UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Lecture

Foundations of Artificial Intelligence

Part 5 – Informed Search

Dr. Mohsen Mesgar

Universität Duisburg-Essen

Organization

- **Exam**
  - **Questions:** English **AND** German.
  - **Answers:** English **OR** German **(only one language you can choose).**
  - **Dictionary:** is allowed (only hard copy).
  - **Anmeldephase** läuft vom **02.05.2022** bis **13.05.2022**

- **Homework:**
  - There are German and English versions of homework.
  - You can **do your homework** in **German.**

Recall …

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

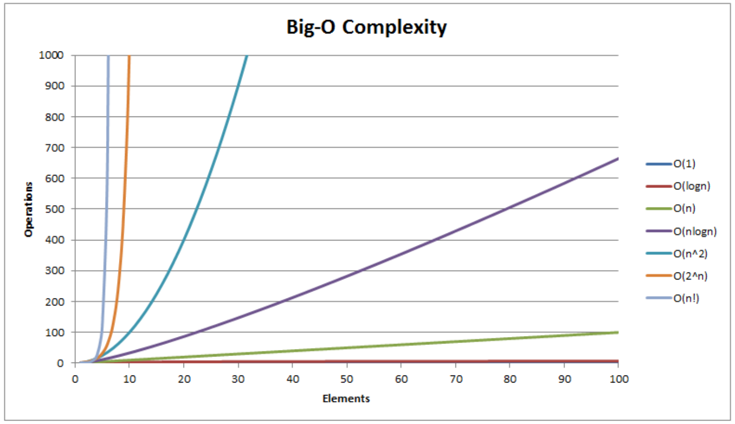
Search strategies

- Breadth-First (BFS),
- Depth-First (DFS),
- Iterative Deepening (ID)

Parameters:

- **b**: maximum branching factor of the search tree
- **d**: depth of the optimal solution
- **m**: maximum depth of the state space (may be ∞)

O Notation



Evaluation of search strategies

- **completeness**
- **optimality**
- time complexity
- space complexity

Evaluation of search strategies

Search strategies are compared along the following dimensions

- completeness: -
- optimality: -
- **time complexity:** How long does the search take?
- **space complexity:** How much memory does the search need?

Time complexity of BFS

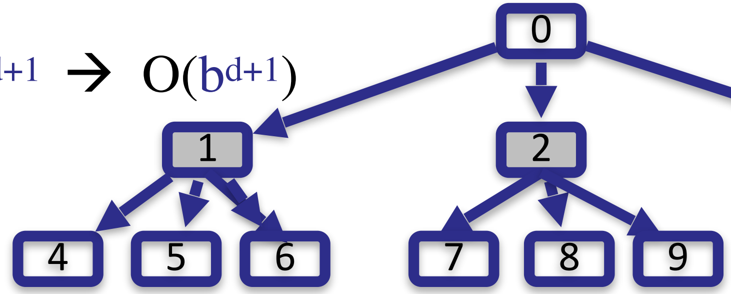
Time complexity: number of nodes that need to be generated

- **b**: maximum branching factor of the search tree
- **d**: **depth of the optimal solution**
- **m**: maximum depth of the state space (may be ∞)

- depth 0 → 1 node
- depth 1 → max b nodes
- depth 2 → max b * b = b^2 nodes
- depth d → max b^d nodes (goal state is here)
- depth d+1 → max b^{d+1} nodes

- Total: 1+ $b + b^2 + b^3 \ldots + b^d + b^{d+1}$ → $O(b^{d+1})$

d = 1
b = 3
m = 2



Time complexity of DFS

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Time complexity: number of nodes that need to be generated

DFS:

- follows each path until maximum depth m

  → $O(b^m)$

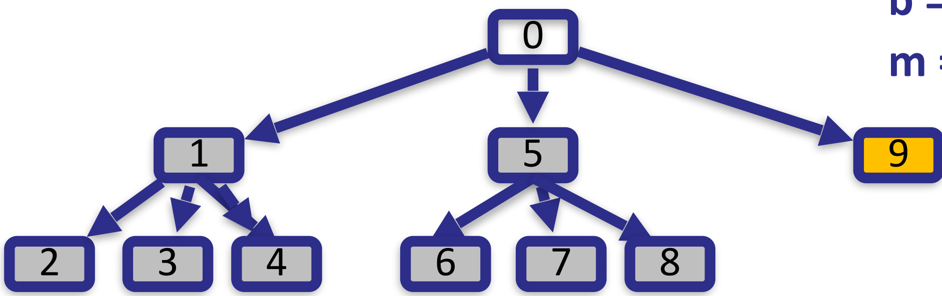
- terrible if m much larger than d

- **b**: maximum branching factor of the search tree
- **d**: depth of the optimal solution
- **m: maximum depth of the state space (may be ∞)**

d = 1

b = 3

m = 2



Time complexity of iterative deepening

Time complexity: number of nodes that need to be generated

Iterative deepening:

- m never goes beyond d

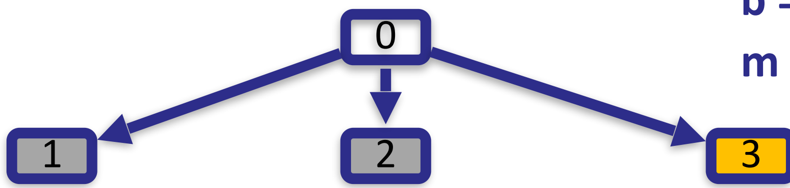
→ $O(b^d)$

- **b**: maximum branching factor of the search tree
- **d**: depth of the optimal solution
- **m**: **maximum depth of the state space (may be ∞)**

d = 1

b = 3

m = 2

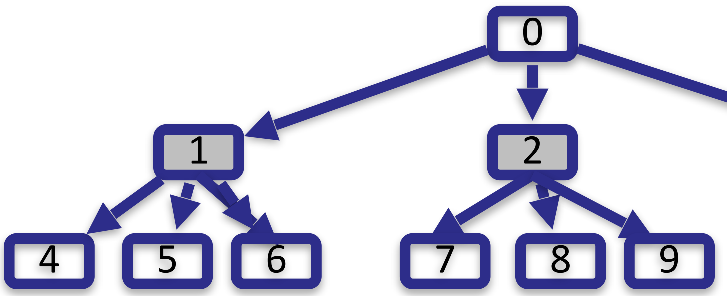


Space complexity of BFS

- **Space complexity:** maximum number of nodes in memory
- **BFS**
  - all nodes in a level should remain in memory

    → $O(b^{d+1})$

d = 1
b = 3
m = 2



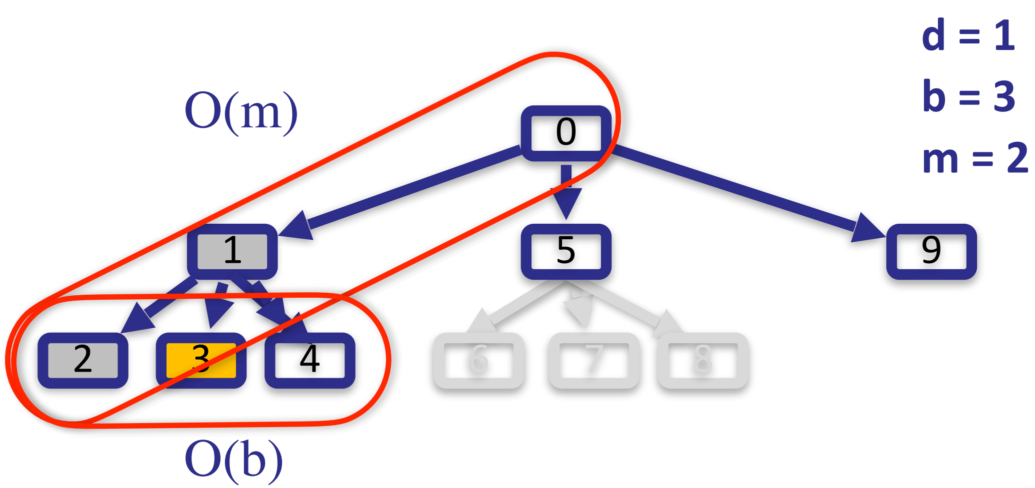
Space complexity of DFS

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

- Space complexity: maximum number of nodes in memory
- **DFS**
  - only nodes in current path and their unexpanded siblings need to be stored

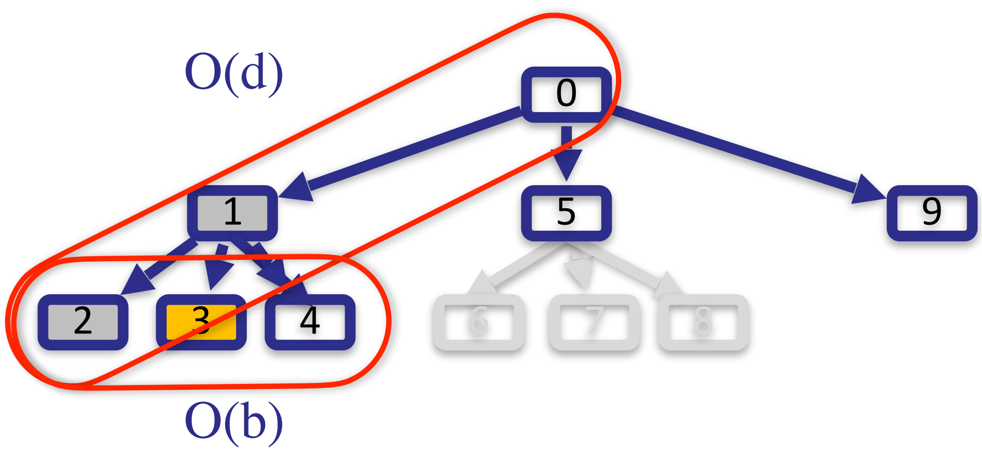
    → O(bm)



Space complexity of iterative deepening

- **Space complexity:** maximum number of nodes in memory
- **ID:**
  - only nodes in **current path** and **their unexpanded siblings** need to be stored

    → $O(bd)$



Performance review summary

- Iterative deepening search uses only **linear space** and not much more time than other **uninformed algorithms**

	BFS	DFS	ID
Complete?	Yes	No	Yes
Optimal?	Yes	No	Yes
Time complexity	b^{d+1}	b^m	b^d
Space complexity	b^{d+1}	bm	bd

Any open questions?

Where are we?

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

- Intelligence & Agents
- Search
  - Uninformed
  - **Informed**
  - Local

In this lecture, you learn …

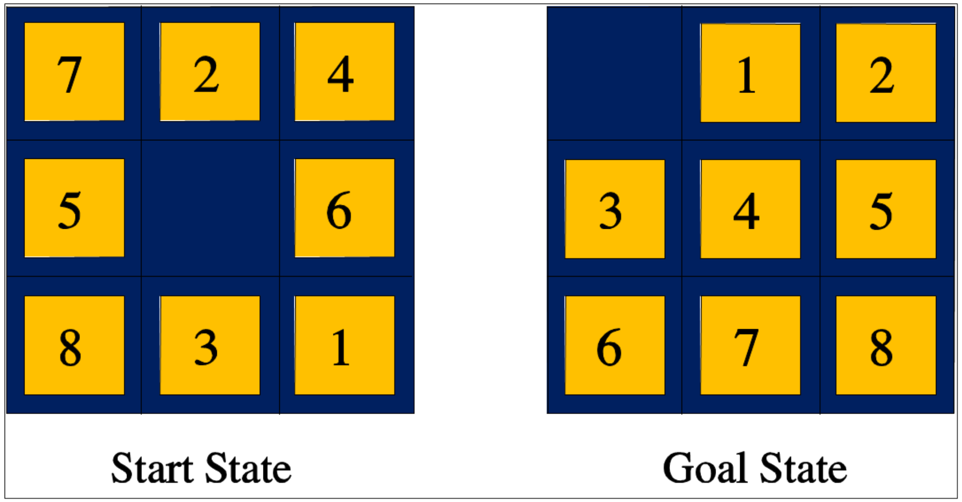
- **Informed search algorithms using heuristics**
  - Greedy BFS
  - A* search
- **Heuristics**
  - Admissible
  - Consistent
  - How do we choose a good heuristic?

Informed Search

Motivation

Uninformed search algorithms are **inefficient**

- they expand far **too many unpromising paths**



- average solution depth = 22
- BFS expands about 3.1 x 10^{10} nodes until depth 22
- Can we make it more efficient?

Idea …

- Instead of **randomly expanding nodes in a level,** …
- expand the **most promising node first**
- How do we determine which node is the most promising one?

→ **heuristics**

Heuristics

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Heuristics are **experience-based approximations** for solving a problem

- **Rule of thumb**
- **Heuristics may go wrong!**

https://upload.wikimedia.org/wikipedia/commons/2/2f/Eureka_arkimedi.jpg

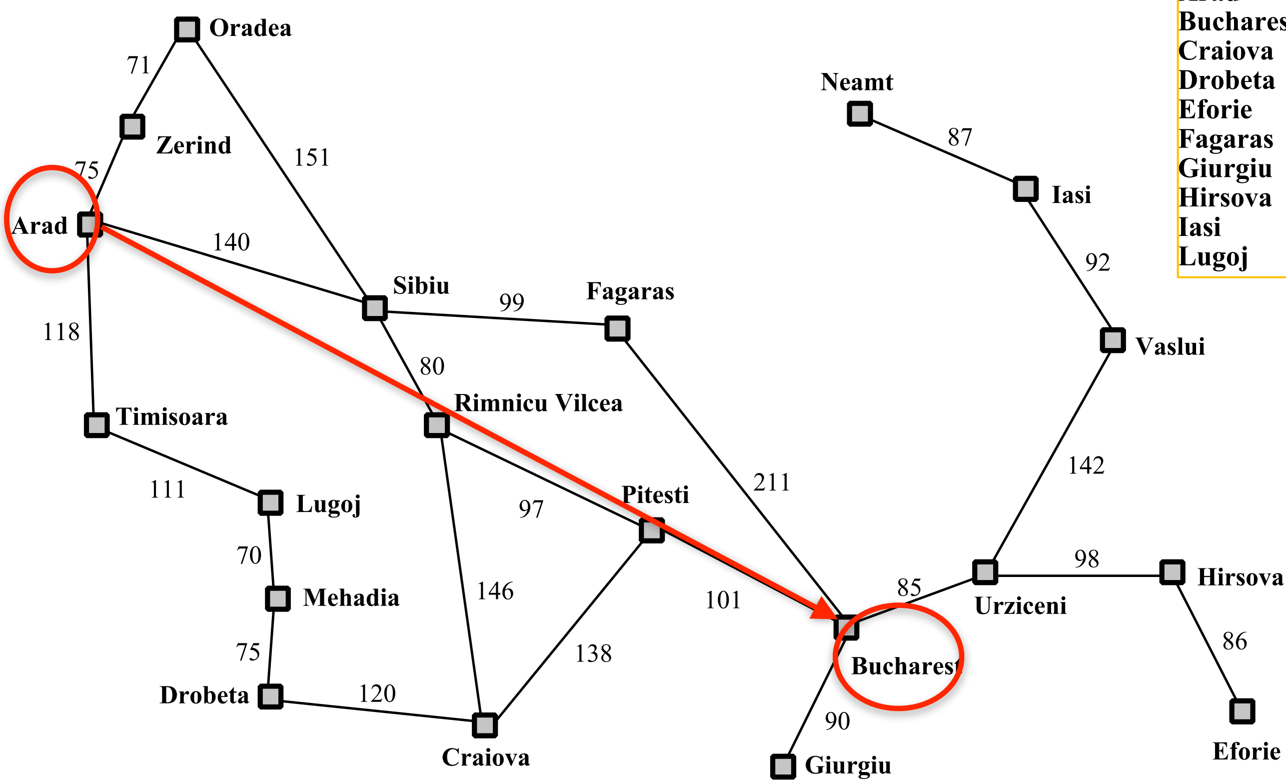
Heuristics

In tree-search algorithms, **a heuristic is a utility function $f(n)$ for each node n**

- it estimates **the 'desirability' of the node's state**

Getting informed

Straight line distances between cities



Straight line distances (SLD) to the goal state

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Let’s refer to city names by their first letters

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

A	366	M	241
B	0	N	234
C	160	O	380
D	242	P	100
E	161	R	193
F	176	S	253
G	77	T	329
H	151	U	80
I	226	V	199
L	244	Z	374

Today

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Greedy BFS

Greedy BFS

Utility function $f(n) = h(n)$

- $h(n)$: **goal-oriented heuristic that estimates the cost from node $\mathbf{n}$ to the goal**

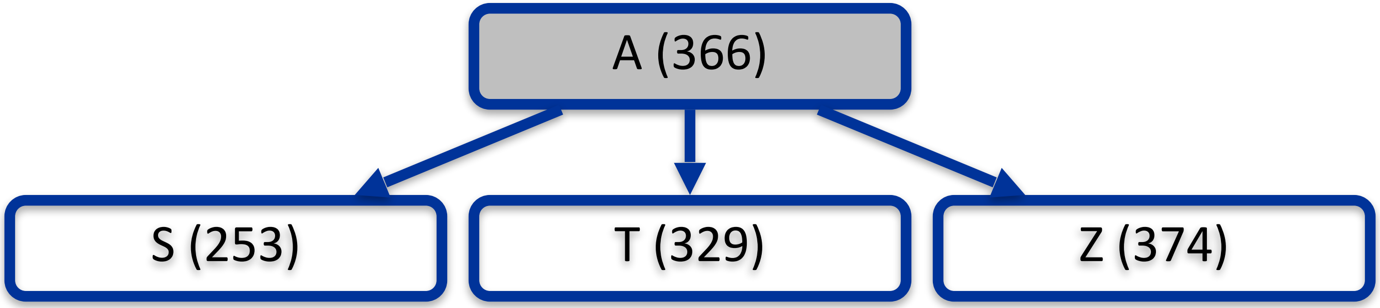
Greedy BFS expands the node that **appears to be nearest to the goal**

Example: $h_{SLD}(n)$ = **S**traight **L**ine **D**istance from $\mathbf{n}$ to Bucharest

Greedy BFS

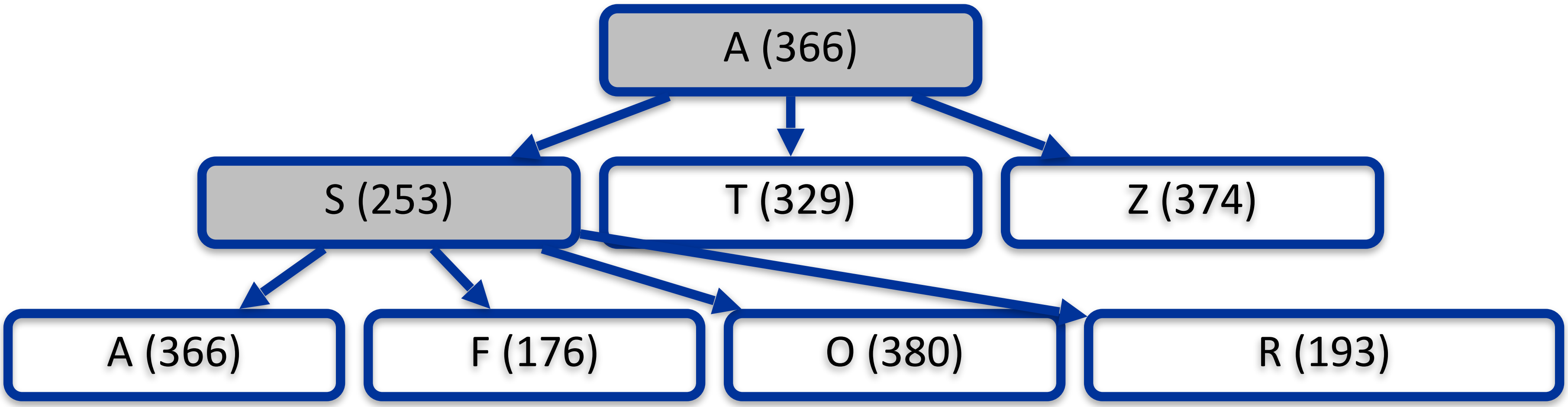
Example: $h_{SLD}(n)$ = **S**traight **L**ine **D**istance from **n** to Bucharest

A	366	**M**	241
B	0	**N**	234
C	160	**O**	380
D	242	**P**	100
E	161	**R**	193
F	176	**S**	253
G	77	**T**	329
H	151	**U**	80
I	226	**V**	199
L	244	**Z**	374



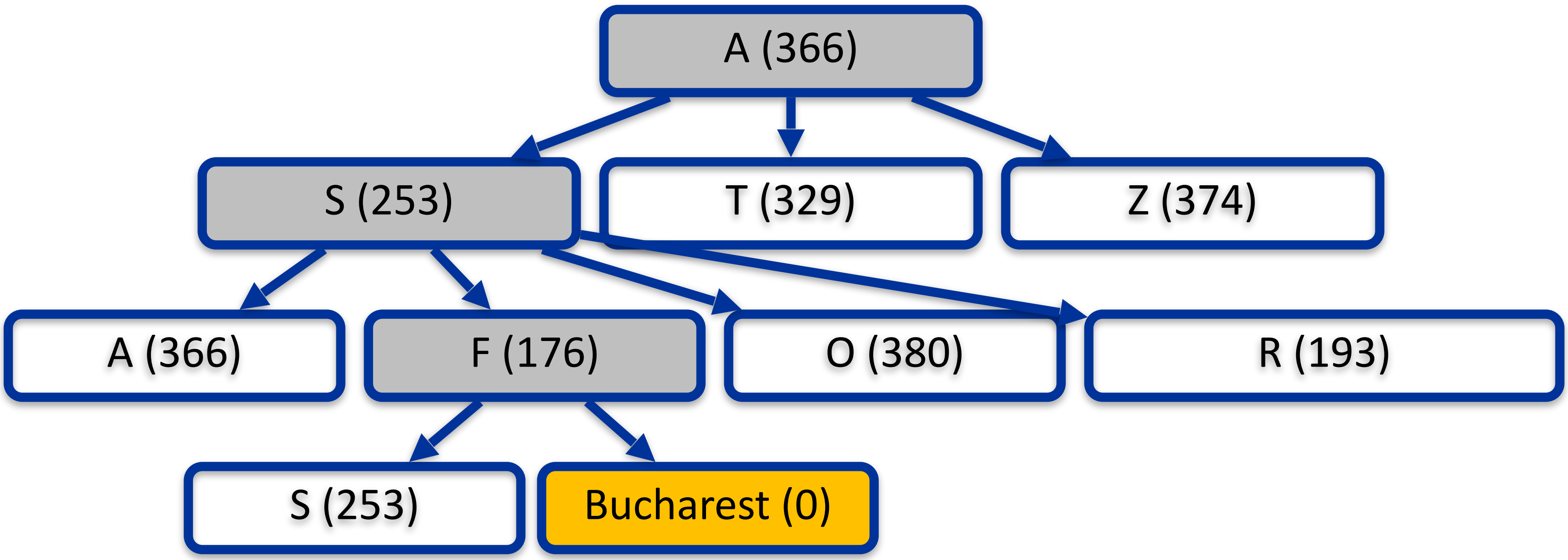
Example: Greedy best-first search

Example: $h_{SLD}(n)$ = straight-line distance from n to Bucharest



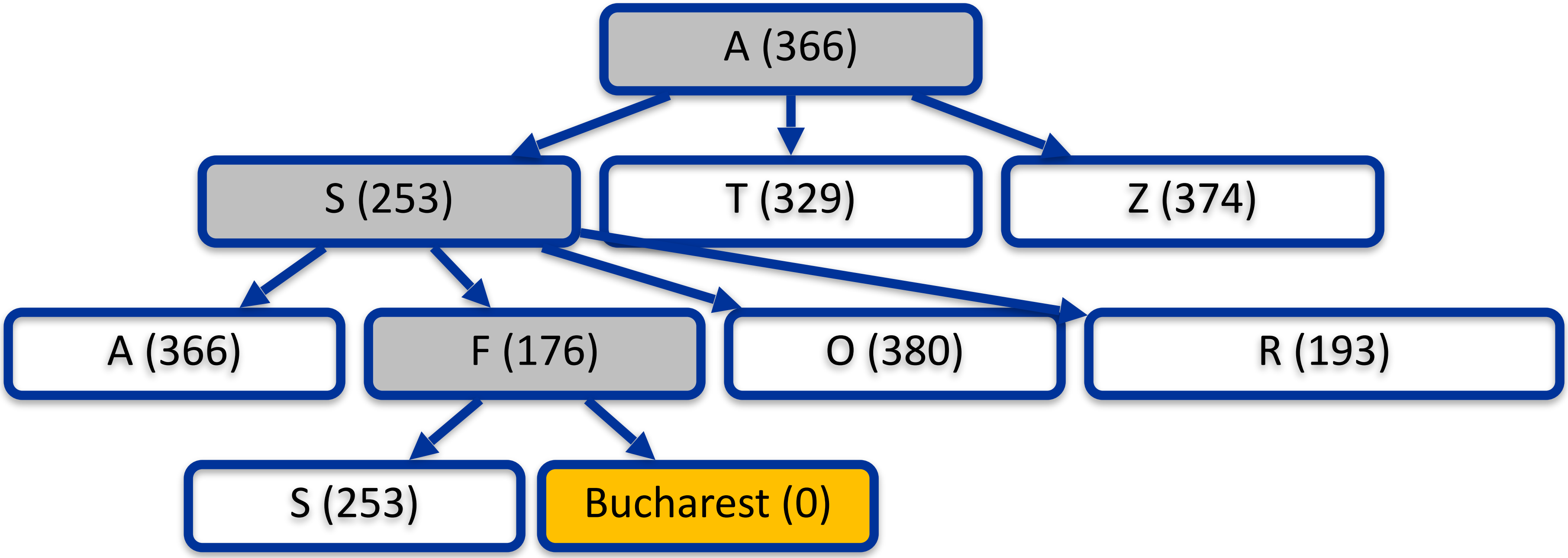
Example: Greedy best-first search

Example: $h_{SLD}(n)$ = straight-line distance from n to Bucharest



Example: Greedy BFS

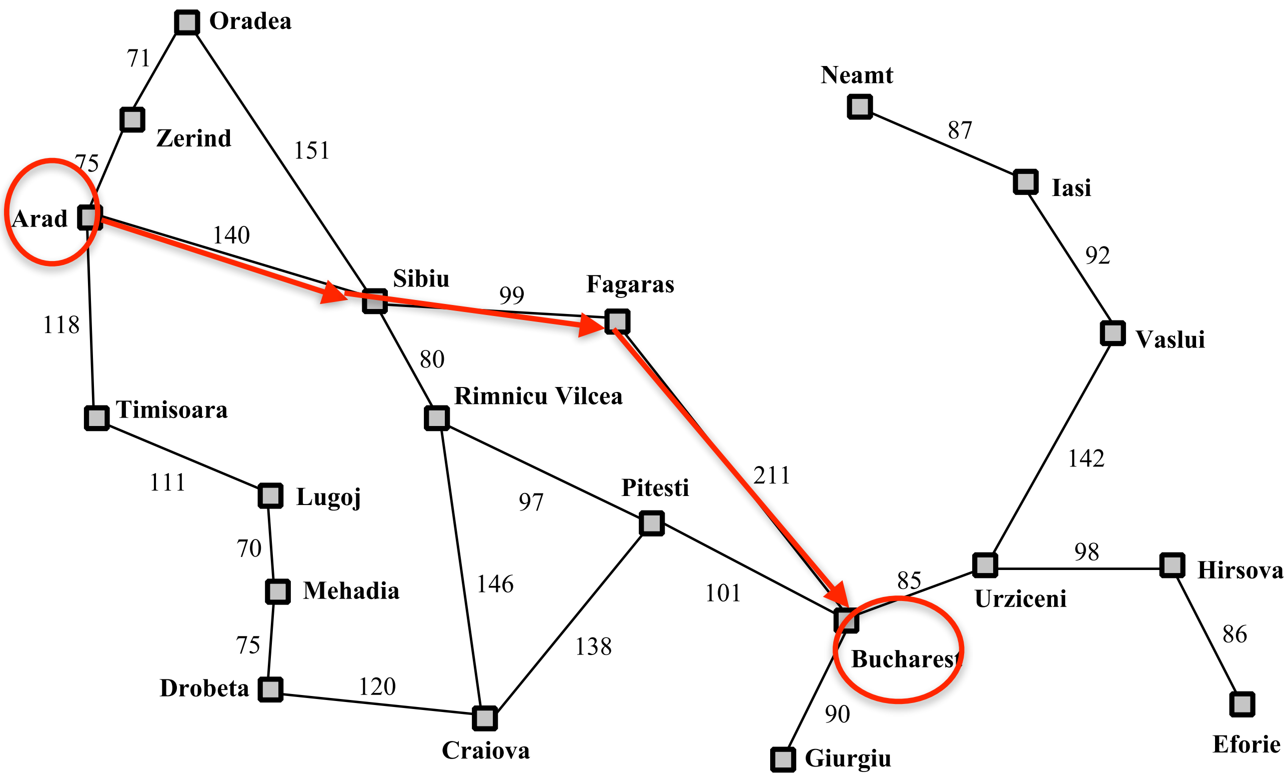
Example: $h_{SLD}(n)$ = straight-line distance from n to Bucharest



A —> S —> F —>B

Here is the solution

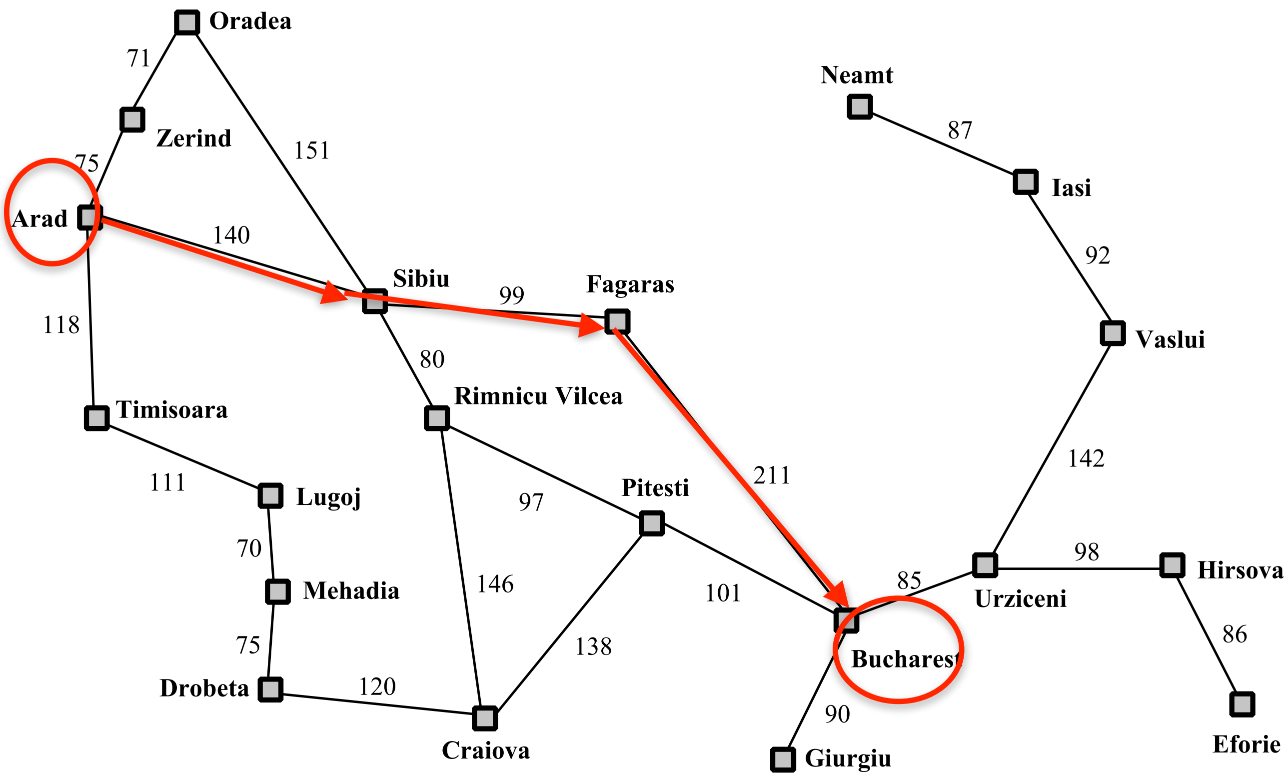
A —> S —> F —>B



Here is the solution

A —> S —> F —>B

What is the cost of this path (solution)?

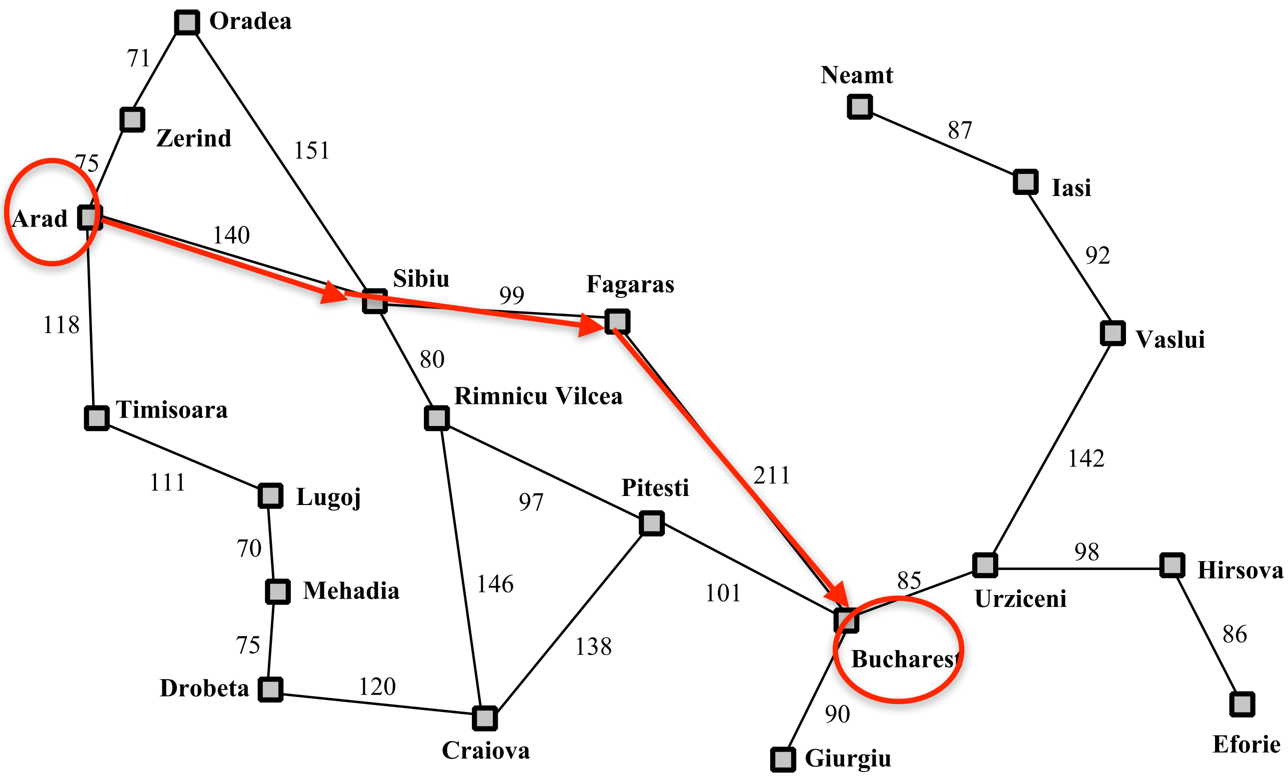


Here is the solution

A —> S —> F —>B

What is the cost of this path (solution)?

140 + 99 + 211 = 450



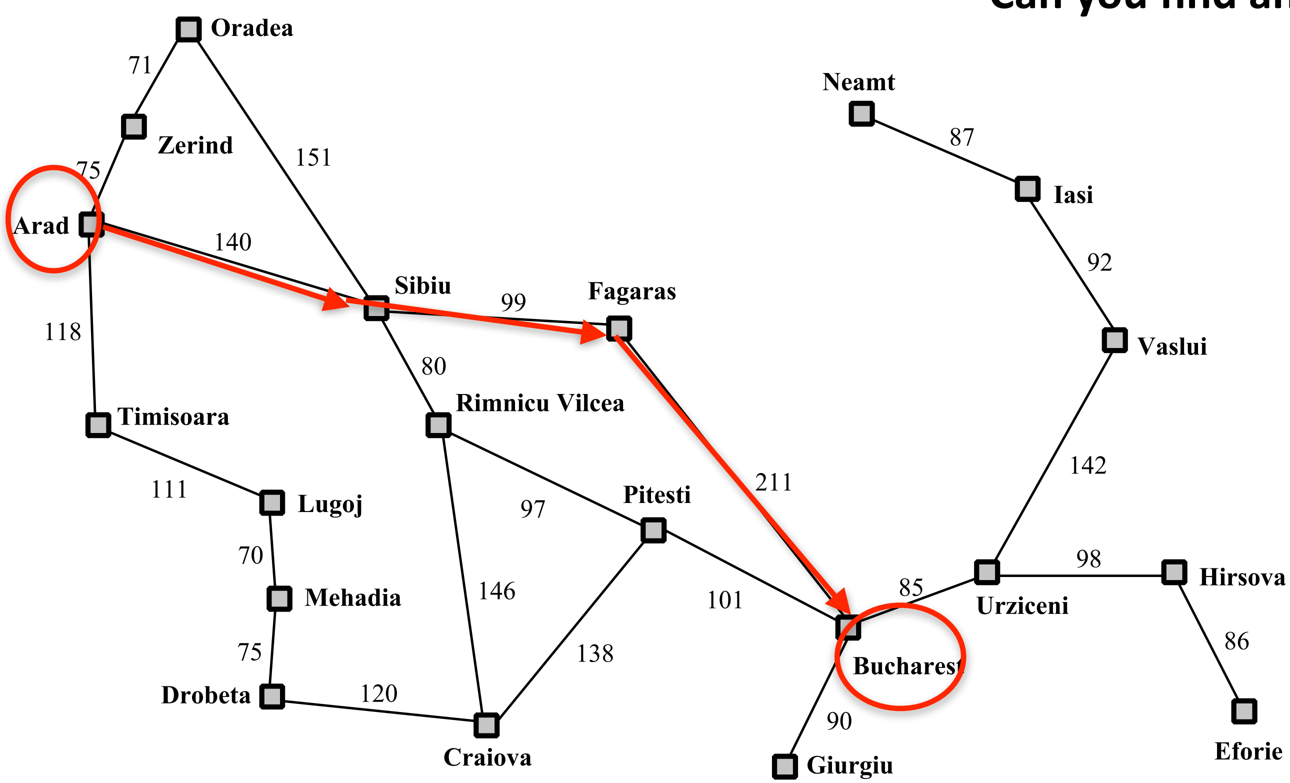
Here is the solution

A —> S —> F —>B

What is the cost of this path (solution)?

140 + 99 + 211 = 450

Can you find another path with a lower cost?



Here is the solution

A —> S —> F —>B

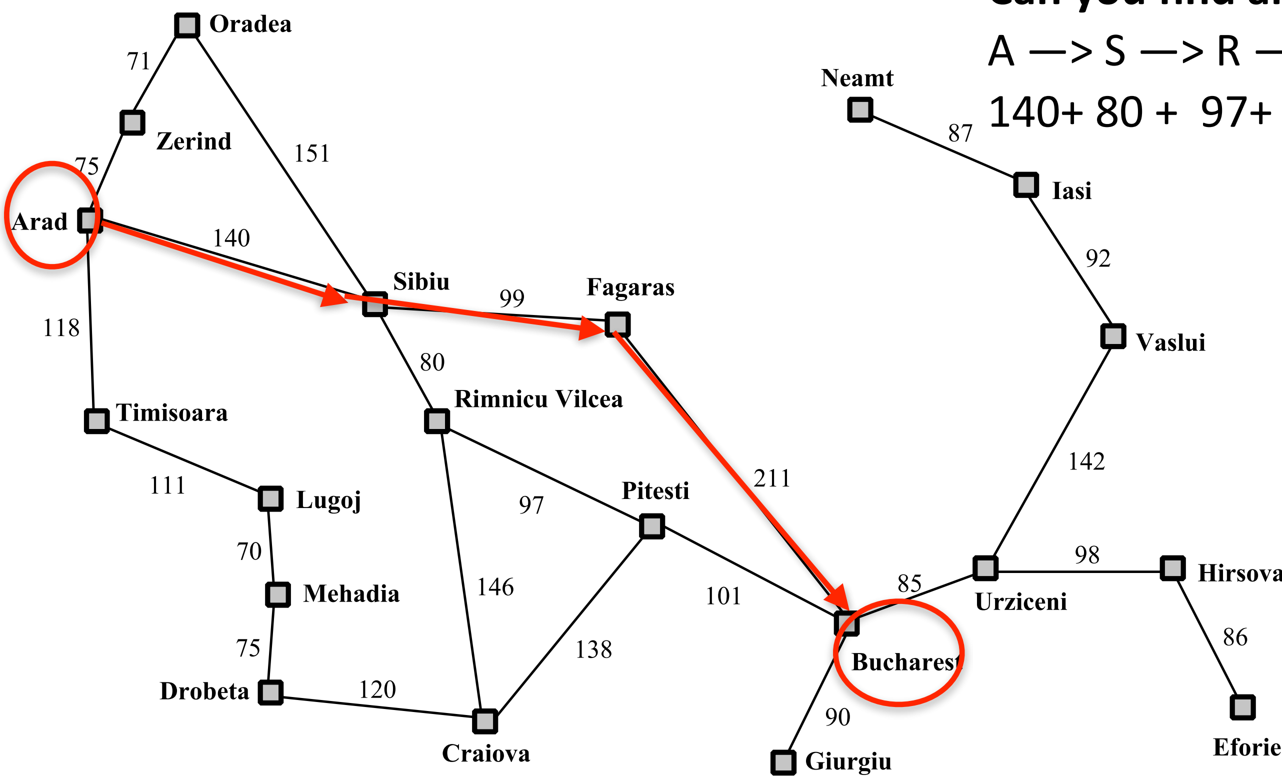
What is the cost of this path (solution)?

140 + 99 + 211 = 450

Can you find another path with a lower cost?

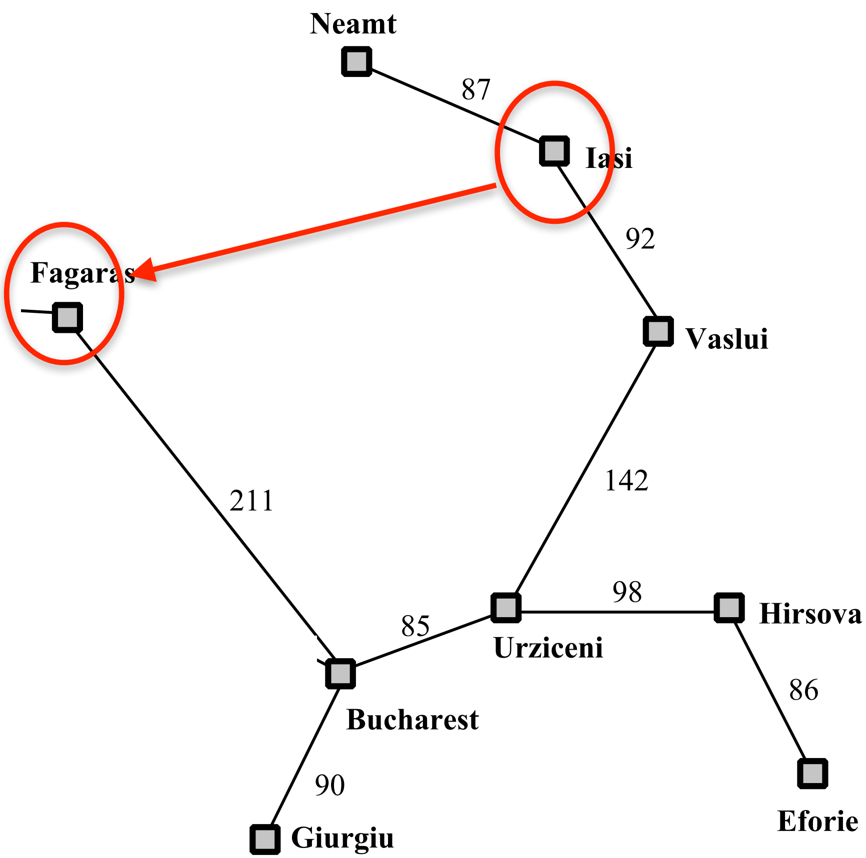
A —> S —> R —> P —> B

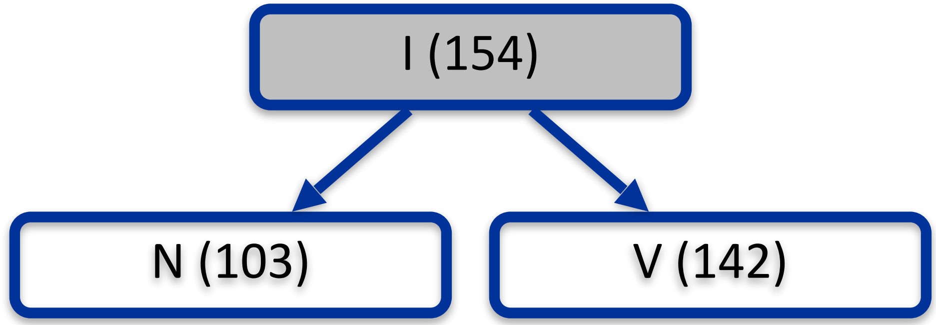
140+ 80 + 97+ 101 = 418



Another example

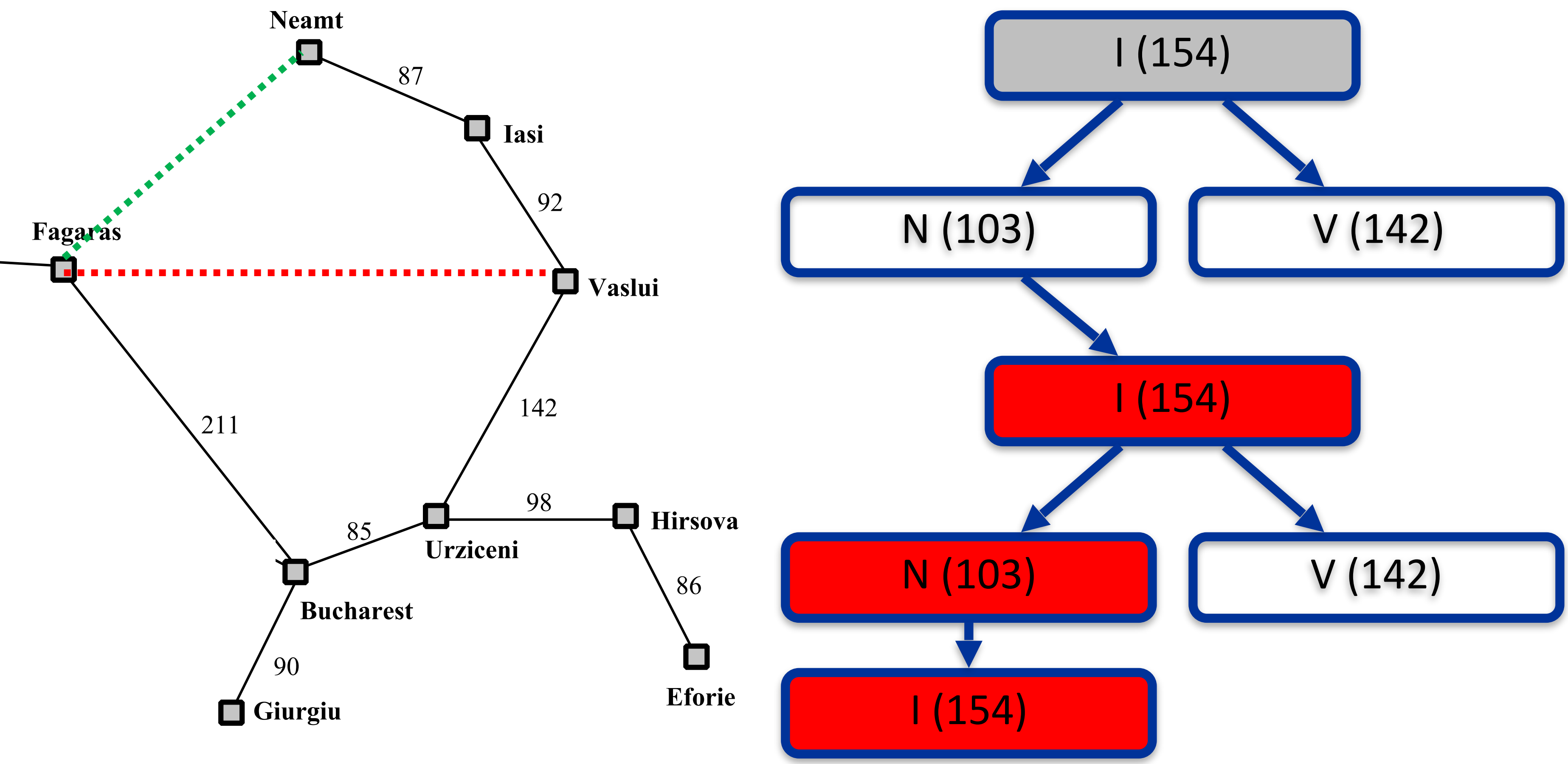
Find a path from **Iasi (I)** to **Fagaras (F)** (with greedy best-first search)





Loops in greedy best-first search

Find a path from Iasi to Fagaras (with greedy best-first search)



Properties of greedy BFS

Completeness: Do we always **find a solution** if one exists?

- **No** –> can get stuck in loops

→ complete in finite state space **if we check for repeated states**

Optimality: Do we always find **an optimal solution**?

- **No**: the path Arad -> Sibiu -> Fagaras -> Bucharest is not optimal

Time complexity:

- **O(b^m)**, **like DFS**
- optimal case: best choice in each step → only *d* steps
- a good heuristic improves chances for encountering optimal case

Space Complexity

- has to keep all nodes in memory → same complexity as BSF

A* Search

A* search

Best-known form of **BFS**

Idea

- **avoid expanding paths** that are **already expensive**

→ **evaluate complete path cost (not only remaining costs)**

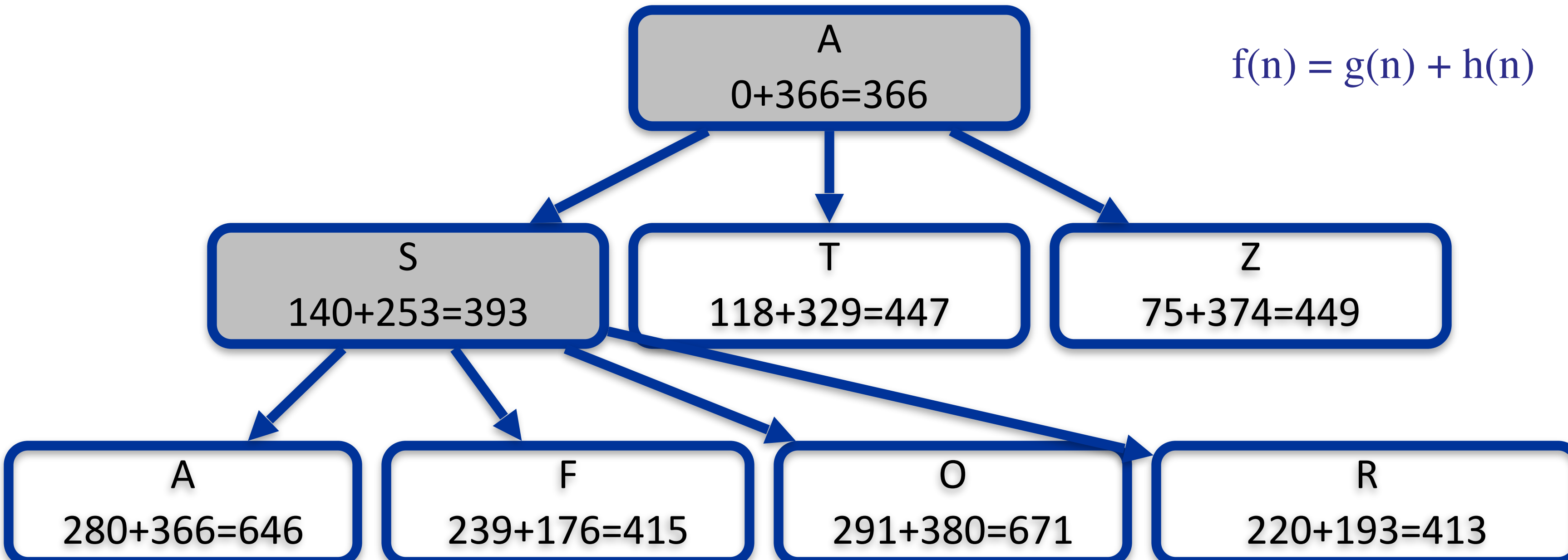
Utility function $f(n) = g(n) + h(n)$

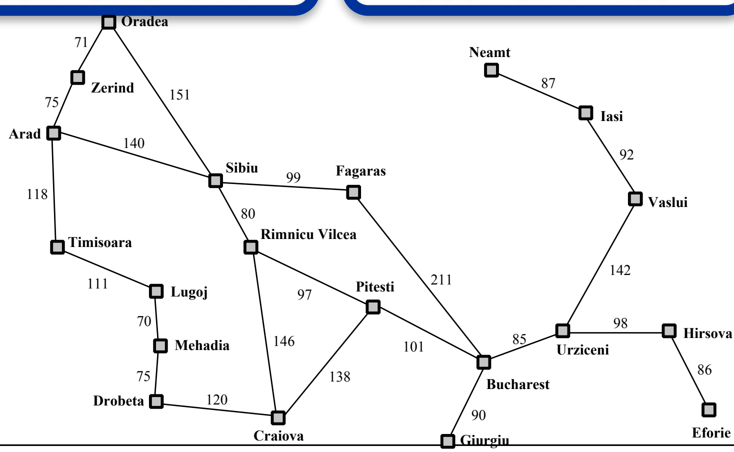
- $g(n)$ = **real cost** to reach node n from source
- $h(n)$ = estimated cost **to get from $\mathbf{n}$ to goal $\mathbf{G}$**
- $f(n)$ = **estimated cost of path to goal via $\mathbf{n}$**

A* search

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

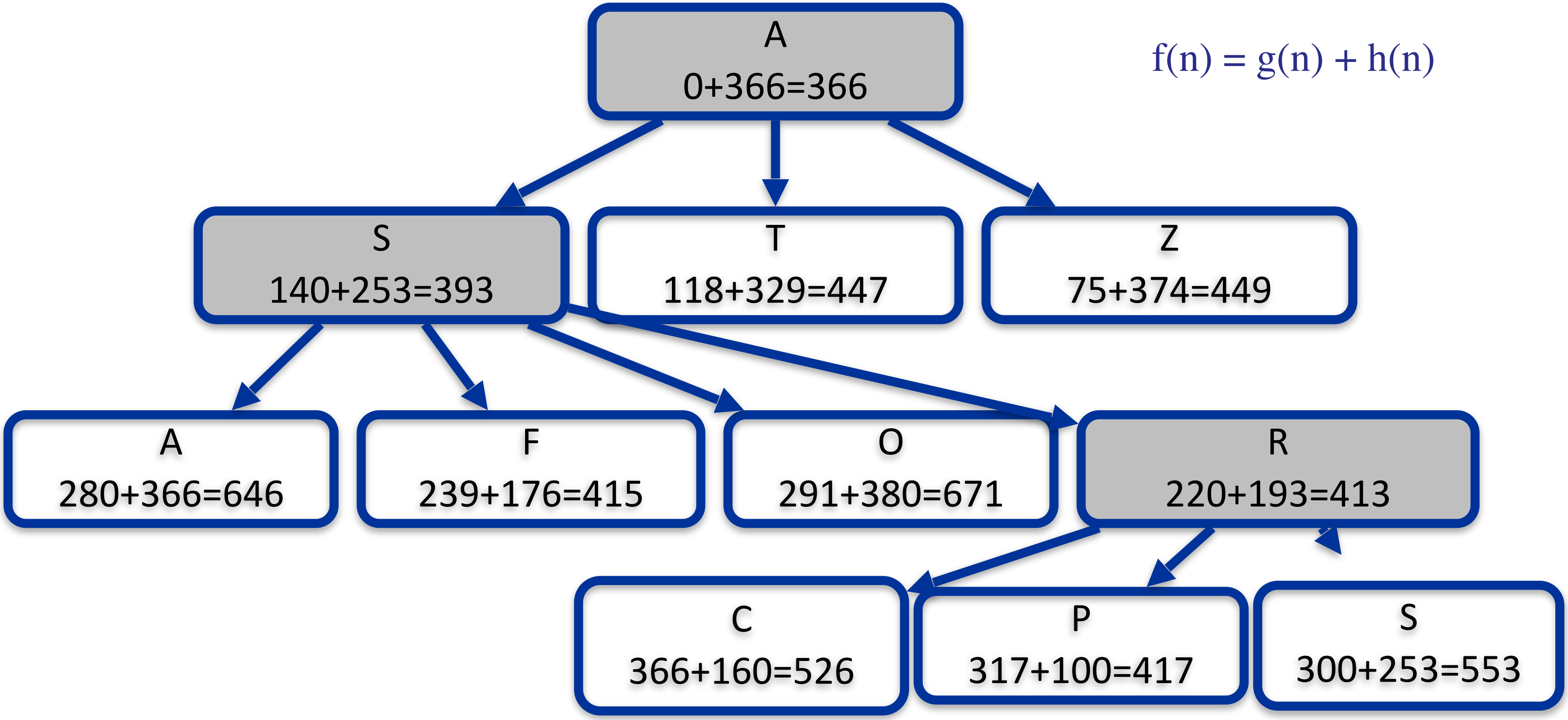
$$f(n) = g(n) + h(n)$$





A	366	M	241
B	0	N	234
C	160	O	380
D	242	P	100
E	161	R	193
F	176	S	253
G	77	T	329
H	151	U	80
I	226	V	199
L	244	Z	374

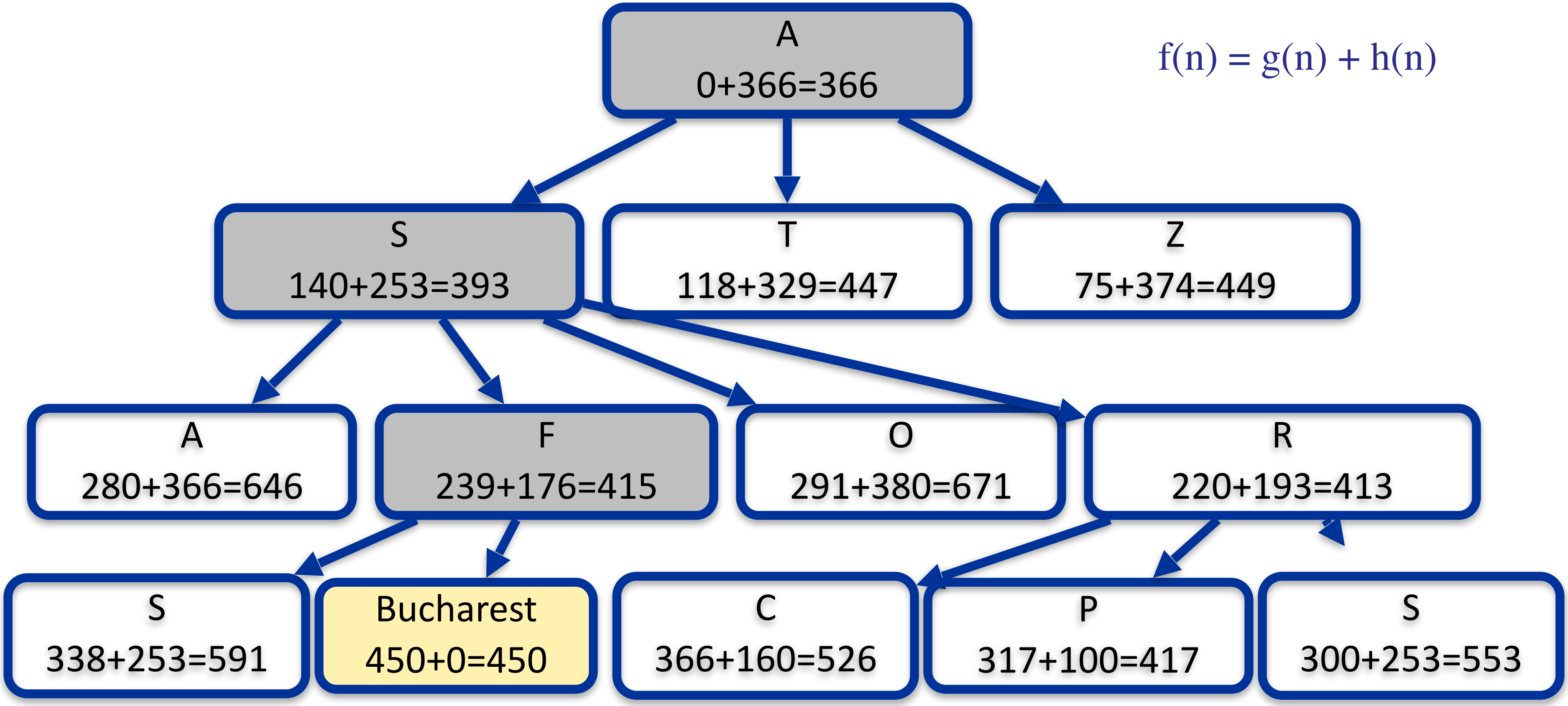
A* search



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

R (Rimnicu Vilcea) will be expanded because it has the lowest value for $f(n)$.

A* search



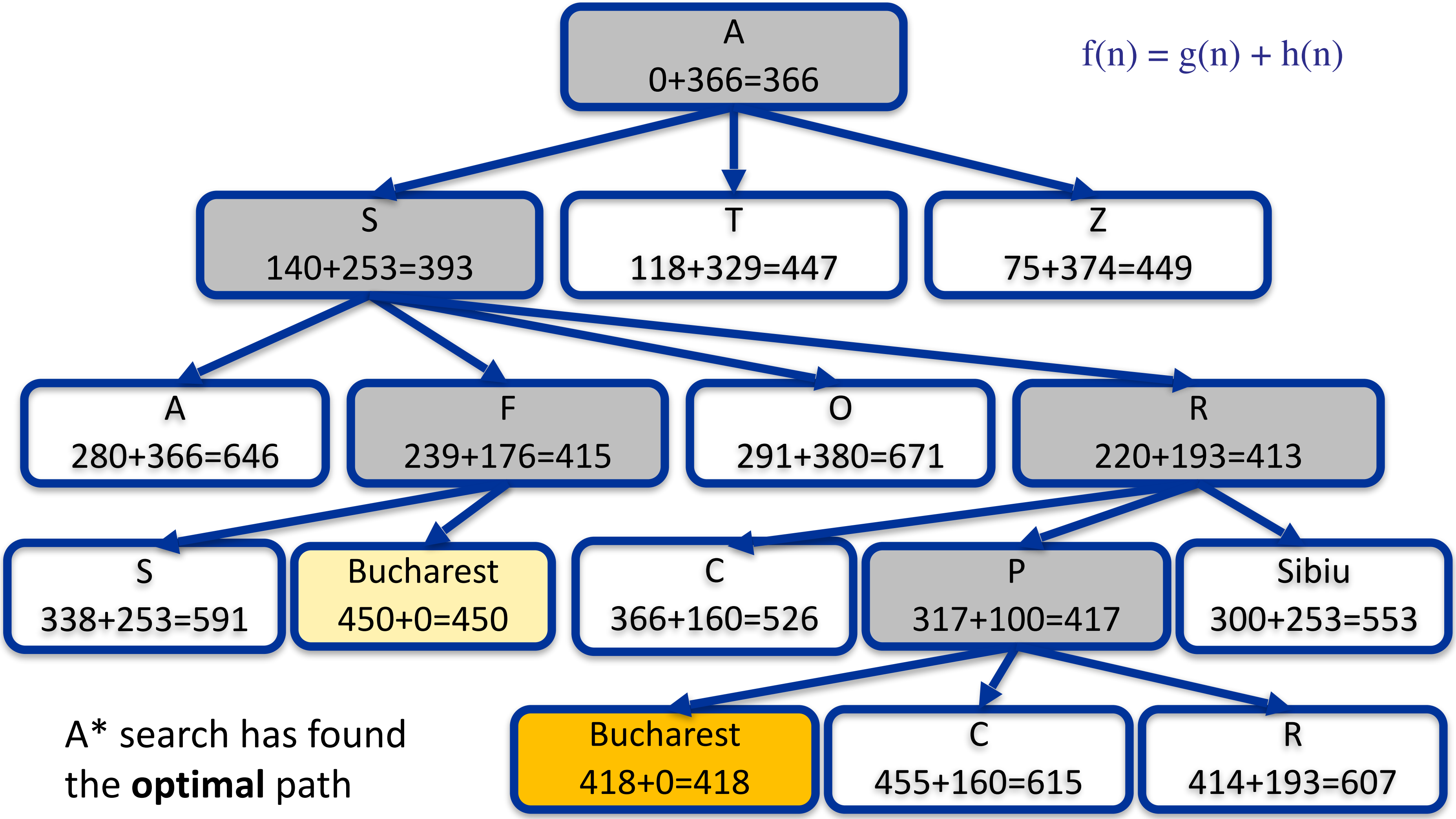
A* search has found one possible path …

But: P will be expanded next although Bucharest is already in the fringe.
Greedy search would **not** do that.

A* search

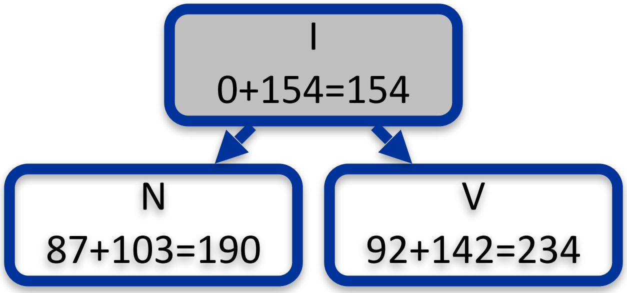
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

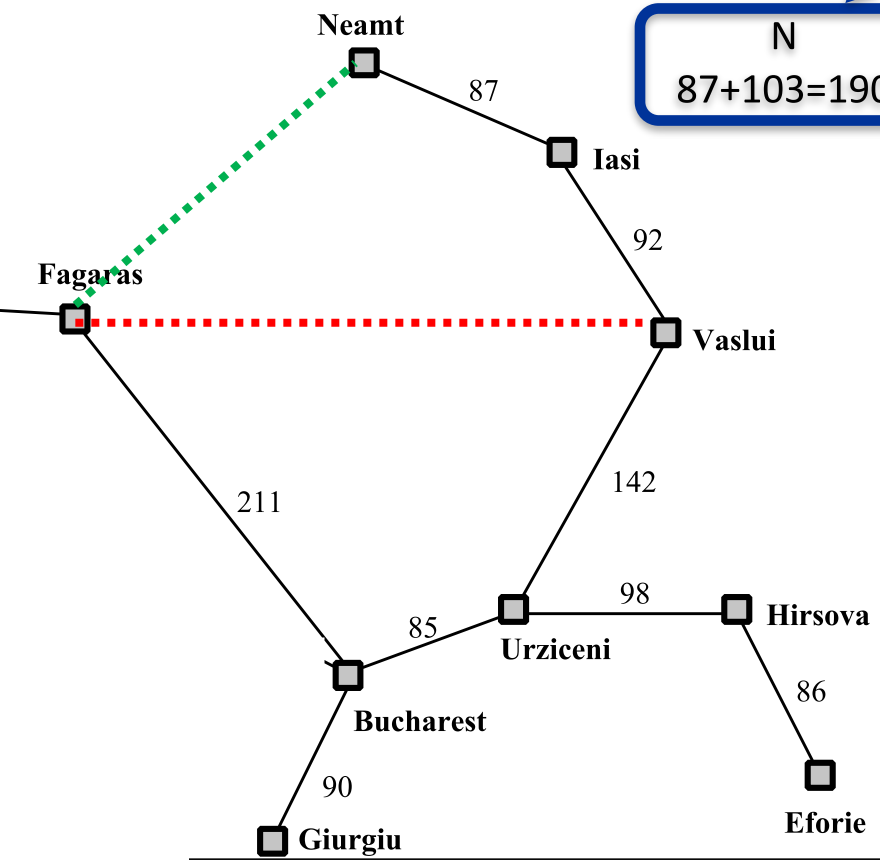
$$f(n) = g(n) + h(n)$$



A* search has found the **optimal** path

Avoiding loops in A* search: Find a path from Iasi to Fagaras





City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176

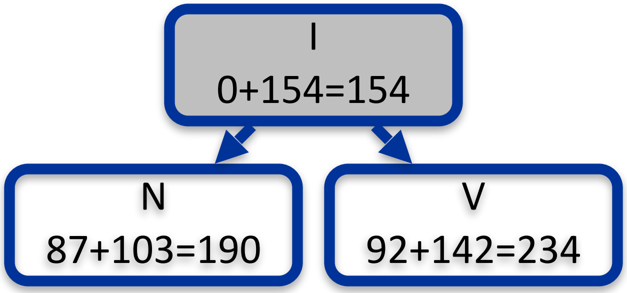
City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

Avoiding loops in A* search: Find a path from Iasi to Fagaras

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

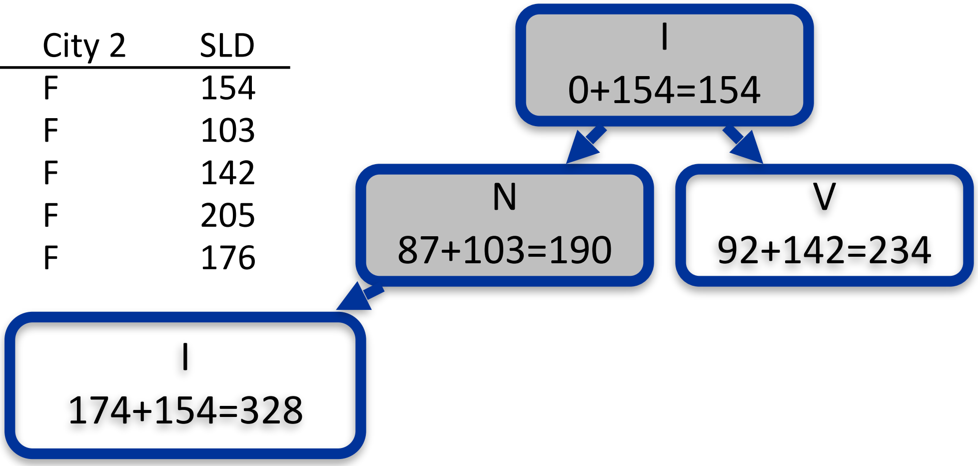
Always expand the node with the lowest value of the utility function

Avoiding loops in A* search: Find a path from Iasi to Fagaras

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

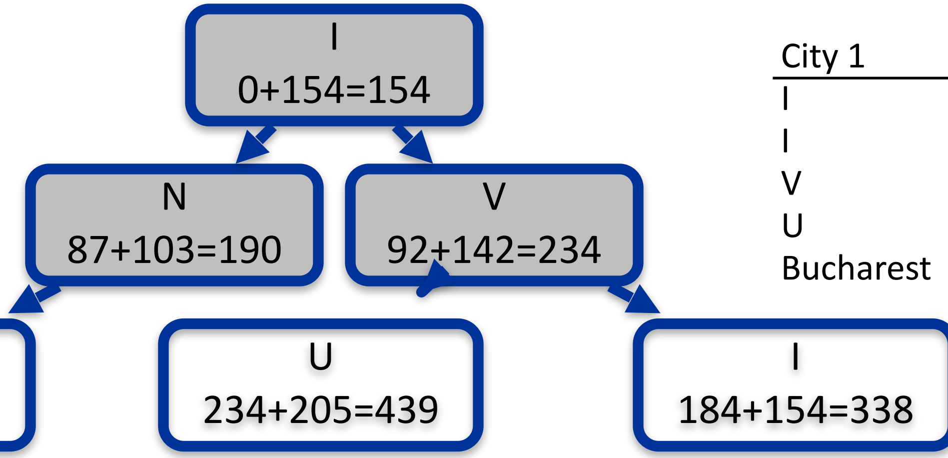
City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

Avoiding loops with A* search: Find a path from Iasi to Fagaras

City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



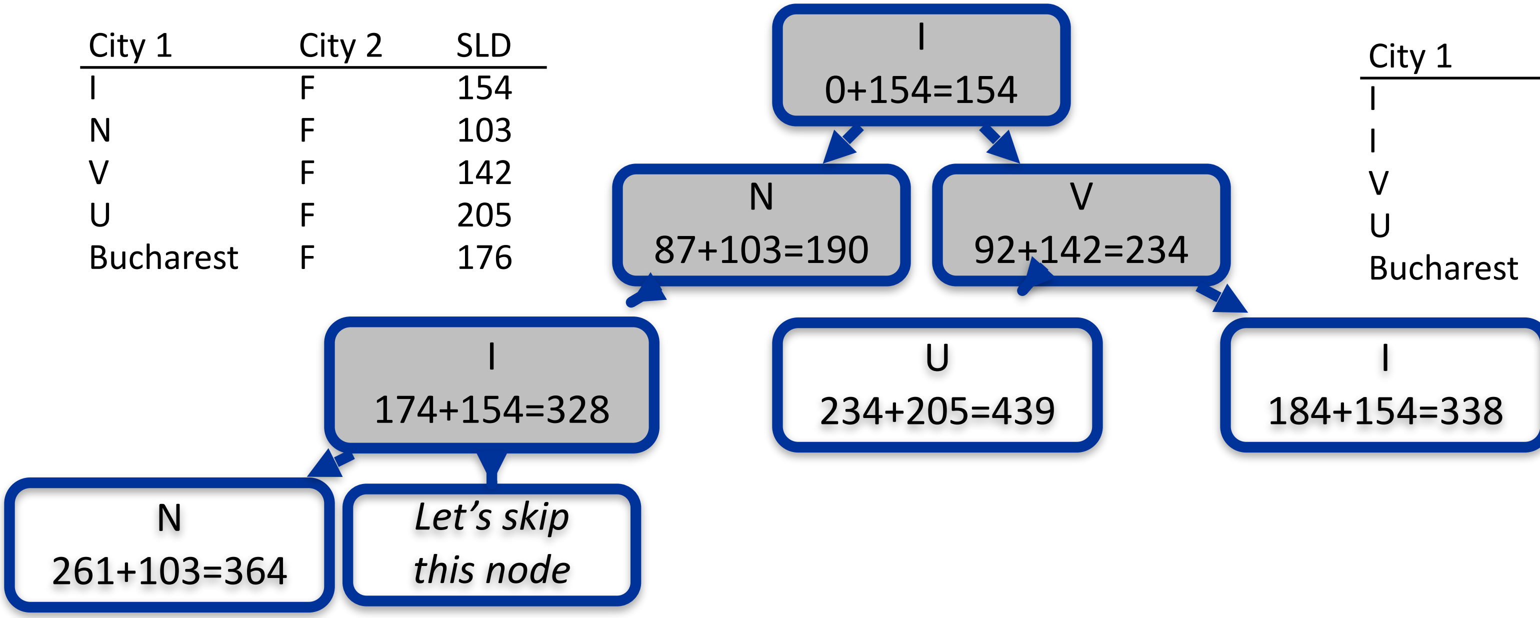
City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

Avoiding loops in A* search: Find a path from Iasi to Fagaras

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

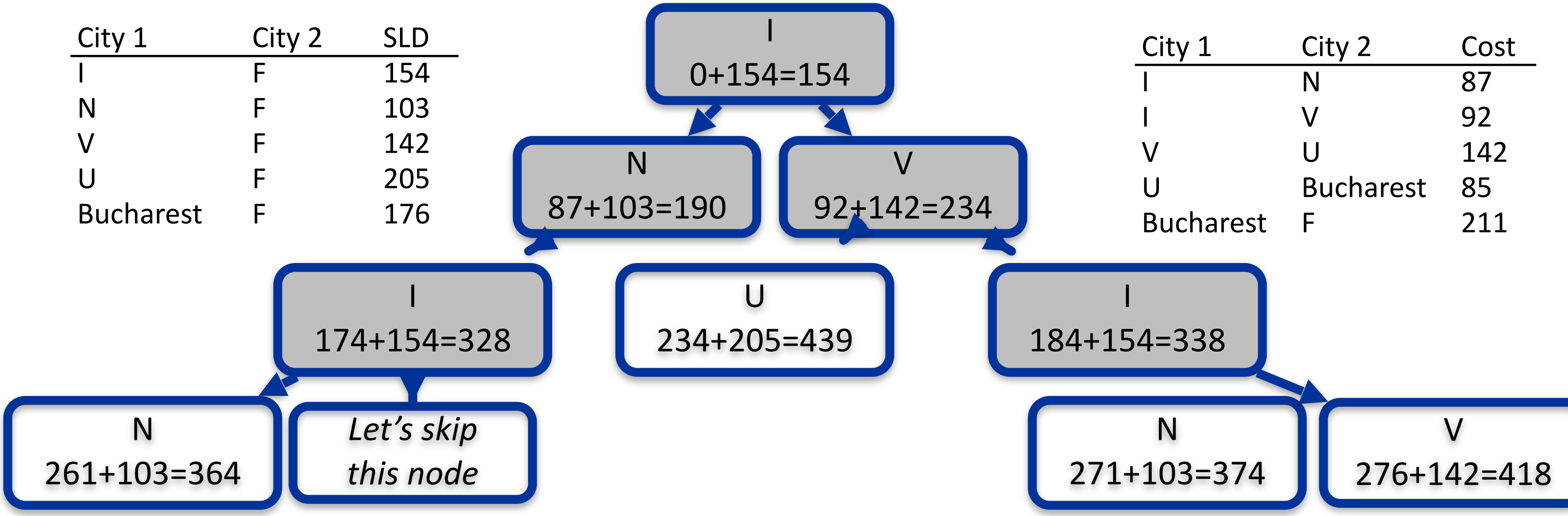
We skip irrelevant nodes to keep the visualization readable.

Avoiding loops in A* search: Find a path from Iasi to Fagaras

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

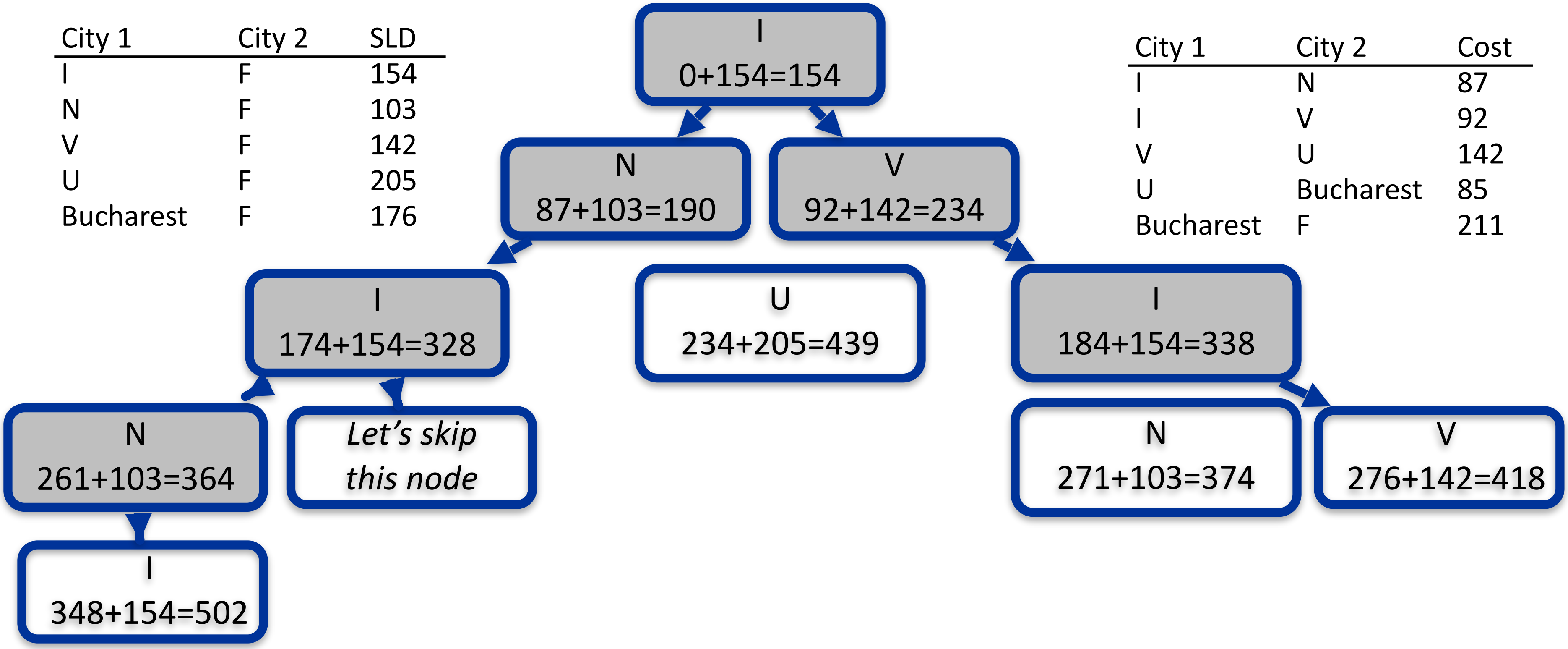
City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

Avoiding loops in A* search: Find a path from Iasi to Fagaras

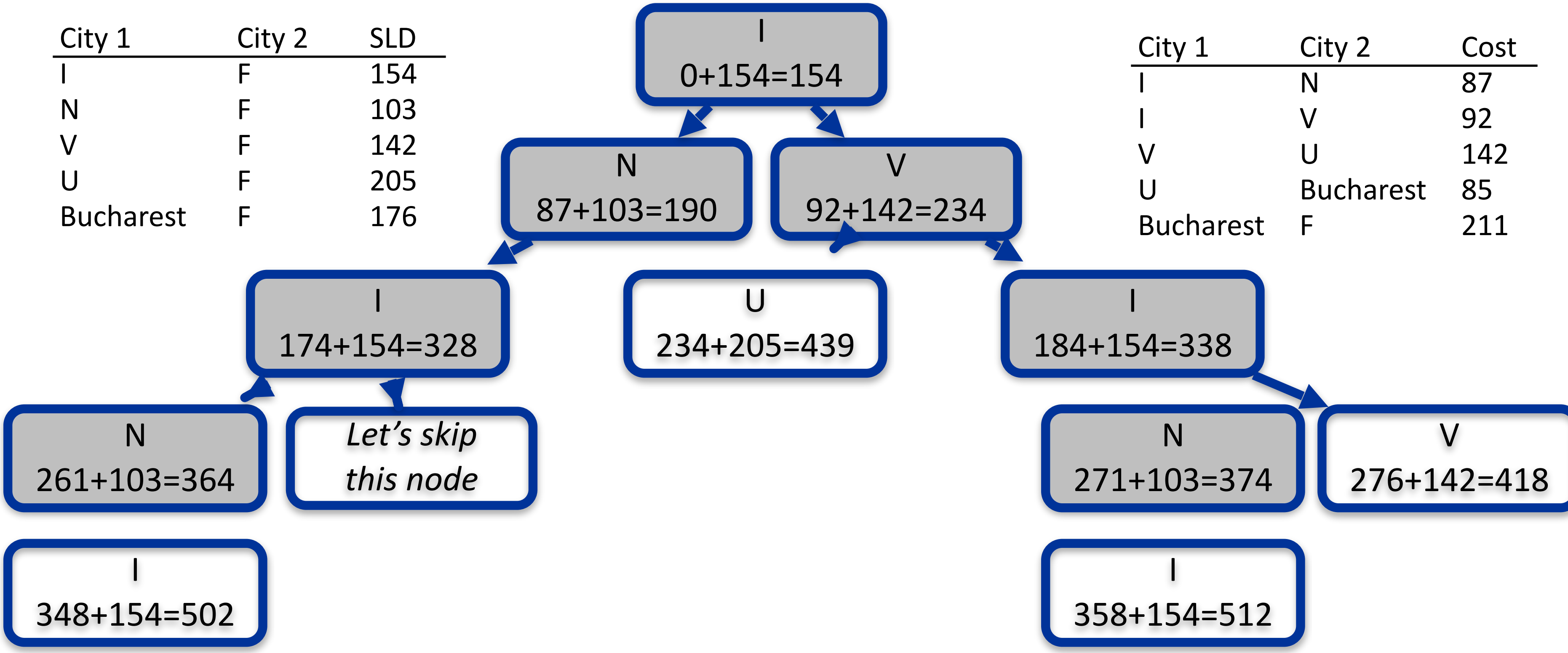
City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

Avoiding loops in A* search: Find a path from Iasi to Fagaras

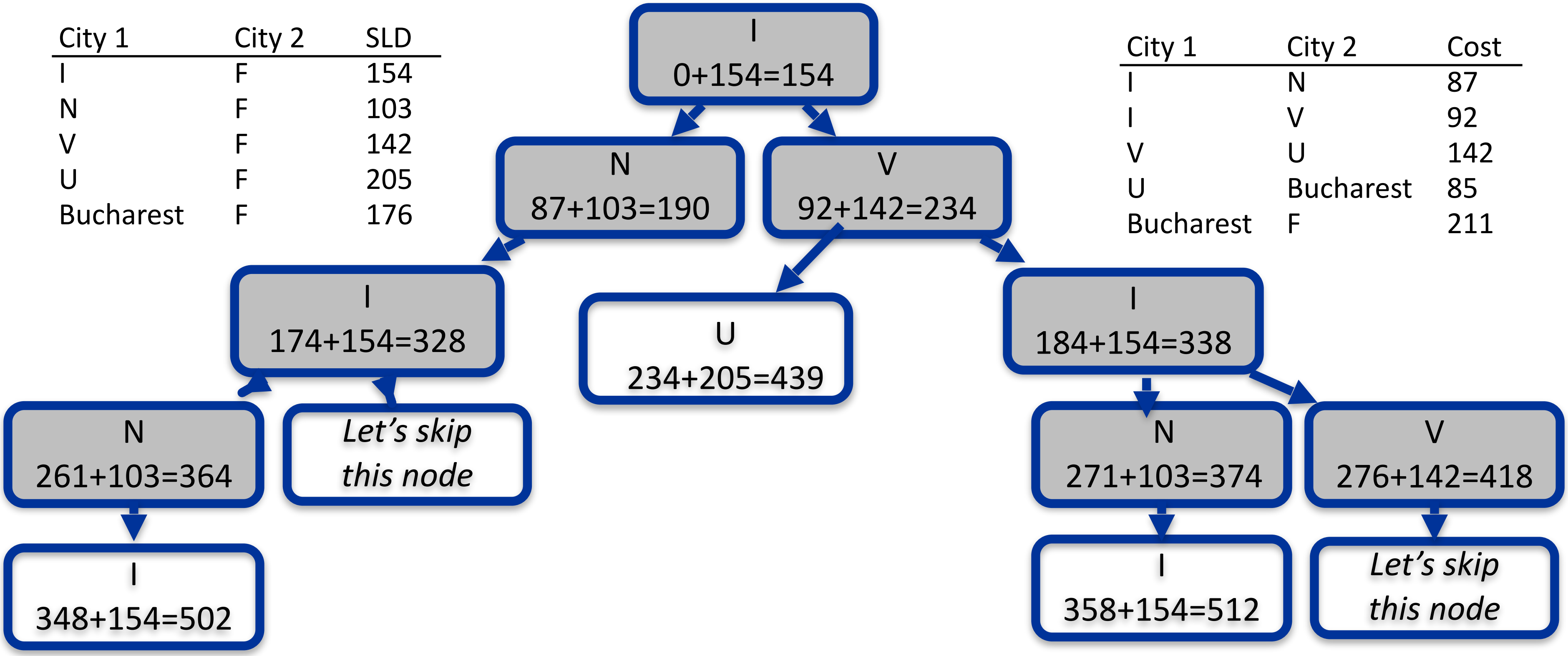
City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

Avoiding loops in A* search:
Find a path from Iasi to Fagaras

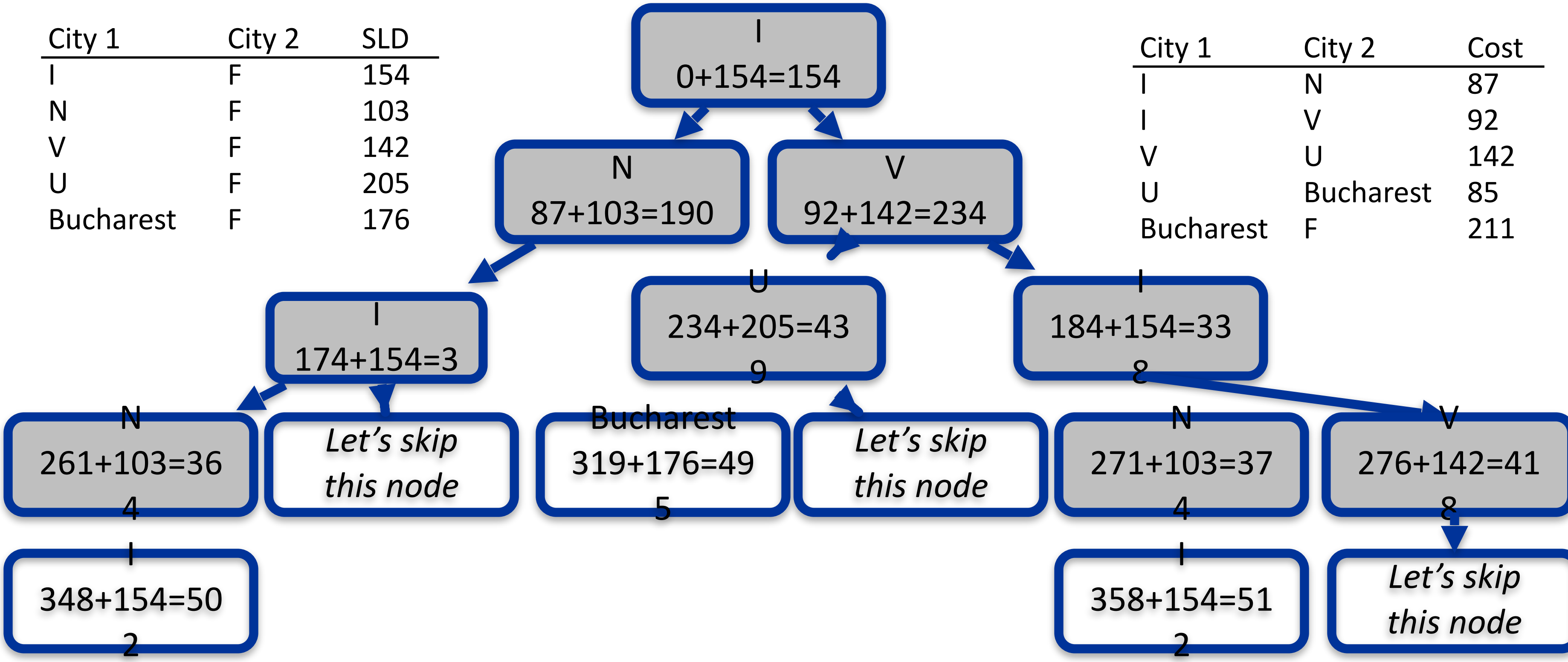
City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

Avoiding loops in A* search: Find a path from Iasi to Fagaras

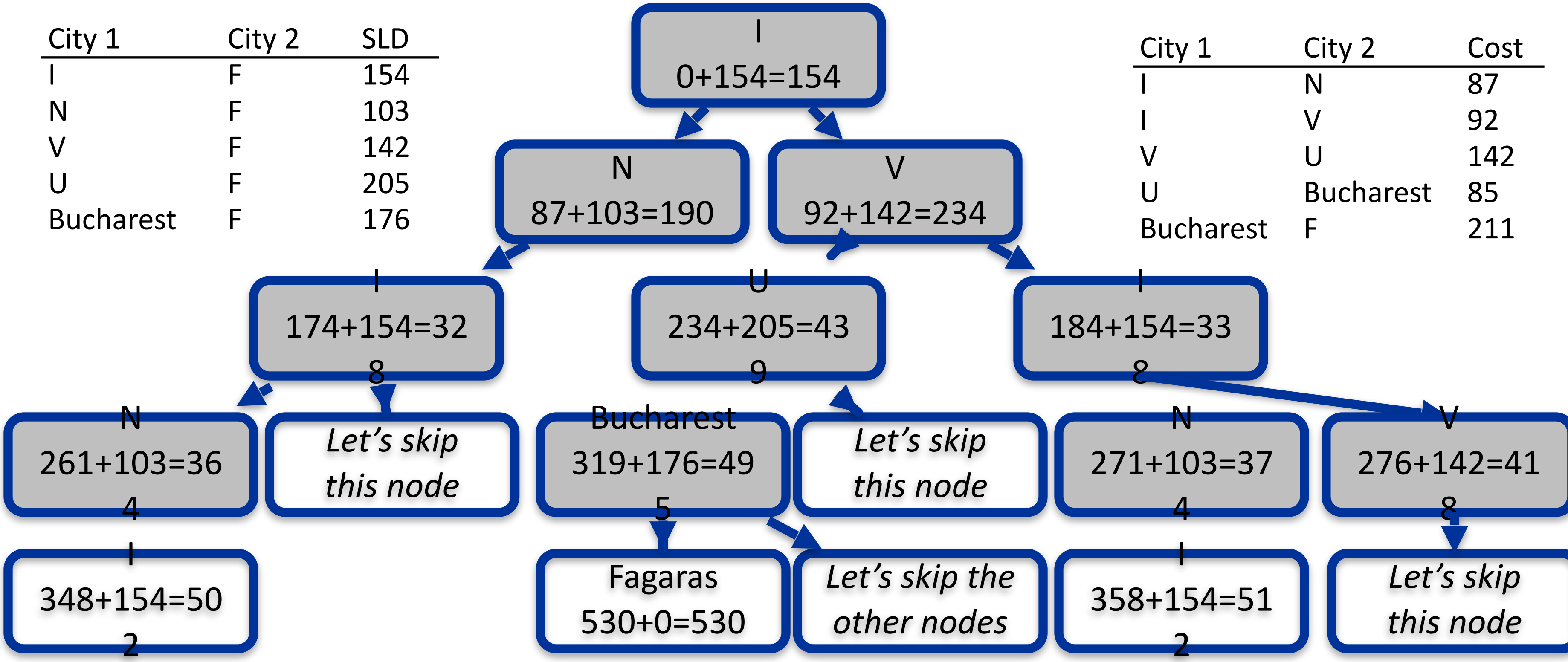
City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

Avoiding loops in A* search: Find a path from Iasi to Fagaras

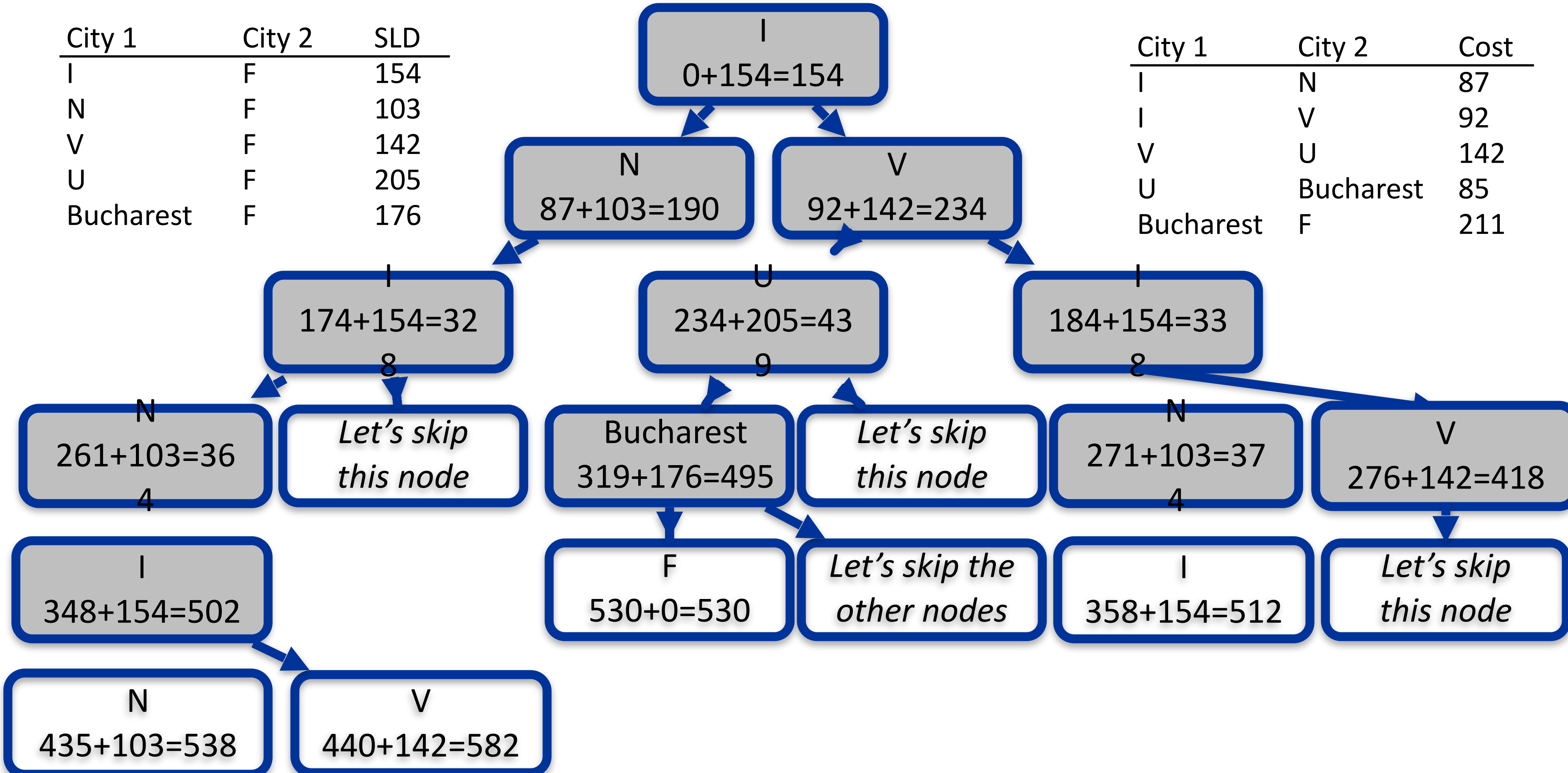
City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

Avoiding loops in A* search: Find a path from Iasi to Fagaras

City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176



City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211

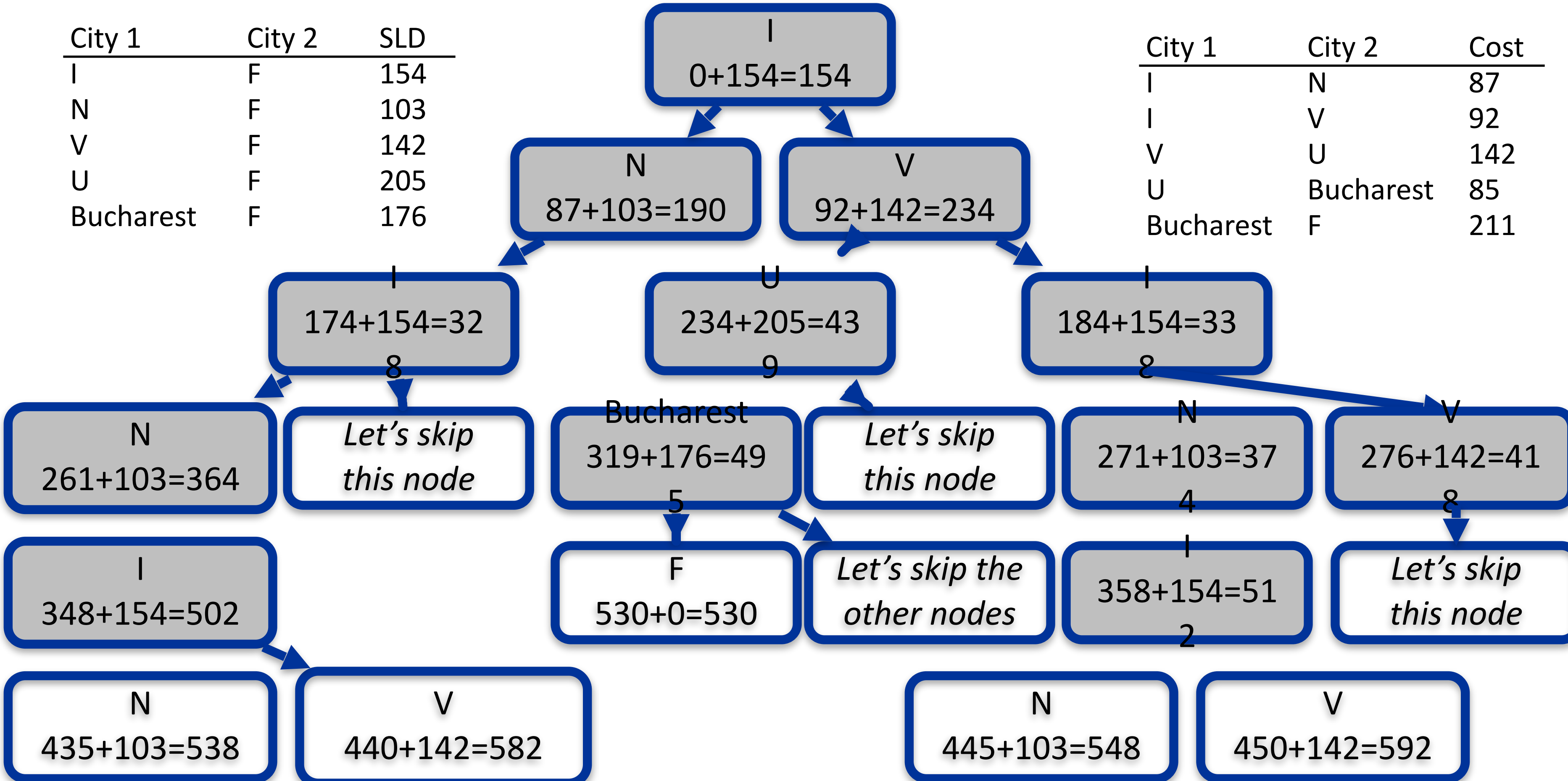
Avoiding loops in A* search: Find a path from Iasi to Fagaras

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176

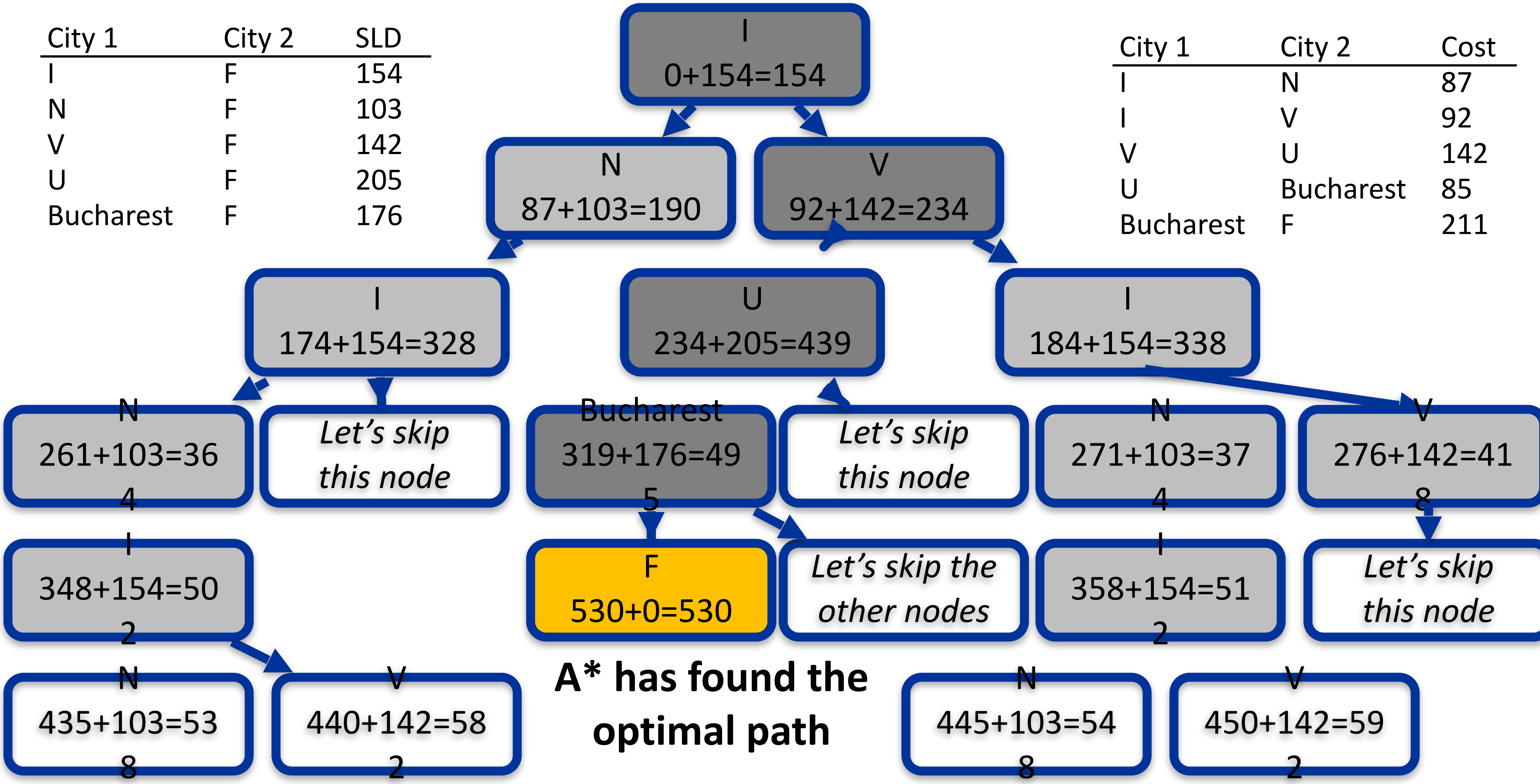
City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211



Avoiding loops in A* search: Find a path from Iasi to Fagaras

City 1	City 2	SLD
I	F	154
N	F	103
V	F	142
U	F	205
Bucharest	F	176

City 1	City 2	Cost
I	N	87
I	V	92
V	U	142
U	Bucharest	85
Bucharest	F	211



All other unexpanded nodes have a higher value

Properties of A*

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Completeness

- **Yes**, unless there are infinitely many nodes with $f(n) < f(Goal)$

Optimality

- depends **on the heuristic** $h(n)$

Time Complexity

- **the number of nodes grows exponentially** unless the error between the true cost $h^*(n)$ and the heuristic $h(n)$ grows logarithmically.

$$\left| h(n) - h^*(n) \right| \leq O(\log h^*(n))$$

Space Complexity

- **has to keep all nodes in memory, typically the main problem with A***

Addressing the space problem

- High memory consumption is the main problem with A* → **memory-bounded algorithms**

- **Iterative-deepening A***
  - like iterative deepening
  - iteratively increase limit of the utility function ($f = g + h$)

- **Recursive BFS**
  - recursive algorithm that attempts to mimic standard BFS with linear space
  - keeps track of the cost of the best alternative path available from any ancestor of the current node

- **(Simple) memory-bounded A***
  - drop the worst leaf node when memory is full

Heuristics

Properties of heuristics for searching

A heuristic $h(n)$ is **admissible**

- if it **never overestimates** the true cost $h^*(n)$

$$\forall\ n:\ h(n) \leq\ h^*(n)$$

Properties of heuristics for searching

A heuristic $h(n)$ is **admissible**

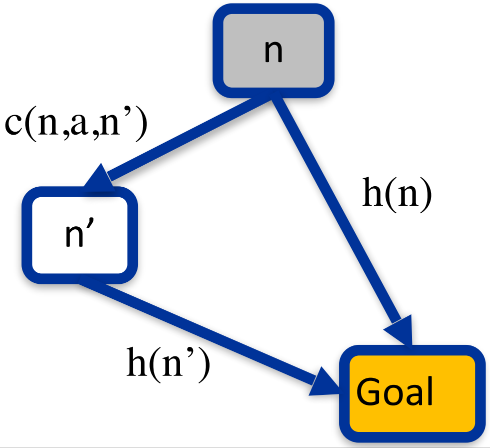
- if it never overestimates the true cost $h^*(n)$

$$\forall\ n:\ h(n) \leq\ h^*(n)$$

A heuristic $h(n)$ is **consistent**

- if **the estimated cost from** n **to the goal** is always less than the sum of **the path cost to any successor** n' **of** n and t**he estimated cost from** n' **to the goal.**

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



→ **Triangle inequality**

Properties of heuristics for searching

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

- Every **consistent** heuristic is **admissible**.
- The reverse case is not always true (it rarely occurs).

Properties of A*

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Completeness

- **Yes**, unless there are infinitely many nodes with $f(n) < f(Goal)$

Optimality

- if $h(n)$ is **consistent**

Time Complexity

- the number of nodes grows exponentially unless the error between the true cost $h^*(n)$ and the heuristic $h(n)$ grows only logarithmically.

$$\left| h(n) - h^*(n) \right| \leq O(\log h^*(n))$$

Space Complexity

- has to keep all nodes in memory, typically the main problem with A*

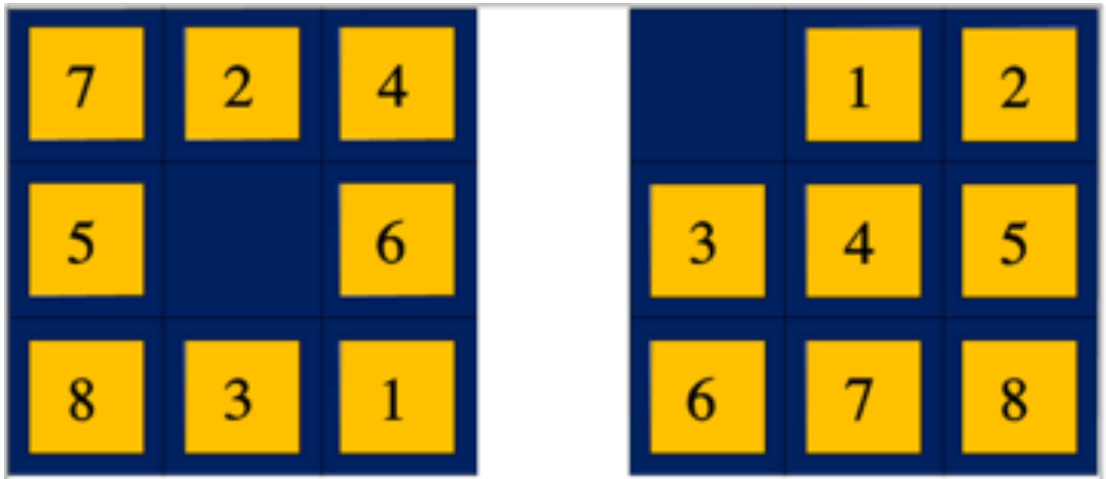
How do we choose a good heuristic?

Try to relax the problem

- A problem with a **fewer restrictions** on the **environment** is called **a relaxed problem**.

- The **cost of an optimal solution to a relaxed problem** is an **admissible heuristic** for the **original problem**.

Admissible heuristic for the 8-puzzle?



Start State Goal State

Admissible heuristic for the 8-puzzle

- **Relax** the rules of the **8-puzzle** so that **a tile can move anywhere**

→ $h_{MIS}(n)$ = **number of misplaced tiles**

- **Relax** the rules of the **8-puzzle** so that **a tile can move to any adjacent square (not only empty ones),**

→ $h_{MAN}(n)$ = **Manhattan distance of each tile to its final position (summed up)**

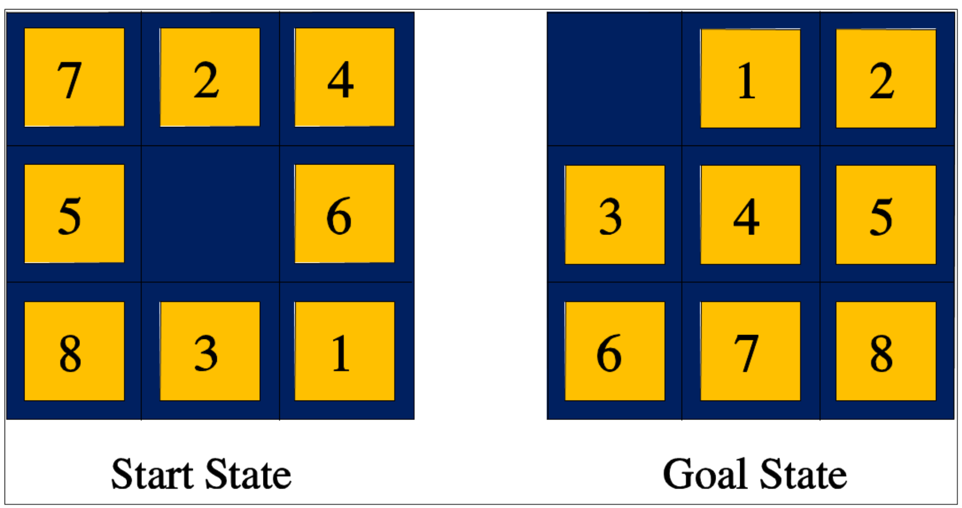
Admissible heuristics for 8-puzzle

$h_{MIS}(n)$ = number of **misplaced tiles**

- **admissible** because **each misplaced tile must be moved at least once**

$h_{MAN}(n)$ = **Manhattan distance to final position (summed up over all tiles)**

- **admissible** because this is **the minimum distance of each tile to its target square**



$h_{MIS}(start)$ = 8

$h_{MAN}(start)$ = 18

$h^*(start)$ = 26

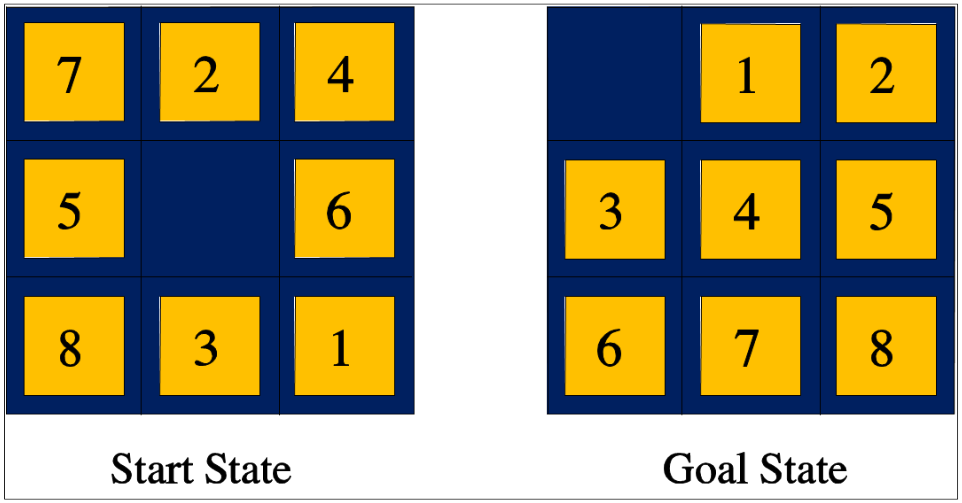
What if more than one admissible h exist

If two heuristics are **admissible, choose the one that dominates the other one.**

Consider two **admissible** heuristics h_1 and h_2 :

> h_2 **dominates** h_1 if h_2 always estimates higher or equal costs than h_1

What if more than one admissible h exist



$h_{MIS}(start)$ = 8

$h_{MAN}(start)$ = 18

$h^*(start)$ = 26

- h_{MAN} dominates h_{MIS} because both are admissible and each misplaced tile needs to be replaced at least once.

- The dominating heuristic is better because it will be closer to the true cost h^*.

Pattern databases

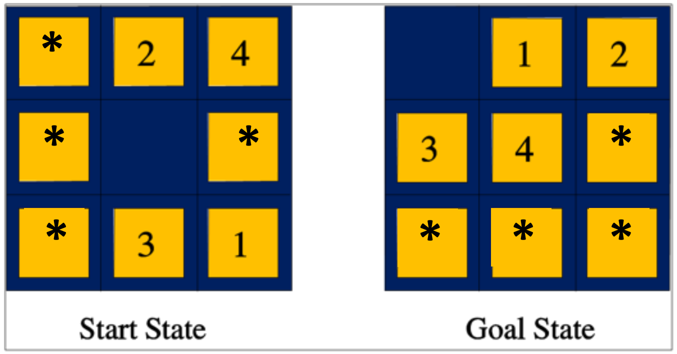
Admissible heuristics can also be derived from the solution cost of a sub-problem of a given problem

- this cost is a lower bound on the cost of the real problem

Pattern databases store the exact solution to sub-problem instances

- constructed once for all by searching backwards from the goal and recording every possible pattern

Example: store exact solution costs for solving 4 tiles of the 8-puzzle



Pattern databases for chess

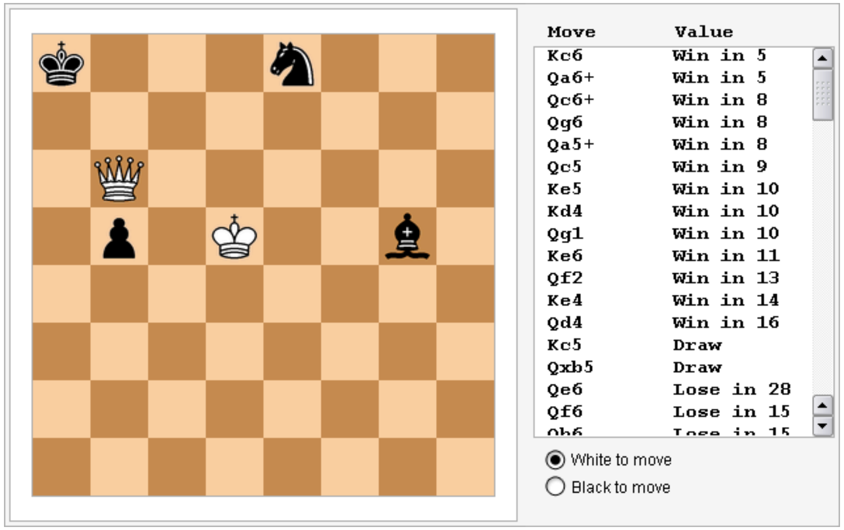
- Storing patterns from the start is infeasible.

Depth	Nodes
0	1
1	20
2	400
3	8,902
4	197,281
5	4,865,609
6	119,060,324
7	3,195,901,860
8	84,998,978,956
9	2,439,530,234,167
10	69,352,859,712,417
11	2,097,651,003,696,806
12	62,854,969,236,701,747
13	1,981,066,775,000,396,239
14	61,885,021,521,585,529,237
15	2,015,099,950,053,364,471,960

◊ Storing patterns by looking at the end?

◊ 2012: database complete for all moves for seven remaining chess pieces

◊ 140 terabyte



https://chessprogramming.wikispaces.com/Perft%20Results

https://de.m.wikipedia.org/wiki/Endspieldatenbank

Summary

- **Informed search algorithms using heuristics**
  - Greedy BFS
  - A* search
- **Heuristics**
  - Admissible
  - Consistent
  - How do we choose a good heuristic?

Readings

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Mandatory

- Russell & Norvig:
  Informed (Heuristic) Search Strategies
  3.5.1 & 3.5.2, p. 92-102

Optional

- Rest of 3.5