भारतीय सूचना प्रौद्योगिकी, अभिकल्पन एवं विनिर्माण संस्थान, जबलपुर
पी.डी.पी.एम.

PDPM
Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

CS 3011: Artificial Intelligence

Solving Problems by Searching

Instructors: Dr. Durgesh Singh

CSE Discipline, PDPM IIITDM, Jabalpur -482005

Informed (Heuristic) Search

Informed (Heuristic) Search Strategies

- It uses domain-specific hints about the location of goals
    - We can find solutions more efficiently than an uninformed strategy.
- The hints come in the form of a heuristic function $h(n)$.
- Heuristic function estimates "how close a state is to the goal state".

  $h(n)$ = estimated cost of cheapest path from node n to a goal

- We can consider $h(n)$ to be arbitrary, non-negative, problem-specific functions, with one constraint: if n is a goal node, then $h(n) = 0$.
- For example – Manhattan distance, Euclidean distance, etc.

Informed (Heuristic) Search Strategies..

- The general approach we will consider is called best-first search.
- Best-first search is an instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on an evaluation function, f(n)
  - The node with the lowest f(n) is expanded first.
  - The implementation of best-first search is identical to that for uniform-cost search except for the use of f instead of g to order the priority queue.

Best-First Search

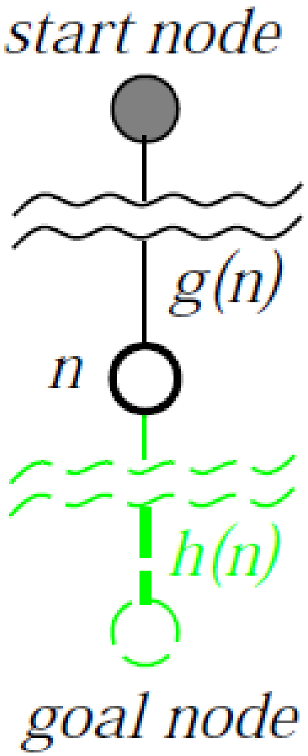
```
function TREE-SEARCH(problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  loop do
     if the frontier is empty then return failure
     choose a leaf node and remove it from the frontier  // Lowest f value
     if the node contains a goal state then return the corresponding solution
     expand the chosen node, adding the resulting nodes to the frontier
```

- Special cases:
  - Uniform Cost Search (uninformed)
  - Greedy best-first Search (informed)
  - A* Search (informed)

Evaluation Function

- g(n) = exact cost so far to reach n
- h(n) = estimated cost of the cheapest path from the state at node n to a goal state.

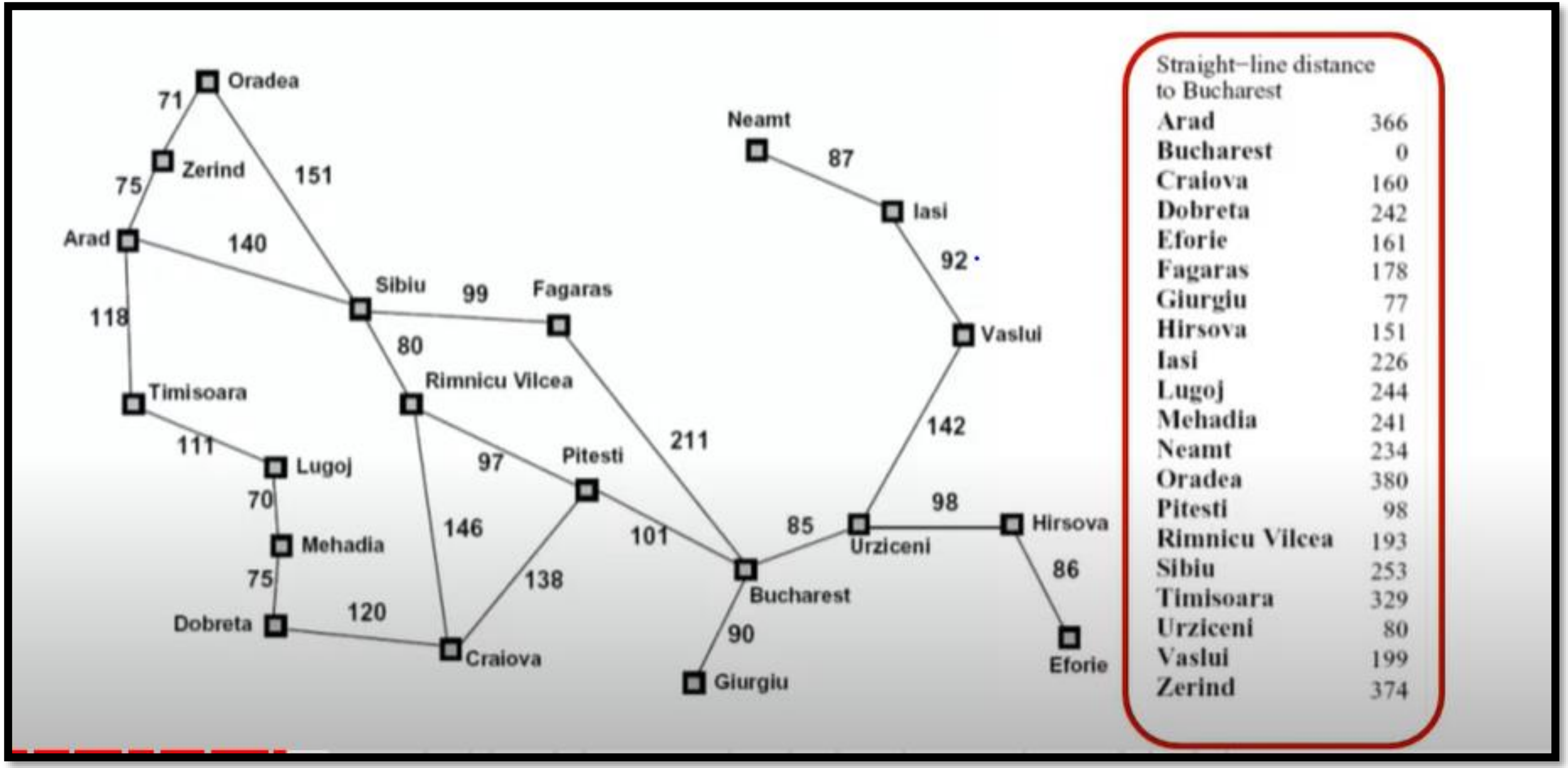
- Special cases:
  - Uniform Cost Search: f(n) = g(n)
  - Greedy best-first Search: f(n) = h(n)
  - A* Search: f(n) = g(n) + h(n)



Greedy Best-First Search

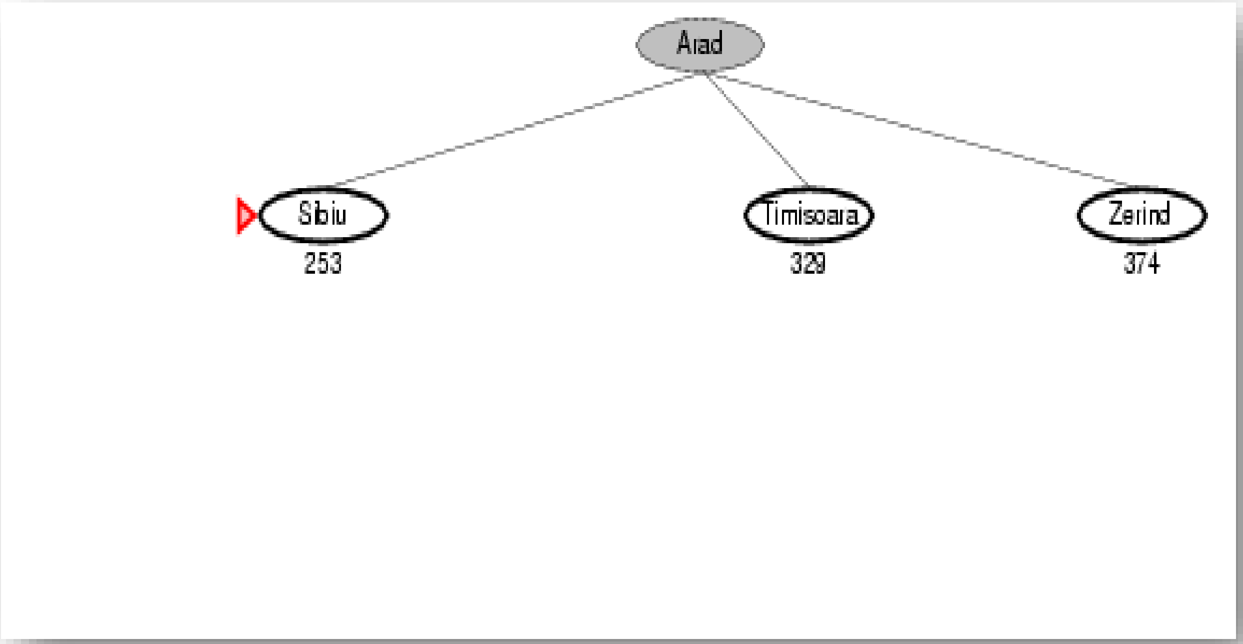
- Greedy best-first search tries to expand the node that is appears to be closest to the goal.
    - On the grounds that this is likely to lead to a solution quickly.
- Thus, it evaluates nodes by using just the heuristic function: $f(n) = h(n)$
- Greedy algorithms often perform very well. They tend to find good solutions quickly, although not always optimal ones.
- E.g., $h_{SLD}(n)$ = straight-line distance from n to Bucharest

Romania - Step Costs in Miles

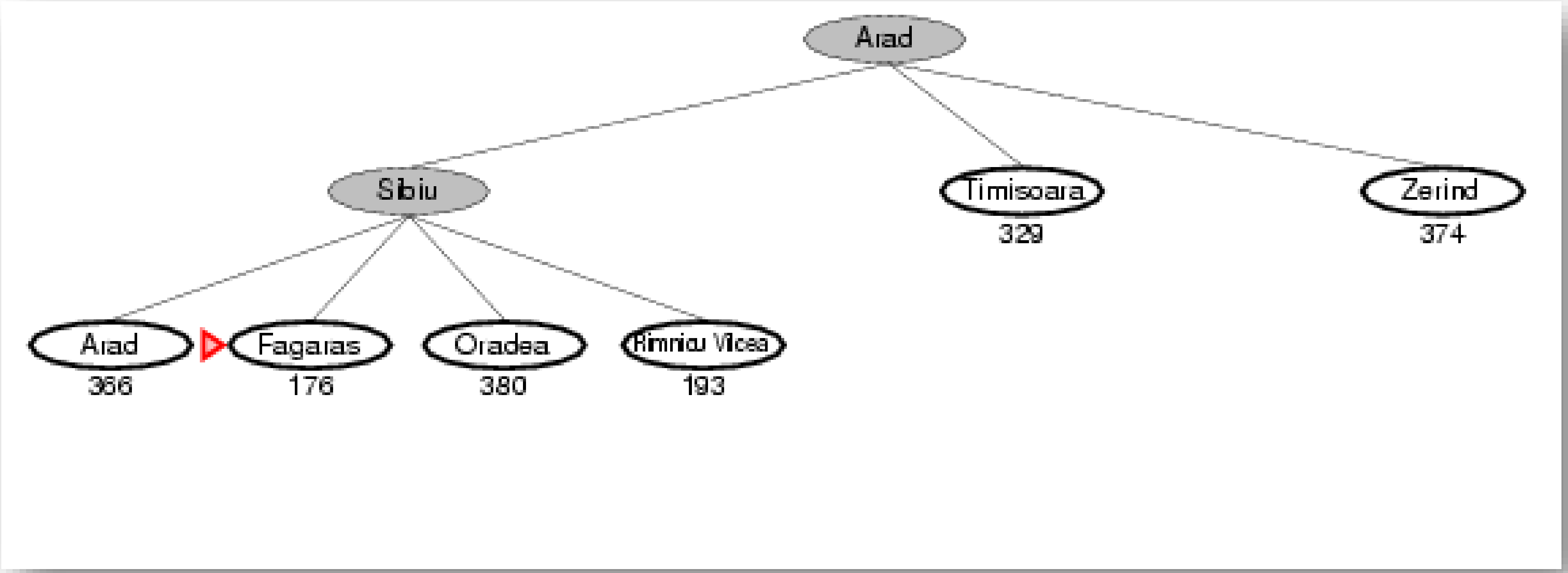


Greedy Best-First search example (Tree Search)

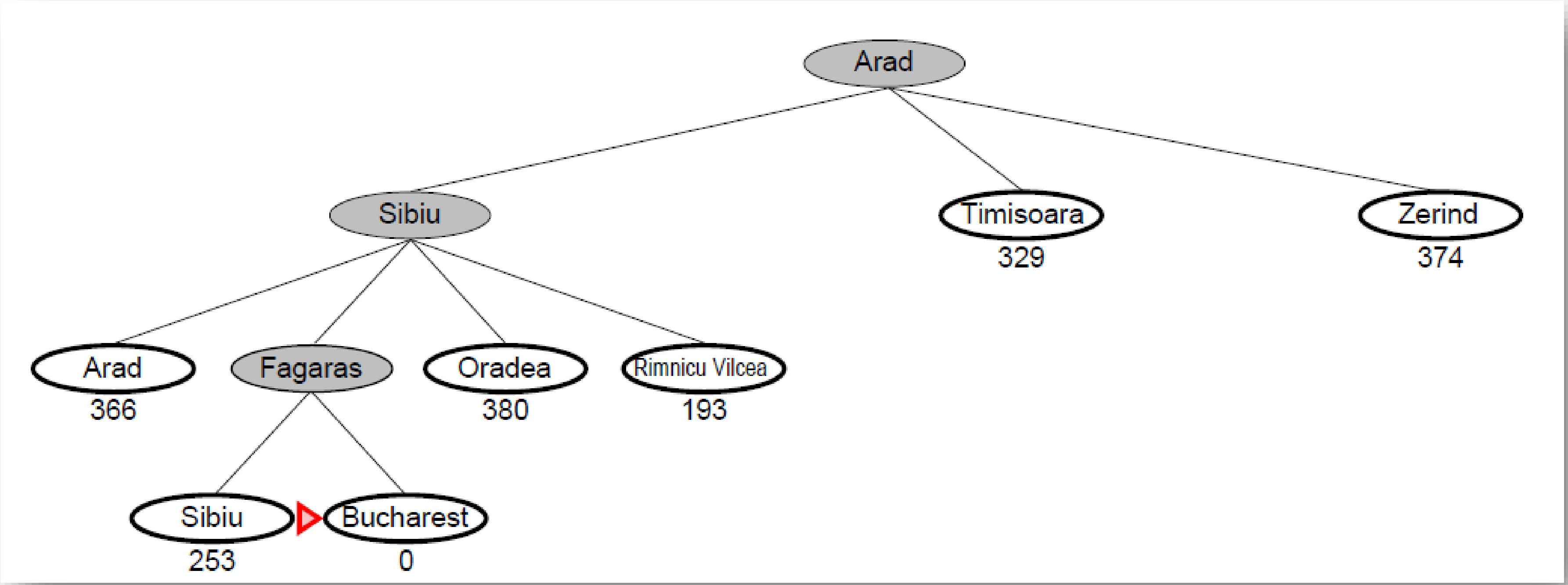
Greedy Best-First search example



Greedy Best-First search example

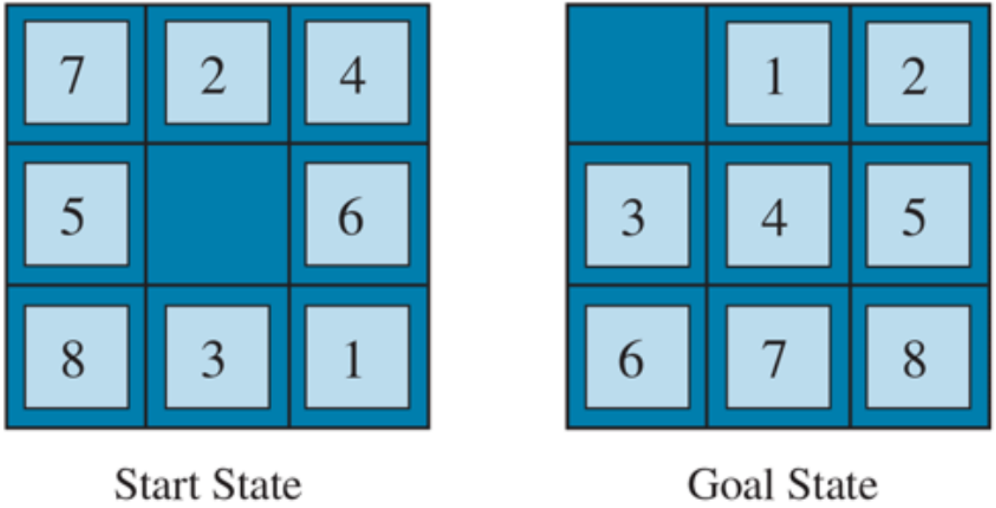


Greedy Best-First search example



Example 2:Gready search on 8 puzzle problem

- h1(n): number of misplaced tiles // blank is not included
- h2(n): sum of the distances of tiles from their goal position ("Manhattan distance")



Properties of Greedy Best-First search

- Complete? No – can get stuck in loops, e.g., with Oradea as goal and start from Iasi:
  - Iasi → Neamt → Iasi → Neamt →
  - Complete in finite space with repeated state checking
- Time? $O(b^m)$, but a good heuristic can give dramatic improvement
- Space? $O(b^m)$ -- keeps all nodes in memory
- Optimal? No.