# INFORMED SEARCH

*Heuristic search*

# INFORMED SEARCH STRATEGIES

- Uses domain-specific hints about the location of goals.

- Greedy best-first search
- A* search
- Search contours
- Satisficing search
- Memory-bound search
- Bidirectional heuristic search

# INFORMED (HEURISTIC) SEARCH

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

- Straight line between Arad to Bucharest!

# GREEDY BEST-FIRST SEARCH

- Tries to expand the node that is closest to the goal.
- Let h(n) be straight line distance heuristic $h_{SLD}(n)$.

# GREEDY BEST-FIRST SEARCH

| | | | |
|---|---|---|---|
| **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 |

**Figure 3.22** Values of $h_{SLD}$—straight-line distances to Bucharest.

# BEST-FIRST SEARCH

CHOOSE A NODE, N, WITH MINIMUM VALUE OF SOME EVALUATION FUNCTION, F(N)=H(N).

```
function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
  node ← NODE(STATE=problem.INITIAL)
  frontier ← a priority queue ordered by f, with node as an element
  reached ← a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s ← child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s] ← child
        add child to frontier
  return failure

function EXPAND(problem, node) yields nodes
  s ← node.STATE
  for each action in problem.ACTIONS(s) do
    s' ← problem.RESULT(s, action)
    cost ← node.PATH-COST + problem.ACTION-COST(s, action, s')
    yield NODE(STATE=s', PARENT=node, ACTION=action, PATH-COST=cost)
```

# GREEDY BEST-FIRST SEARCH

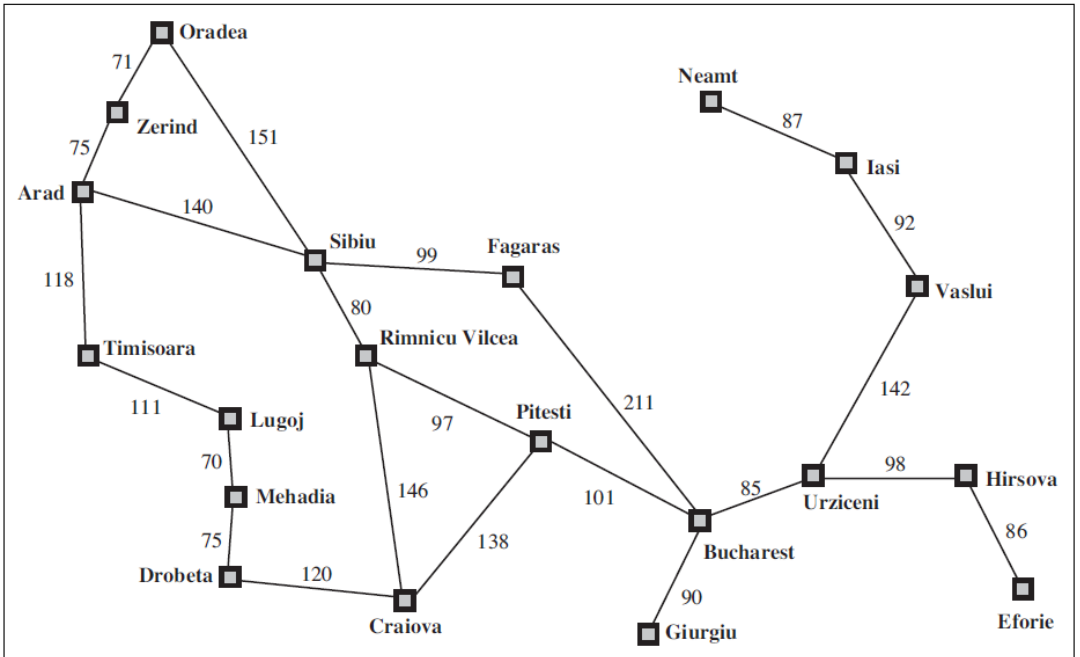


**Figure 3.2** A simplified road map of part of Romania.

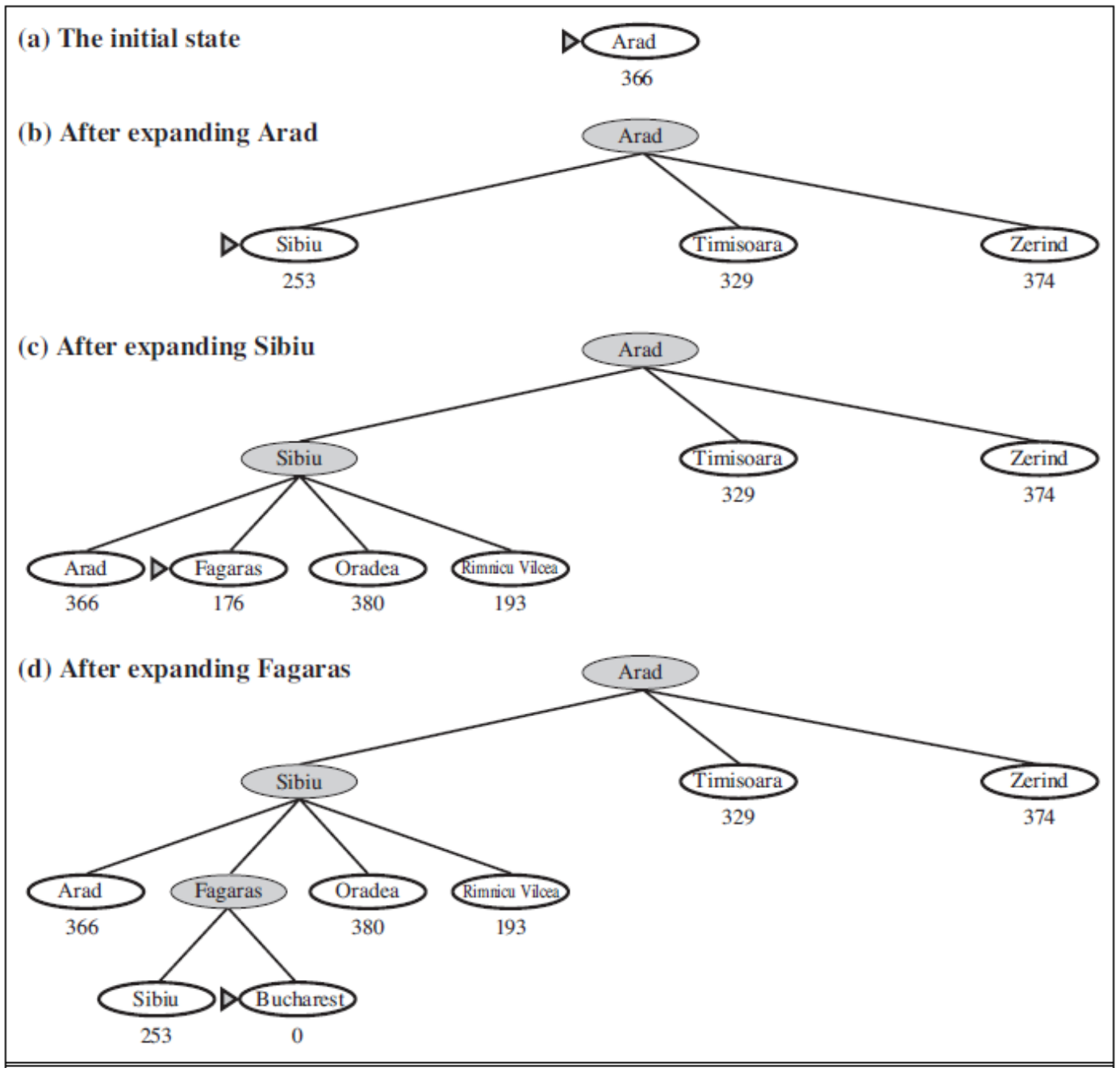


**Figure 3.23** Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic $h_{SLD}$. Nodes are labeled with their $h$-values.

# GREEDY BEST-FIRST SEARCH

- Challenges:
- Not optimal
- Tree search is not optimal and incomplete.
- Worst-case time and space complexity for Tree version is $O(b^m)$. m is maximum depth.

- Graph search version is complete in finite space.