A heuristic evaluation function is a function that when applied to anode gives a value that represent a good estimate of the distance of the node from a goal . For two nodes *m , n and heuristic function f , if f(m) < f(n), then it should be the case that m is more likely to be an optimal path to the goal node n.*

Informed Search Strategies

- Greedy search
- A* search
- IDA* search
- Hill climbing
- Simulated annealing

- Also known as **heuristic search**
  - require heuristic function

Algorithm: Simple Hill Climbing

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.
2. Loop until a solution is found or until there are no new operators left to be applied in the current state:
   (a) Select an operator that has not yet been applied to the current state and apply it to produce a new state.
   (b) Evaluate the new state.
      (i) If it is a goal state, then return it and quit.
      (ii) If it is not a goal state but it is better than the current state, then make it the current state.
      (iii) If it is not better than the current state, then continue in the loop.

Review: Best-first search

Basic idea:

- ***select node for expansion*** **with minimal** ***evaluation function*** $f(n)$
  - where $f(n)$ is some function that includes ***estimate heuristic*** $h(n)$ of the remaining distance to goal

- **Implement using priority queue**
- **Exactly UCS with** $f(n)$ **replacing** $g(n)$

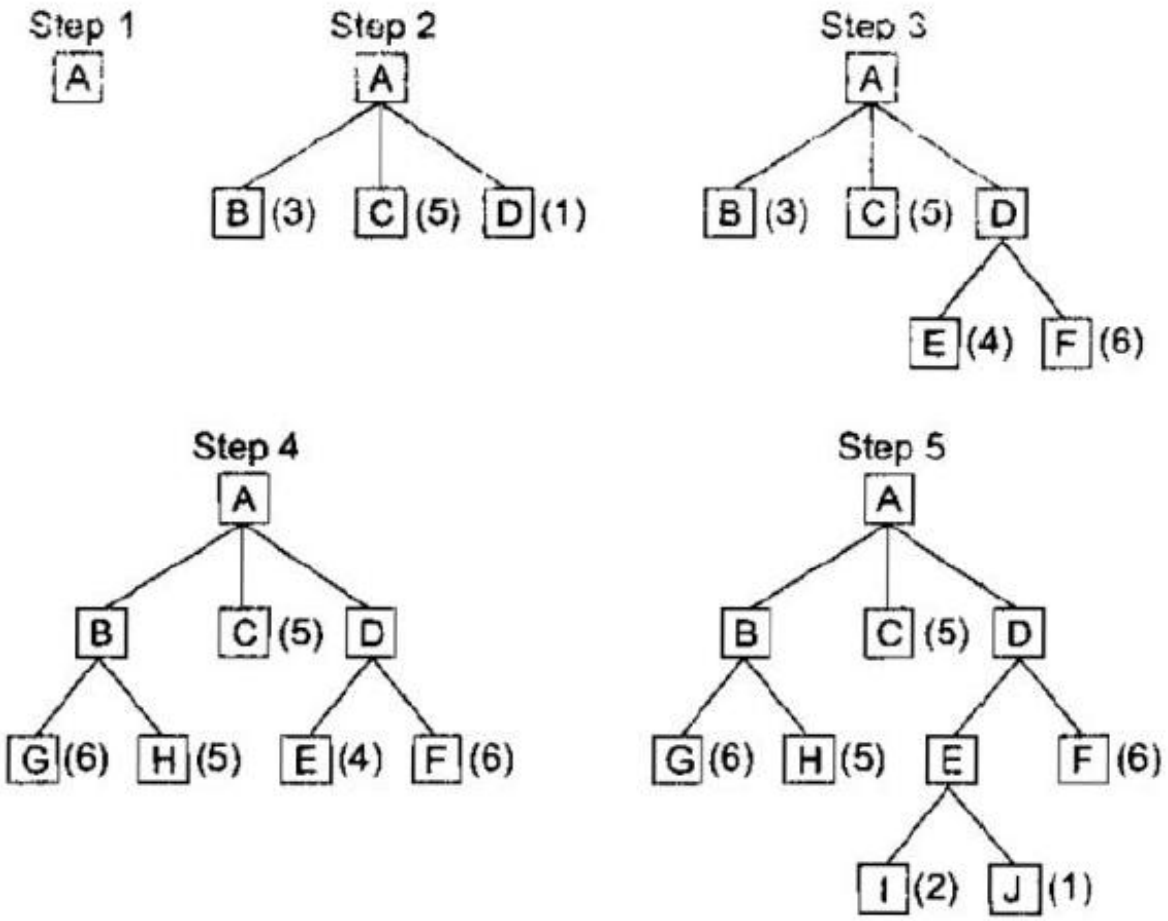


Fig. 3.3 *A Best-First Search*

Greedy best-first search: $f(n) = h(n)$

- Expands the node that *is estimated* to be closest to goal
- Completely ignores *g(n):* the cost to get to *n*
- Here, $h(n) = h_{SLD}(n)$ = straight-line distance from to Bucharest

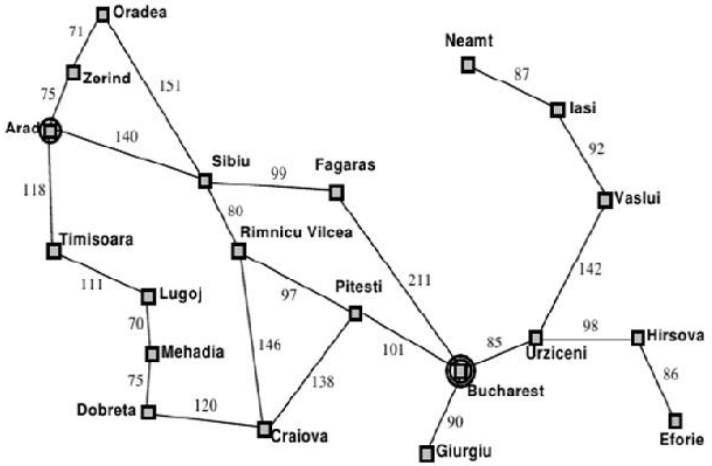
Greedy best-first search example

Frontier queue:

Arad 366

- **Initial State = Arad**
- **Goal State = Bucharest**

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



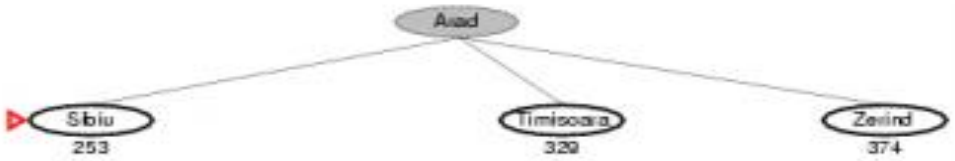
Greedy best-first search example

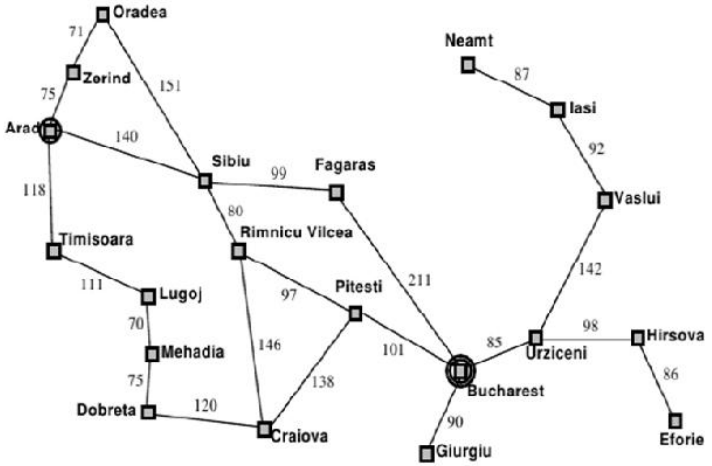
Frontier queue:

Sibiu 253

Timisoara 329

Zerind 374





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

Greedy best-first search example

Frontier queue:

Fagaras 176

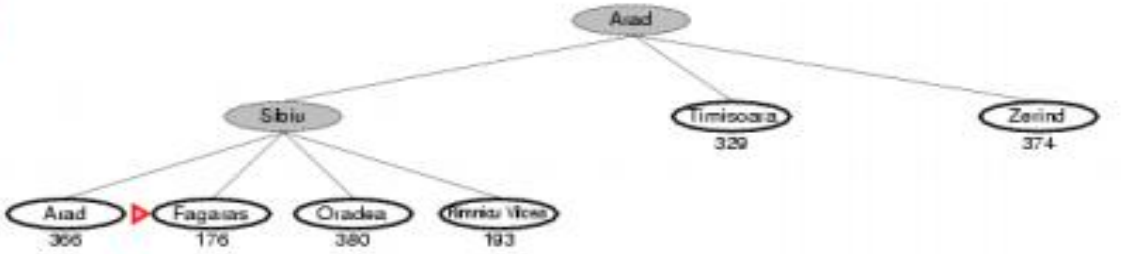
Rimnicu Vilcea 193

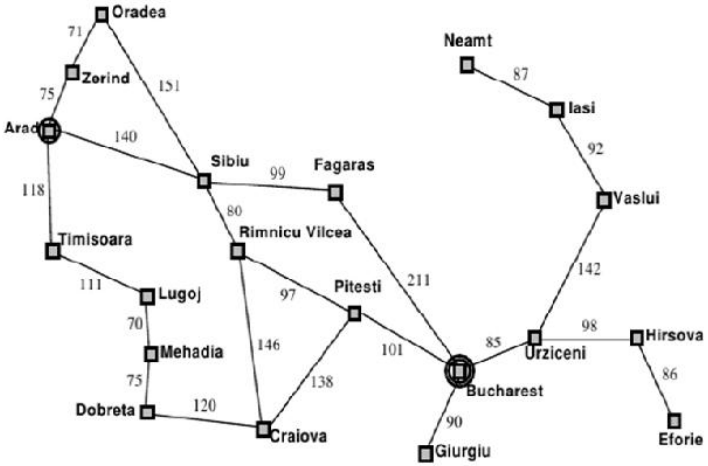
Timisoara 329

Arad 366

Zerind 374

Oradea 380





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

Greedy best-first search example

Frontier queue:

Bucharest 0

Rimnicu Vilcea 193

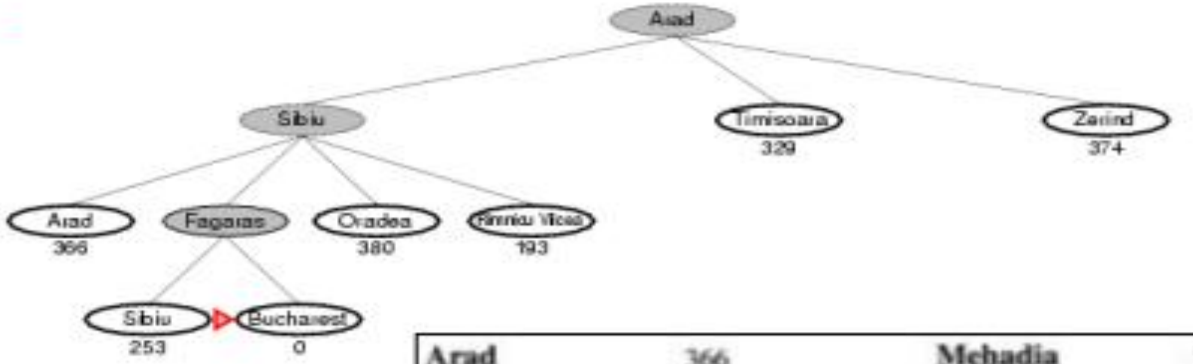
Sibiu 253

Timisoara 329

Arad 366

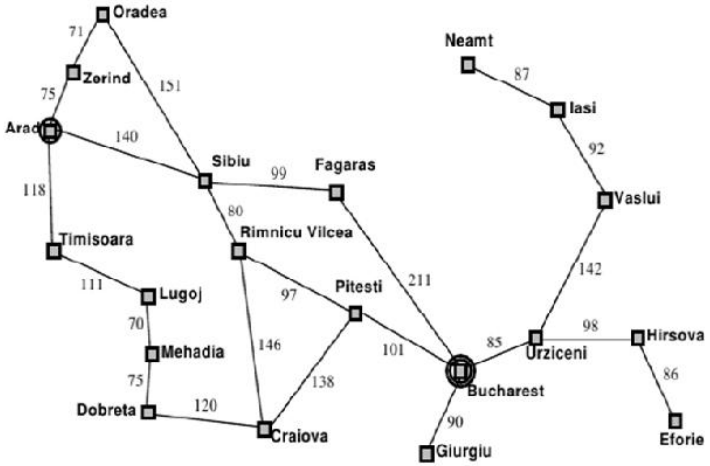
Zerind 374

Oradea 380



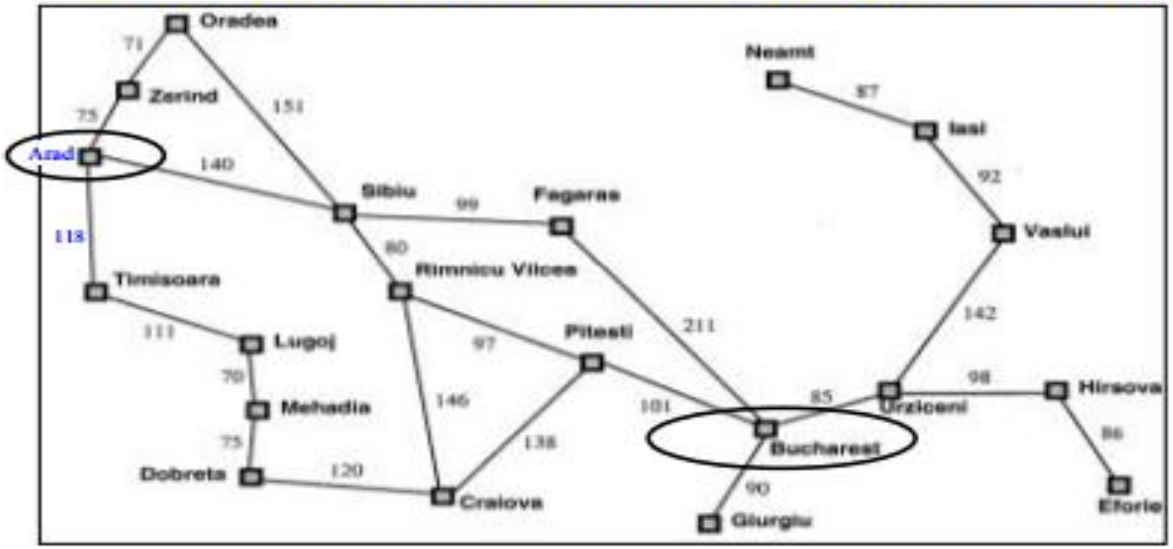
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Goal reached !!



Properties of greedy best-first search

- ***Optimal?***
  - No!
    - —Found: *Arad → Sibiu → Fagaras → Bucharest (450km)*
    - —Shorter: *Arad → Sibiu → Rimnicu Vilcea → Pitesti → Bucharest (418km)*



140+99+211= 450
140+80+97+101= 418 (Arad- Sibiu – Rimnicu Vilcea – Pitesti – Bucharest)
140+80+146+138+101= 605
118+111+70+75+120+138+101= 733
118+111+70+75+120+146+97+101=838
75+71+151+99+211=607
75+71+151+80+97+101=575
75+71+151+80+146+138+101=762

Properties of greedy best-first search

- ***Complete?***
  - No – can get stuck in loops,
  - e.g., Iasi → Neamt → Iasi → Neamt →…

Properties of greedy best-first search

- ***Complete?*** **No – can get stuck in loops,**
  - e.g., Iasi → Neamt → Iasi → Neamt → …
- ***Time?*** ***O(b^m) – worst case*** **(like Depth First Search)**
  - But a good heuristic can give dramatic improvement of *average cost*
- ***Space?*** ***O(b^m)*** **– priority queue, so worst case: keeps all (unexpanded) nodes in memory**
- ***Optimal?*** **No**

A* search

- **Best-known form of best-first search.**
- **Key Idea: avoid expanding paths that are already expensive, but expand most promising first.**
- ***Simple idea*: $f(n)=g(n)+h(n)$**
  - $g(n)$ the cost (so far) to *reach* the node
  - $h(n)$ estimated cost to *get from the node to the goal*
  - $f(n)$ estimated *total cost* of path through *n* to goal
- **Implementation: Frontier queue as priority queue by increasing $f(n)$ *(as expected…)***

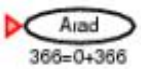
Admissible heuristics

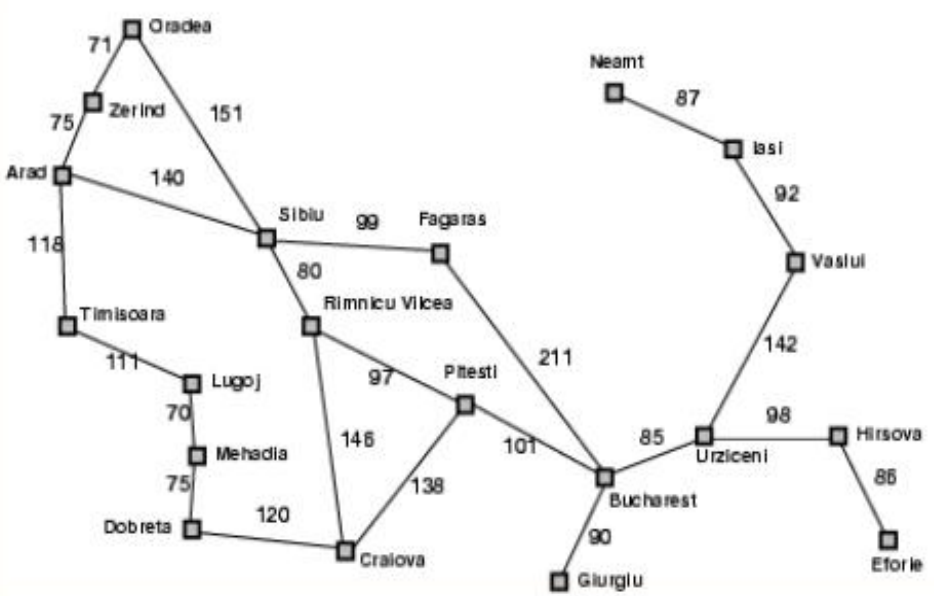
- **A heuristic $h(n)$ is *admissible* if it *never overestimates* the cost to reach the goal; i.e. it is *optimistic***
  - Formally: $\forall n$, n a node:
    1. $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from n
    2. $h(n) \geq 0$ so $h(G)=0$ for any goal G.

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

Theorem: If $h(n)$ is *admissible*, A* using Tree Search is *optimal*

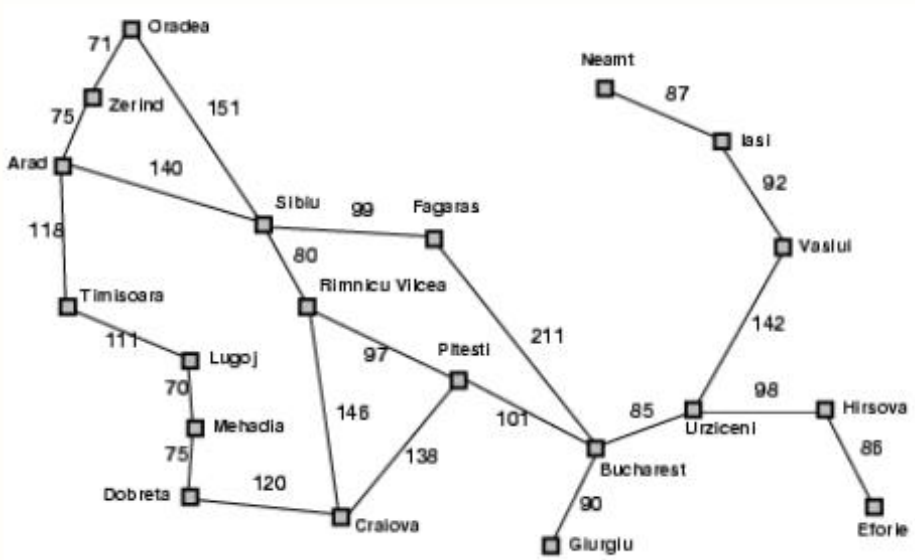
A* search example





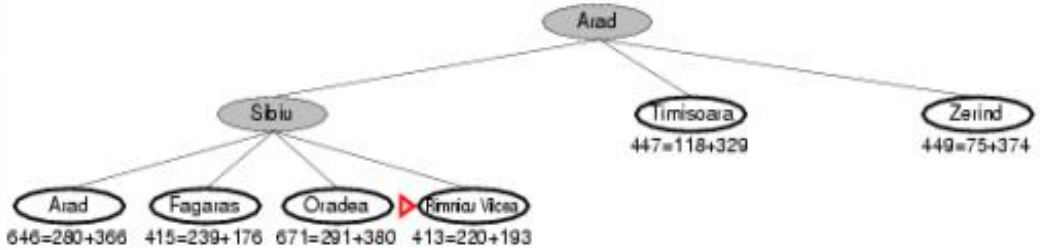
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

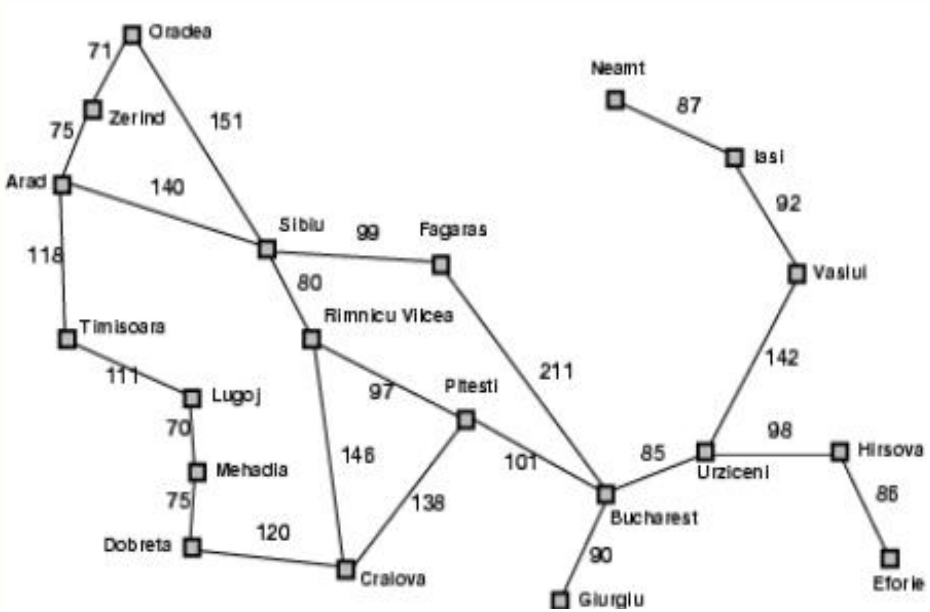
A* search example



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

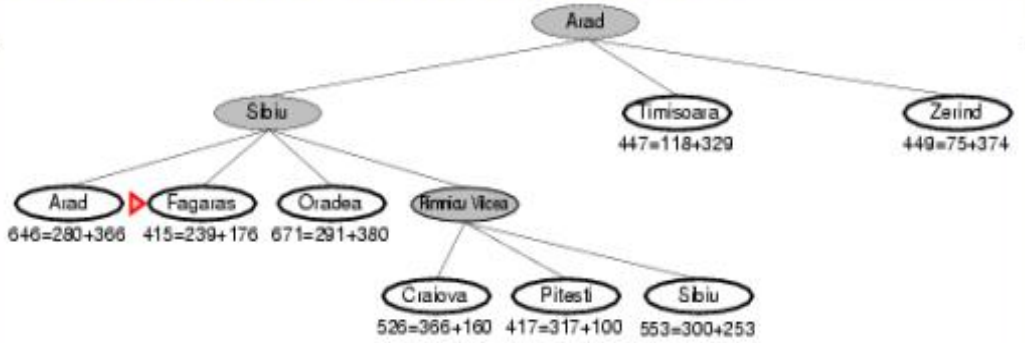
A* search example

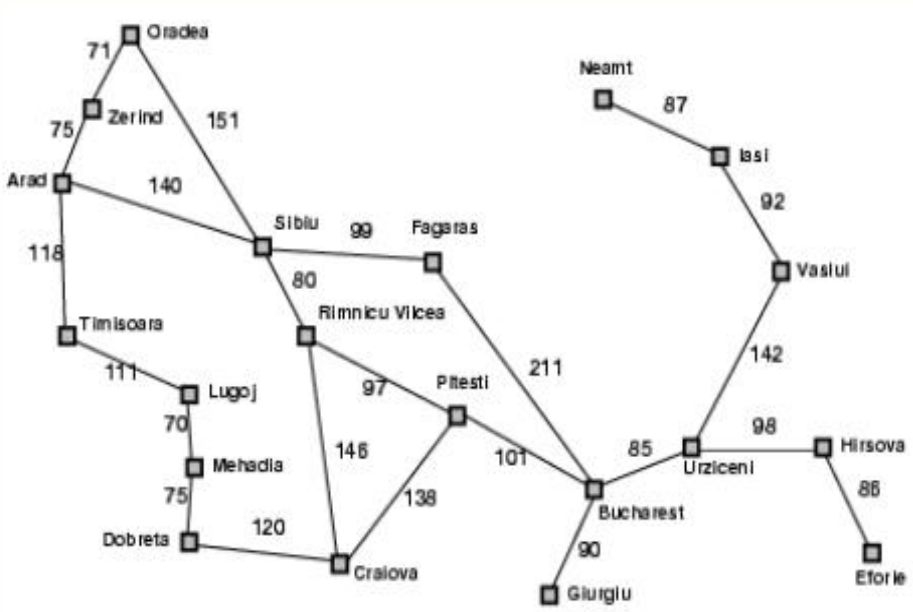




Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

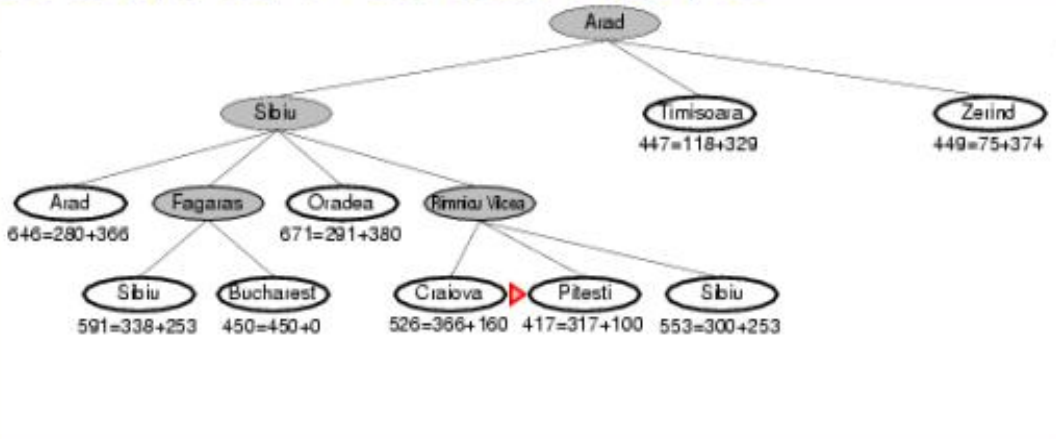
A* search example

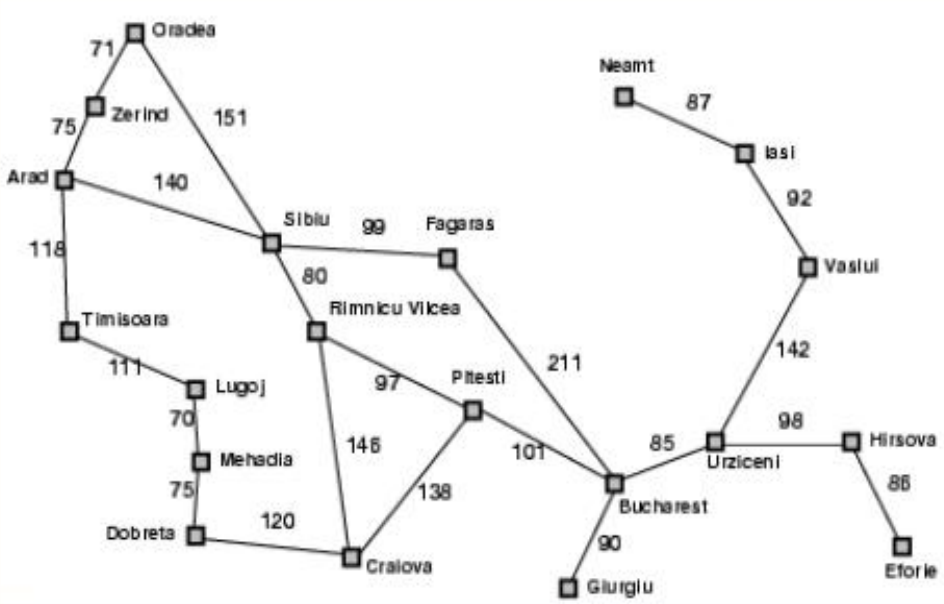




Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

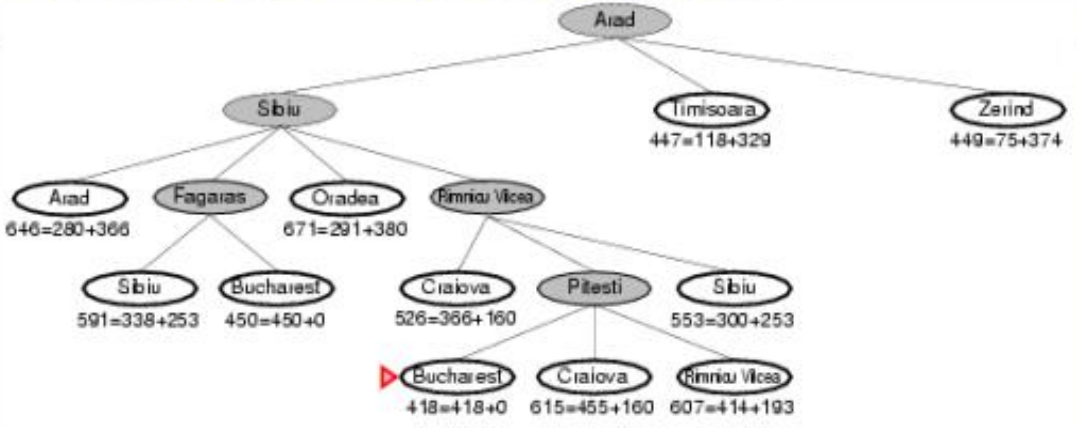


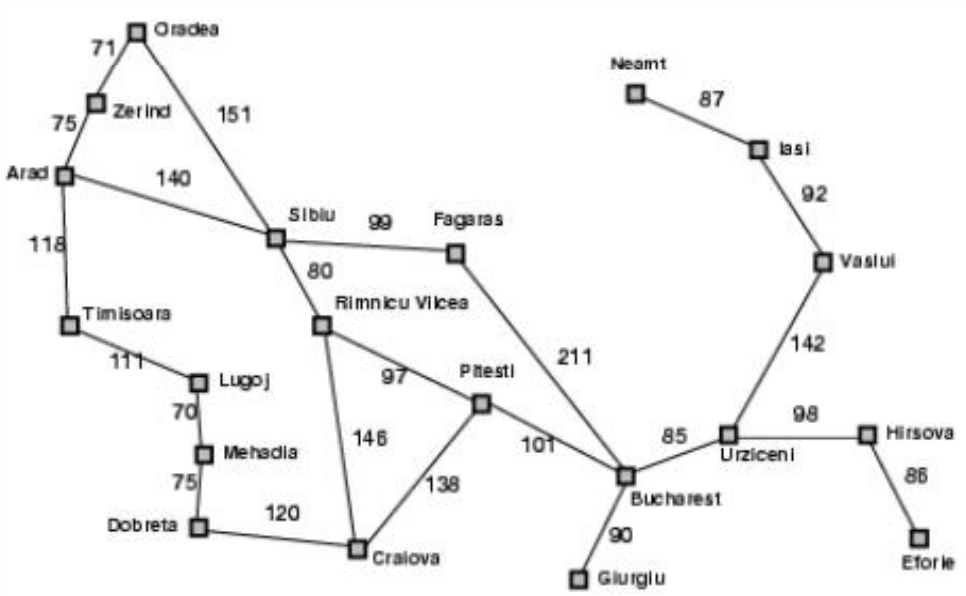


Straight-line distance to Bucharest

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

Properties of A*

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

  except if: $|h(n) - h^*(n)| \leq O(\log h^*(n))$
- Optimal? Yes
- *Optimally Efficient*: Yes (no algorithm with the same heuristic is guaranteed to expand fewer nodes)

A* search, evaluation

- **Completeness: YES**
  - Since bands of increasing *f* are added
  - As long as *b* is finite
    - (guaranteeing that there aren't infinitely many nodes n with $f(n) < f(G)$)

A* search, evaluation

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

A* search, evaluation

- **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