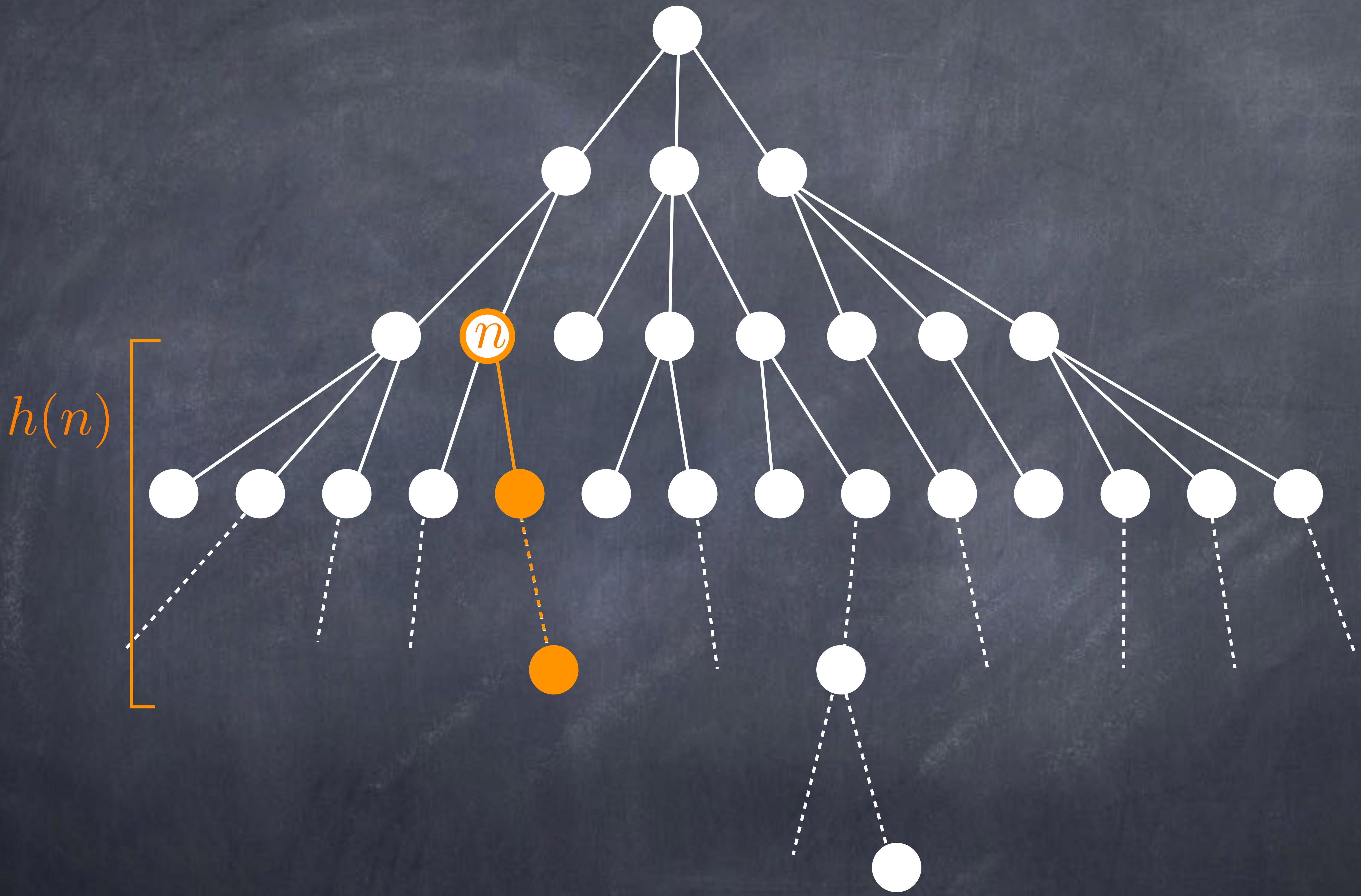
$$\cancel{f(n)}$$

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

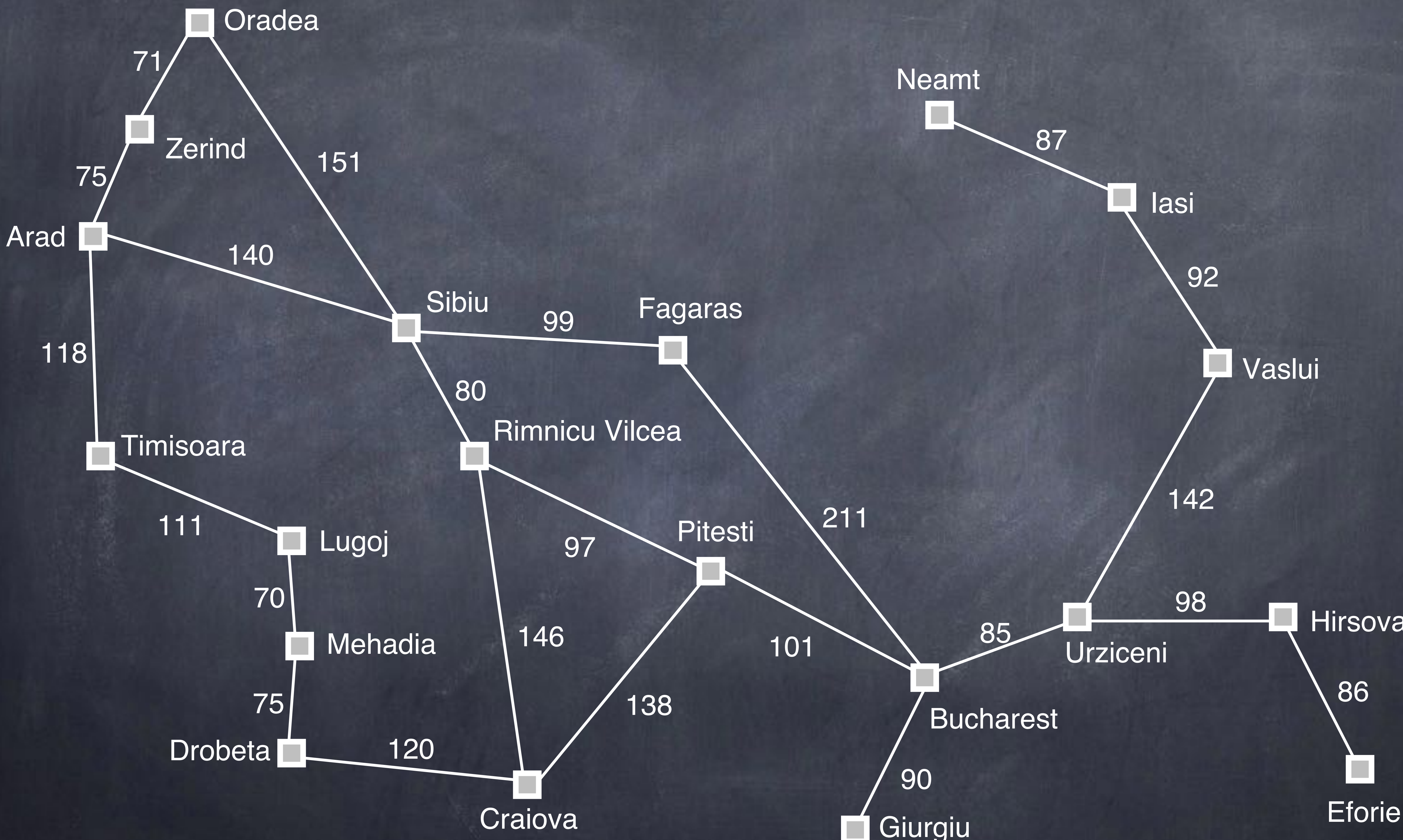

$h(n)$
n

Frontier:
(Heap)

Heuristic function

$$h(n)$$

Estimated cost of cheapest path from n to a goal node



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

$$h_{SLD}$$

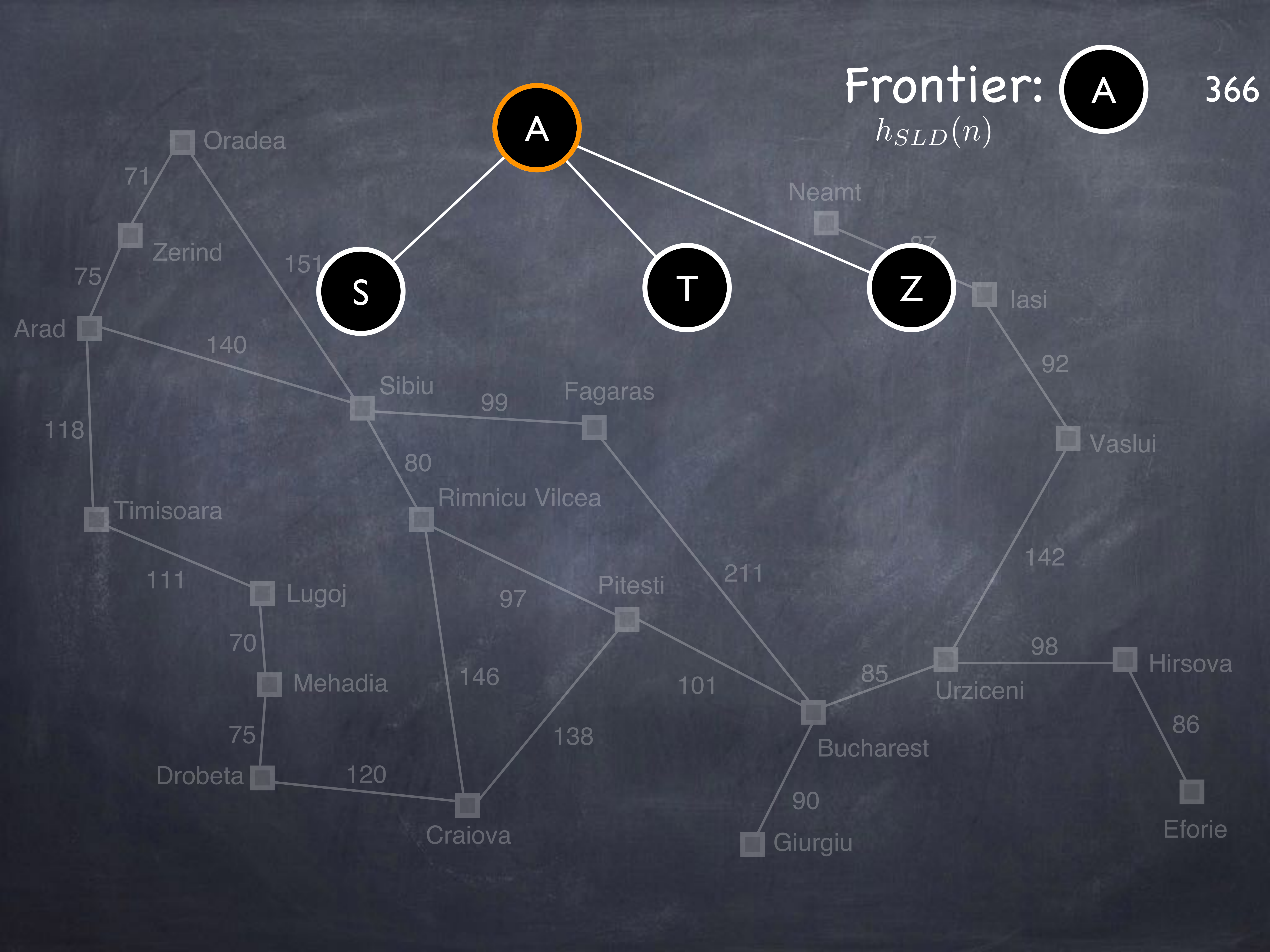
(straight-line distance to Bucharest)

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

$h_{SLD}(n)$

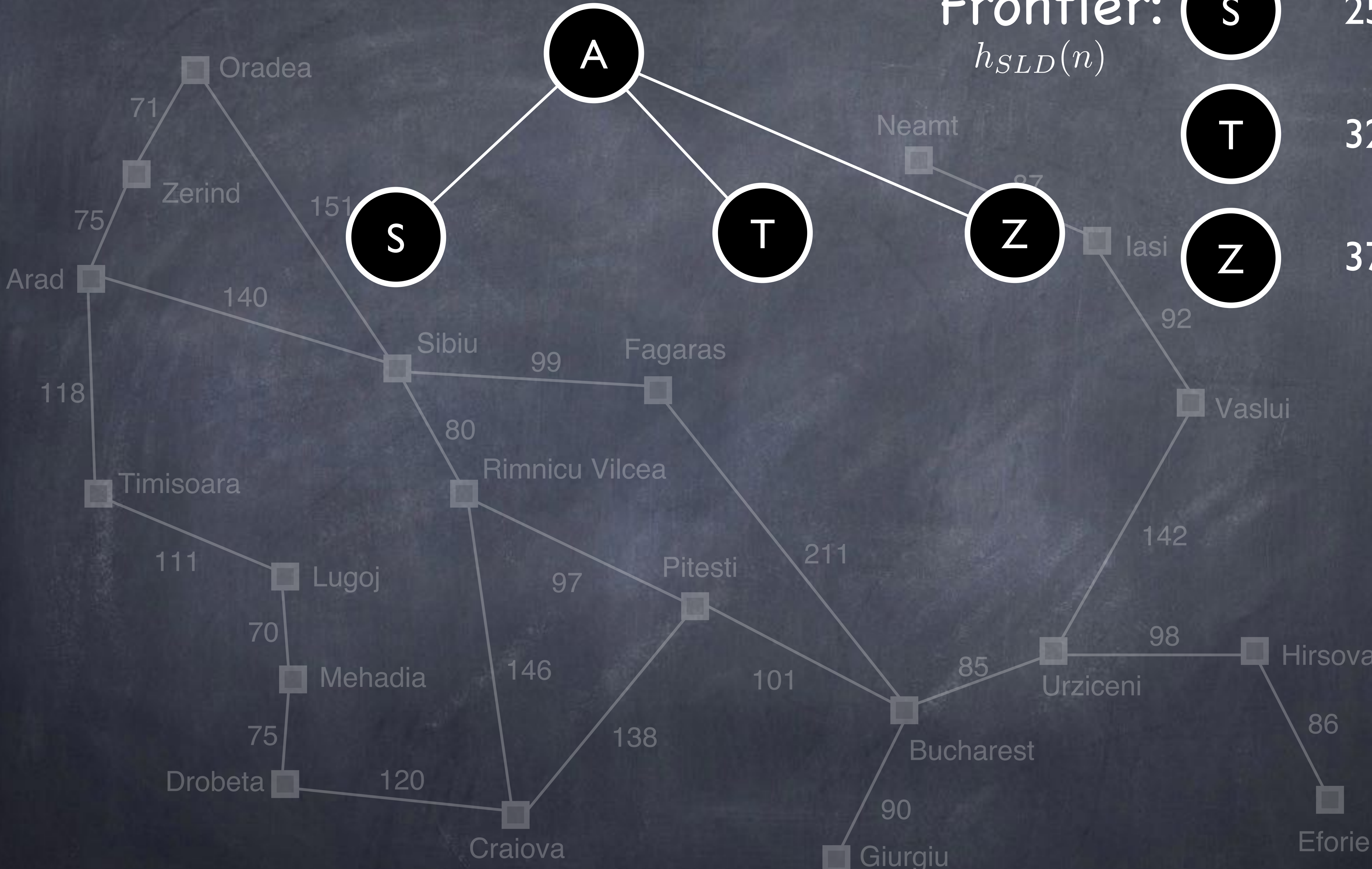


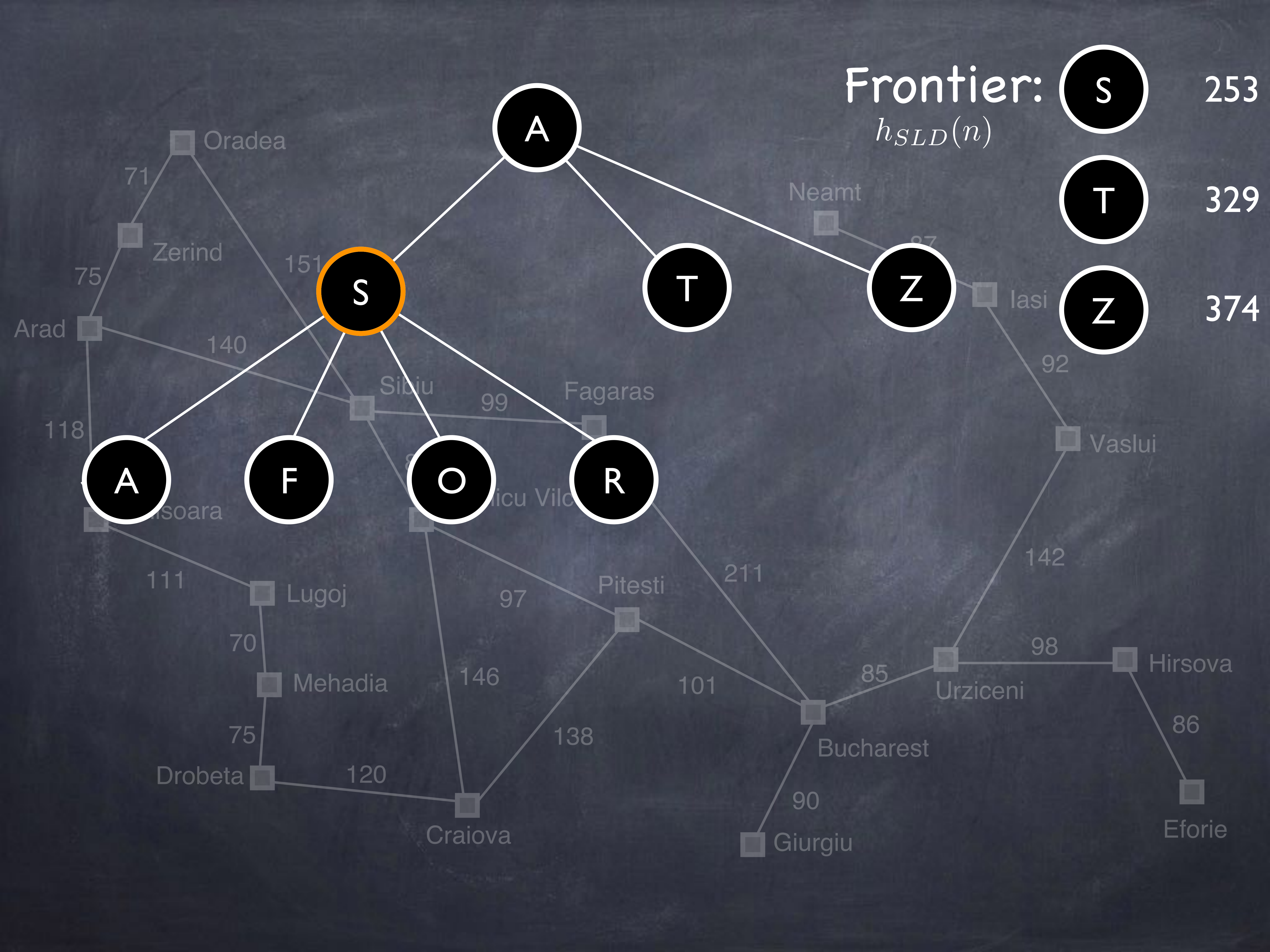
Frontier:
A
366
$h_{SLD}(n)$
A
S
T
Z
Oradea
Zerind
Arad
Sibiu
Fagaras
Neamt
Iasi
Vaslui
Timisoara
Lugoj
Mehadia
Drobeta
Craiova
Rimnicu Vilcea
Pitesti
Bucharest
Giurgiu
Urziceni
Hirsova
Eforie
71
75
151
140
118
99
80
211
92
142
111
70
75
120
146
97
138
101
85
98
86
90

Frontier:

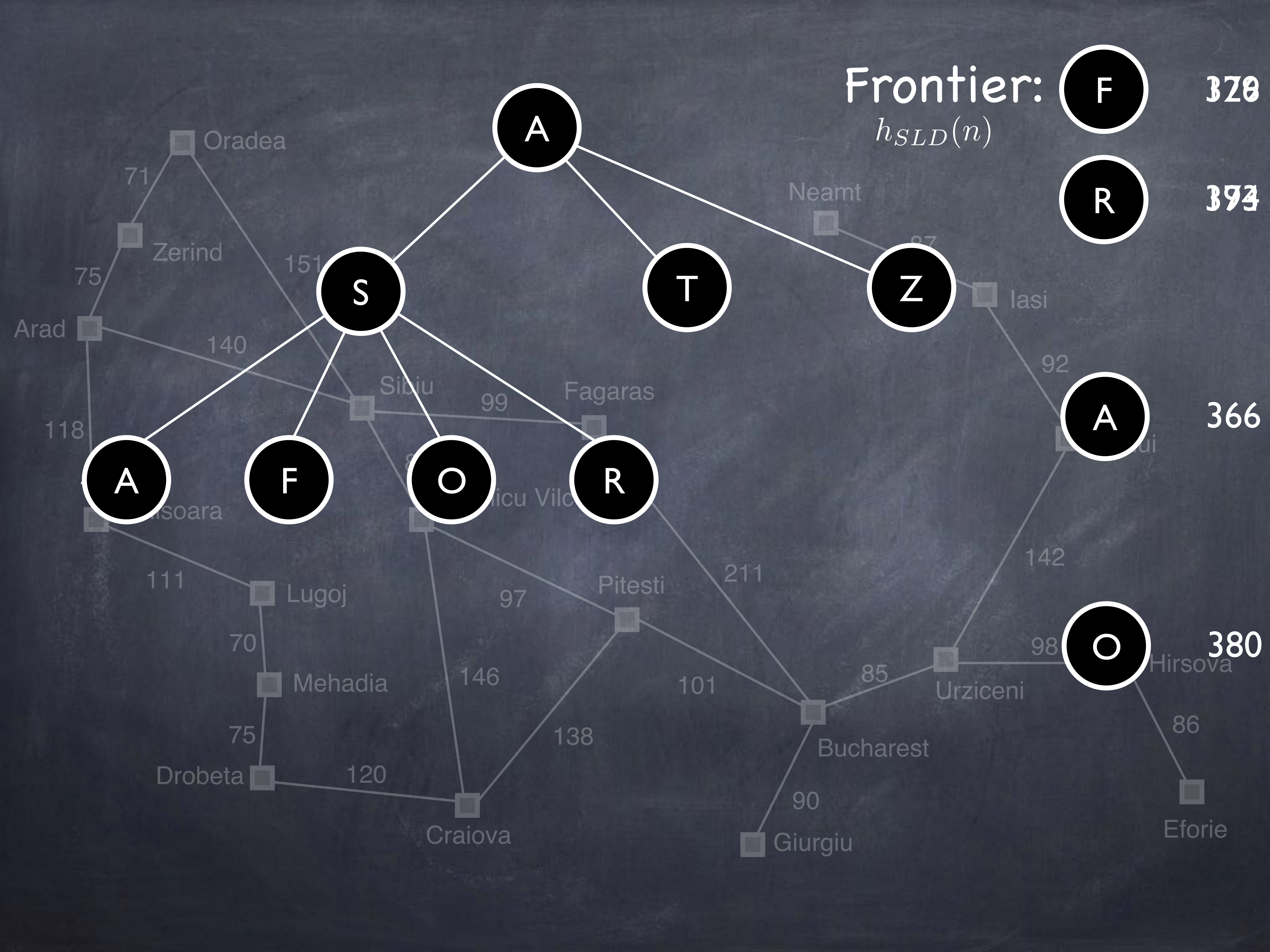
$h_{SLD}(n)$

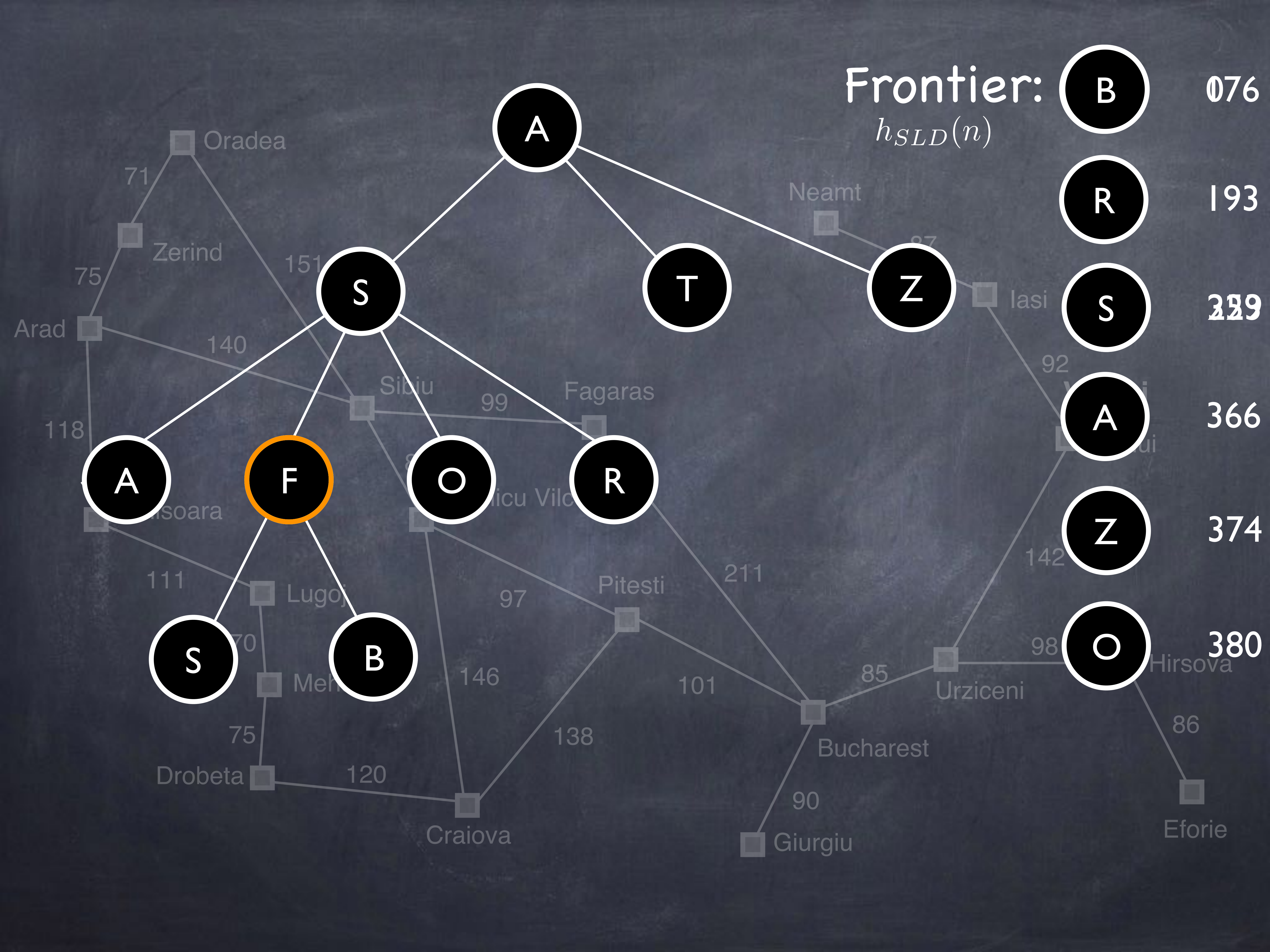
S	253
T	329
Z	374

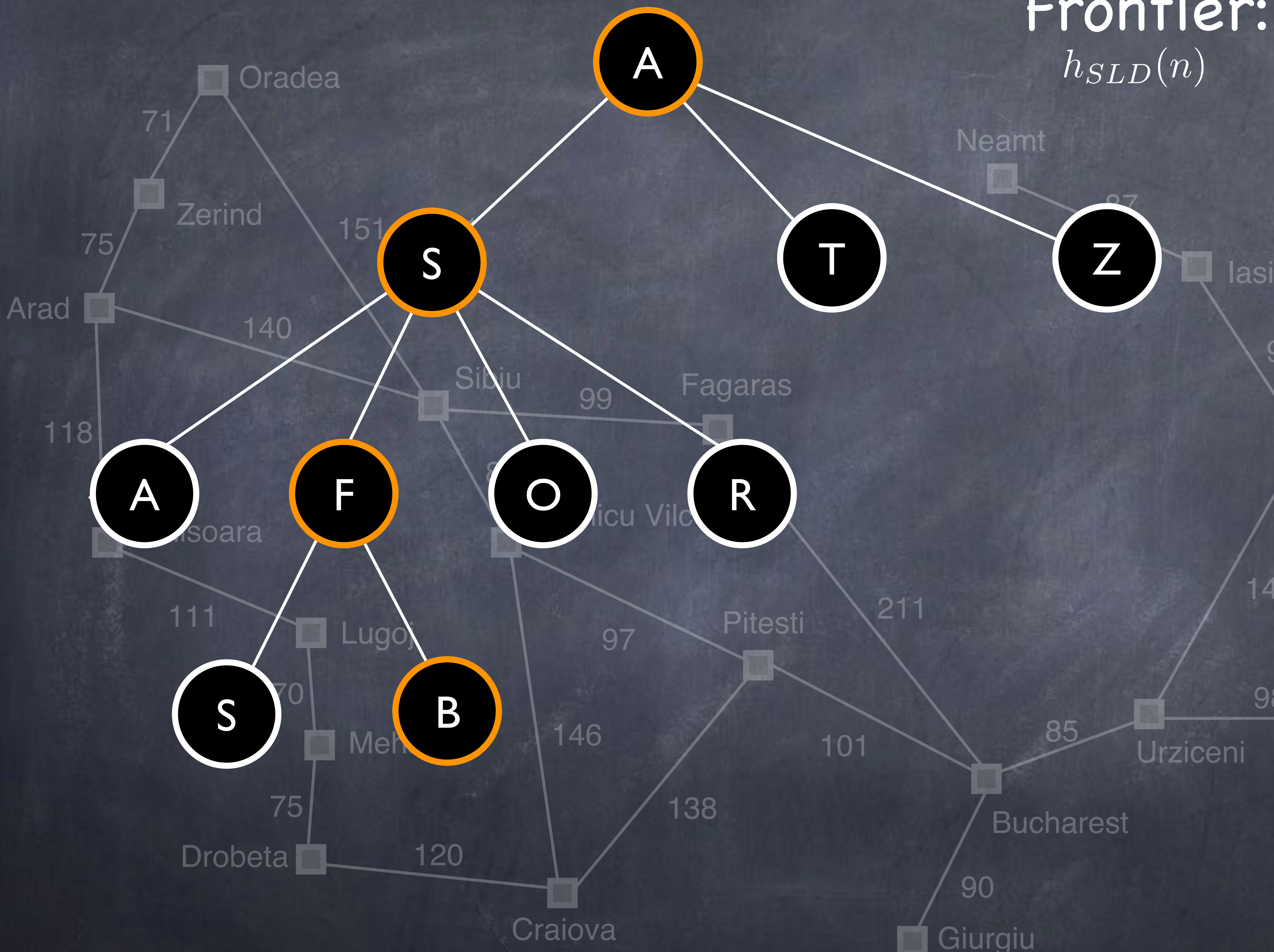




Frontier:
$h_{SLD}(n)$
S
253
T
329
Z
374
A
S
T
Z
A
F
O
R
Oradea
Zerind
Arad
Sibiu
Fagaras
Neamt
Iasi
Vaslui
Lugoj
Mehadia
Drobeta
Craiova
Pitesti
Bucharest
Giurgiu
Urziceni
Hirsova
Eforie
71
75
151
140
118
99
92
111
70
75
120
146
97
138
101
211
85
98
142
86
90


Frontier:
$h_{SLD}(n)$
F
R
A
366
O
380
A
S
T
Z
A
F
O
R
Oradea
Zerind
Arad
Sibiu
Fagaras
Neamt
Iasi
Lugoj
Mehadia
Drobeta
Craiova
Pitesti
Bucharest
Giurgiu
Urziceni
Hirsova
Eforie
71
75
151
140
118
99
87
92
111
70
75
120
146
97
138
101
211
85
90
98
142
86

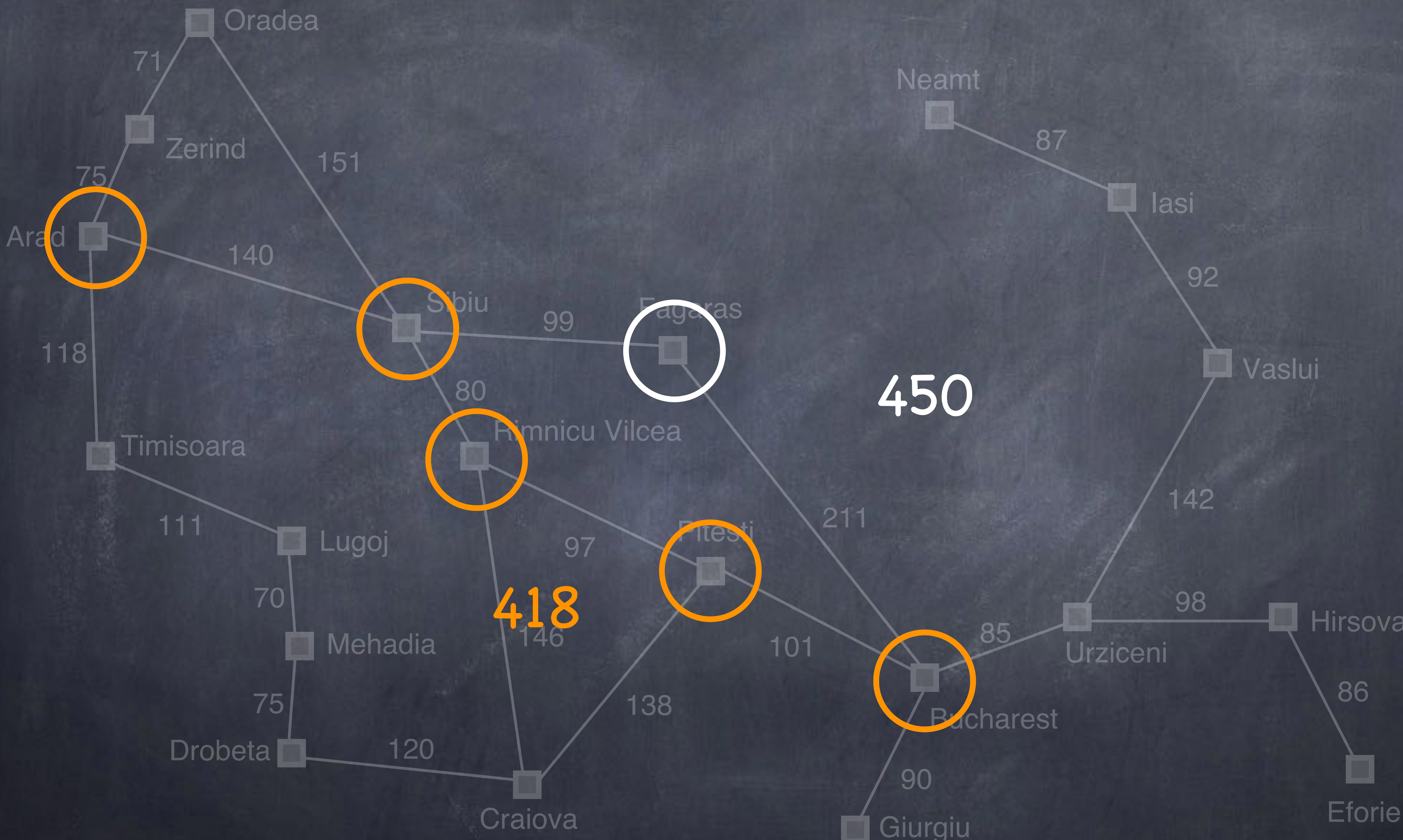

Frontier:
$h_{SLD}(n)$
B
R 193
S
A 366
Z 374
O 380
A
S
T
Z
A
F
O
R
S
B
Oradea
71
Zerind
75
151
Arad
140
118
Sibiu
99
Fagaras
Neamt
Iasi
92
111
Lugoj
97
Pitesti
211
146
Mehadia
75
Drobeta
120
Craiova
138
101
Bucharest
85
Urziceni
142
98
Hirsova
86
Eforie
90
Giurgiu



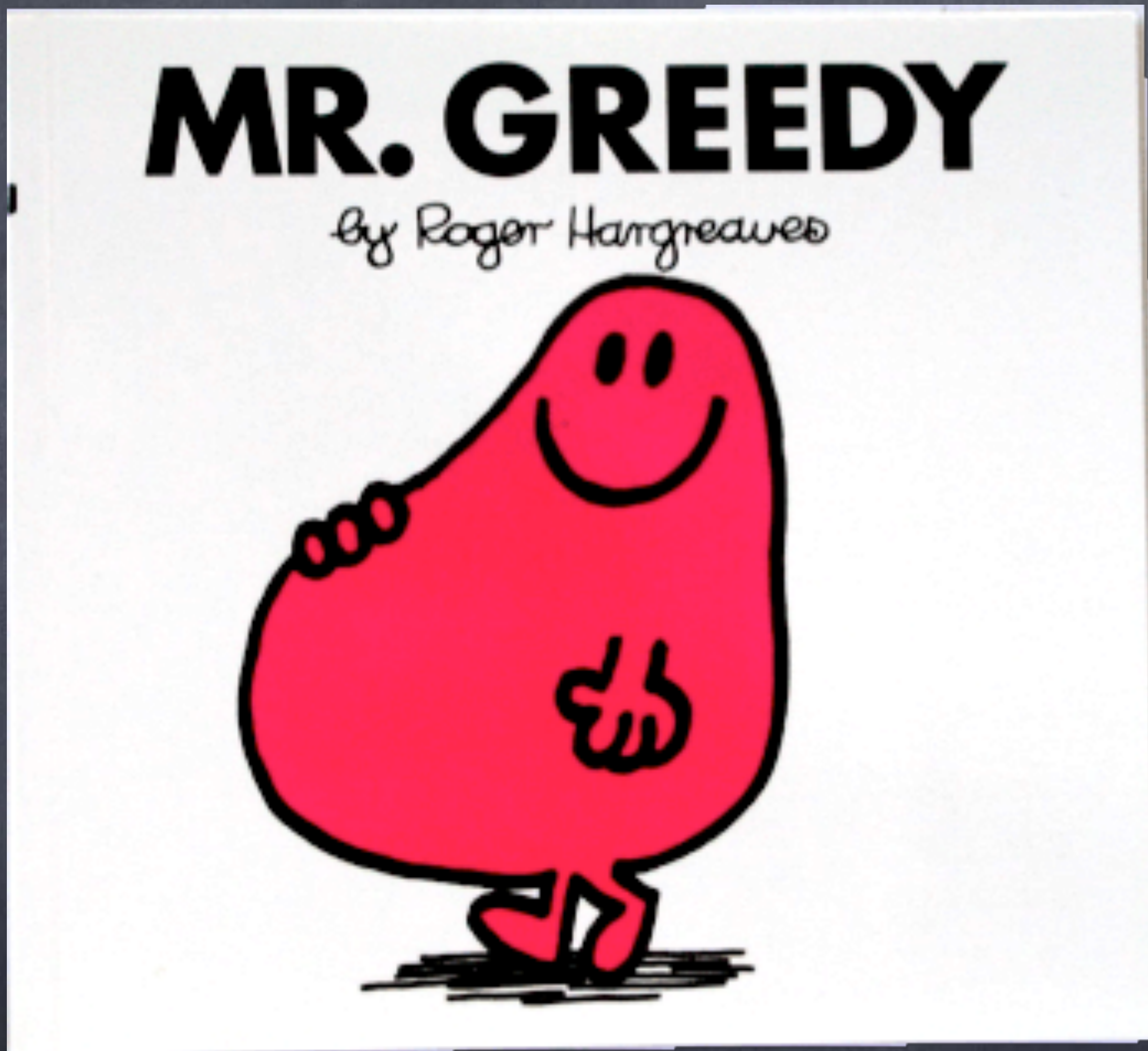
Frontier:

$h_{SLD}(n)$

B	0
R	193
S	253
T	329
A	366
Z	374
O	380


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

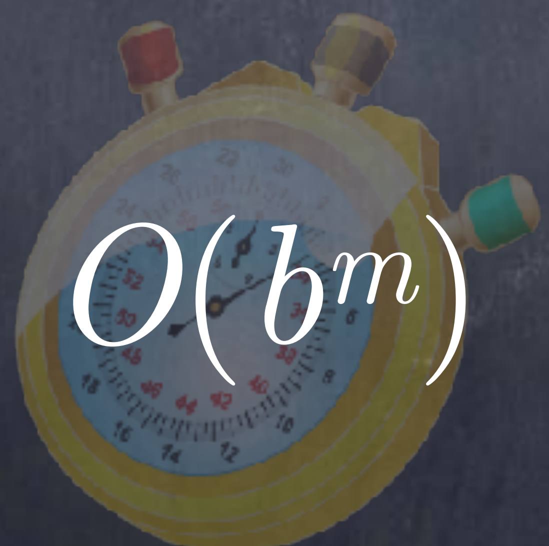

Oradea
71
Zerind
151
75
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
98
Hirsova
85
Urziceni
Bucharest
86
90
Giurgiu
Eforie


MR. GREEDY
by Roger Hargreaves

Greedy Best-First Search

Completeness

Optimality



$O(b^m)$

Time Complexity

$O(b^m)$

Space Complexity

Evaluation function

$$f(n)$$

Evaluation function

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

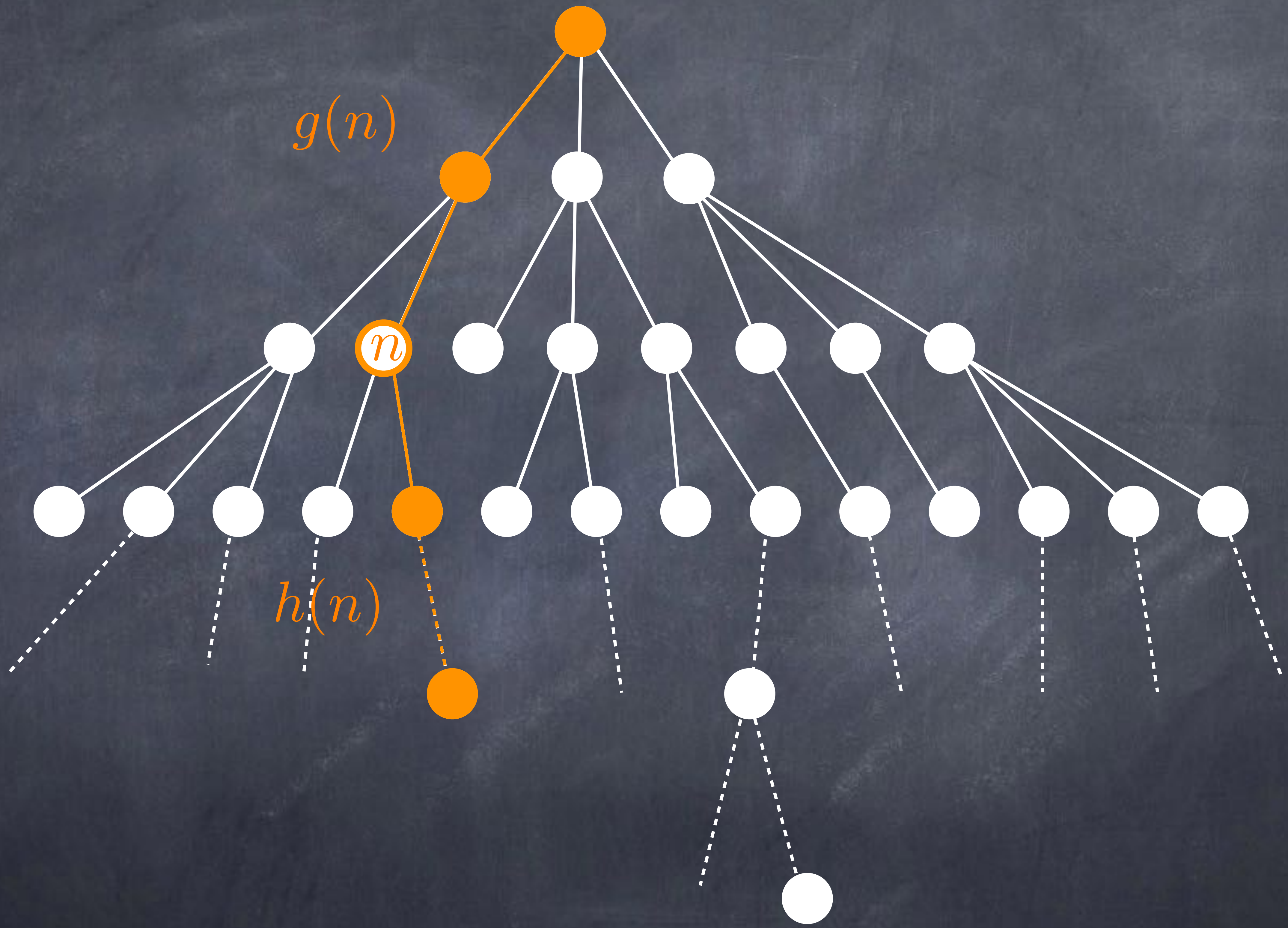
Estimated cost of cheapest path from n to a goal node

Evaluation function

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

True cost of path from start node to node n

Estimated cost of cheapest path from n to a goal node



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

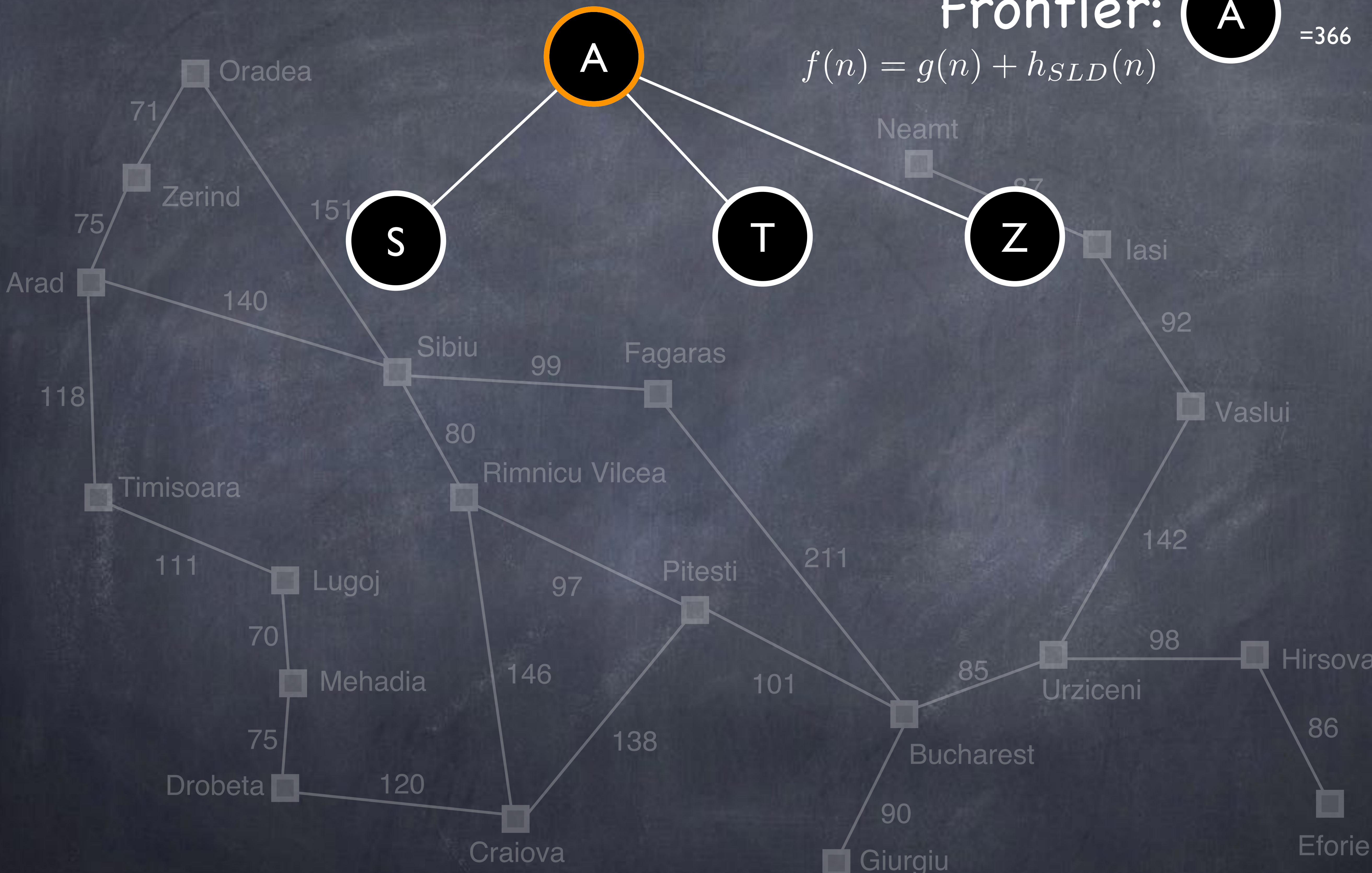
Evaluation function

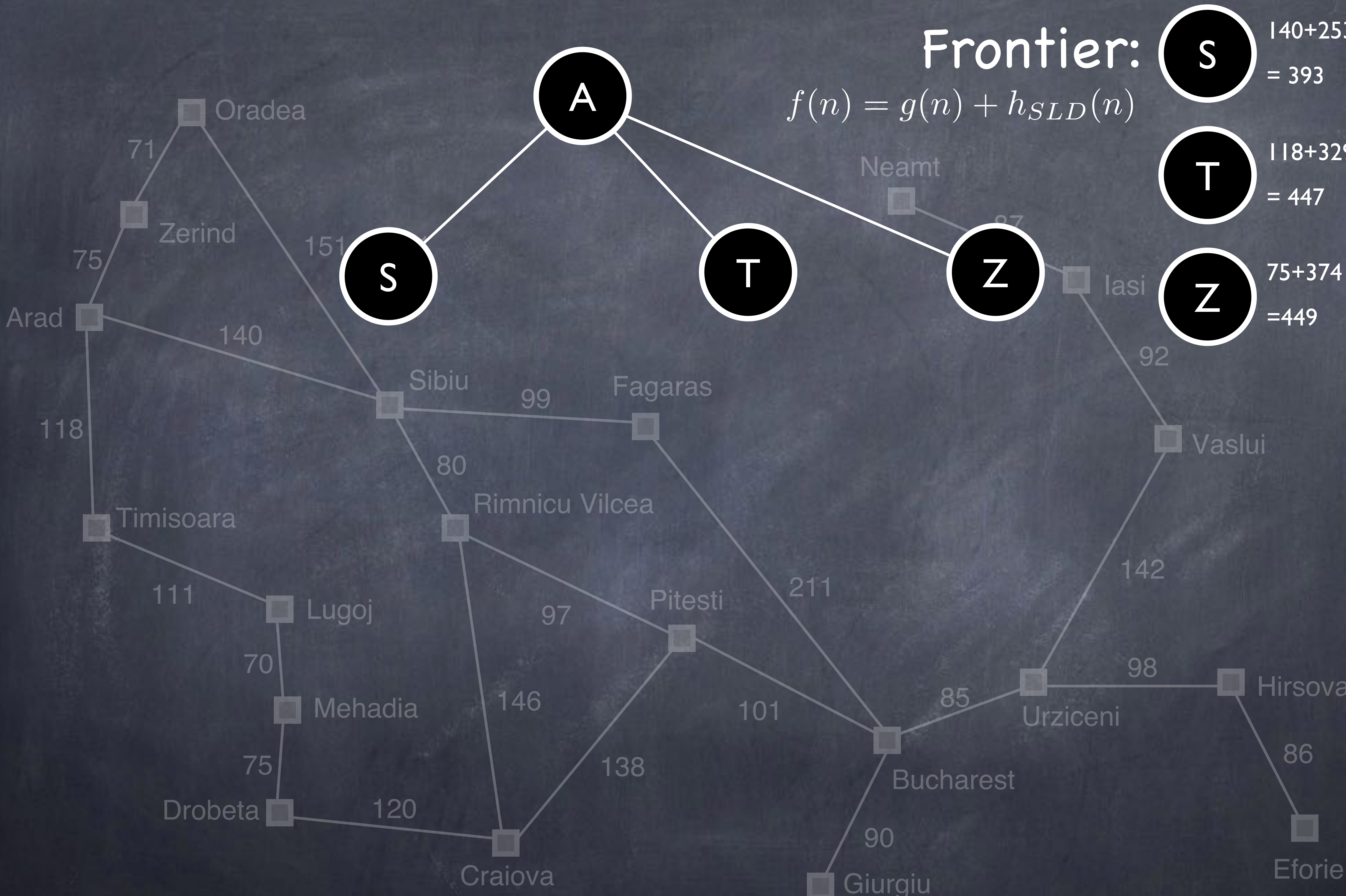
$$f(n) = g(n) + h(n)$$

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

Frontier: A 0+366 =366

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




Frontier:
$f(n) = g(n) + h_{SLD}(n)$
A
S
T
Z
S
140+253
= 393
T
118+329
= 447
Z
75+374
=449
Oradea
71
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
Neamt
Iasi
92
Vaslui
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
142
70
Mehadia
146
101
85
Urziceni
98
Hirsova
75
Drobeta
120
138
Bucharest
86
90
Craiova
Giurgiu
Eforie

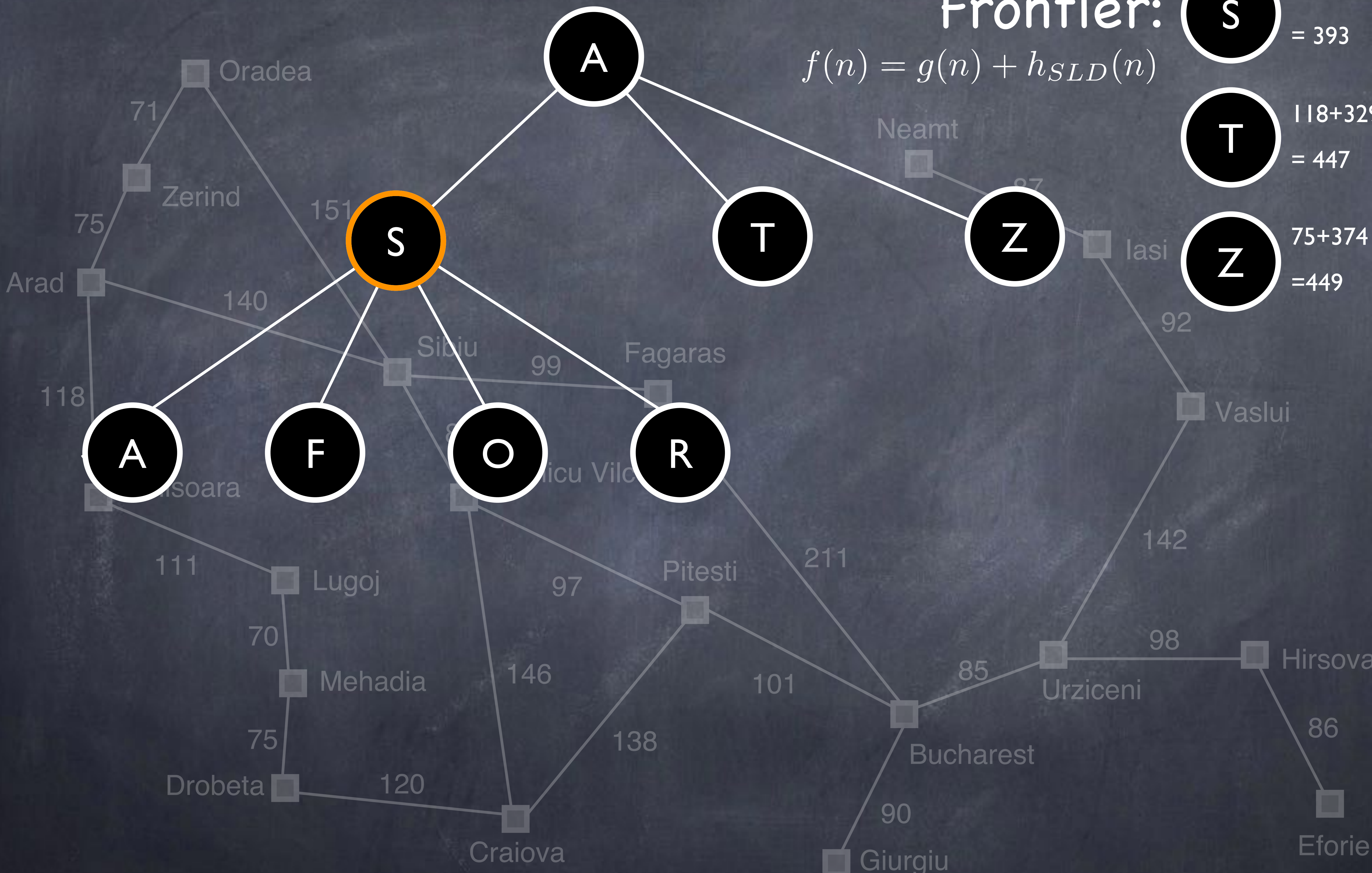
Frontier:

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

S 140+253 = 393

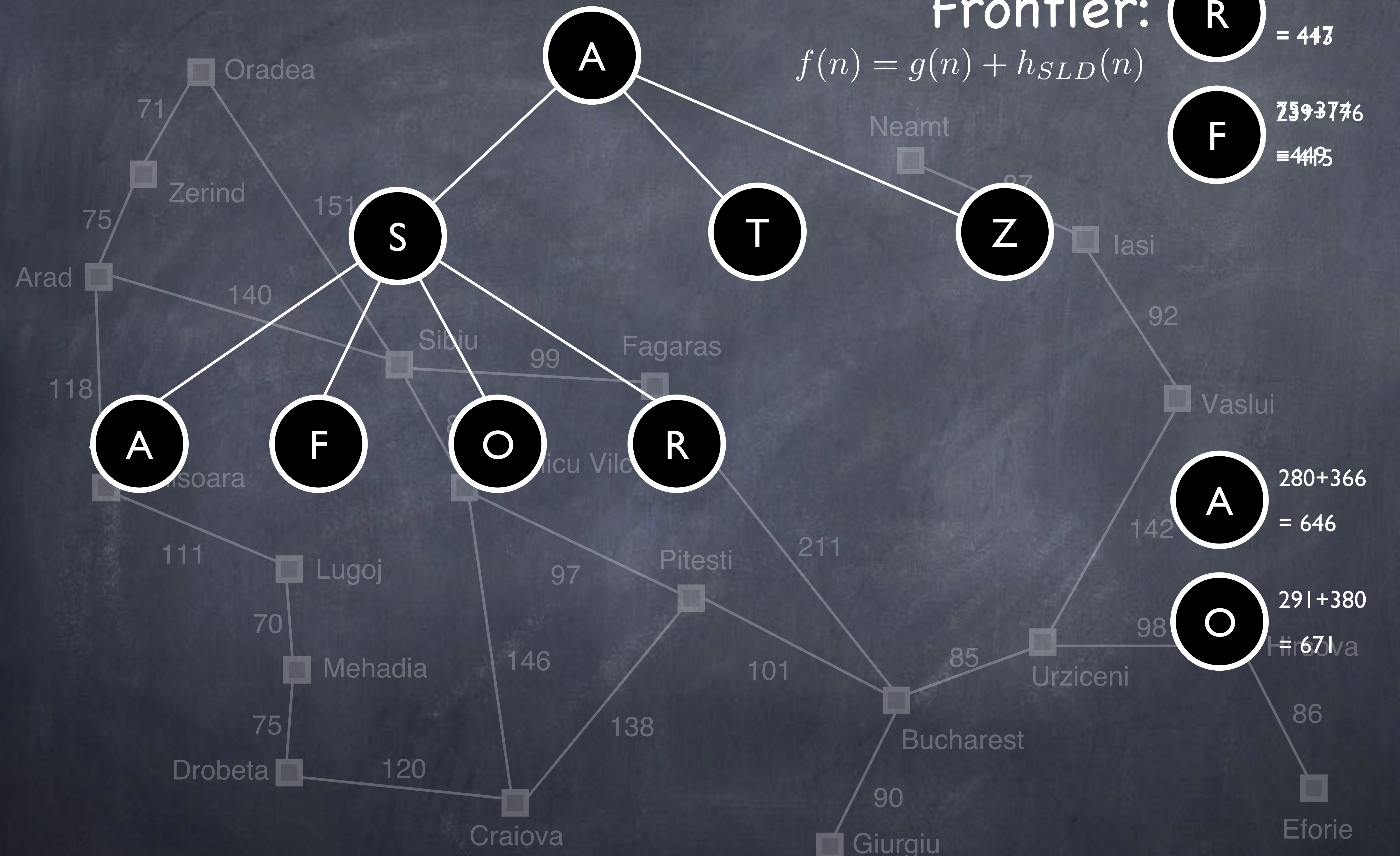
T 118+329 = 447

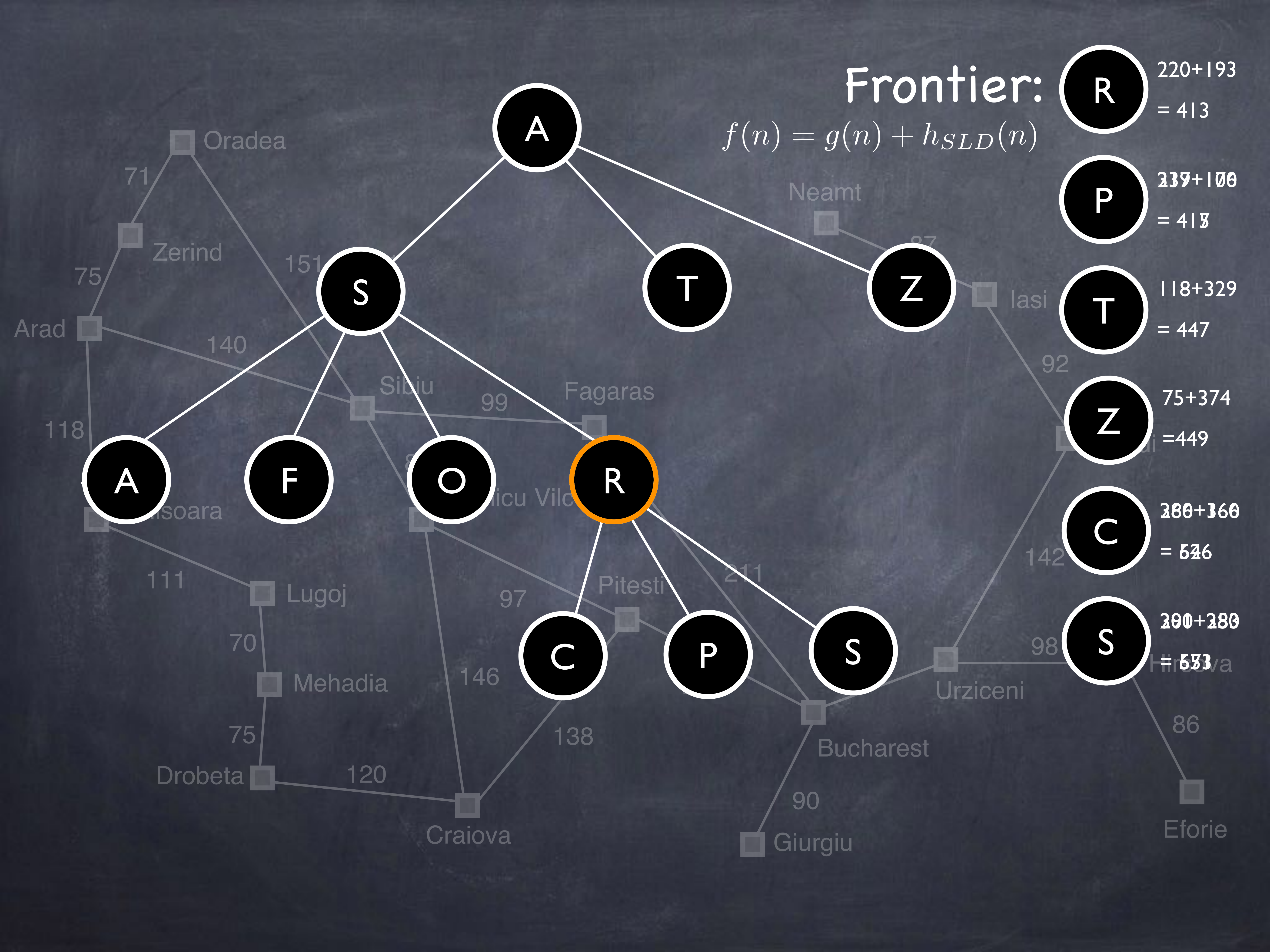
Z 75+374 =449



Frontier:

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

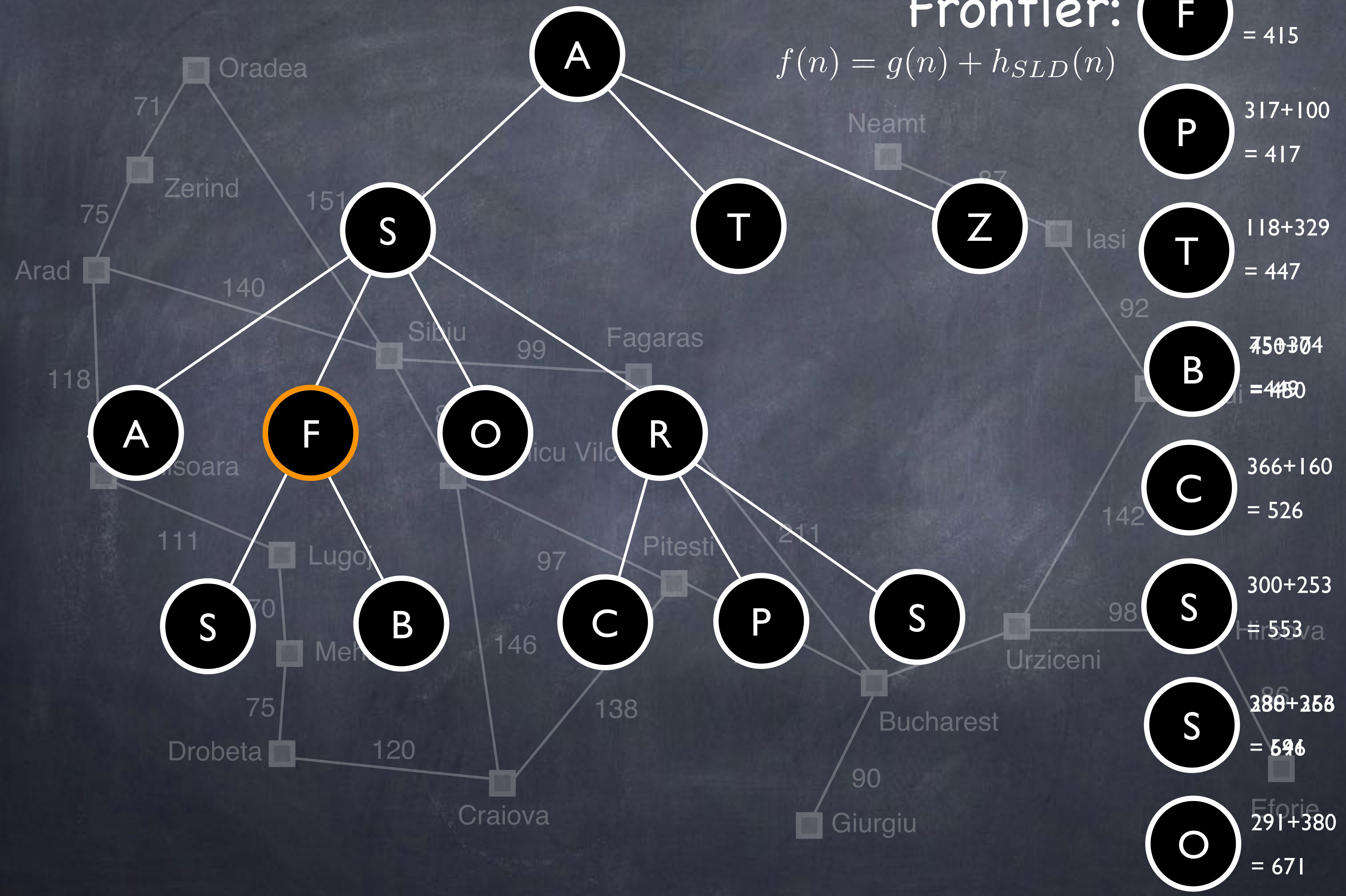


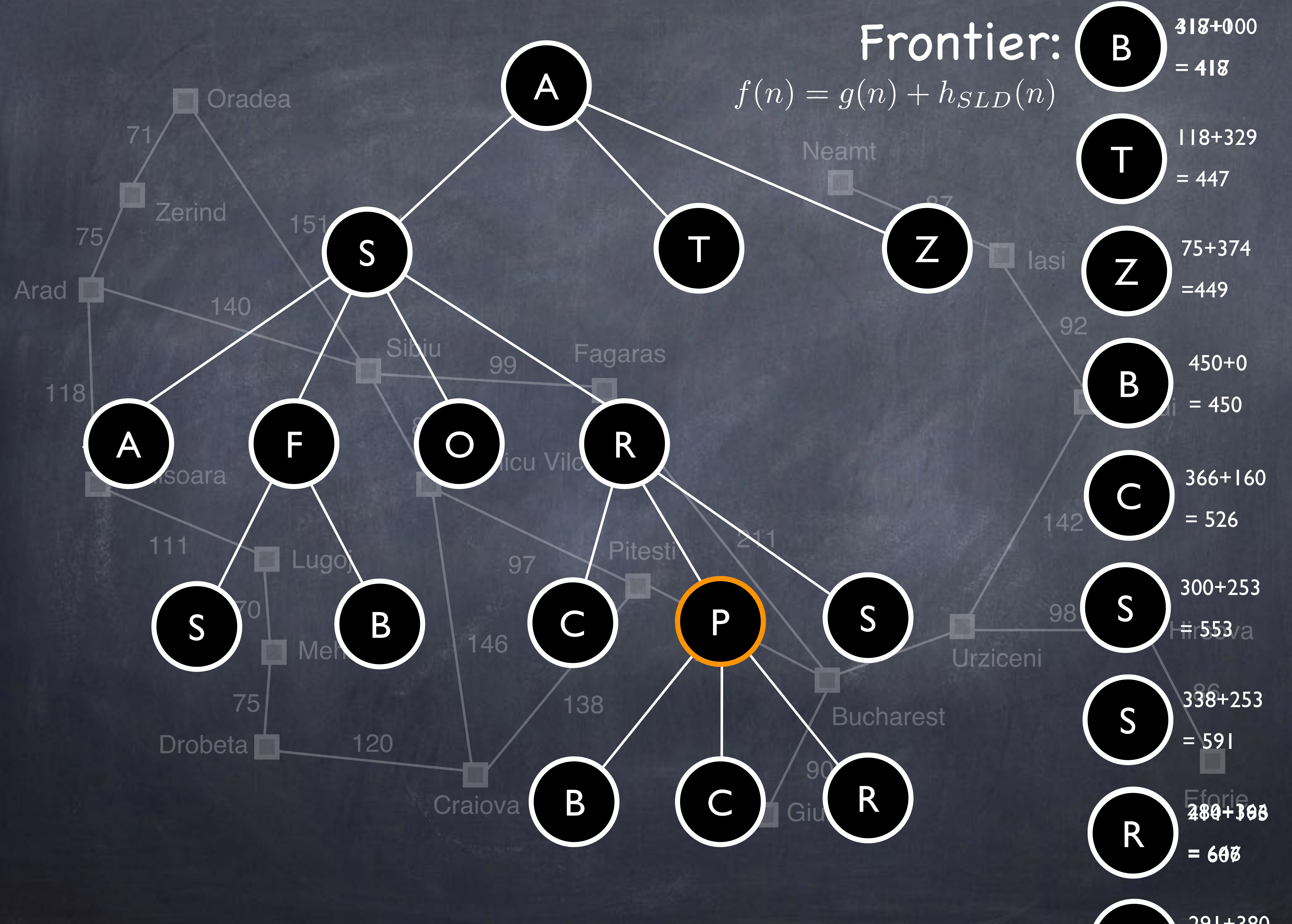


Frontier:
$f(n) = g(n) + h_{SLD}(n)$
R
220+193
= 413
P
T
118+329
= 447
Z
75+374
=449
C
S
A
S
T
Z
A
F
O
R
C
P
S
Oradea
Zerind
Arad
Sibiu
Fagaras
Neamt
Iasi
Lugoj
Mehadia
Drobeta
Craiova
Pitesti
Bucharest
Giurgiu
Urziceni
Eforie
71
75
151
140
118
99
111
70
75
120
146
97
138
211
90
142
92
98
86

Frontier:

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

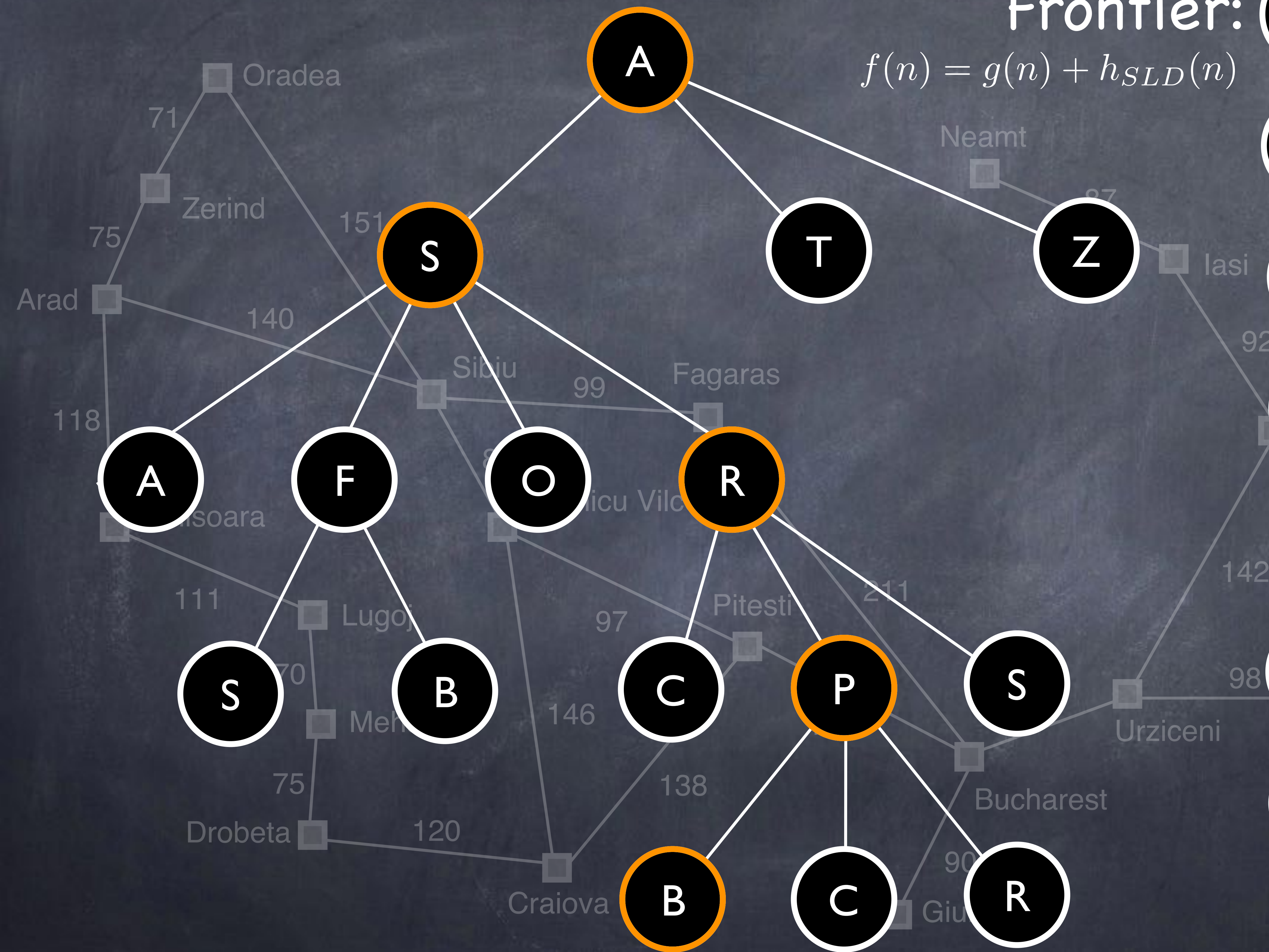




Frontier:
$f(n) = g(n) + h_{SLD}(n)$
A
S
T
Z
A
F
O
R
S
B
C
P
S
B
C
R
T 118+329 = 447
Z 75+374 =449
B 450+0 = 450
C 366+160 = 526
S 300+253 = 553
S 338+253 = 591
R
291+380
Oradea
Zerind
Arad
Sibiu
Fagaras
Neamt
Iasi
Pitesti
Urziceni
Bucharest
Drobeta
Craiova
71
75
151
140
118
99
92
211
97
111
146
138
120
75
98
142
90

Frontier:

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



B 418+0 = 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 414+193 = 607

280+366

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

A* Search

If $h(n)$ is admissible

Completeness

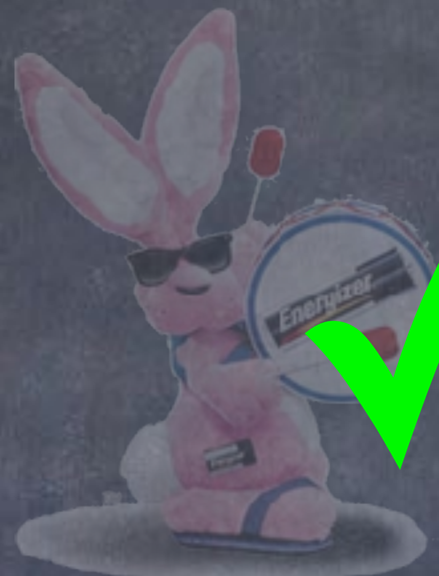
Optimality

Admissible Heuristic

Never overestimates the true cost of a solution

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

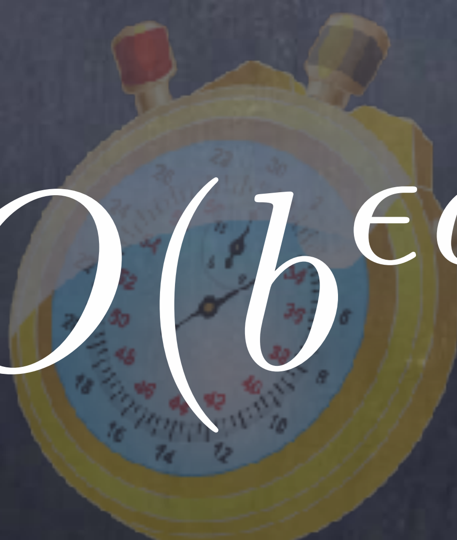
A* Search



If $h(n)$ is admissible

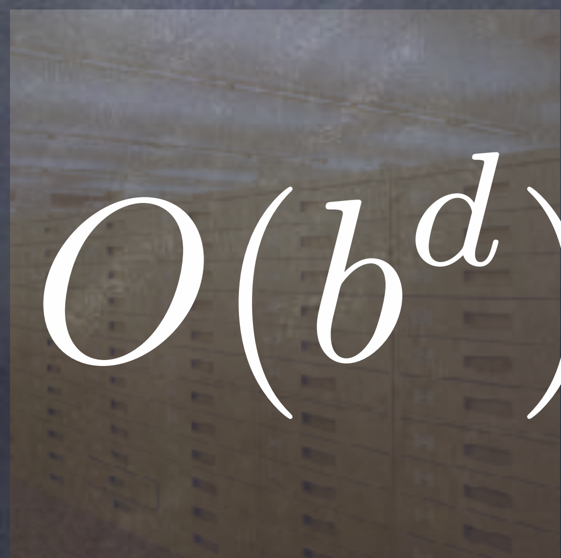
Completeness ✓

Optimality ✓



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

Time Complexity



$$O(b^d)$$

Space Complexity

Heuristic Functions

Where do good heuristic functions come from?

Good question...

(See Section 3.6 for some ideas)

Search Strategies

Uninformed / Informed

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

* If step costs are identical

† With an admissible heuristic

For next time:
AIMA 5.0–5.2.1