Lecture 5

Informed Search

Lusi Li
Department of Computer Science
ODU

Reading for This Class:
Chapter 3, Russell and Norvig

Review

- **Last Class**
  - **Implementation of Blind Search strategies (never looks ahead to goal)**
    - **Breadth-first search**
    - **Depth-first search**
    - **Uniform-cost search**
  - **Analysis of Blind Search Strategy**

- **This Class**
  - **Heuristic Search**

- **Next Class**
  - **Local Search**

Heuristic Search

- **Informed Search (Heuristic Search): use <u>problem-specific knowledge</u> beyond the definition of the problem itself to look ahead to goal**

- **General approach of heuristic search:**
  - **Best-first Search**
    - **Idea: use an evaluation function f(n) for each node**
    - **f(n) is constructed as a cost estimate, and the node with the lowest f(n) is expanded first**
    - **Implemented using a priority queue based on f(n) not g(n)**
    - **Goal-test when node is expanded**

  - **The choice of f(n) determines the search strategy**
    - **Greedy best-first search**
    - **A* search**

Heuristic Function

- **Path-cost Function**
  - g(n) = path cost from root node to node n

- **Heuristic Function**
  - h(n) = an estimated cost from node n to goal
  - nonnegative, problem-specific functions, with one constraint: if n is a goal node, then h(n)=0
  - If h(n1) < h(n2), we guess it is cheaper to reach the goal from n1 than n2

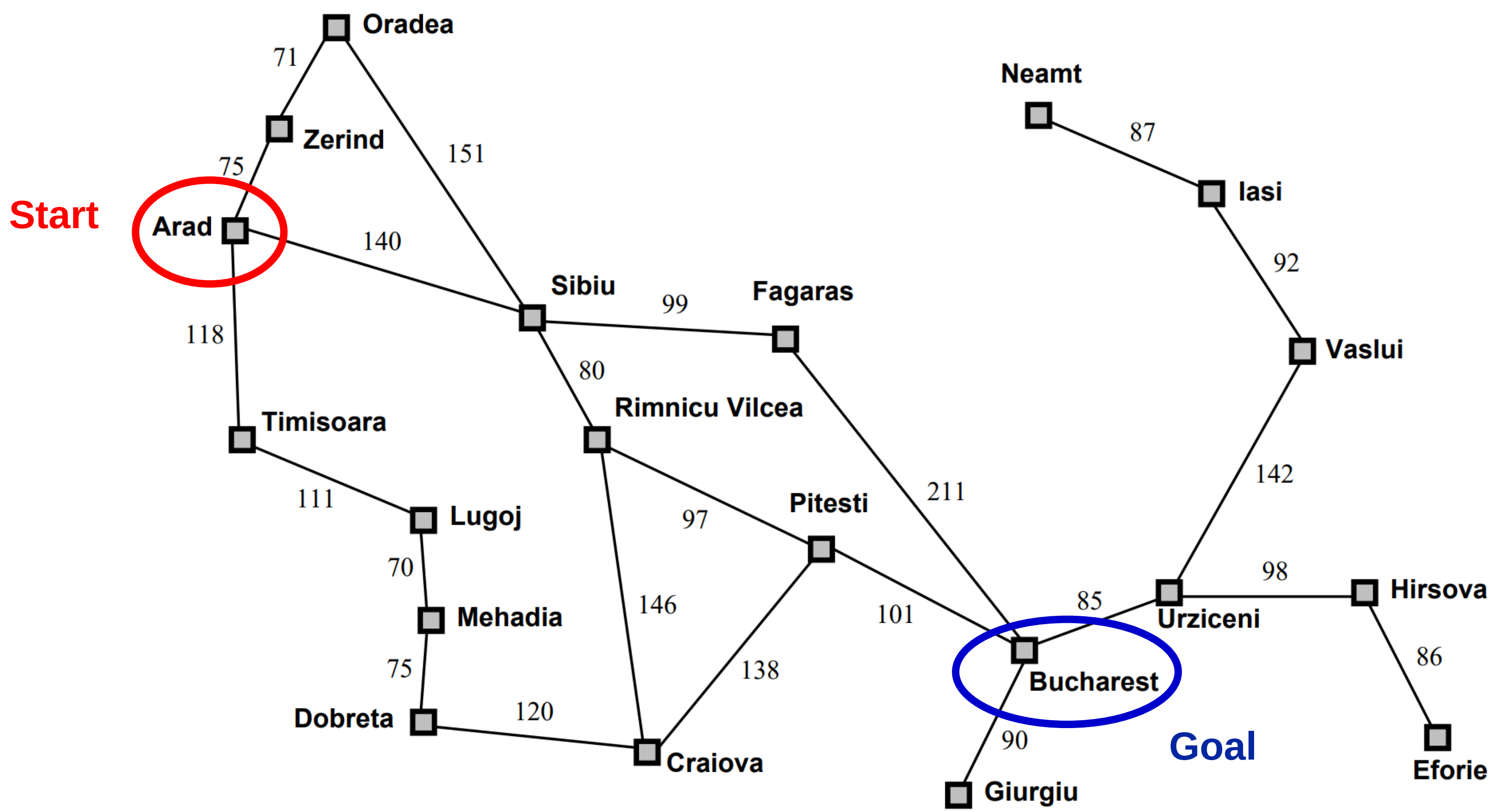
- **Evaluation Function**
  - f(n) = h(n), resulting in greedy best-first search
  - f(n) = g(n) + h(n), resulting in A* search

Greedy Best-First Search

- **Strategy**
  - Ignore the cost to get to node n, g(n)
  - Try to expand the node that is closest to the goal
  - The node is likely to lead to a solution quickly
  - Evaluation function is the heuristic function
    - $f(n) = h(n)$

Example: Route Finding

- **Suppose we start from Arad and our goal is to be in Bucharest**
- **Need to design a heuristic function**



Example: Route Finding Heuristic Function

- **Heuristic Function**
  - Using the straight-line distance h_{SLD}
  - h_{SLD} cannot computed from the problem description itself and is from experience
  - We require $h_{SLD}(In(Bucharest)) = 0$

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

Values of h_{SLD}—straight-line distances to Bucharest.

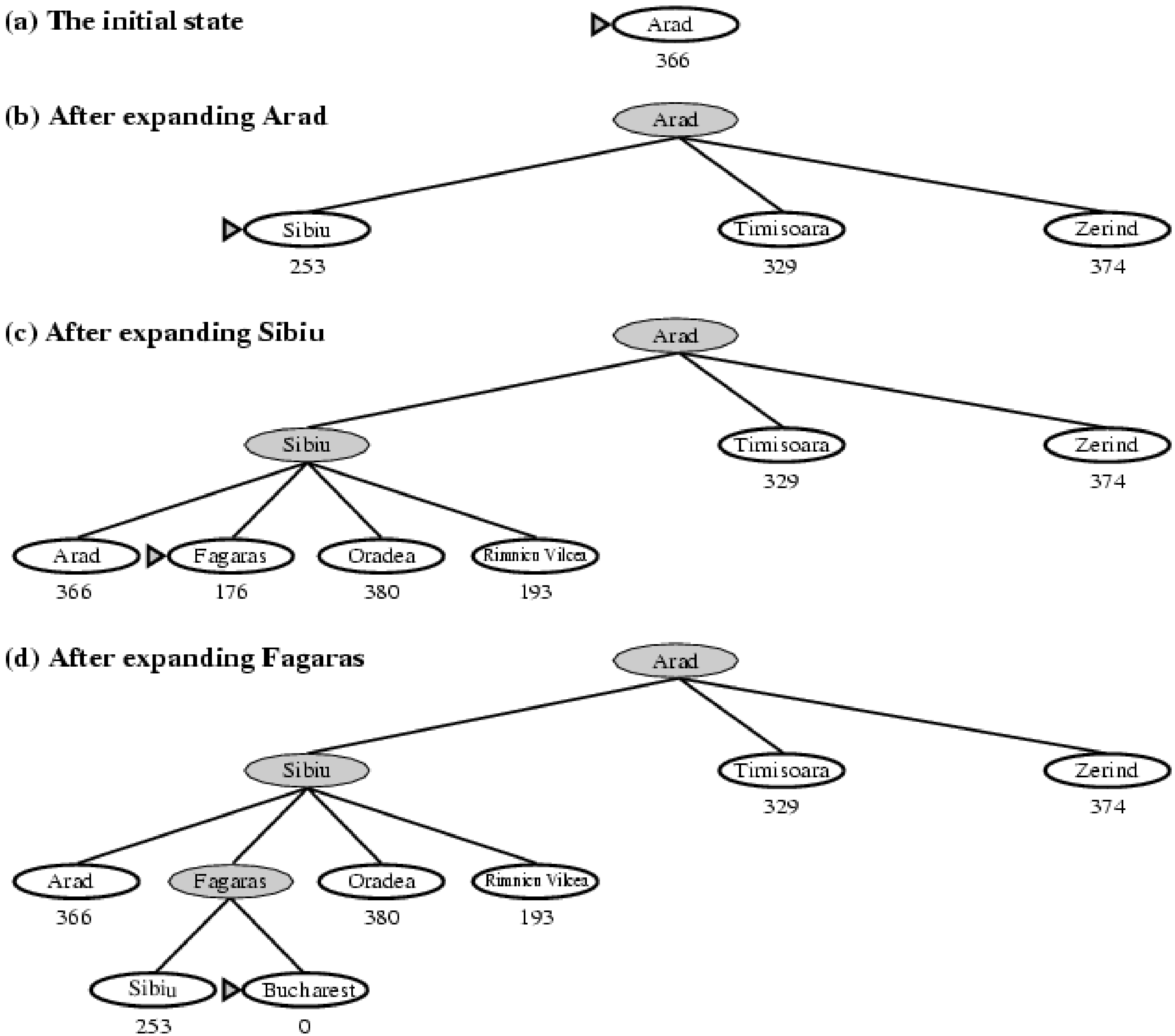
Stages in a Greedy Best-First Tree Search

Frontier queue:
Arad 366

Frontier queue:
Sibiu 253
Timisoara 329
Zerind 374

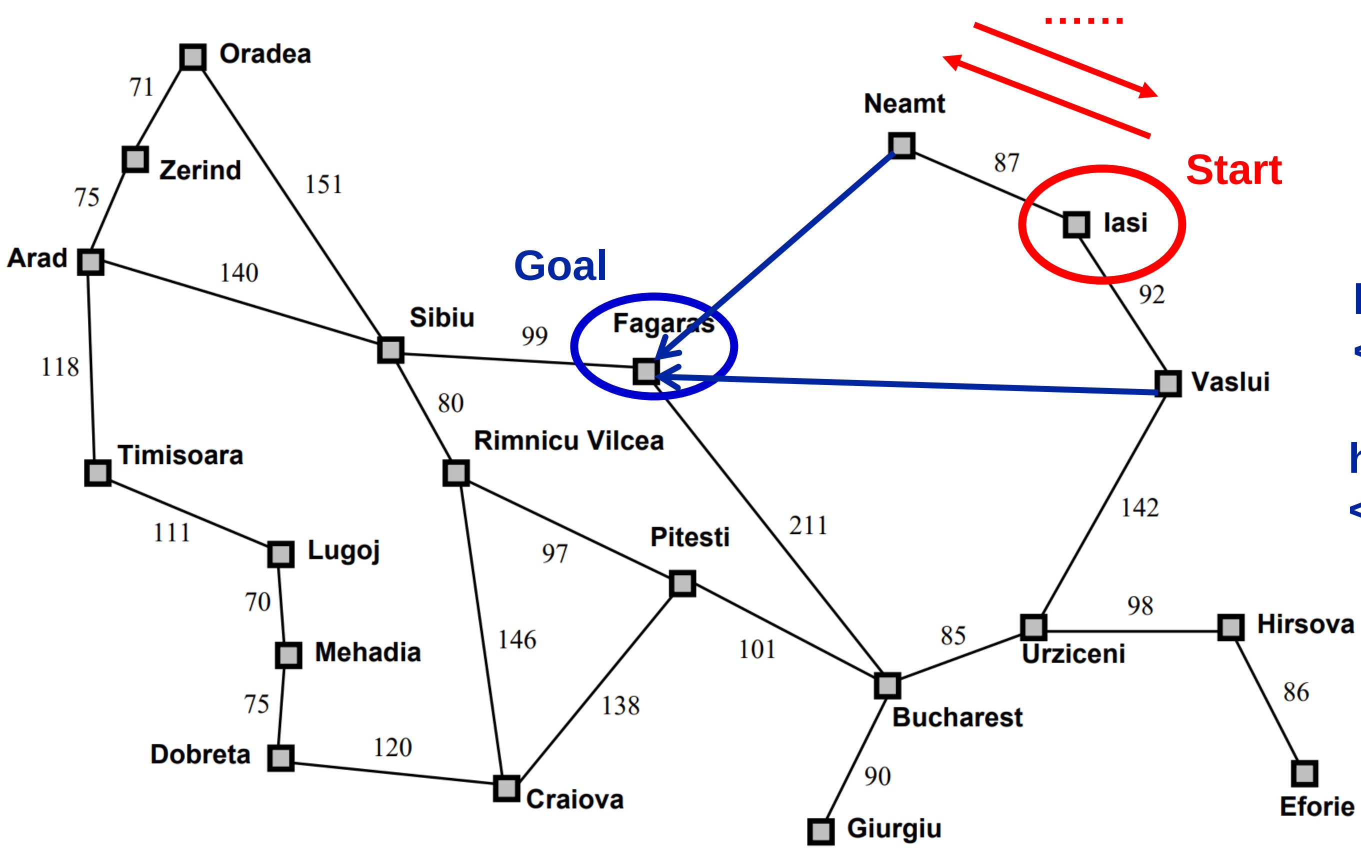
Frontier queue:
Fagaras 176
Rimnicu Vilcea 193
Timisoara 329
Arad 366
Zerind 374
Oradea 380

Frontier queue:
Bucharest 0
Rimnicu Vilcea 193
Sibiu 253
Timisoara 329
Arad 366
Zerind 374
Oradea 380



Analysis of Greedy Best-first Search

- Complete?
  - No, tree search can get stuck in loops, E.g., going from Iasi to Fagaras
  - Graph search is complete in finite space



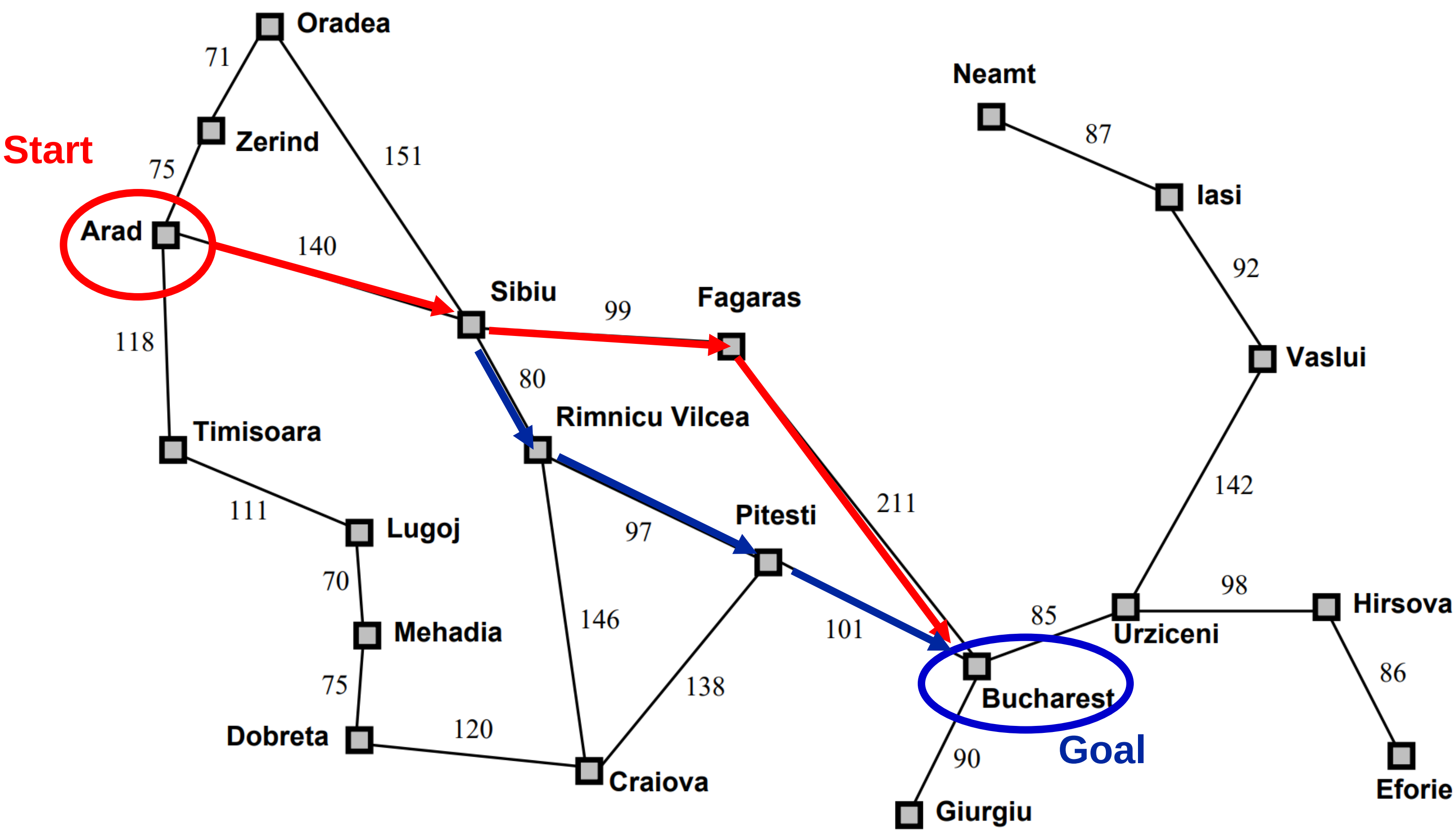
$h_{SLD}(In(Neamt)) < h_{SLD}(In(Vaslusi))$

$h_{SLD}(In(Iasi)) < h_{SLD}(In(Vaslusi))$

Consider the problem of getting from Iasi to Fagaras

Analysis of Greedy Best-first Search

- **Optimal? No (A -> S -> R -> P -> B)**



Analysis of Greedy Best-first Search

- Requirement of Greedy Best-first Search
  - We need additional information to design the heuristic function
    - E.g., straight line distance in our route-finding example
- Performance
  - Complete?
    - No, tree search version can get stuck in loops
    - Yes for graph search in finite space
  - Not optimal: can have a better solution
  - Time complexity: $O(b^m)$ in the worst case
    - but a good heuristic can give dramatic improvement
  - Space complexity: $O(b^m)$ in the worst case
    - keep all nodes in memory
- Conclusion
  - Good Heuristic is the Key
    - The complexity can be reduced substantially
  - Bad Heuristic can make the problem worse

Greedy Algorithms

- **Step-by-step algorithms**
- **Sometimes works well for optimization problems**
- **A greedy algorithm works in phases. At each phase:**
  - **You take the best you can get right now, without regard for future consequences**
  - **You hope that by choosing a *local* optimum at each step, you will end up at a *global* optimum**

Uniform Cost Search vs. Greedy Best First Search

- **Same Strategy**
  - Implementation
    - A priority queue
- **Difference**
  - Uniform Cost Search
    - Cares for the path cost g(n)
    - Expand nodes with the lowest path cost
  - Greedy Best First Search
    - Cares for the heuristic h(n)
    - Expand nodes with the lowest heuristic value
- **Thinking**
  - Can we combine the two search strategies?
  - A* Search

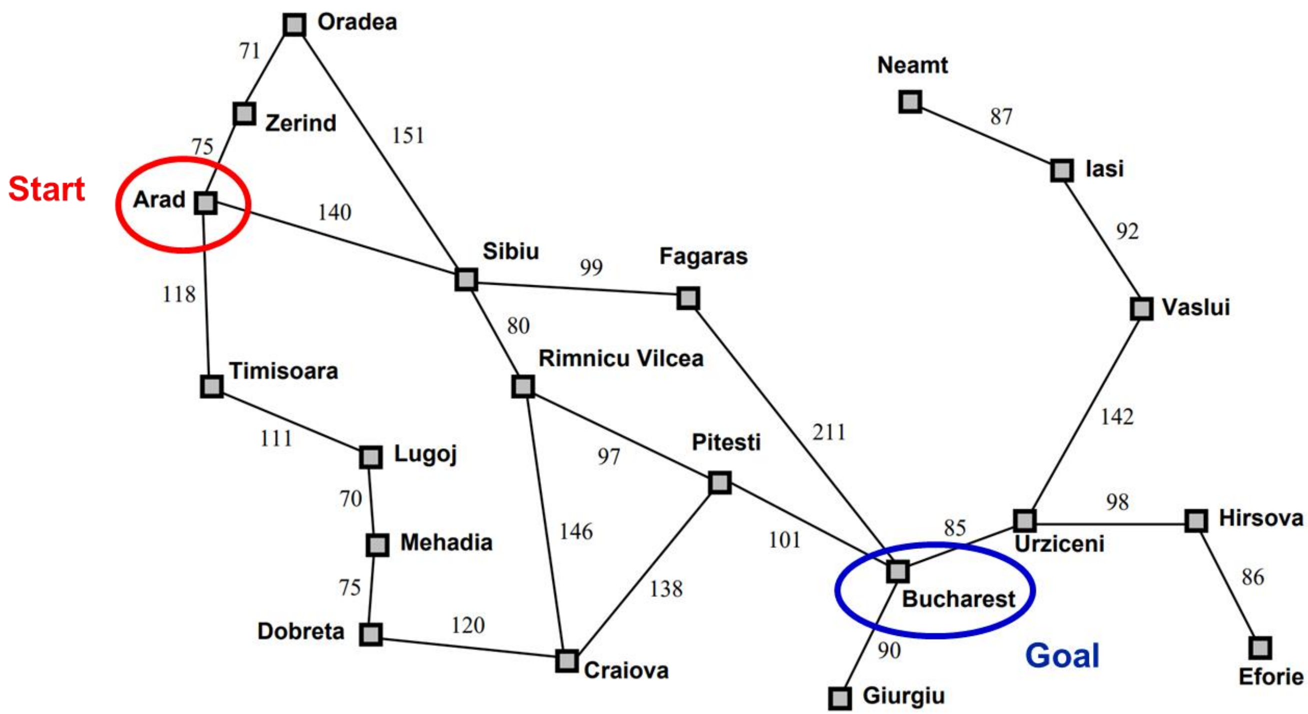
A* Search

- **Goal**
  - Minimizing the total estimated solution cost
- **Evaluation function**
  - f(n)
    - Estimated cost of the cheapest solution through n
    - f(n)=g(n)+h(n)
    - A* search sorts frontier by f(n)
  - g(n)
    - History
    - Gives the path cost from the start node to node n
    - Uniform cost search sorts frontier by g(n)
  - h(n)
    - Estimated cost of the cheapest path from node n to the goal
    - Heuristic function
    - Greedy best first search sorts frontier by h(n)

Example: Route Finding
A* Search

- **Heuristic Function**
  - Using the straight-line distance
- **Path Cost Function**
  - Using the path cost from the graph

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	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



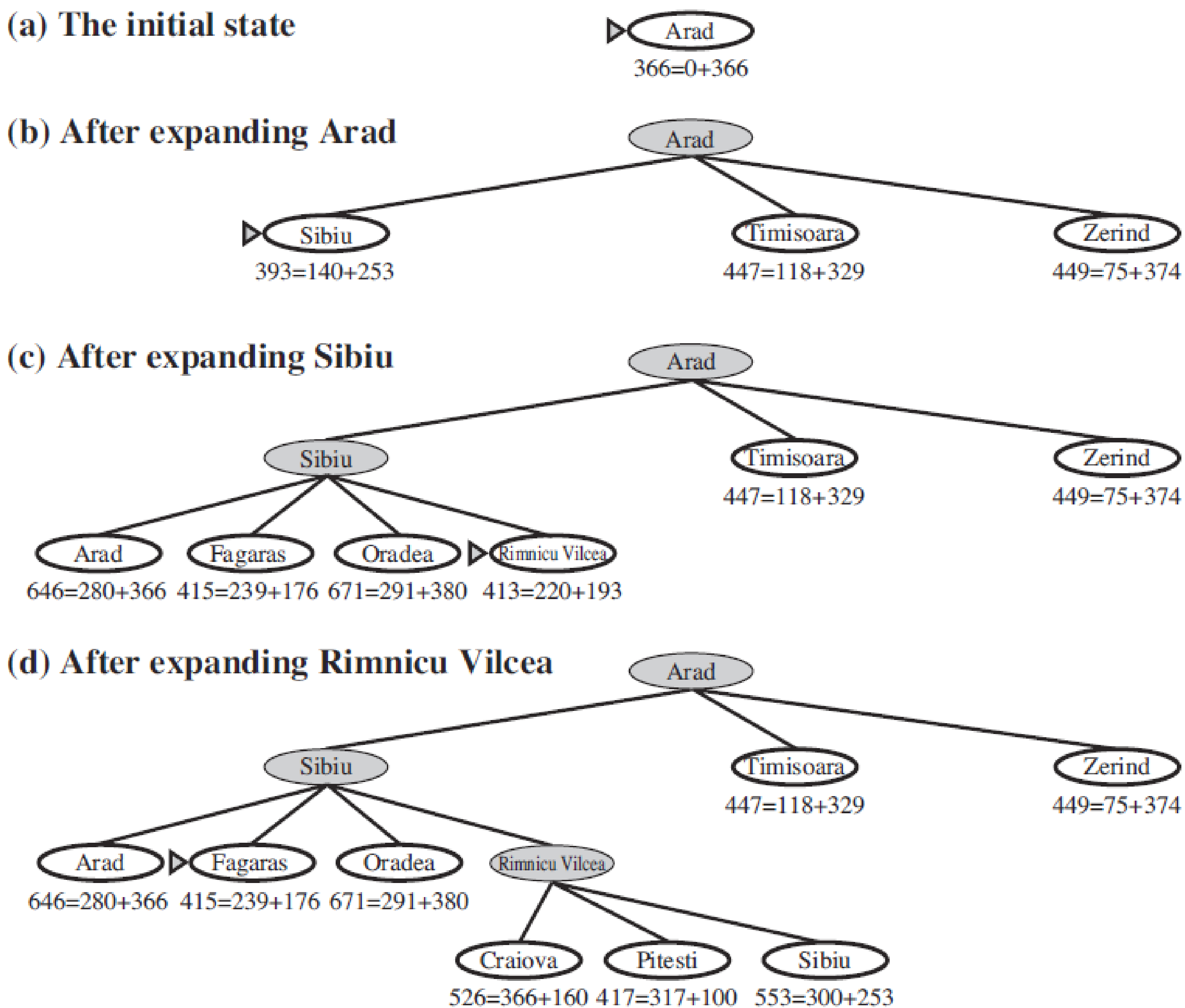
Progress in A* Tree Search

Frontier queue:
Arad 366

Frontier queue:
Sibiu 393
Timisoara 447
Zerind 449

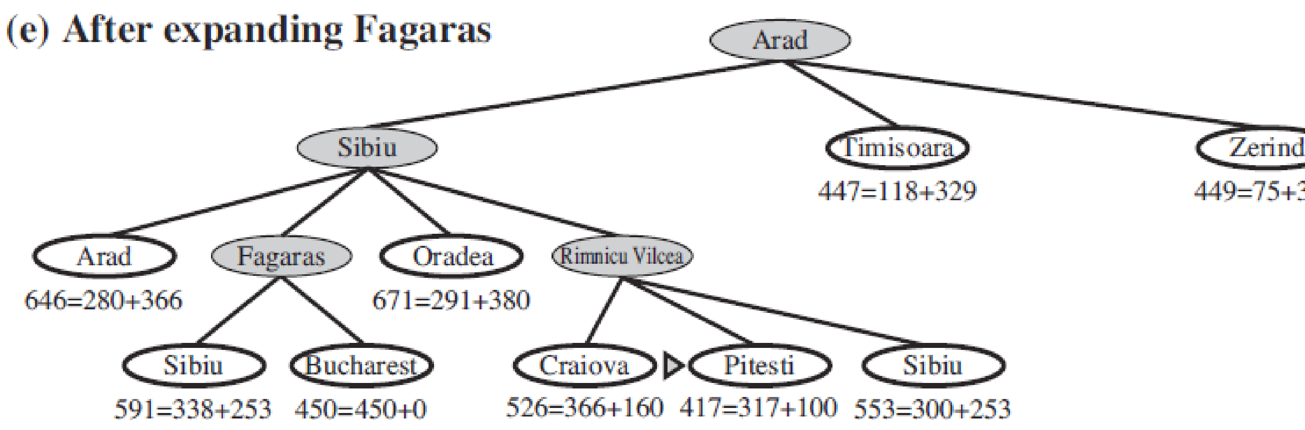
Frontier queue:
Rimricu Vicea 413
Fagaras 415
Timisoara 447
Zerind 449
Arad 646
Oradea 671

Frontier queue:
Fagaras 415
Pitesti 417
Timisoara 447
Zerind 449
Craiova 526
Sibiu 553
Arad 646
Oradea 671



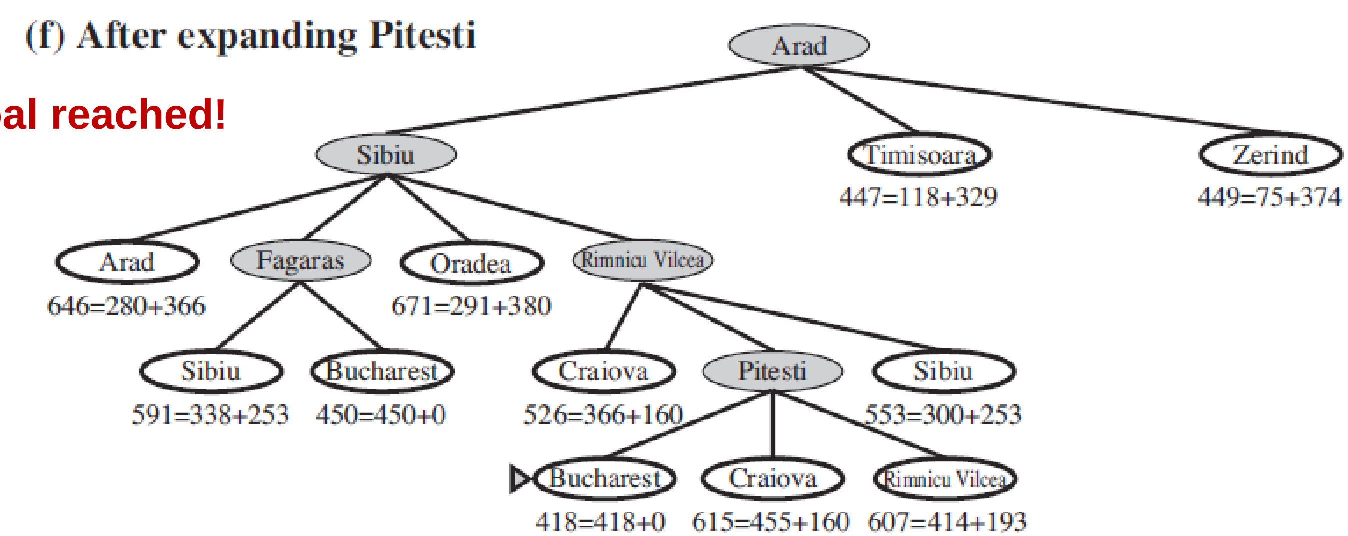
Progress in A* Tree Search

Frontier queue:
Pitesti 417
Timisoara 447
Zerind 449
Bucharest 450
Craiova 526
Sibiu 553
Sibiu 591
Arad 646
Oradea 671



Frontier queue:
Bucharest 418
Timisoara 447
Zerind 449
Bucharest 450
Craiova 526
Sibiu 553
Sibiu 591
Rimricu Vicea 607
Craiova 615
Arad 646
Oradea 671

Goal reached!



Another Example of A* Search
8-puzzle

- Develop a useful heuristic function
- A good heuristic function can reduce the search process

- 8-puzzle
- Size of reachable state space: 9!/2 = 181, 440 states
- Use heuristics

8-Puzzle

- **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

$h_1(n)$ =?

8-Puzzle

- **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

$$h_1(n) = \text{number of misplaced tiles}$$
$$= 6$$

8-Puzzle

- **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

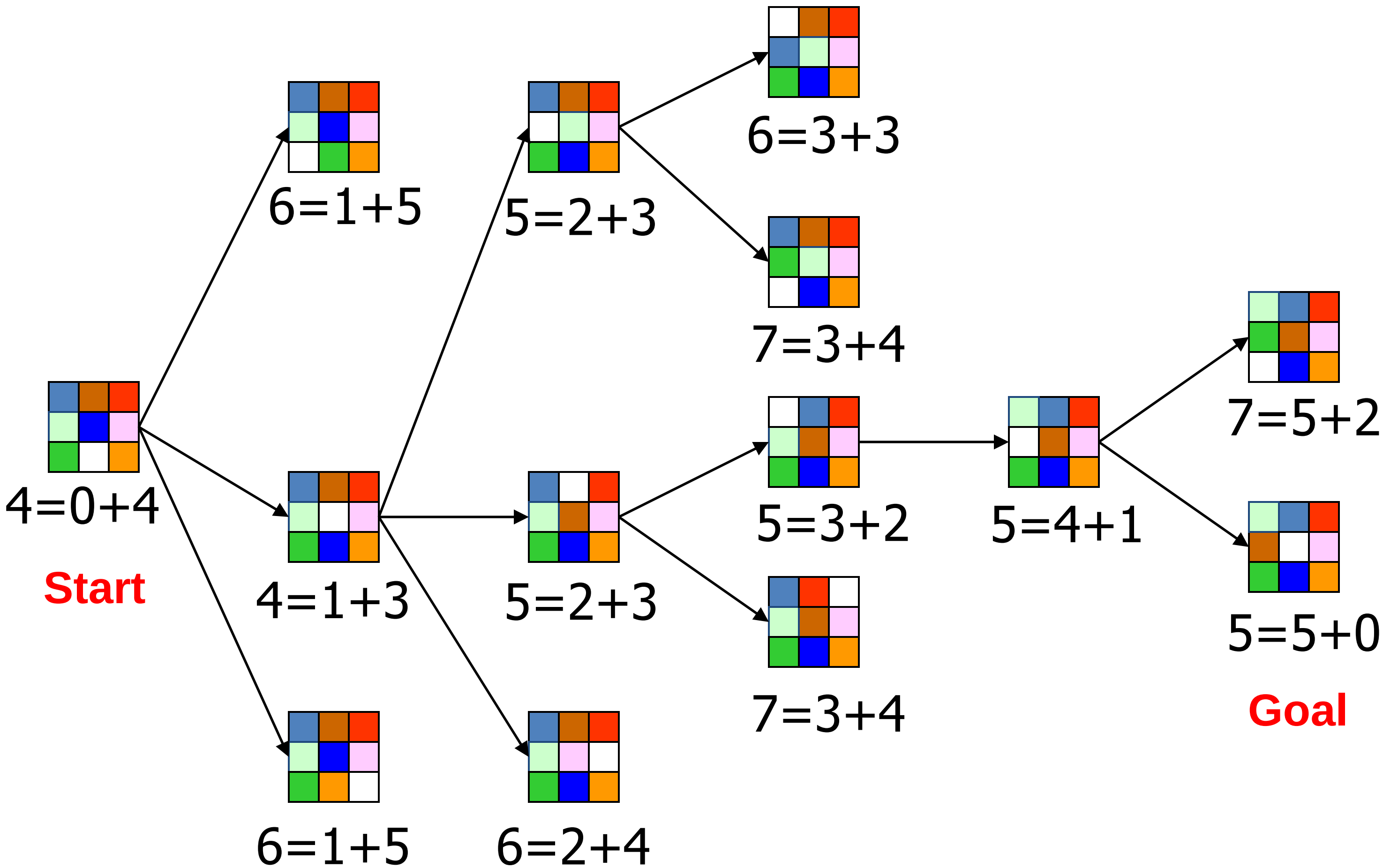
$h_2(n)$ = total Manhattan distance (sum of the distances of every numbered tile to its goal position)

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

$= 13$

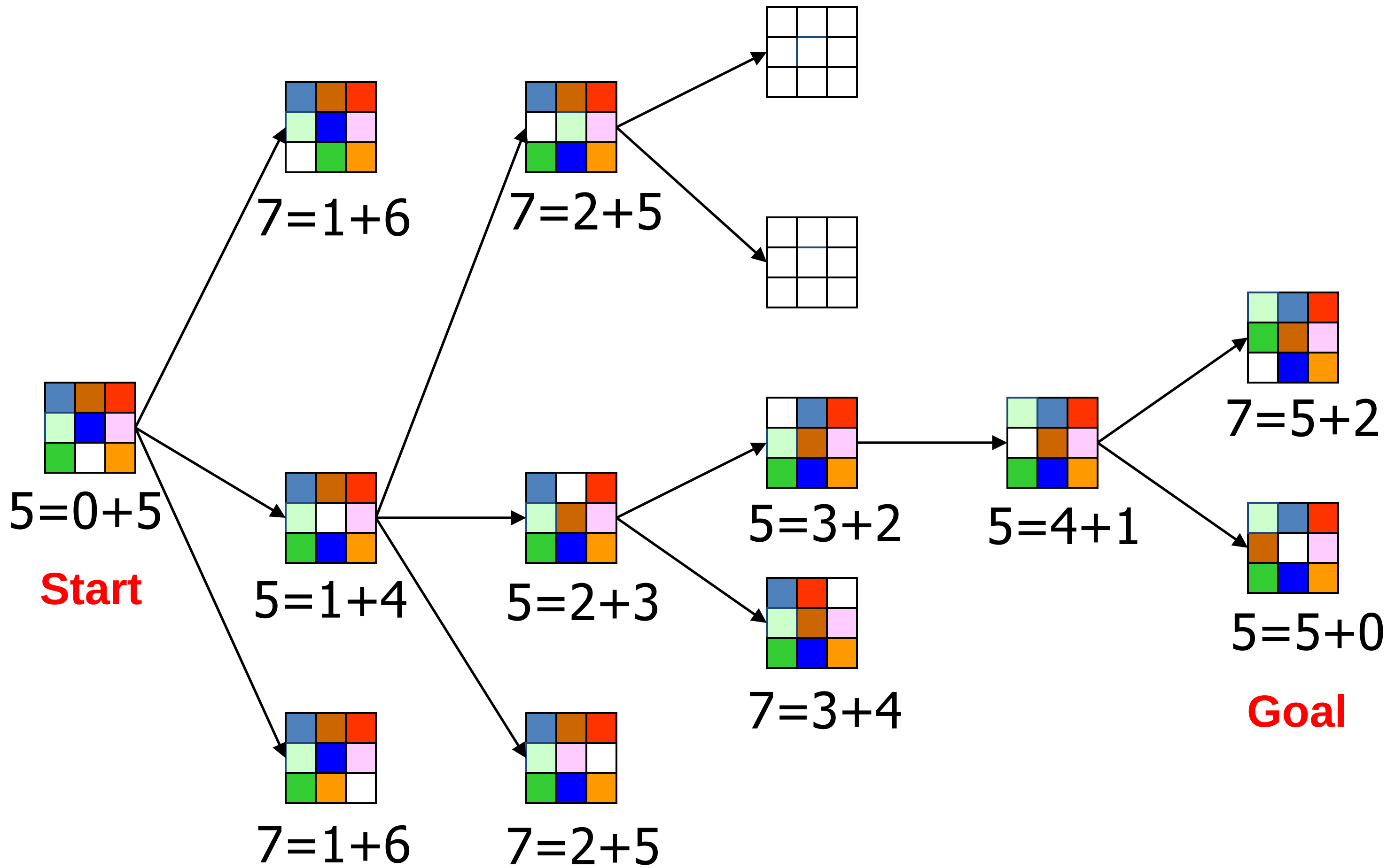
8-Puzzle in Graph Search

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



8-Puzzle in Graph Search

$f(n) = g(n) + h_2(n)$ with $h_2(n) = \Sigma$ distances of tiles to goal



Heuristic Function

- **Relaxed problem:**
  - **A problem with fewer restrictions on the actions than the original**
  - **The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem**
  - **If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ gives the shortest solution**
  - **If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution**

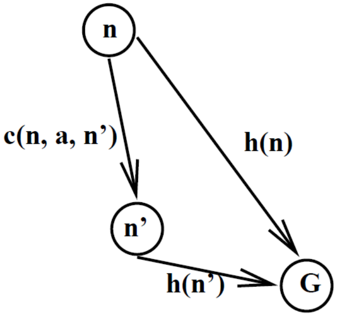
- **Define h(n) = cost of an exact solution to a relaxed problem (fewer restrictions on operator)**
  - **If $h_2(n) \geq h_1(n)$ for all n then h_2 dominates h_1**
  - **So h_2 is optimistic and more accurate than h_1**
  - **h_2 is therefore better for search because fewer nodes will be expanded**

- **Key point: the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem (never overestimate)**

Analysis of A* Search

- **Complete?**
  - Yes, in finite state space
- **Optimal?**
  - Yes, with tree search if h(n) is admissible
  - Yes, with graph search if h(n) is both admissible and consistent
    - admissible heuristic
      - **Provided that h(n) never overestimates the cost to reach the goal**
      - **$h(n) \le h^*(n)$, where $h^*(n)$ is the true cost from node n to the goal**
      - **E.g., h_{SLD} never overestimates actual road distance**
    - consistent heuristic (triangle inequality)
      - **For every node n and every successor n' of n generated by any action a**
      - **the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n'**

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



- **Time complexity: $O(b^d)$ (worst case)**
  - but a good heuristic can give dramatic improvement
- **Space complexity: $O(b^d)$, keep all nodes in memory**

Analysis of A* Search

- **Drawback**
  - **Space requirement**
    - **Keep all generated nodes in memory**
    - **Easy to run out of memory**
    - **Not practical for many large-scale problems**

Summary

- Heuristics and Heuristic Functions
- Heuristic Algorithm
- Greedy Algorithm
- A* Algorithm

What I want you to do

- **Review Chapter 3**
- **Work on your assignment**