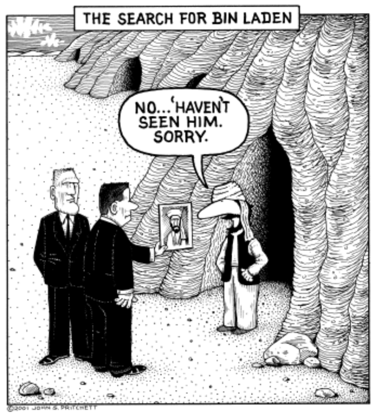
CSE 473

Chapter 4

Informed Search



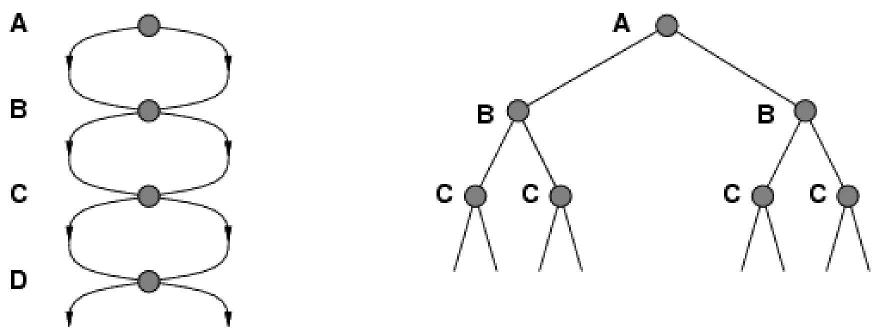
Last Time

Blind Search

- BFS
- UC-BFS
- DFS
- DLS
- Iterative Deepening
- Bidirectional Search

Repeated States

- **Failure to detect repeated states can turn a linear problem into an exponential one! (e.g., repeated states in 8 puzzle)**



- **Graph search algorithm: Store expanded nodes in a set called *closed* and only add new nodes to the fringe**

3

Graph Search

```
function GRAPH-SEARCH( problem, fringe) returns a solution, or failure
   closed ← an empty set
   fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
   loop do
       if fringe is empty then return failure
       node ← REMOVE-FRONT(fringe)
       if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
       if STATE[node] is not in closed then
           add STATE[node] to closed
           fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

4

Can we do better?

All these methods are slow (blind)

Solution ⇒ use problem-specific knowledge to guide search ("heuristic function")

⇒ "informed search"

5

Best-first Search

- Generalization of breadth first search
- Priority queue of nodes to be explored
- Evaluation function f(n) used for each node

```
Insert initial state into priority queue
While queue not empty
        Node = head(queue)
        If goal(node) then return node
        Insert children of node into pr. queue
```

6

Who’s on (best) first?

- Breadth first is best first
  With f(n) = depth(n)

- Dijkstra’s Algorithm is best first
  With f(n) = g(n)
  where g(n) = sum of edge costs from start to n

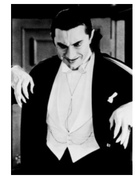
7

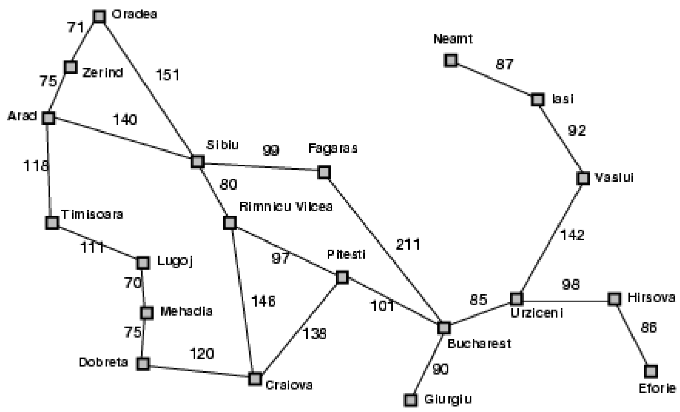
Greedy best-first search

- Evaluation function $f(n) = h(n)$ (heuristic) = estimate of cost from *n* to *goal*
- e.g., Route finding problems: $h_{SLD}(n)$ = straight-line distance from *n* to destination
- Greedy best-first search expands the node that appears to be closest to goal

8

Example: Lost in Romania



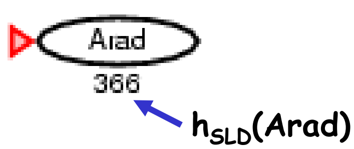


Straight–line distance to Bucharest

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

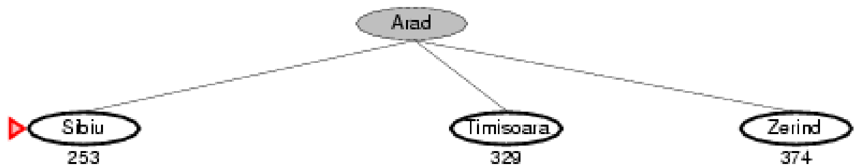
9

Example: Greedily Searching for Bucharest



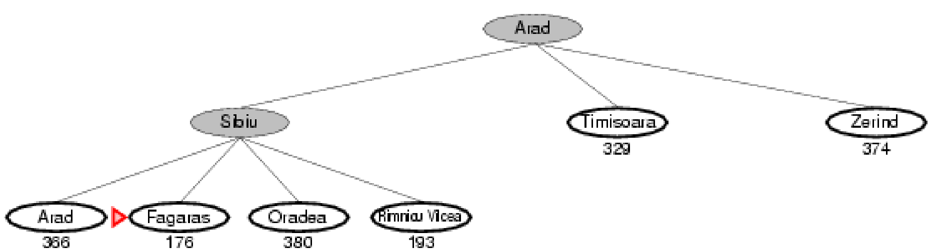
10

Example: Greedily Searching for Bucharest



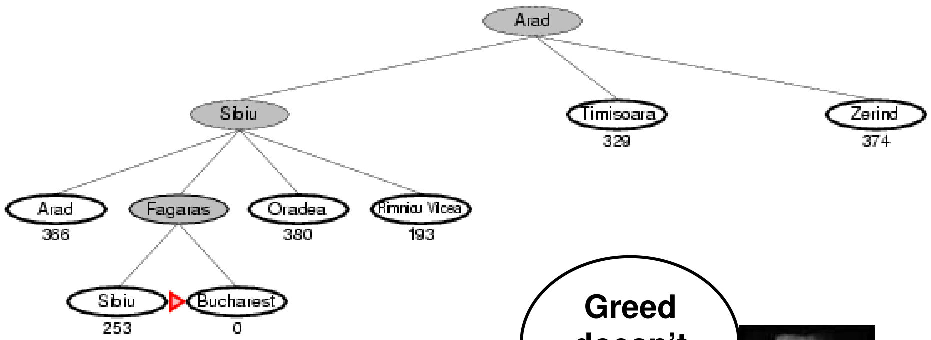
11

Example: Greedily Searching for Bucharest



12

Example: Greedily Searching for Bucharest



Not optimal!
Arad, Sibiu, Rimnicu Vilcea, Pitesti, Bucharest shorter

Properties of Greedy Best-First Search

- Complete? No – can get stuck in loops, e.g.,

  Iasi $\rightarrow$ Neamt $\rightarrow$ Iasi $\rightarrow$ Neamt $\rightarrow$

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

A* Search

(Hart, Nilsson & Rafael 1968)

Best first search with f(n) = g(n) + h(n)

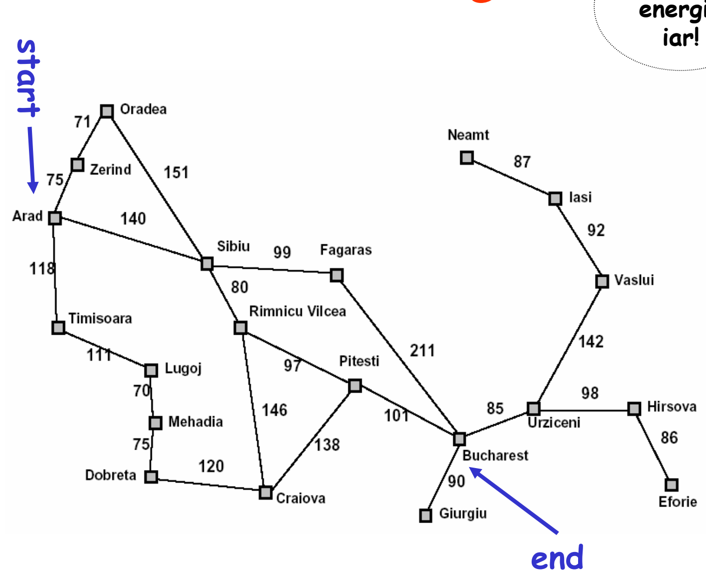
g(n) = sum of edge costs from start to n
+ heuristic function h(n) = estimate of lowest cost path from n to goal

If h(n) is "**admissible**" then search will be optimal

Underestimates cost
of any solution which
can be reached from node

15

Back in Romania Again

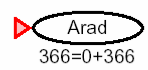


Straight−line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

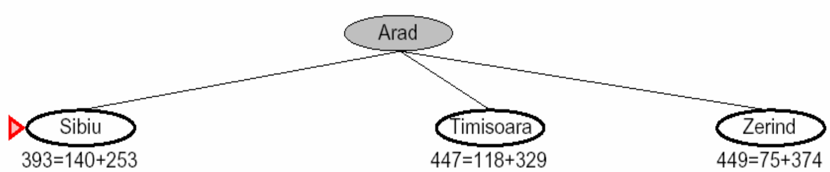
16

A* Example



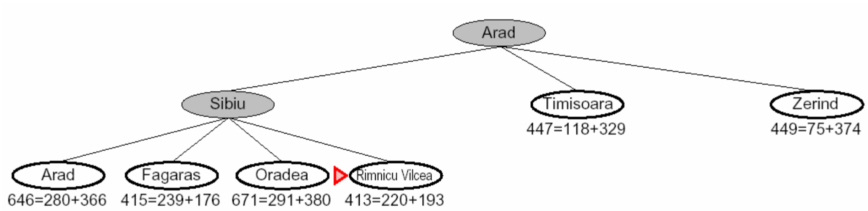
17

A* Example



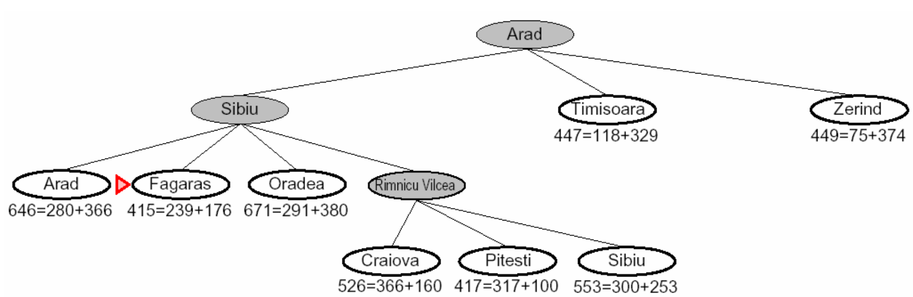
18

A* Example



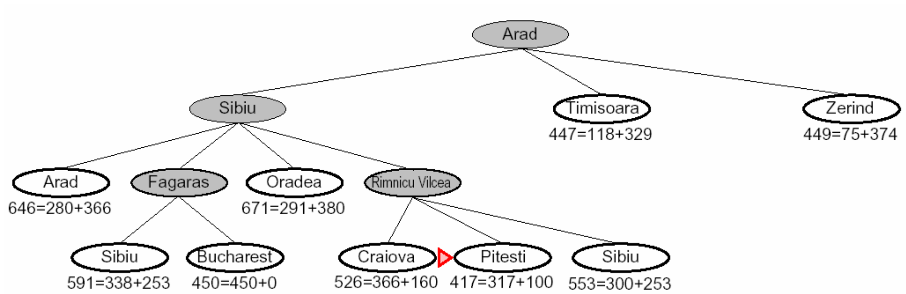
19

A* Example



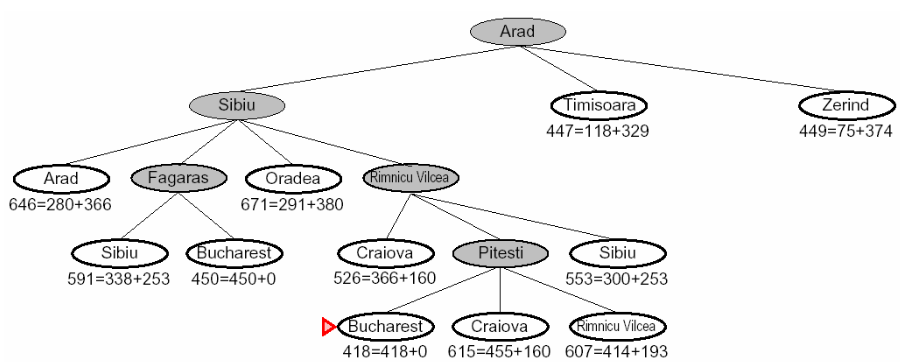
20

A* Example



21

A* Example



22

Admissible heuristics

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

  $$h(n) \leq h^*(n)$$

  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**

23

Admissible Heuristics

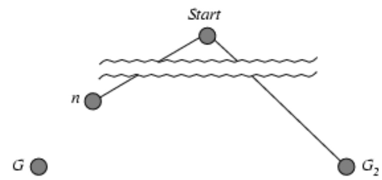
- Example: Straight Line Distance heuristic $h_{SLD}(n)$ is *admissible* (never overestimates the actual road distance)

- Theorem: If $h(n)$ is admissible, A^* using TREE-SEARCH is optimal.

24

Optimality of A* (proof)

- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let *n* be an unexpanded node in the fringe such that *n is on a shortest path to an optimal goal G.*

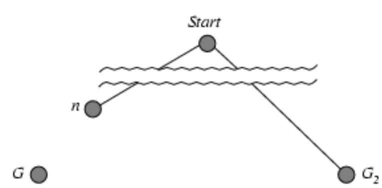


- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $> g(G)$ since G_2 is suboptimal
- $f(G) = g(G)$ since $h(G) = 0$
- $f(G_2) > f(G)$ from above

25

Optimality of A* (cont.)

- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let *n* be an unexpanded node in the fringe such that *n is on a shortest path to an optimal goal G.*



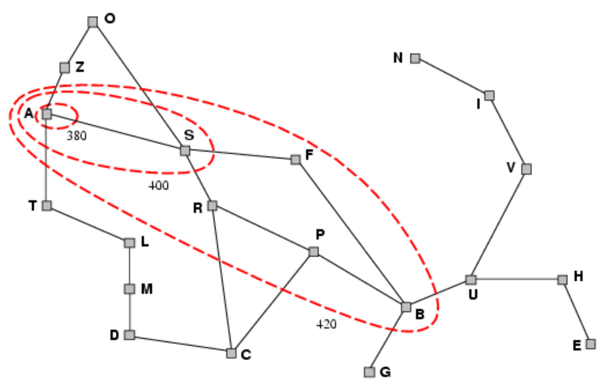
- $f(G_2) > f(G)$ from above
- $h(n) \leq h^*(n)$ since h is admissible
- $g(n) + h(n) \leq g(n) + h^*(n)$
- $f(n) \leq f(G)$

Hence $f(n) < f(G_2)$, *i.e.*, A^* will never select G_2 for expansion.

26

Optimality of A*

- A* expands nodes in order of increasing *f* value
- Gradually adds "*f*-contours" of nodes



27

Okay, proof is done! Time to wake up…

28

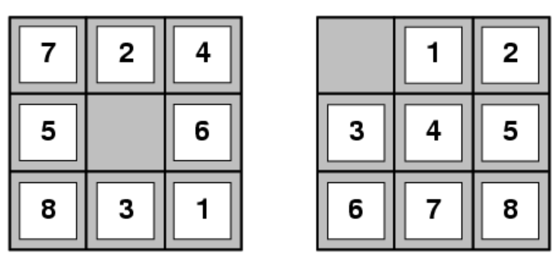
Properties of A*

- Complete? Yes (unless there are infinitely many nodes with $f \leq f(G)$)
- Time? Exponential (for most heuristic functions in practice)
- Space? Keeps all generated nodes in memory (exponential number of nodes)
- Optimal? Yes

Admissible heuristics

E.g., for the 8-puzzle, what are some heuristic functions?

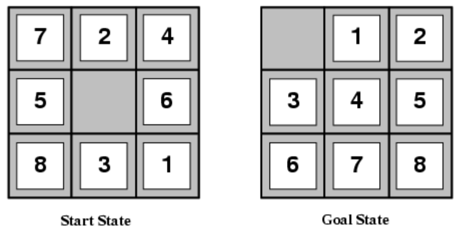
- $h_1(n) = ?$
- $h_2(n) = ?$



Start State Goal State

Admissible heuristics

E.g., for the 8-puzzle:
- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance (no. of squares from desired location of each tile)



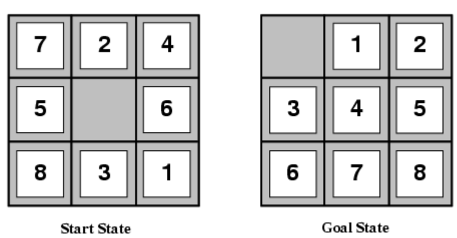
S

- $h_1(S)$ = ?
- $h_2(S)$ = ?

31

Admissible heuristics

E.g., for the 8-puzzle:
- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance (no. of squares from desired location of each tile)



- $h_1(S)$ = ? 8
- $h_2(S)$ = ? 3+1+2+2+2+3+3+2 = 18

32

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible) then h_2 dominates h_1
- h_2 is better for search

Dominance

- E.g., for 8-puzzle heuristics h_1 and h_2, typical search costs (average number of nodes expanded for solution depth d):

- *d=12* IDS = 3,644,035 nodes
  $A^*(h_1)$ = 227 nodes
  $A^*(h_2)$ = 73 nodes
- *d=24* IDS = too many nodes
  $A^*(h_1)$ = 39,135 nodes
  $A^*(h_2)$ = 1,641 nodes

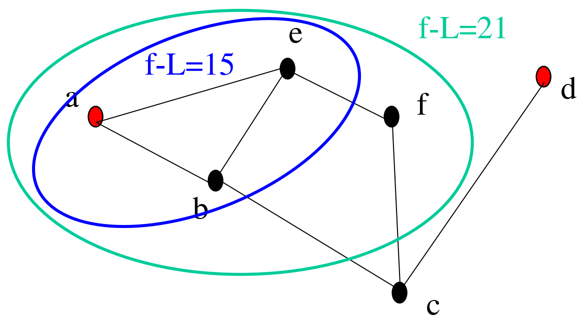
Iterative-Deepening A*

- Like iterative-deepening search, but...
- Depth bound modified to be an **f-limit**

  Start with limit = h(start)

  Prune any node if f(node) > f-limit

  Next f-limit=min-cost of any node pruned



35

Next Time

- How to climb hills
- How to reach the top by annealing
- How to simulate and profit from evolution

36