Lecture 13
Informed (Heuristic) Search Strategies

Artificial Intelligence

Dr. Ahmed Mateen

Today’s Agends

- Informed Search
  - Heuristic Functions
  - Best First Search
    - Difference between Best First Search & Uniform Cost Search
  - Greedy Best First Search
    - Difference between Best First Search & Greedy Best First Search
  - A* Search

Blind Search

- Depth-first search and breadth-first search are examples of *blind* (or uninformed) search strategies.
- Breadth-first search produces an optimal solution (eventually, and if one exists), but it still searches blindly through the state-space.
- Neither uses any knowledge about the specific domain in question to search through the state-space in a more directed manner.
- If the search space is big, blind search can simply take too long to be practical, or can significantly limit how deep we're able to look into the space.

Informed (Heuristic) Search Strategies

- Uses *problem-specific* knowledge
- Can find solutions more *efficiently* than an uninformed strategy.
- Searches less of the state space

- Ideally, we'd like a **search strategy** which is both
    - admissible
    - informed

Informed (Heuristic) Search Strategies

- A search strategy which searches the most promising branches of the state-space first can:
  - find a solution more quickly,
  - find solutions even when there is limited time available,
  - often find a *better* solution, since more profitable parts of the state-space can be examined, while ignoring the unprofitable parts.

- A search strategy which is better than another at identifying the most promising branches of a search-space is said to be more *informed*.

Informed vs Uniformed Search Strategies

Basis of Comparison	Informed Search	Uninformed Search
Basic knowledge	Uses knowledge to find the steps to the solution.	No use of knowledge
Efficiency	Highly efficient as consumes less time and cost.	Efficiency is mediatory
Cost	Low	Comparatively high
Performance	Finds the solution more quickly.	Speed is slower than the informed search.
Algorithms	Best-first search, Greedy Best First Search and A* search	Depth-first search, breadth-first search, and lowest cost first search

Heuristics

- Heuristic -> guess
  - Which node will reach to goal.
  - Information that can be readily obtained about a node.
  - Can be learned based on experiences.

- Heuristic Function
  - Expands most promising node according to some heuristic
  - heuristic function *h(n)* -> estimated cost of the cheapest path from the state at node *n* to a goal state.
    - which takes a node *n* and returns a non-negative real number that is an estimate of the path cost from node *n* to a goal node.
    - The function *h(n)* is an *underestimate* if

$$h(n) <= g(n)$$

Best First Search

- Idea: use an evaluation function *f(n)* for each node
  - f(n) provides an estimate for the total cost.

  → Expand the node n with smallest f(n).

- Implementation:

  Order the nodes in fringe increasing order of cost.

  Makes use of Priority Queue

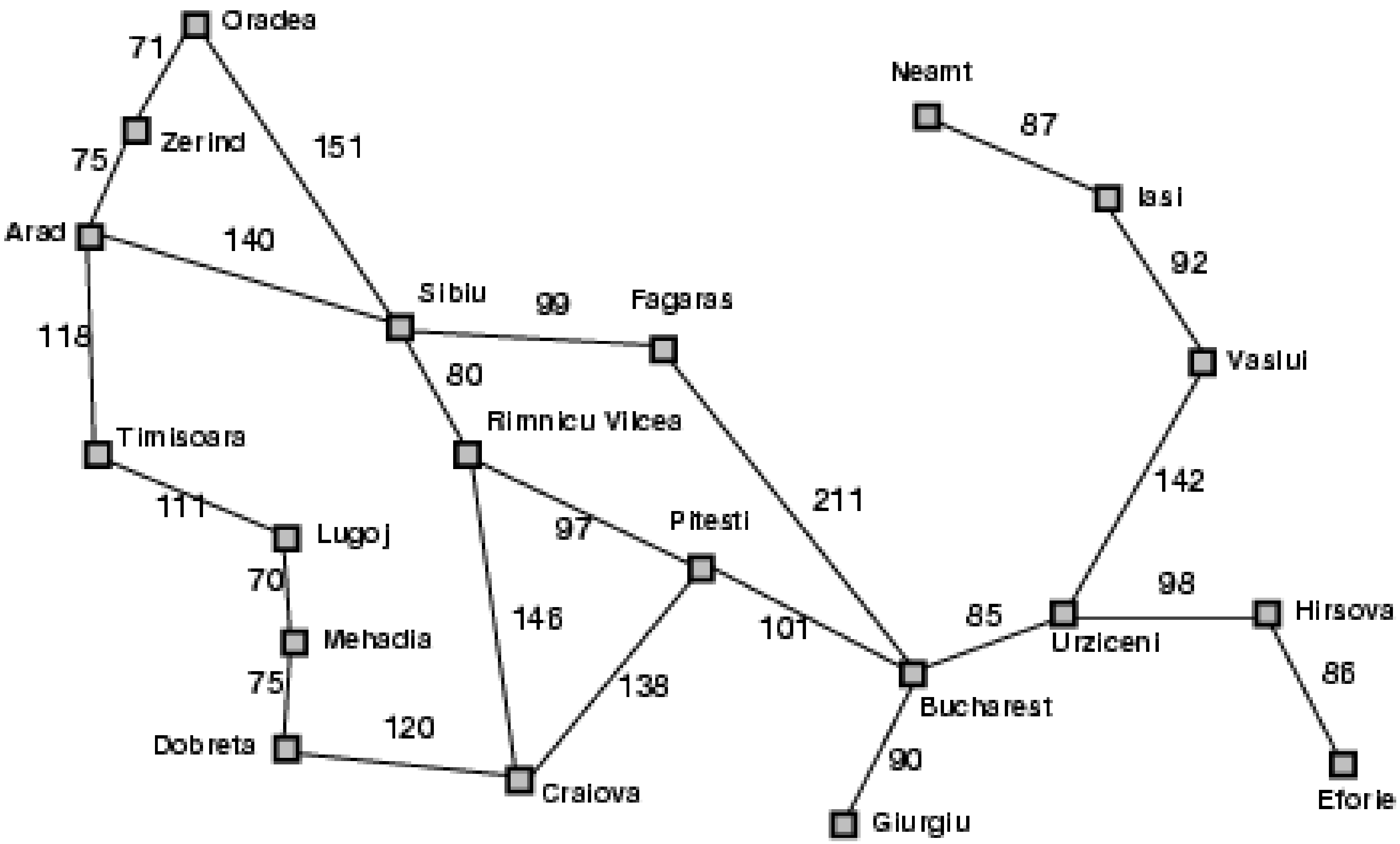
  Makes use of heuristic value

- Special cases:
  - greedy best-first search
  - A^* search

Best First Search

- Similar Case
  - Uniform Cost Search
- Difference
  - Best First Search uses of *f* instead of *g* to order the priority queue.
  - *lowest path cost* g(n).

Romania with straight-line Distance



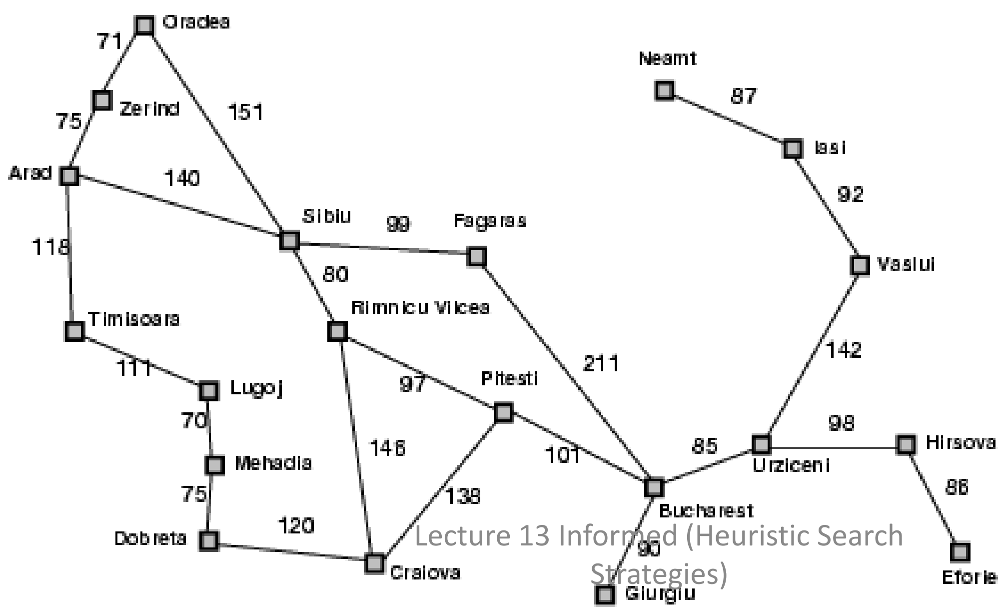
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best First Search

- f(n) = estimate of cost from *n* to *goal*
- e.g., $f_{SLD}(n)$ = straight-line distance from *n* to Bucharest
- Greedy best-first search expands the node that appears to be closest to goal.
- It evaluates nodes by using just the heuristic function;

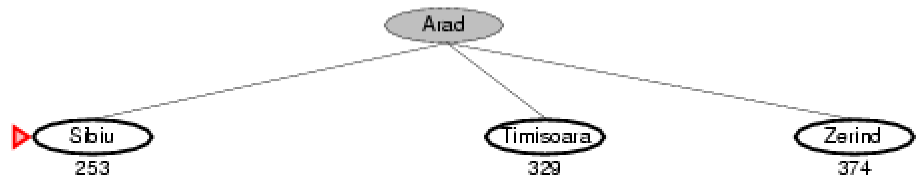
$$f(n) = h(n)$$

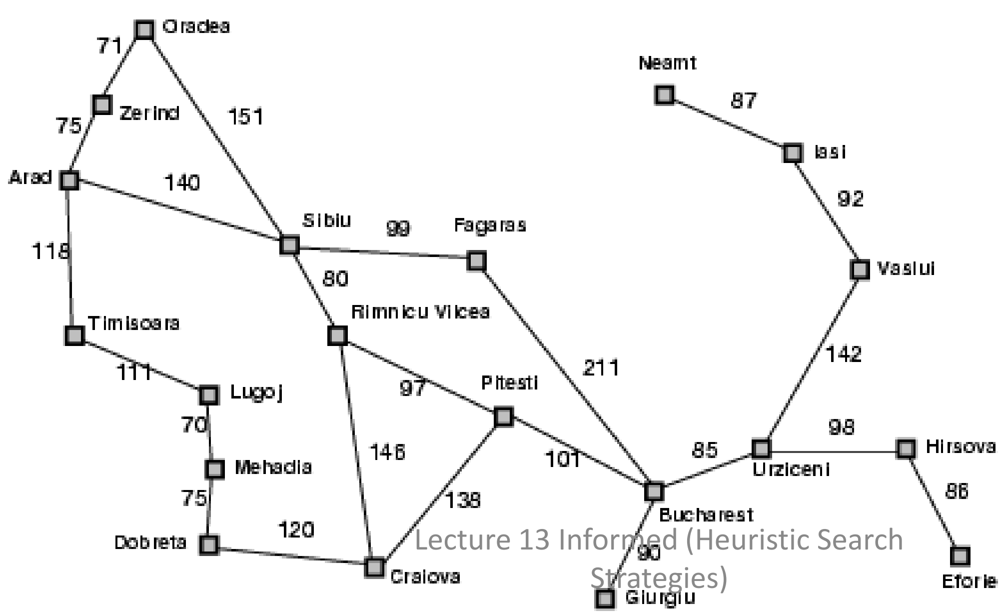
Greedy best-first search example



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

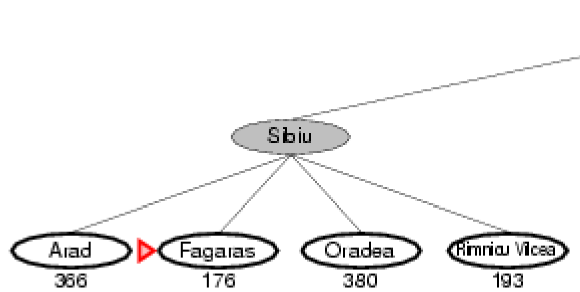
Greedy best-first search example

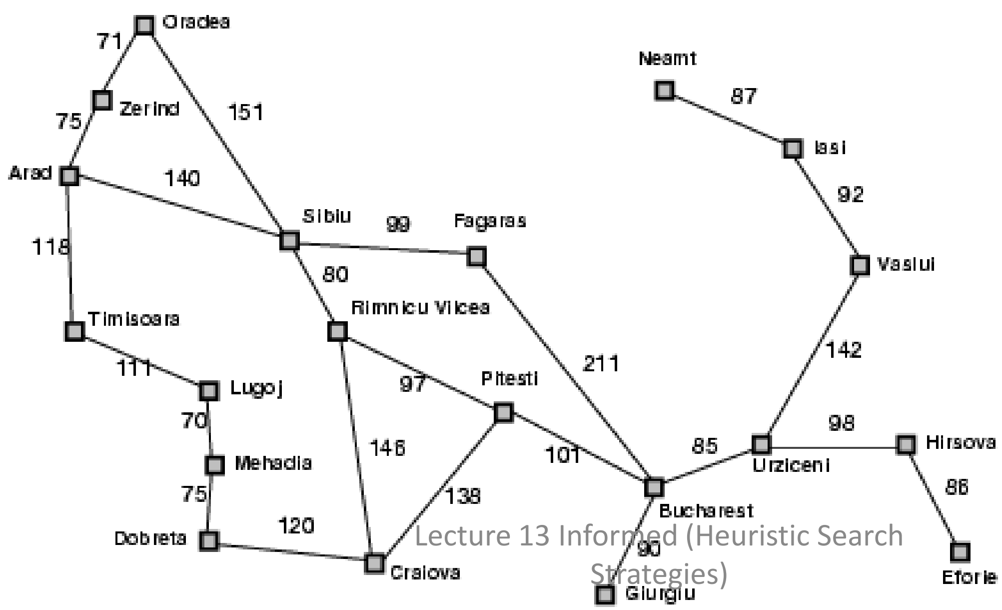




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search example

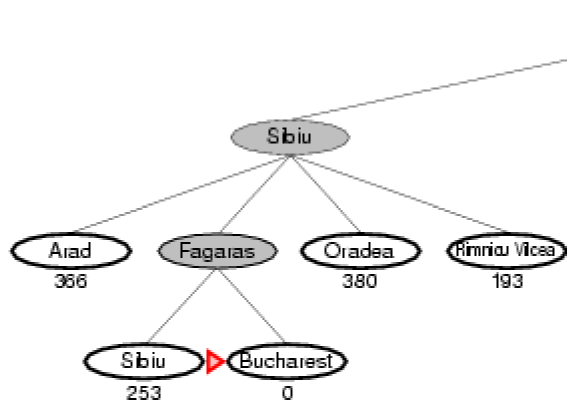


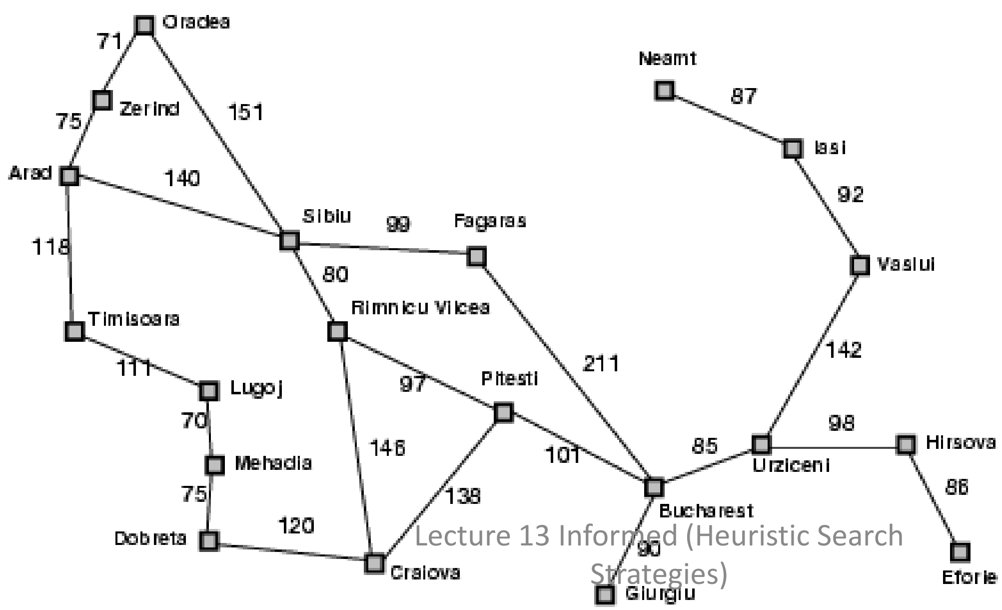


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search example





Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Properties of greedy best-first search

- Complete? No – can get stuck in loops.
- Time? $O(b^m)$, but a good heuristic can give dramatic improvement
- Space? $O(b^m)$ - keeps all nodes in memory
- Optimal? No

  e.g. Arad→Sibiu→Rimnicu Virea→Pitesti→Bucharest is shorter!

Difference between Best First Search & Greedy Best First Search

Best First Search	**Greedy Best First Search**
Keeps track of path cost	Ignores path cost
Allows revising decisions	In a greedy algorithm, the decisions should be final, and not revised.
Always results in optimal solution	Often results in accepting suboptimal solutions.
Slow as compared to greedy best first search	Improves running time.
Uses evaluation function *f(n)*	Uses current heuristic value *f(n)* = *h(n)*

A* Search

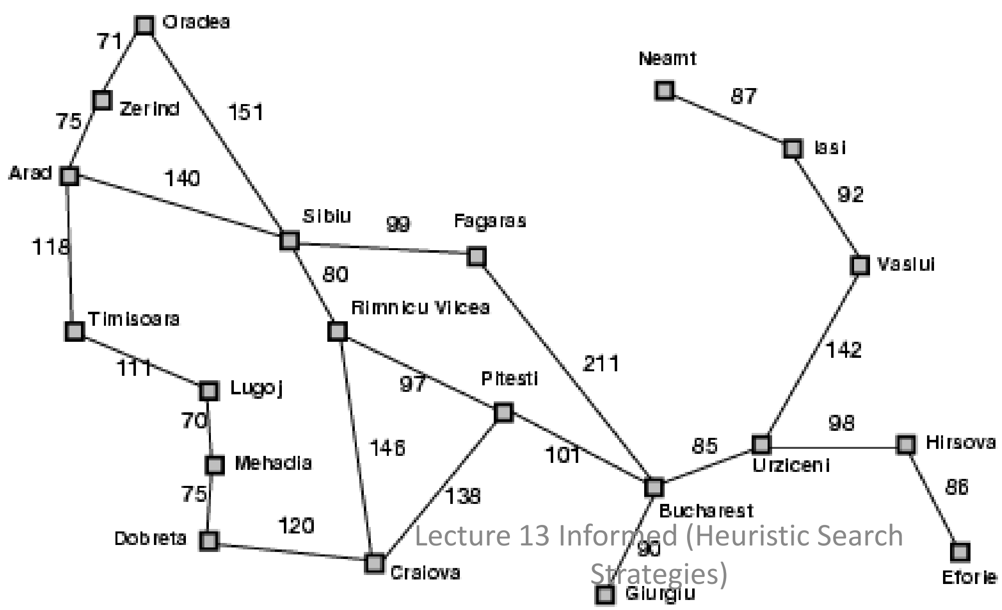
- **A*** A **search** (pronounced “A-star search”).
- Most widely known form of Best First Search

- **A*** Score
  - cost of path
  - heuristics

A^* search

- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = actual cost so far to reach n
- $h(n)$ = estimated cost from n to goal
- $f(n)$ = estimated total cost of path through n to goal
- Best First search has $f(n)=h(n)$

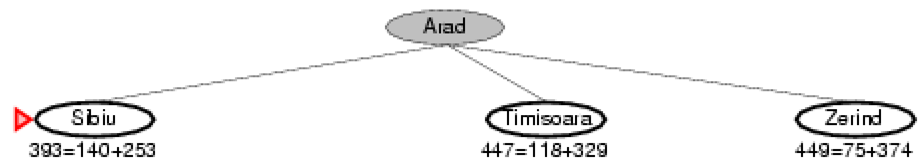
A* search example

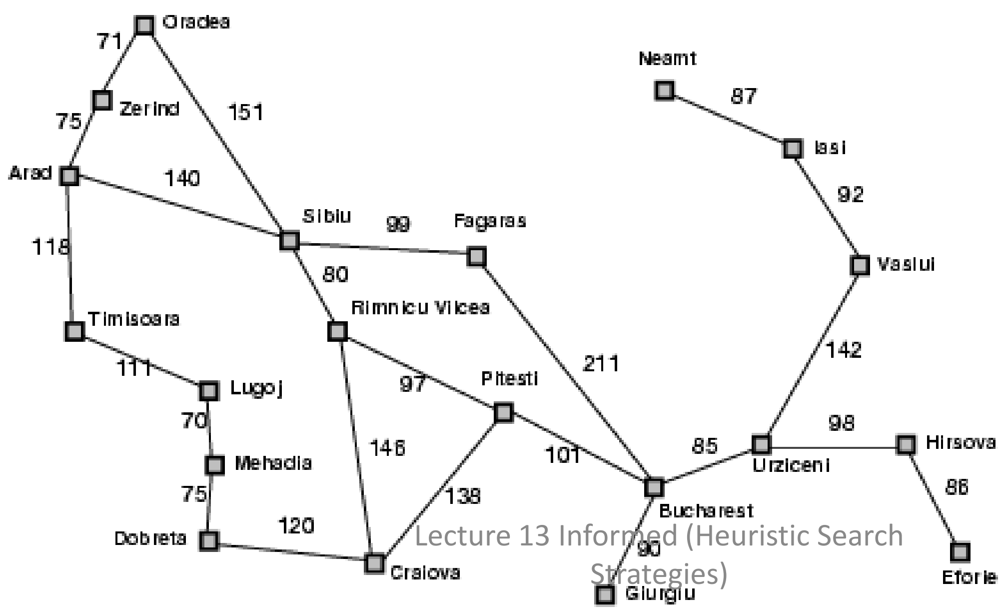


Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

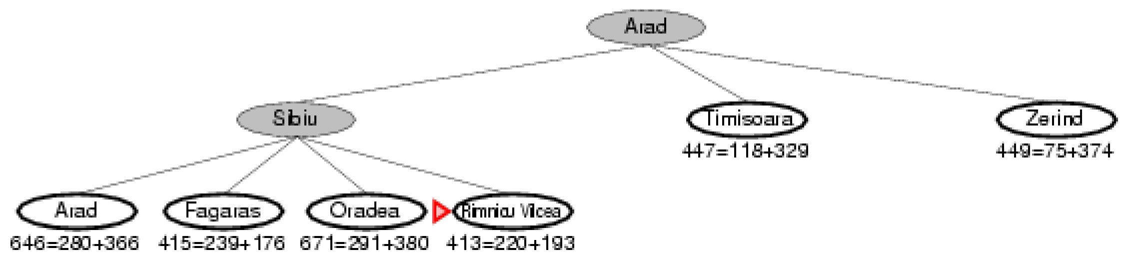
A* search example

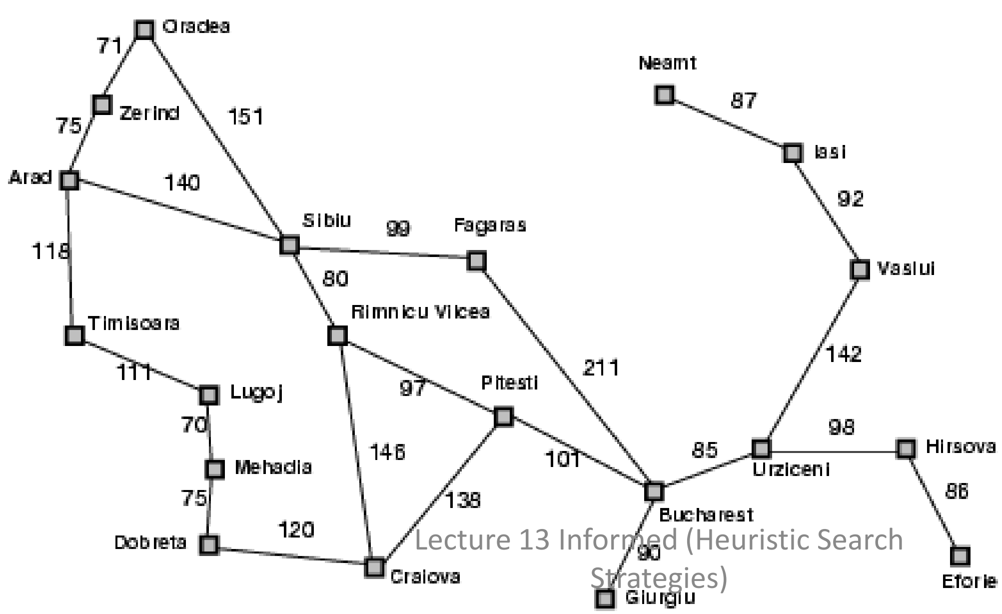




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

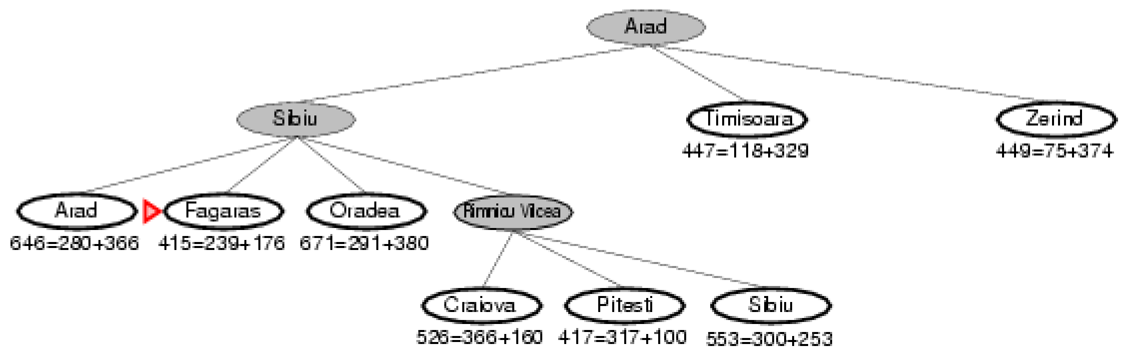


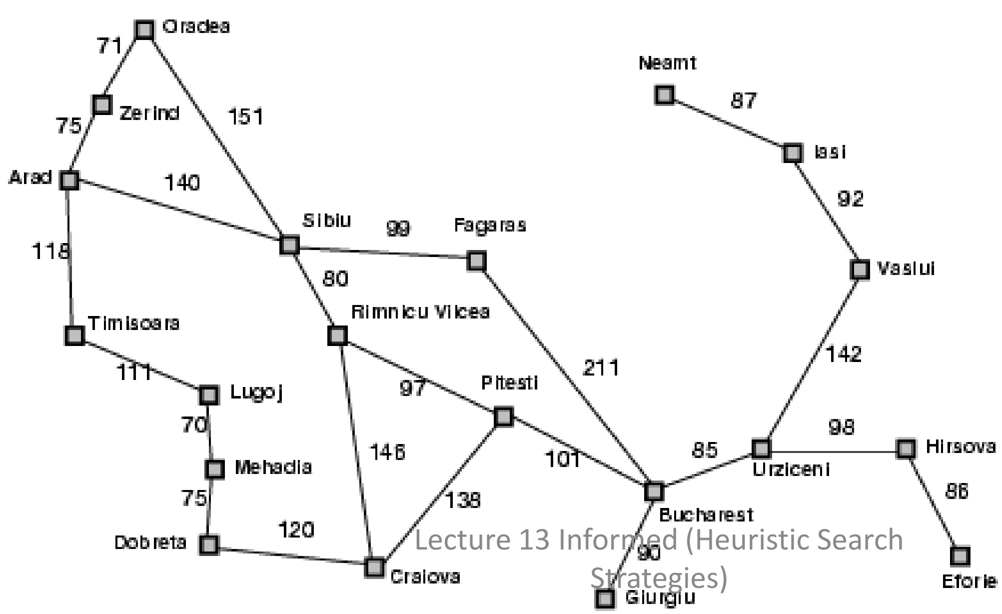


Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A^* search example

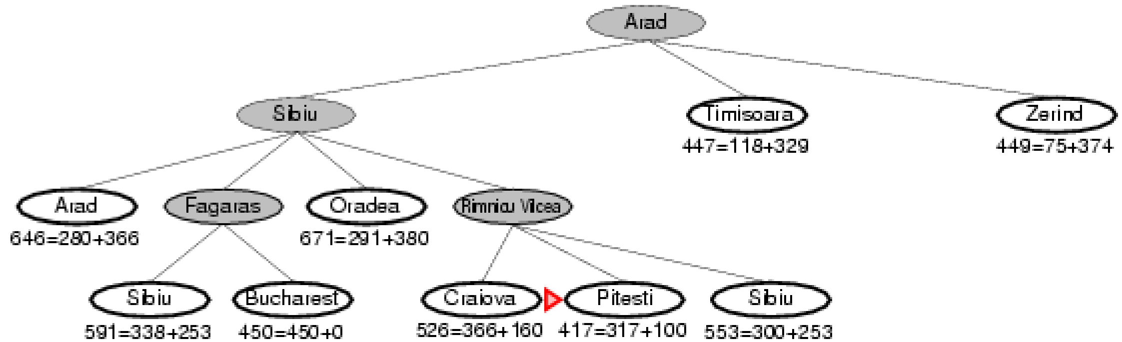


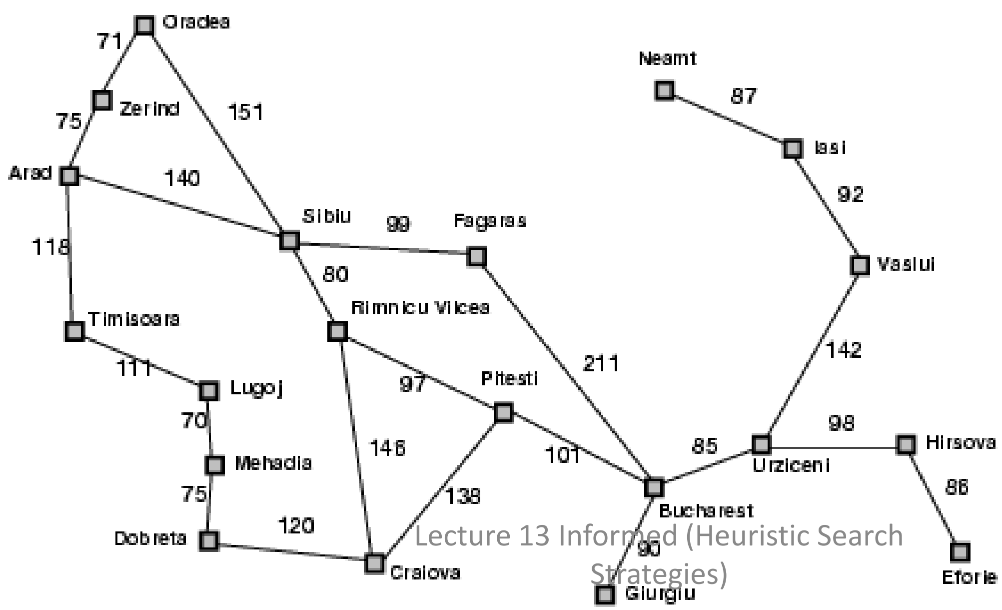


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	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

A* search example

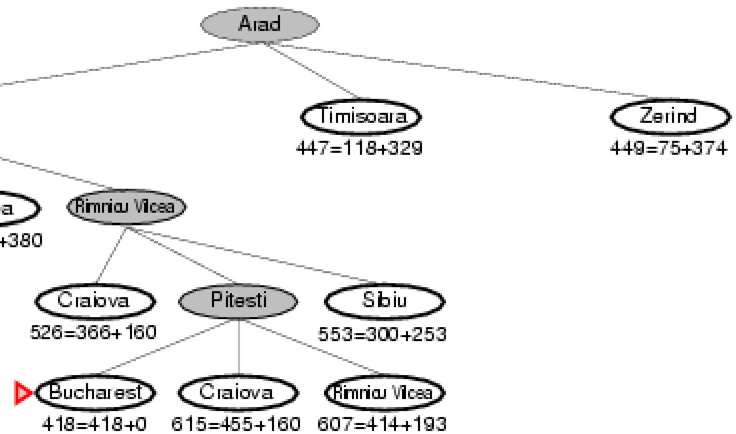


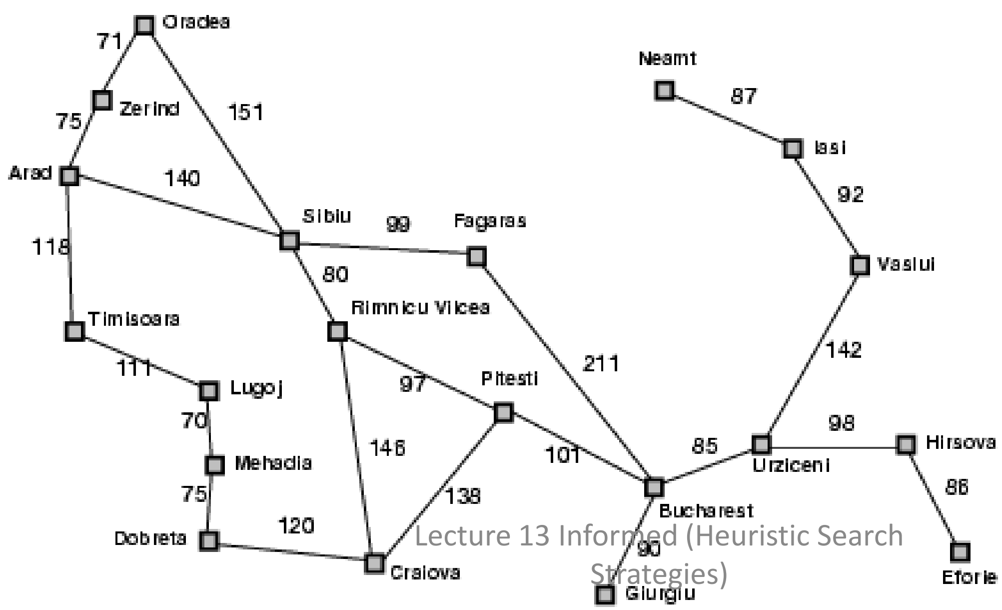


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example





Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Admissible heuristics

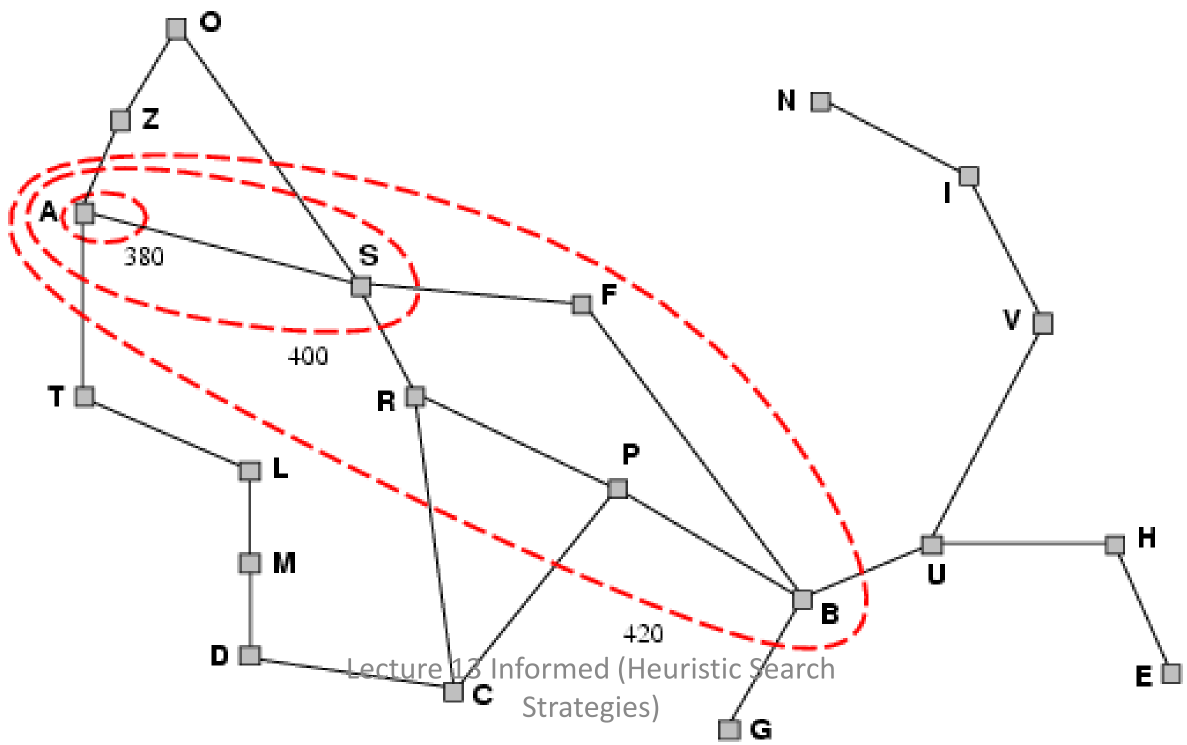
- A heuristic $h(n)$ is admissible if for every node n, if

$$h(n) \leq g(n);$$

where $g(n)$ is the true cost to reach the goal state from n

- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- 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*

- A^* expands nodes in order of increasing f value
- Gradually adds "f-contours" of nodes
- Contour i contains all nodes with $f \le f_i$ where $f_i < f_{i+1}$
- 

Properties of A*

- Complete? Yes (unless there are infinitely many nodes with f $\leq f(G)$, i.e. path-cost > ε)
- Time/Space? Exponential

$$b^d$$

- Optimal? Yes
- *Optimally Efficient*: Yes (no algorithm with the same heuristic is guaranteed to expand fewer nodes)