Informed search

Classes of Search

Blind (uninformed)	**Depth-First** **Breadth-First** **Iterative-Deepening**	Systematic exploration of whole tree until the goal is found.
Heuristic (informed)	**Hill-Climbing** **Best-First** **Beam**	Uses heuristic measure of goodness of a node,e.g. estimated distance to goal.
Optimal (informed)	**Branch&Bound** **A***	Uses path "length" measure. Finds "shortest" path. A* also uses heuristic

INFORMED SEARCH STRATEGIES

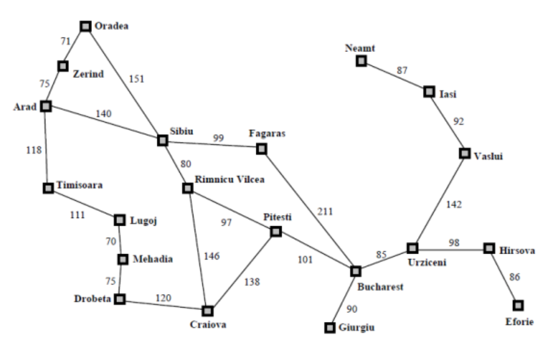
- One that uses problem specific knowledge beyond the definition of the problem itself.
- Find solutions more efficiently than an uninformed strategy.

Best-first search

- Uses 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 increasing order of cost.
- Special cases:
  - Greedy best-first search
  - A^* search

- Most best-first algorithms include as a component of **f** a **heuristic function denoted by h(n)**
- **h(n)** = estimated cost of the cheapest path from the **state at node n** to a **goal state**

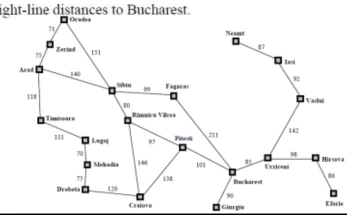
A simplified road map of part of Romania



Romania with straight-line distance

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

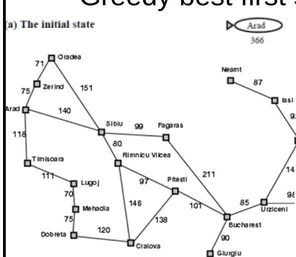
Values of h_{SLD}—straight-line distances to Bucharest.



Greedy Best first search

- f(n) = estimate of cost from n to goal
- **f(n) = straight-line distance from n to Bucharest**
- Greedy best-first search expands the node that appears **to be closest to goal.**

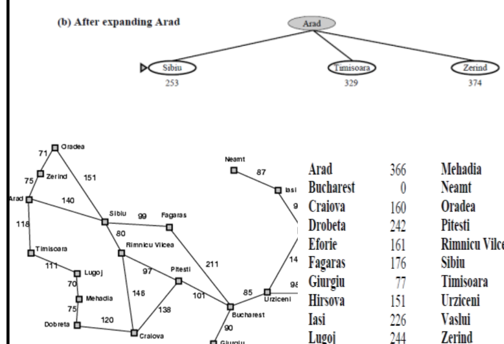
Greedy best first search example



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

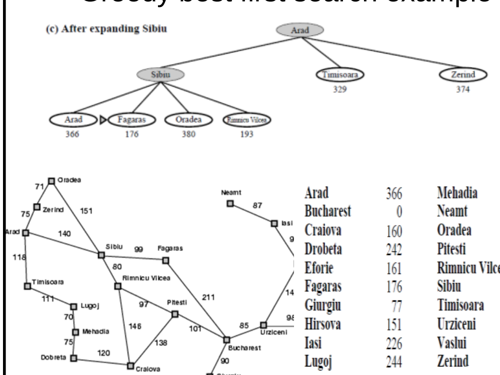
Values of h_{SLD}—straight-line distances to Bucharest.

Greedy best first search example



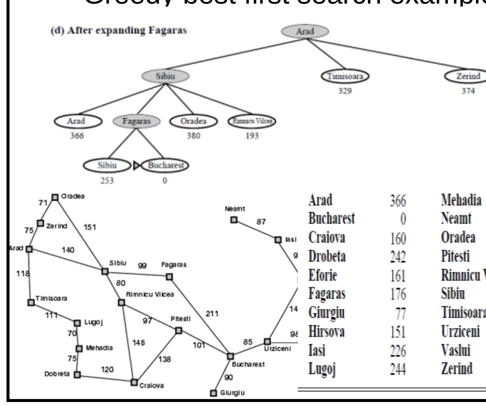
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Greedy best first search example



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

Greedy best first search example

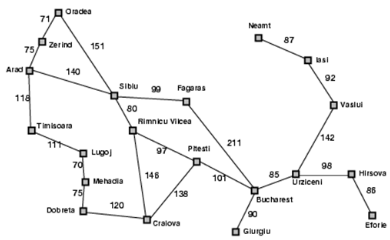


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Properties of greedy best-first search

- Can stuck in loops.
- Optimal? No

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



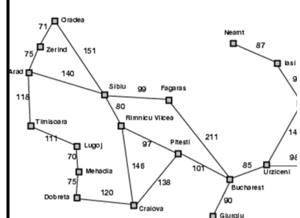
A* search

- Idea: avoid expanding paths that are already expensive
- Evaluation function f(n) = g(n) + h(n)
- g(n) = 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)

Admissible heuristics

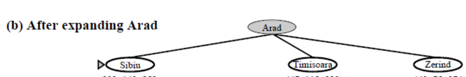
- A heuristic h(n) is admissible if for every node n, $h(n) \le h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n.
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

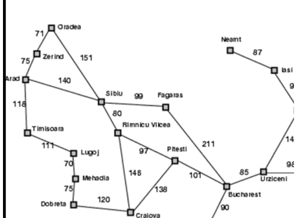
A* search example



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

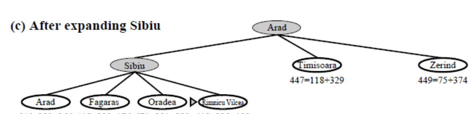
A* search example

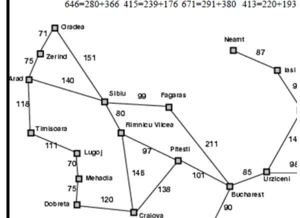




Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

A* search example

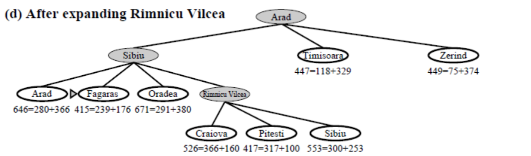


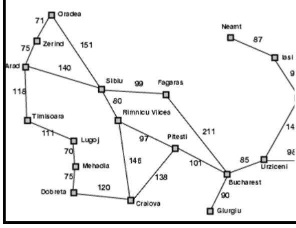


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

A* search example

(d) After expanding Rimnicu Vilcea

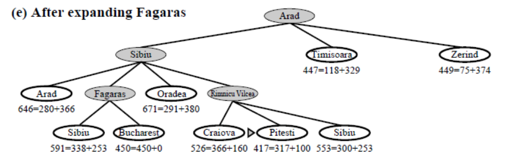


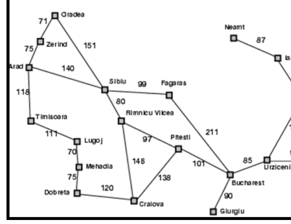


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

A* search example

(e) After expanding Fagaras

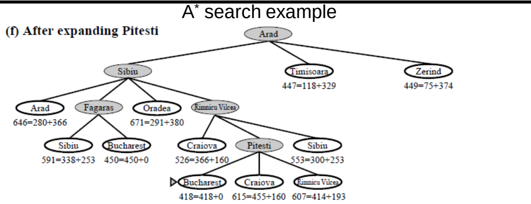


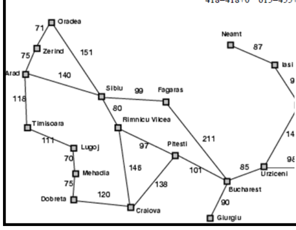


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

A* search example

(f) After expanding Pitesti





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

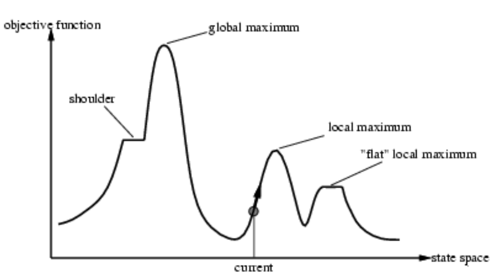
Hill-climbing search

- If there exists a **successor state s** for the **current state n** such that
  - h(s) < h(n)
  - h(s) <= h(t) for all the successors t of n,

  then move from n to s. Otherwise, halt at n.
- Looks one step ahead to determine if any successor is better than the current state; if there is, move to the best successor.
- Similar to Greedy search, it uses h but does not allow backtracking or jumping to an alternative path since it doesn't "**remember**" where it has been.
- Not complete since the search will terminate at "local minima", "plateaus", and "ridges".

Hill-climbing search

- Problem: depending on initial state, can get stuck in local maxima



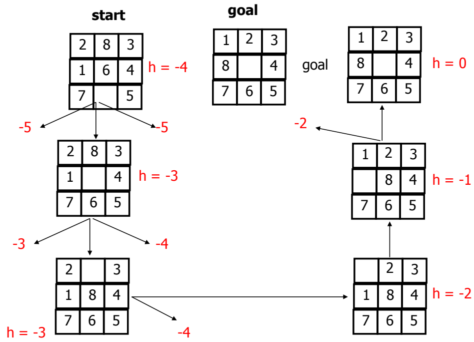
Hill-climbing search

- "Like climbing Everest in thick fog with amnesia (memory loss)

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
       neighbor ← a highest-valued successor of current
       if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
       current ← neighbor
```

Hill climbing example



f(n) = -(number of tiles out of place)

Drawbacks of hill climbing

- Problems:
  - **Local Maxima**: peaks that aren't the highest point in the space
  - **Plateaus:** the space has a broad flat region that gives the search algorithm no direction
  - **Ridges:** flat like a plateau, but with dropoffs to the sides; steps to the North, East, South and West may go down, but a step to the NW may go up.
- Remedy:
  - Random restart.
- Some problem spaces are great for hill climbing and others are terrible.