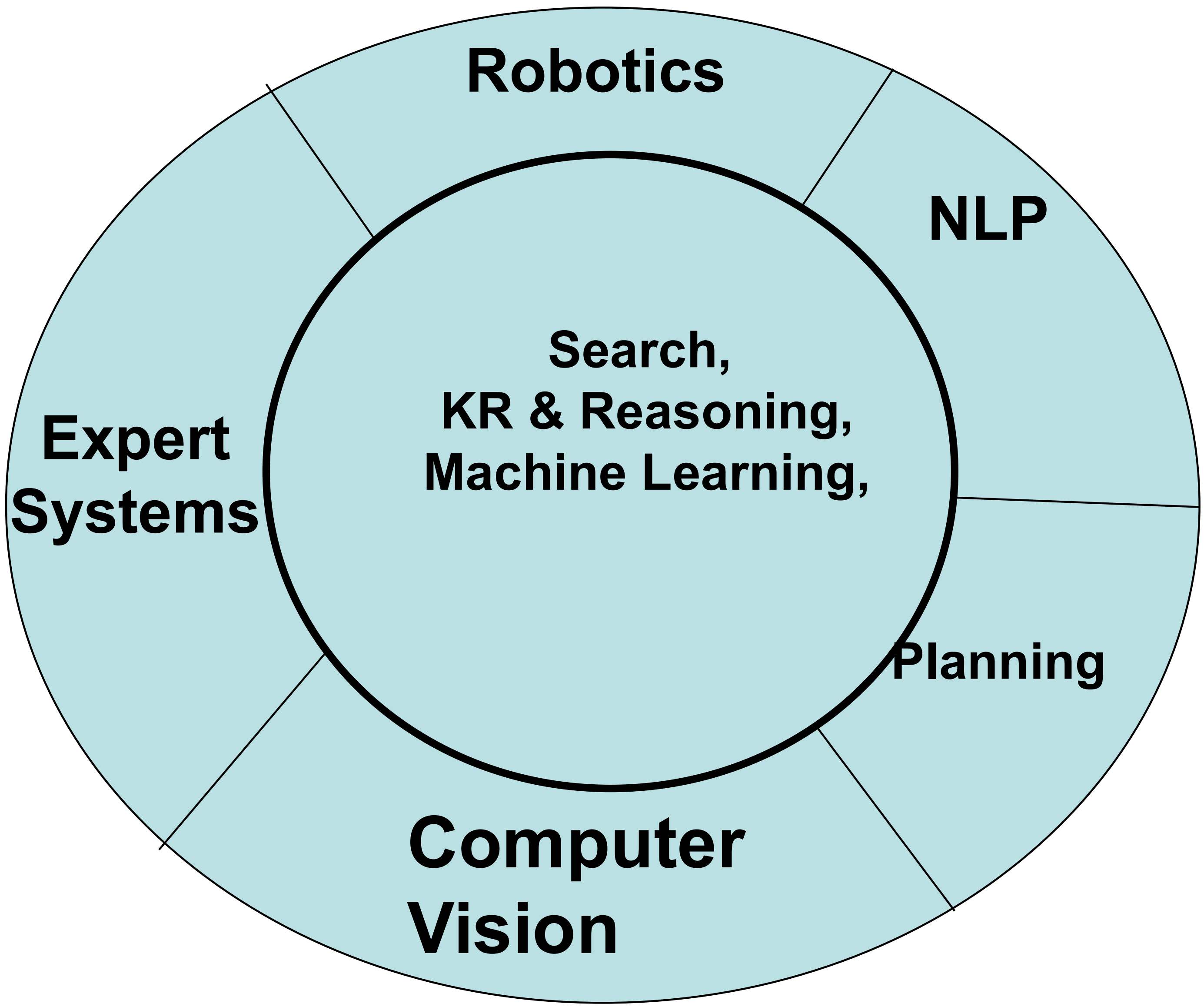
Artificial Intelligence Techniques

Informed Search

Outline

- Search Problems
- Search trees and state space graphs
- Uninformed search
  - Depth-first, Breadth-first, Uniform cost
  - Search graphs
- Informed search
  - Greedy search, A* search
  - Heuristics, admissibility
- Local search and optimization
  - Hill-climbing
  - Simulated annealing
  - Genetic algorithms

Disciplines which form the core of AI- inner circle
Fields which draw from these disciplines- outer circle.



Uninformed vs. informed search

- Uninformed search (or, blind search)
  - Uses only the information available in the problem definition
  - Searches through the space of possible solutions
  - Uses no knowledge about which path is likely to be best
  - E.g. BFS, DFS, uniform cost etc.
- Informed search (or, heuristic search)
  - Knows whether a non-goal state is more promising
  - E.g. Greedy search, A* Search etc.

What are heuristics?

Problem-specific knowledge that reduces expected search effort

Heuristic

Webster's Revised Unabridged Dictionary (1913)

Heuristic \Heu*ris"tic\, a. [Greek. to discover.] Serving to discover or find out.

The Free On-line Dictionary of Computing

heuristic 1. <programming> A rule of thumb, simplification or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood. Unlike algorithms, heuristics do not guarantee feasible solutions and are often used with no theoretical guarantee. 2. <algorithm> approximation algorithm.

From WordNet (r) 1.6

heuristic adj 1: (computer science) relating to or using a heuristic rule 2: of or relating to a general formulation that serves to guide investigation [ant: algorithmic] n : a commonsense rule (or set of rules) intended to increase the probability of solving some problem [syn: heuristic rule, heuristic program]

Heuristics (some examples)

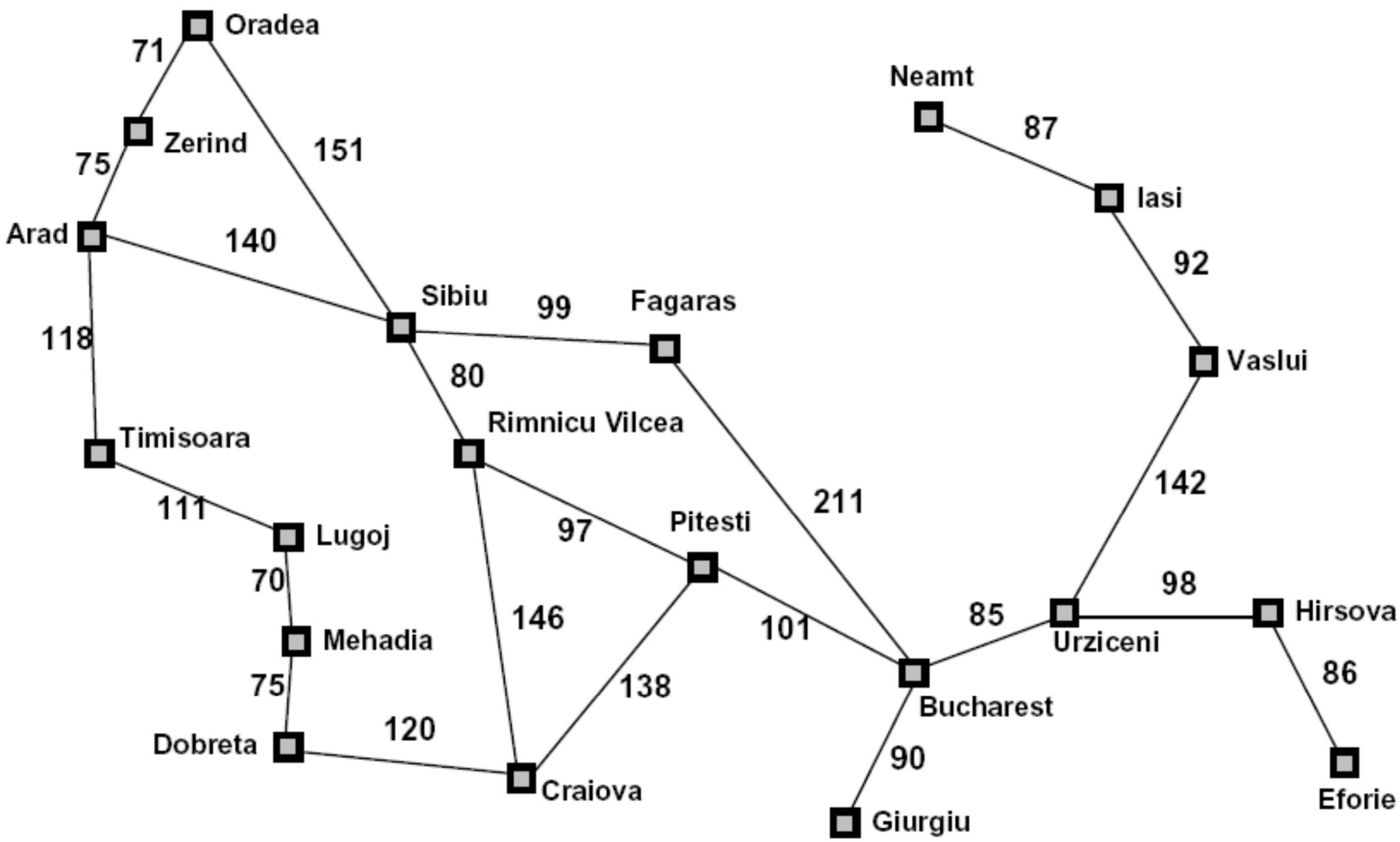
- Travel planning
  - Euclidean distance
- 8-puzzle
  - Manhattan distance (how far the goal is?)
  - No. of misplaced tiles
- Travelling salesman problem
  - Minimum spanning tree

Best-first search

- An instance of the general tree search
- Idea: use an evaluation function *f(n)* for each node
  - Estimate of "desirability“
  - Expand most desirable unexpanded node, i.e. node with lowest f(n)
- Implementation:
  - Priority queue: Order the nodes in fringe in ascending order of *f-values*

- Key component: heuristic function (*h(n)*)
  - h(n)= estimated cost of the cheapest path from node n to the goal
  - Most common form in which additional knowledge of the problem is imparted
- Special cases:
  - greedy best-first search
  - A^* search

Heuristics



Straight−line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search

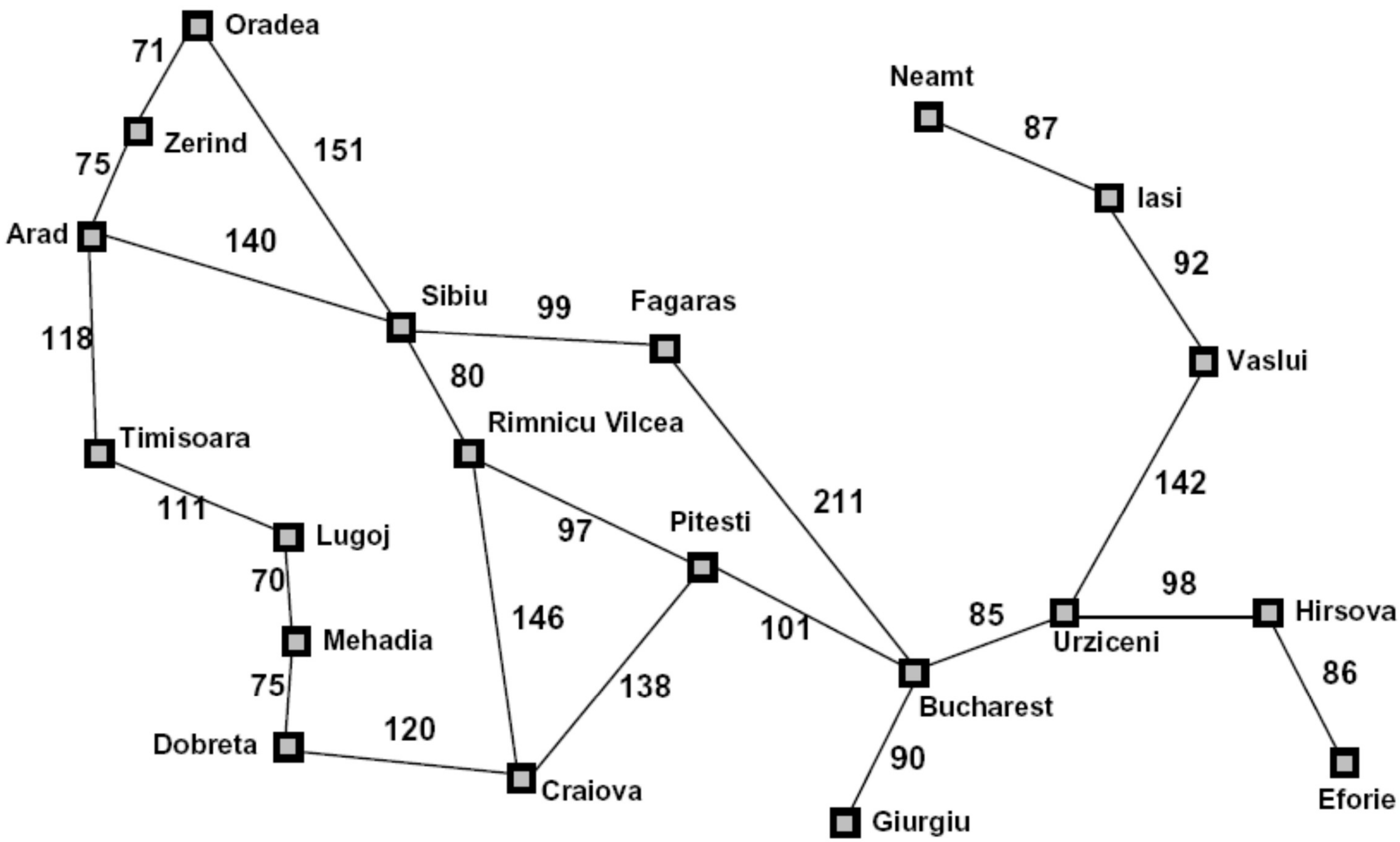
- Greedy best-first search expands the node that appears to be closest to goal
- Finds the solution quickly

- Evaluation function $f(n) = h(n)$ (heuristic)

= estimate of cost from n to *goal*

e.g., $h_{SLD}(n)$ = straight-line distance in the route-finding problem

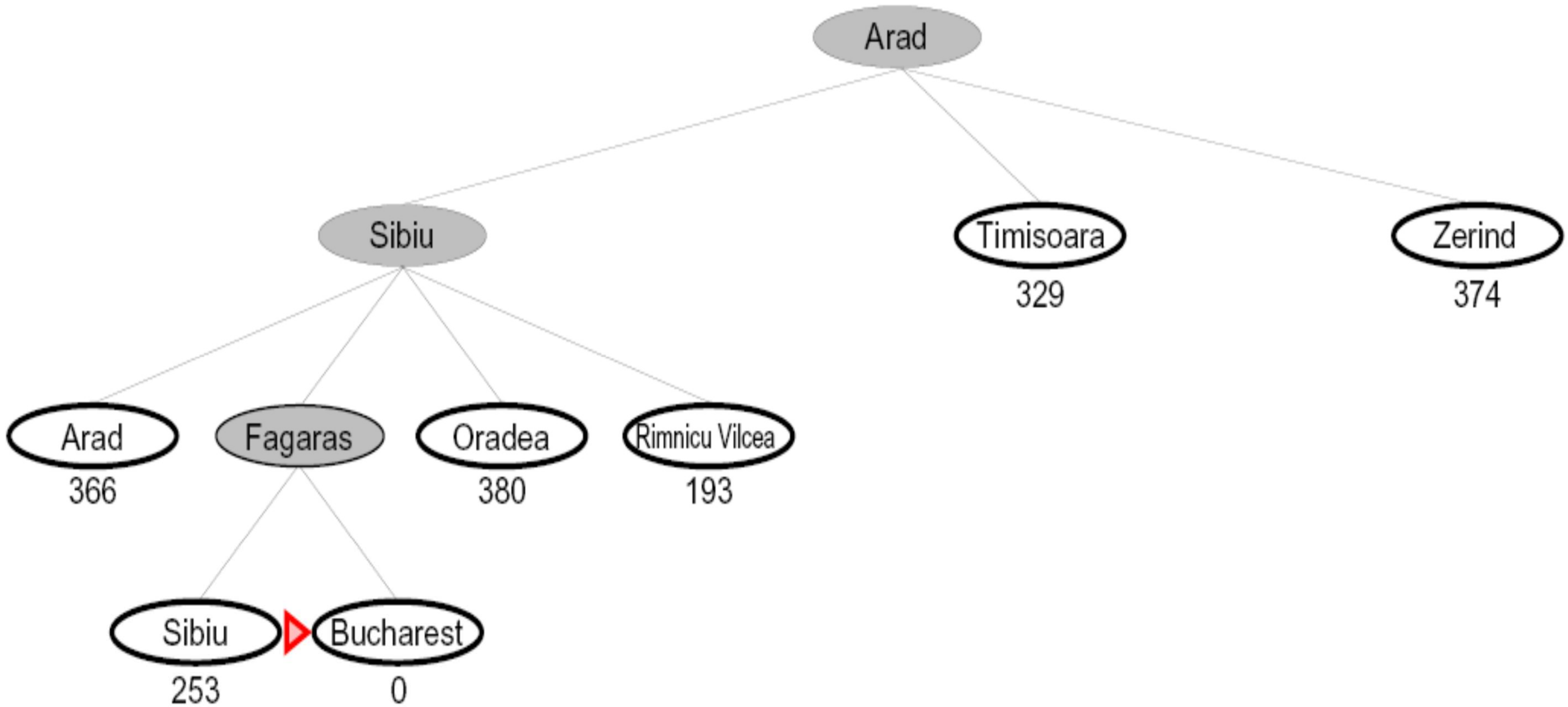
Heuristics



Straight−line distance to Bucharest

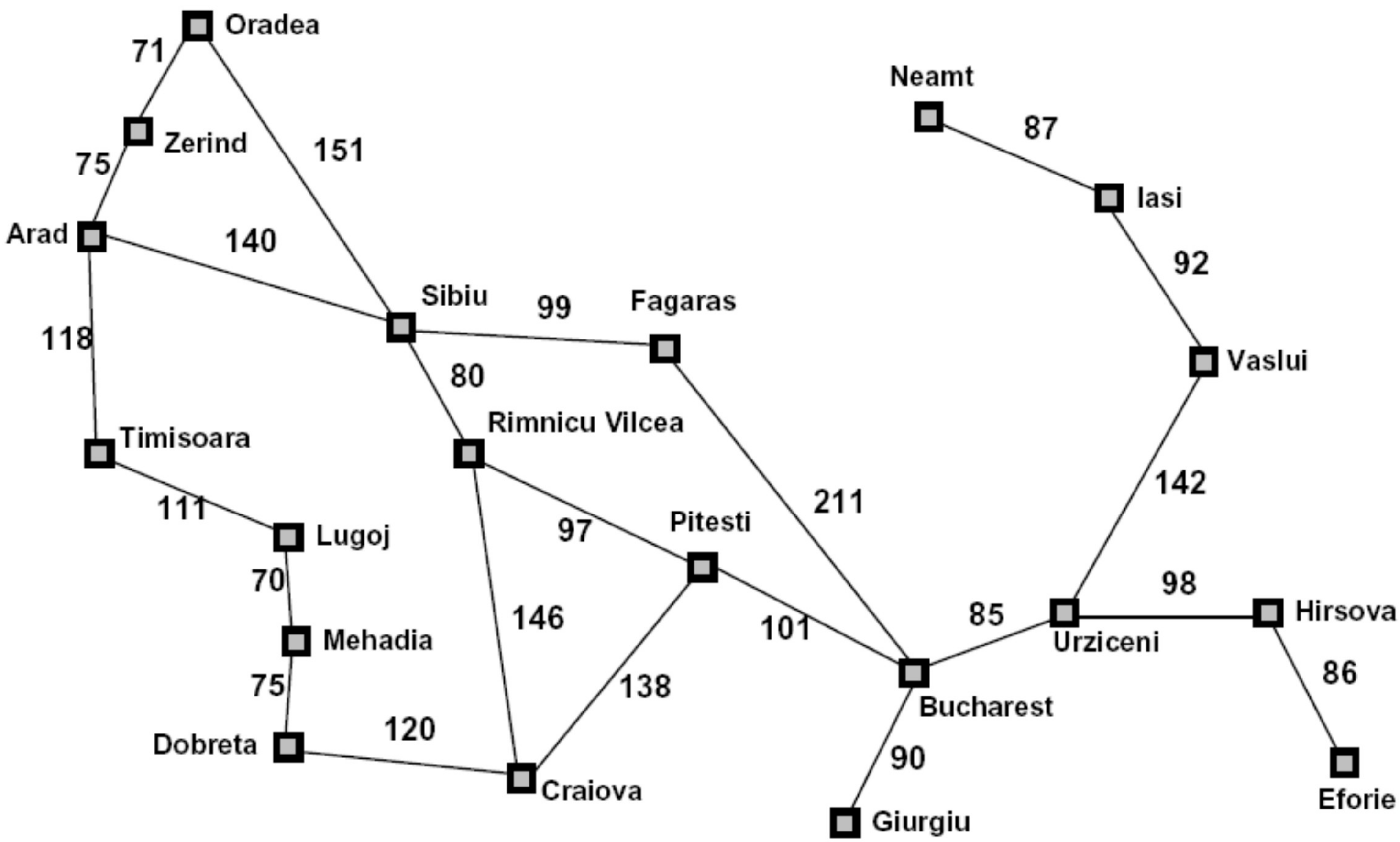
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best First Search



- What can go wrong?

Heuristics



Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best First Search

- A common case:
  - Can guide to non-optimal solution
  - Path via *Sibiu → Fagars →Bucharest* is 32 km longer than through *Rimnicu Vilcea→Pitesi*

    (99+211)-(80+97+101)=32
- Minimizing h(n) is susceptible to false starts (*unnecessary nodes are expanded*)

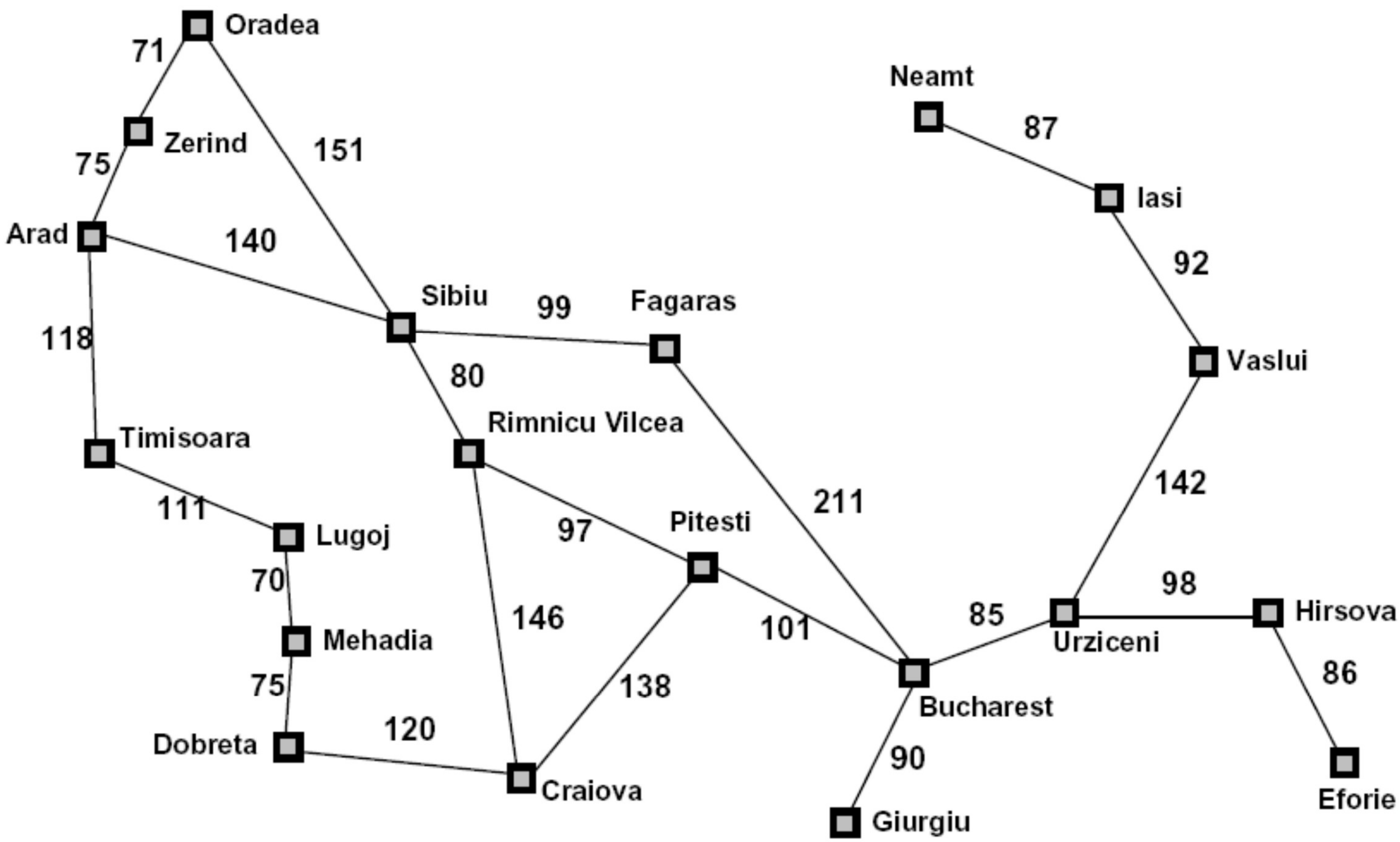
  E.g. Iasi→Fagras

  According to heuristic, Neamt is expanded but it is dead end!

  Solution: Vaslui (farther from goal)→Urziceni→Bucharest→Fagras

- Worst-case: like a badly-guided DFS
  - Can explore everything
  - Can get stuck in loops if no cycle checking
    - e.g., Iasi → Neamt → Iasi → Neamt
  - Can go infinite path and never try other possibilities

Heuristics

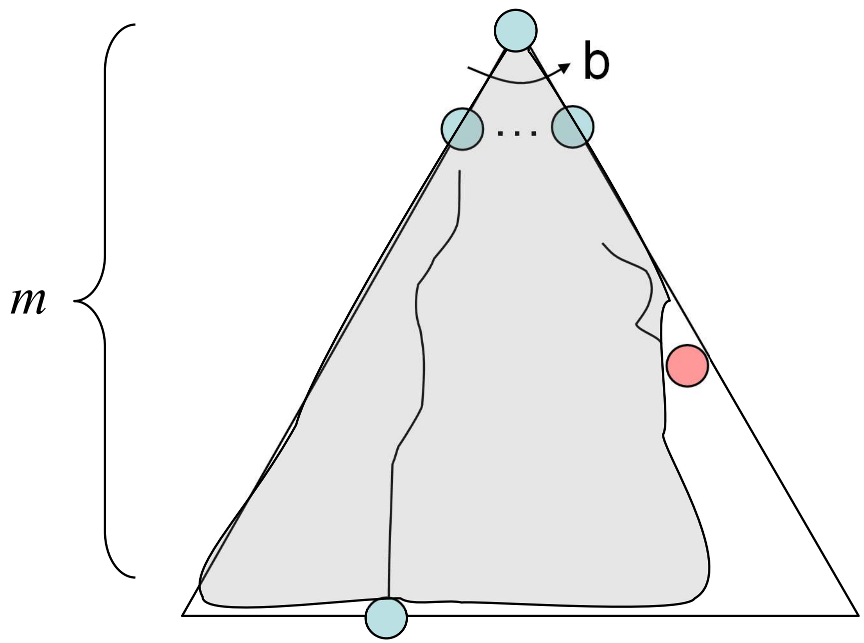


Straight−line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Best First Greedy Search

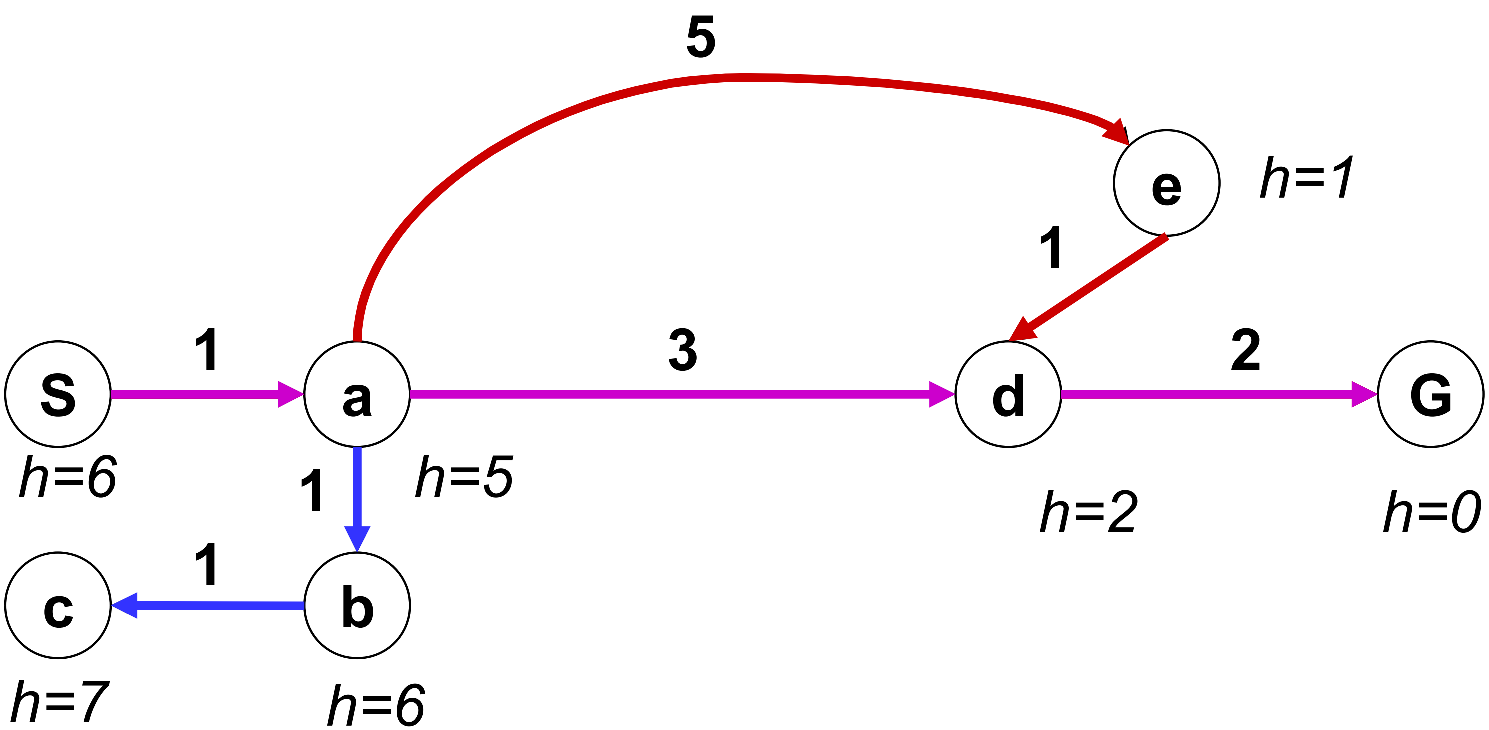
Algorithm	Complete	Optimal	Time	Space
Greedy Best-First Search	Y*	N	$O(b^m)$	$O(b^m)$



- What do we need to do to make it complete? (*repeated cycle check*)
- Can we make it optimal?

Combining UCS and Greedy

- Uniform-cost orders by path cost, or *backward cost* **g(n)**
- Best-first orders by distance to goal, or *forward cost* **h(n)**



- A* Search orders by the sum: **f(n) = g(n) + h(n)**

Algorithm A

- A function f is maintained with each node

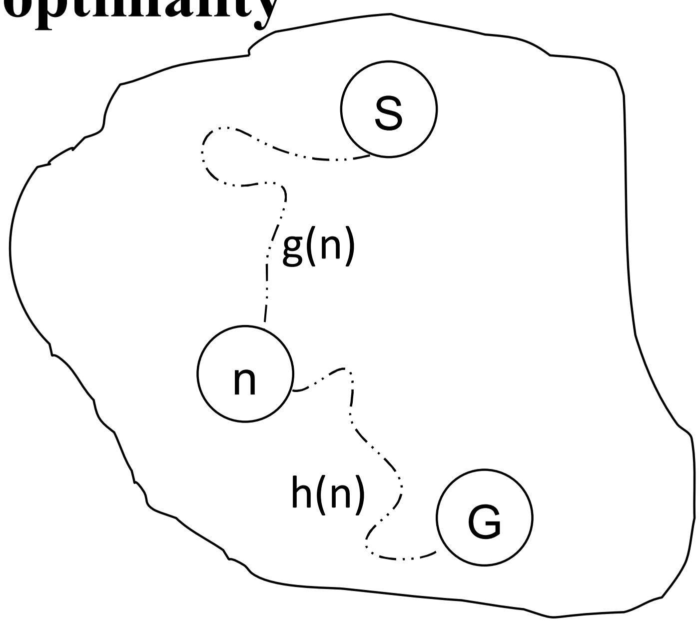
  $f(n) = g(n) + h(n)$, n is the node in the open list

- Node chosen for expansion is the one with least f value
- For BFS: $h = 0$, g = number of edges in the path to S
- For DFS: $h = 0$, $g = \dfrac{1}{\text{No of edges in the path to S}}$

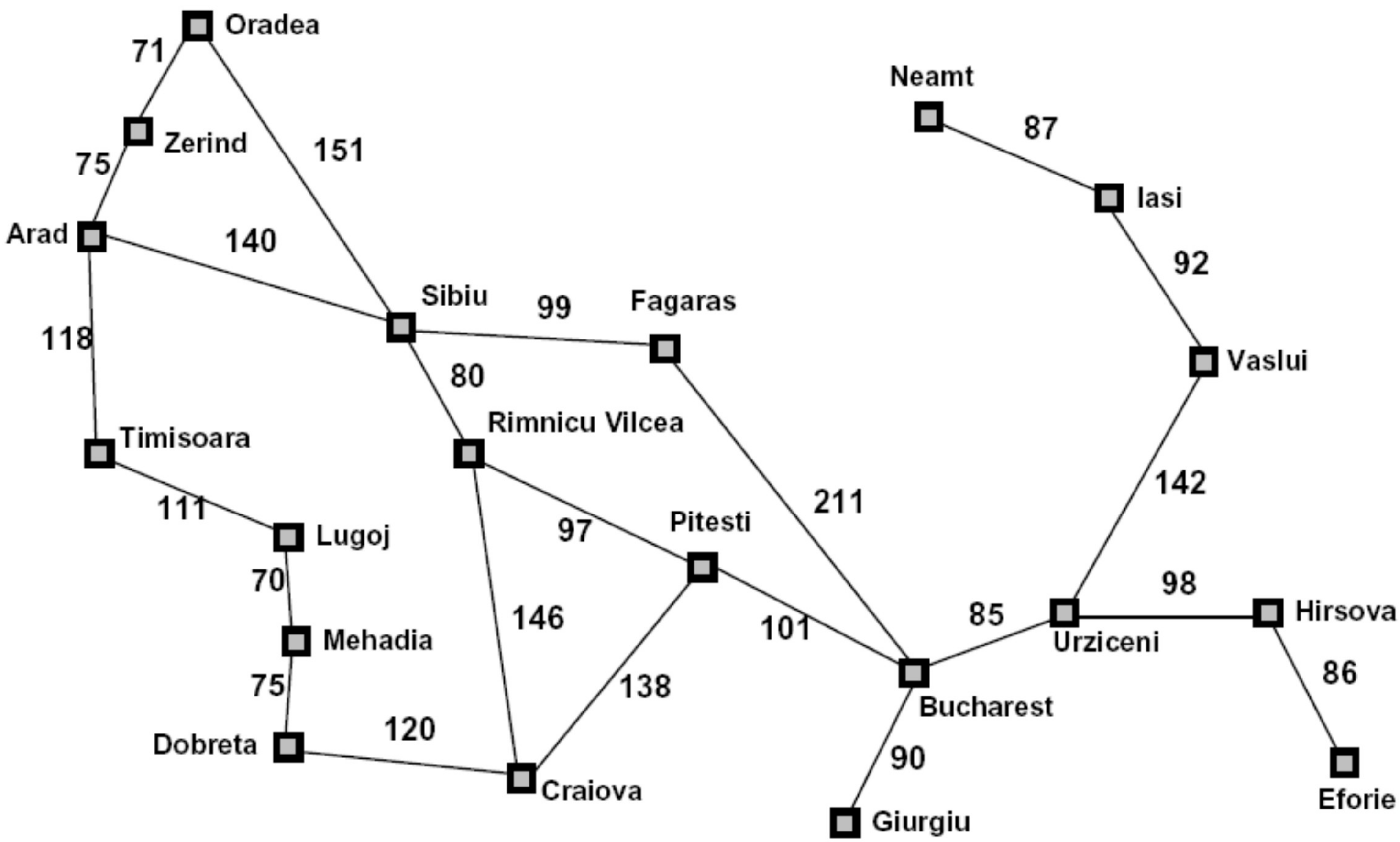
Algorithm A*

- One of the most important advances in AI
  - Idea: *avoid expanding paths that are already expensive*
- $g(n)$ = least cost path to n from S found so far
- $h(n) <= h^*(n)$ where $h^*(n)$ is the actual cost of optimal path to G(node to be found) from n

"Optimism leads to optimality"



Heuristics

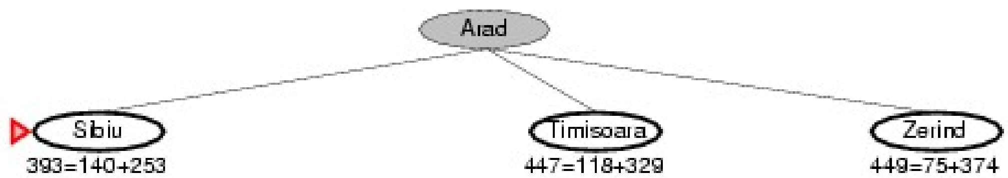


Straight−line distance to Bucharest

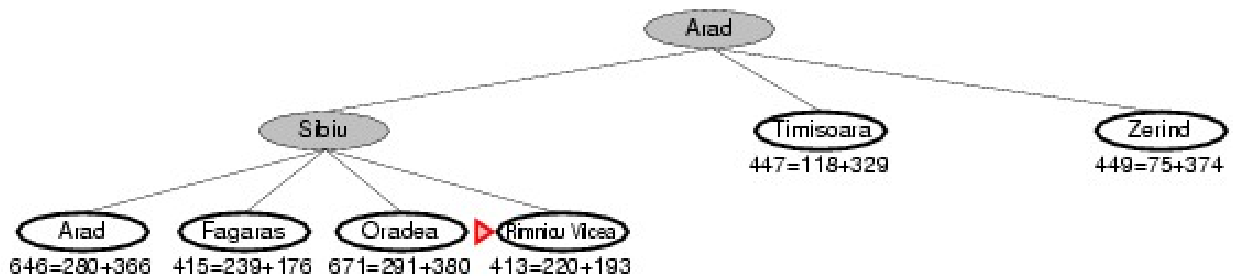
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

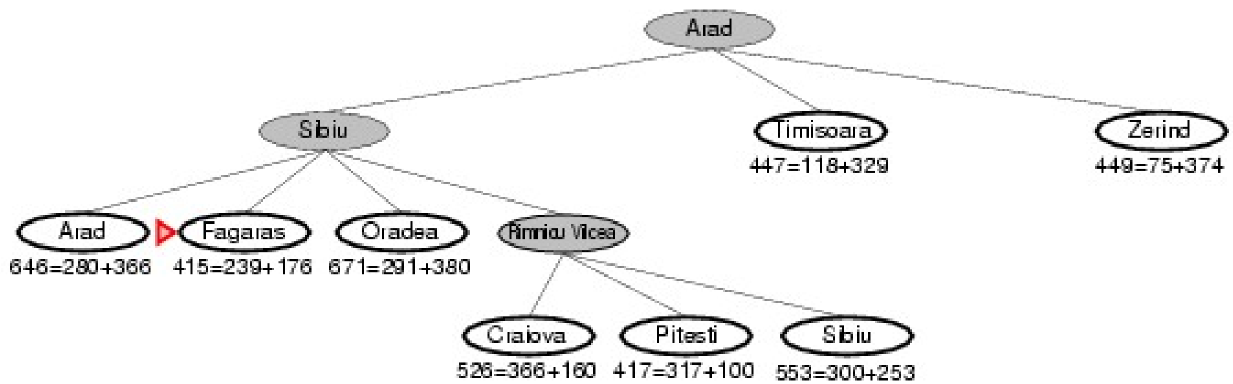
A* search example



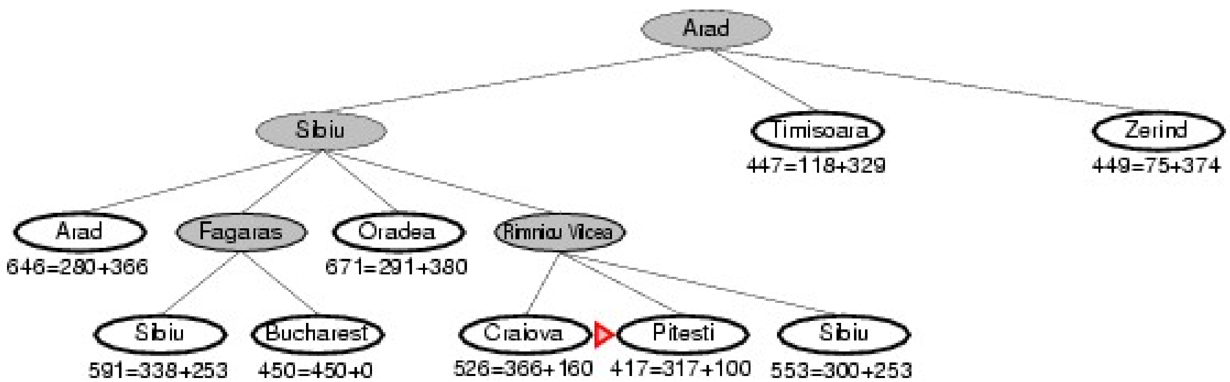
A* search example



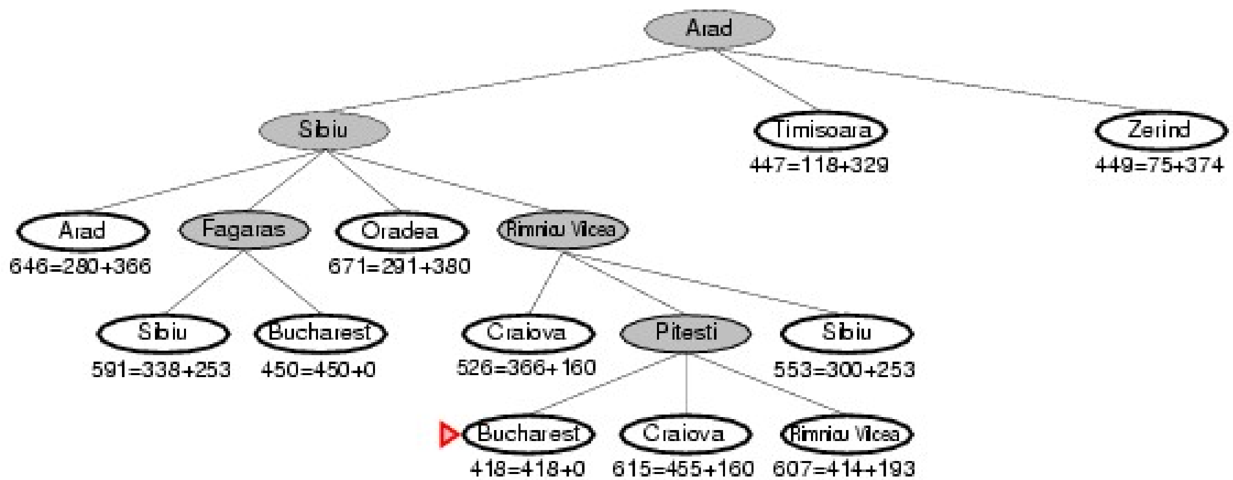
A* search example



A* search example

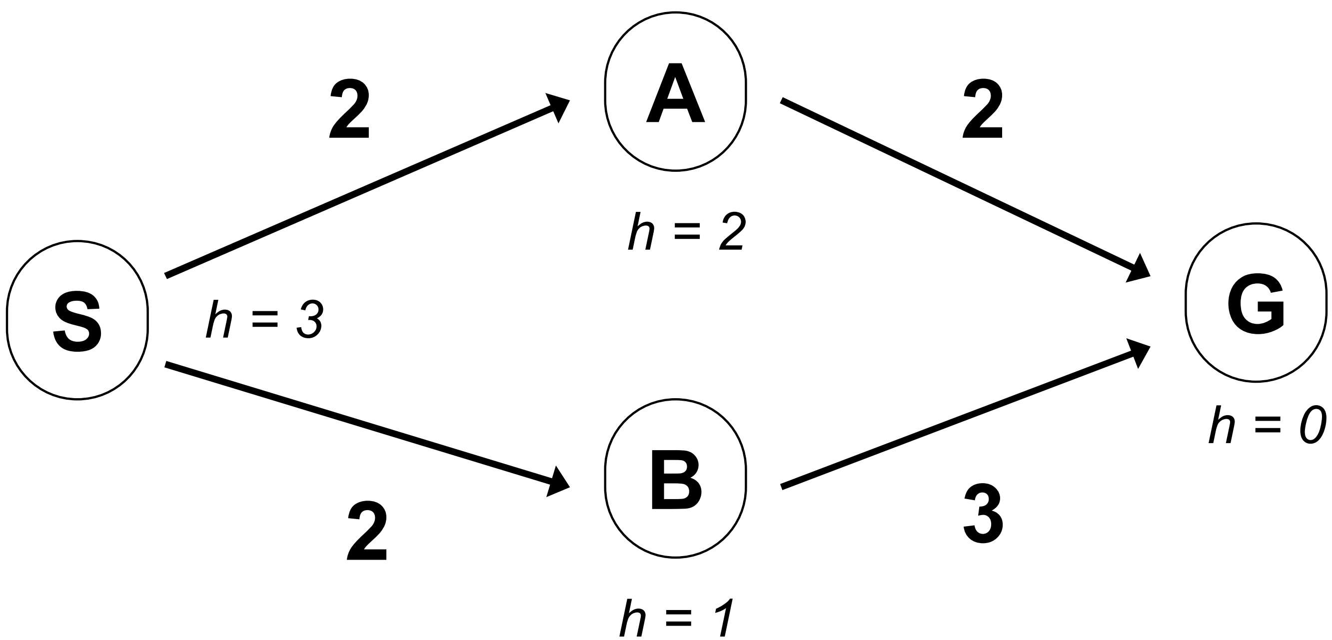


A* search example



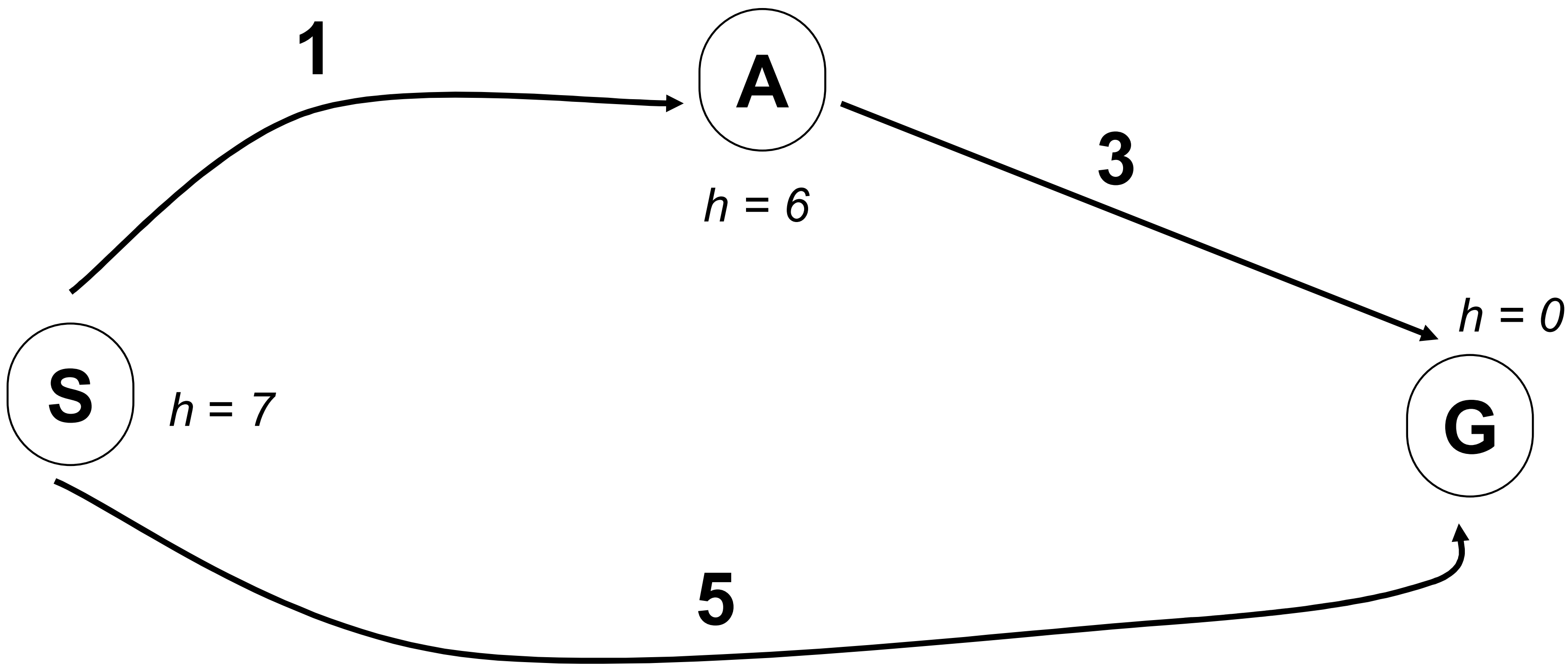
When should A* terminate?

- Should we stop when we enqueue a goal?



- No: only stop when we dequeue a goal

Is A* Optimal?



- What went wrong?
- Actual bad path cost (5) < estimated good path cost (1+6)
- We need estimates (h=7) to be less than actual (5) costs!

Admissible Heuristics

- A heuristic h is *admissible* (optimistic) if:

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

where $h^*(n)$ is the true cost to a nearest goal

Never overestimate!

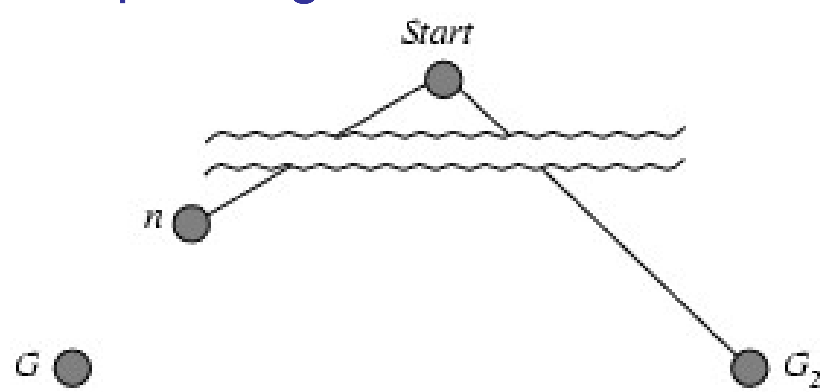
Optimistic: *cost of solution is less than the actual cost!*

Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

Theorem: If h(n) is admissible, A^* using TREE-SEARCH is optimal

Optimality of A* (proof)

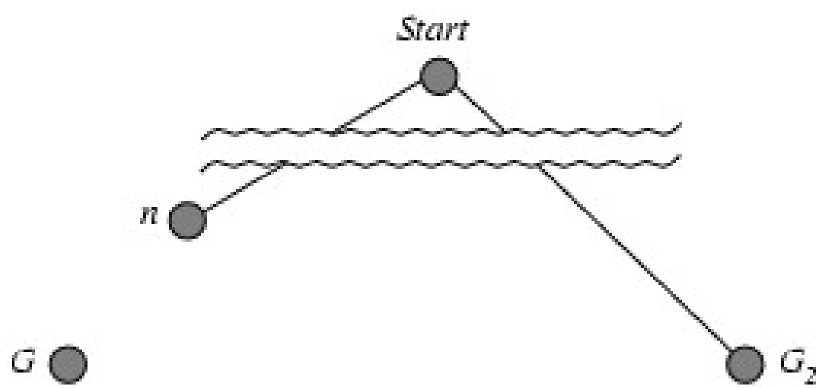
- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on a shortest path to an optimal goal G



- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $g(G_2) > g(G)$ since G_2 is suboptimal
- $f(G) = g(G)$ since $h(G) = 0$
- $f(G_2) > f(G)$ from above

Optimality of A* (proof)

- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let *n* be an unexpanded node in the fringe such that *n* is on a shortest path to an optimal goal *G*



- f(n)=g(n)+h(n) ≤ f(G) since h is admissible

Hence *f(G₂) > f(n),* and A* will never select G_2 for expansion

BUT … graph search

- Discards new paths to repeated state
  - Previous proof breaks down
  - First one may not lead to the optimal one and so can be a problem

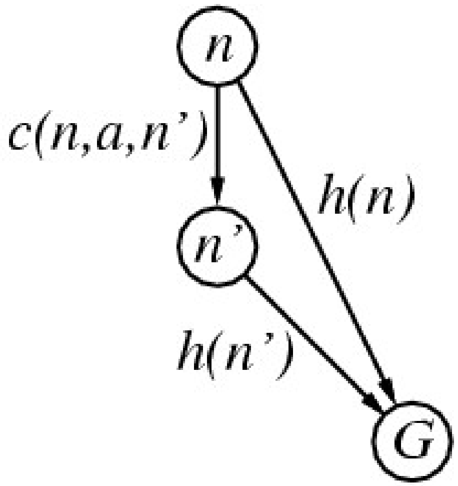
- Solution:
  - Add extra bookkeeping i.e. remove more expensive of two paths
  - Ensure that optimal path to any repeated state is always first followed
    - Extra requirement on *h(n)*: consistency (*monotonicity*)

Consistent heuristics

- A heuristic is consistent (or, monotonic) if for every node *n*, every successor *n'* of *n* generated by any action *a*,

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

  -*general form of triangle inequality*
  -*straight line distance*



If *h* is consistent, we have

$$
\begin{aligned}
f(n') &= g(n') + h(n') \\
&= g(n) + c(n,a,n') + h(n') \\
&\geq g(n) + h(n) \\
&= f(n)
\end{aligned}
$$

i.e., *f(n)* is non-decreasing along any path (*so the first goal node expanded is always least expensive*)

- Theorem: If *h(n)* is consistent, A* using `GRAPH-SEARCH` is optimal

A* search (properties)

- Completeness: YES
  - Since bands of increasing *f* are added
  - Unless there are infinitely many nodes with $f<f(G)$

A* search (properties)

- Completeness: YES
- Time complexity:
  - Number of nodes expanded is still exponential in the length of the solution

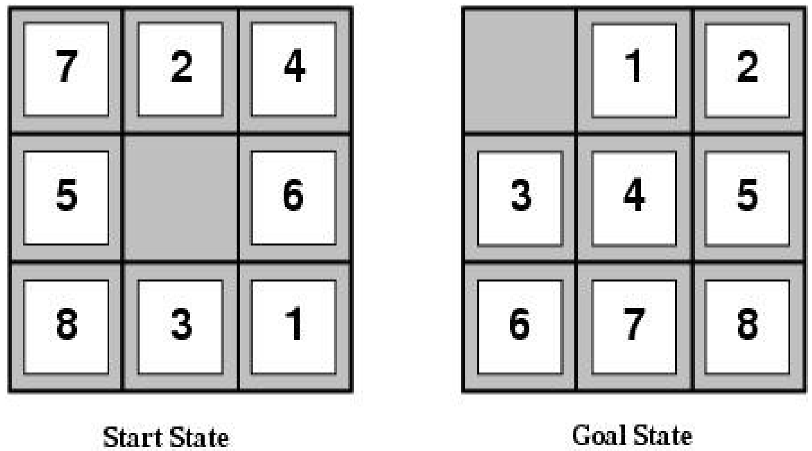
A* search (properties)

- Completeness: YES
- Time complexity: (exponential with path length)
- Space complexity:
  - It keeps all generated nodes in memory
  - Hence space is the major problem not time

A* search (properties)

- Completeness: YES
- Time complexity: exponential with path length
- Space complexity: all nodes are stored
- Optimality: YES
  - Cannot expand f_{i+1} until f_i is finished.
  - A* expands all nodes with $f(n) < C^*$
  - A* expands some nodes with $f(n)=C^*$
  - A* expands no nodes with $f(n)>C^*$
- *Optimally efficient*
  - *No other algorithm guaranteed to expand fewer nodes than A**

Heuristic functions



- E.g for the 8-puzzle
  - Avg. solution cost is about 22 steps (branching factor +/- 3)
  - Exhaustive search to depth 22: 3^{22} (3.1×10^{10}) states (*for tree*)
  - 9!/2 (=181,440) states are actually reachable (*for graph*)
  - A good heuristic function can reduce the search process

Heuristic Function (Admissible heuristic)

- Function h(N) that estimates the cost of the cheapest path from node N to goal node.
- Example: 8-puzzle

5		8
4	2	1
7	3	6

N

1	2	3
4	5	6
7	8	

goal

$h_1(N)$ = number of misplaced tiles
= 6

Heuristic Function (Admissible heuristic)

- Function h(N) that estimate the cost of the cheapest path from node N to goal node.
- Example: 8-puzzle

5		8
4	2	1
7	3	6

N

1	2	3
4	5	6
7	8	

goal

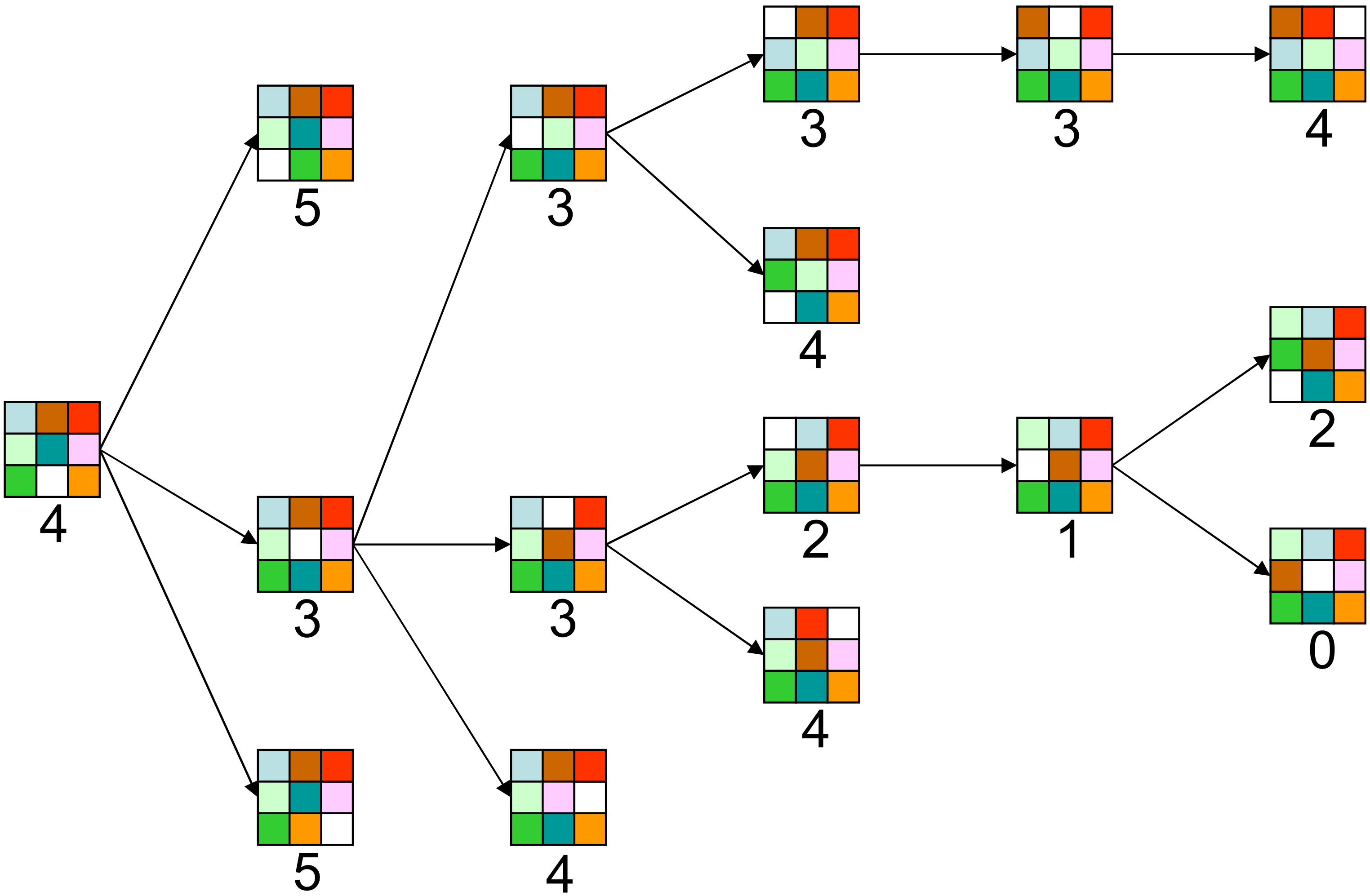
total Manhattan distance $=h_2(N)$ = sum of the distances of every tile to its goal position

$= 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1$

$= 13$

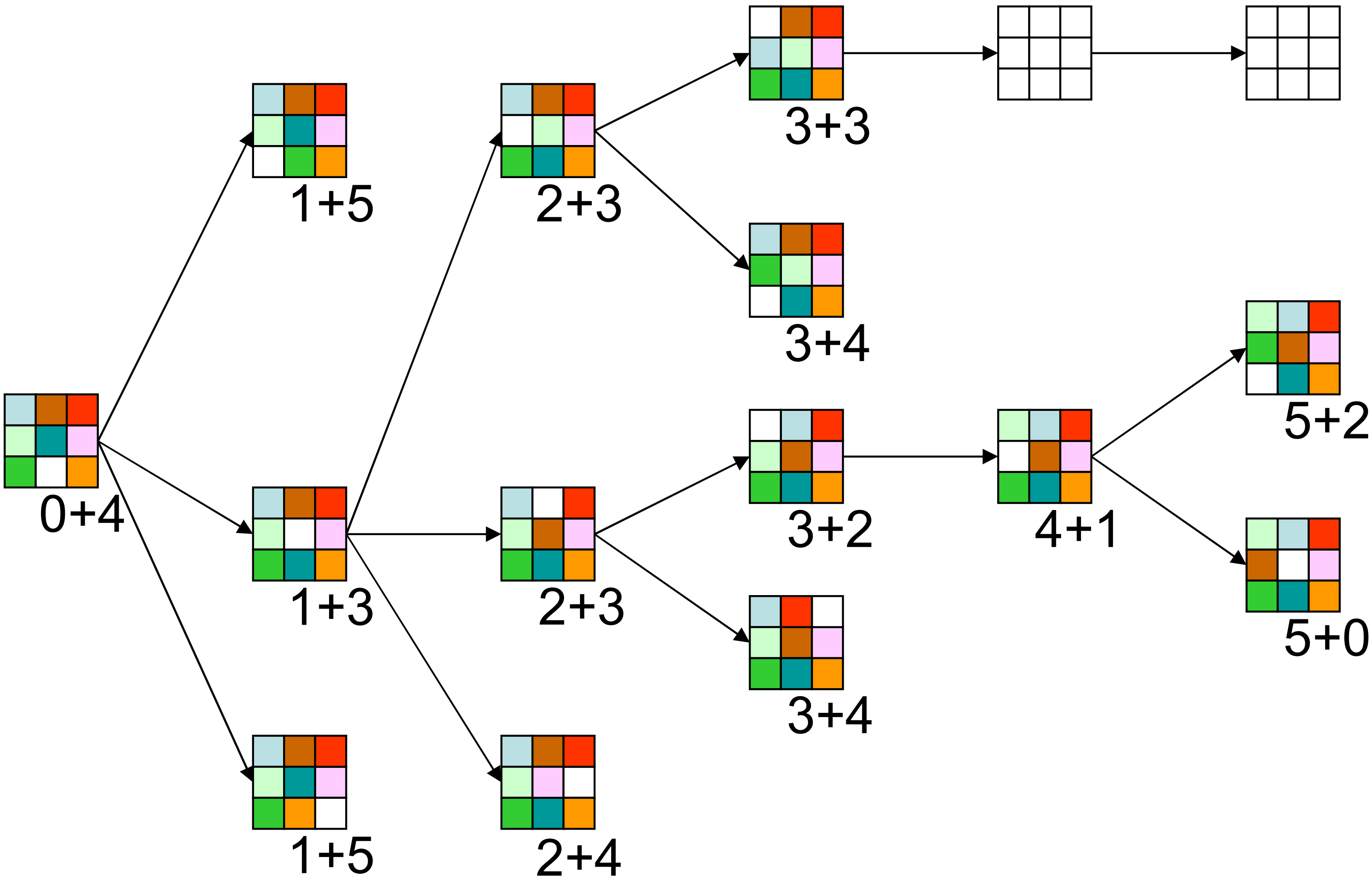
8-Puzzle

$f(N) = h_1(N)$ = number of misplaced tiles



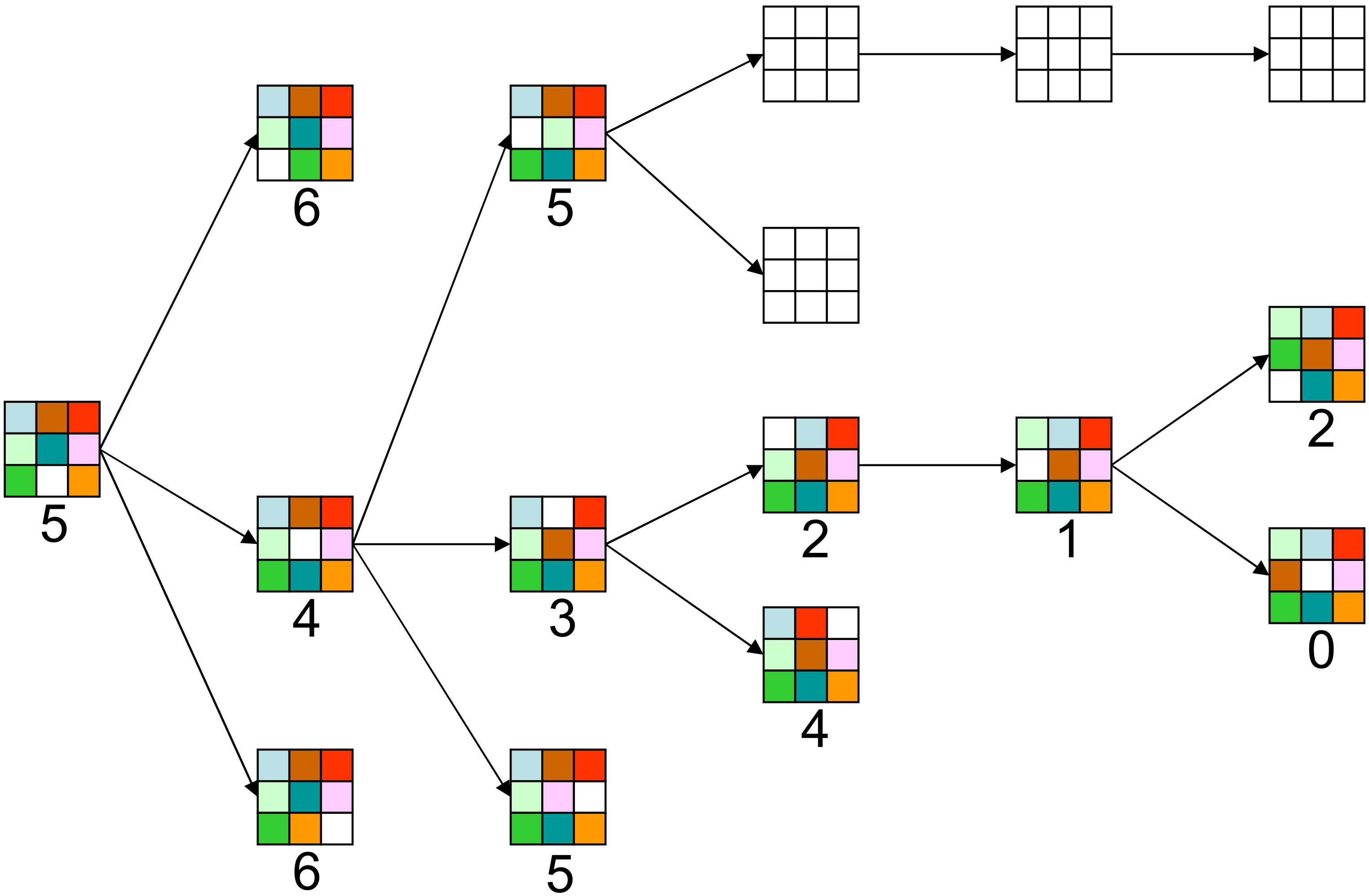
8-Puzzle

$f(N) = g(N) + h(N)$
with $h_1(N)$ = number of misplaced tiles



8-Puzzle

$f(N) = h_2(N) = \Sigma$ distances of tiles to goal



Thank You