CSE 473

Lecture 4

Informed Search

Last Time

Blind Search

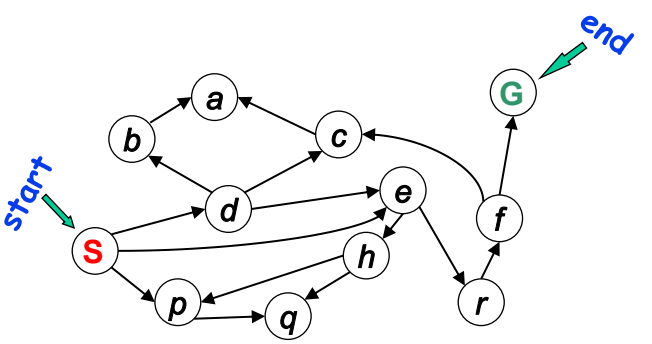
- BFS
- UC-BFS
- DFS
- DLS
- Iterative Deepening

Summary of algorithms

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes*	Yes*	No	Yes, if $l \geq d$	Yes
Time	b^d	$b^{\lceil C^*/\epsilon \rceil}$	b^m	b^l	b^d
Space	b^d	$b^{\lceil C^*/\epsilon \rceil}$	bm	bl	bd
Optimal?	Yes*	Yes*	No	No	Yes

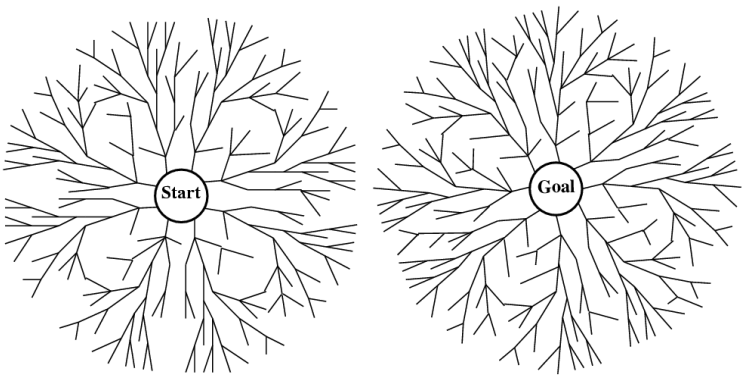
3

Forwards *vs.* Backwards Search



4

Bidirectional Search



Motivation: Search time $b^{d/2} + b^{d/2} << b^d$
(E.g., $10^8+10^8 = 2 \cdot 10^8 << 10^{16}$)
Can use breadth-first search or uniform-cost search
Hard for implicit goals e.g., goal = "checkmate" in chess

5

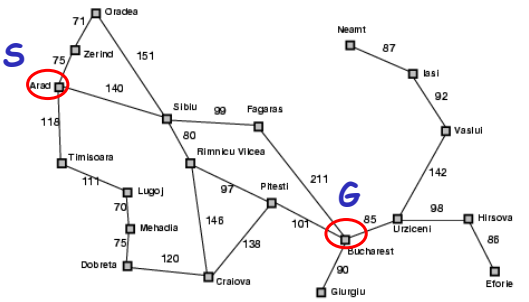
Can we do better?

Can we use problem-specific knowledge to speed up search and maintain optimality?

6

Informed Search

- General search problem: Actions have different costs



- Want to minimize *total cost* from start to goal
  Not just minimizing *path cost* like Uniform-cost search
- Idea: Use problem-specific knowledge to guide search by using "heuristic function"

7

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

8

Who's on (best) first?

Examples of best-first search:

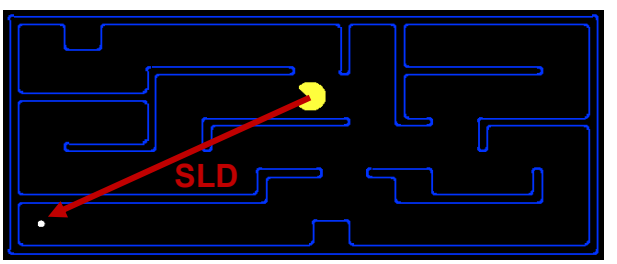
- Breadth-first search is best-first
  With f(n) = depth(n)

- Uniform-cost search is best-first
  With f(n) = g(n)
  where g(n) = path cost (sum of edge costs from start to n)

9

Greedy best-first search

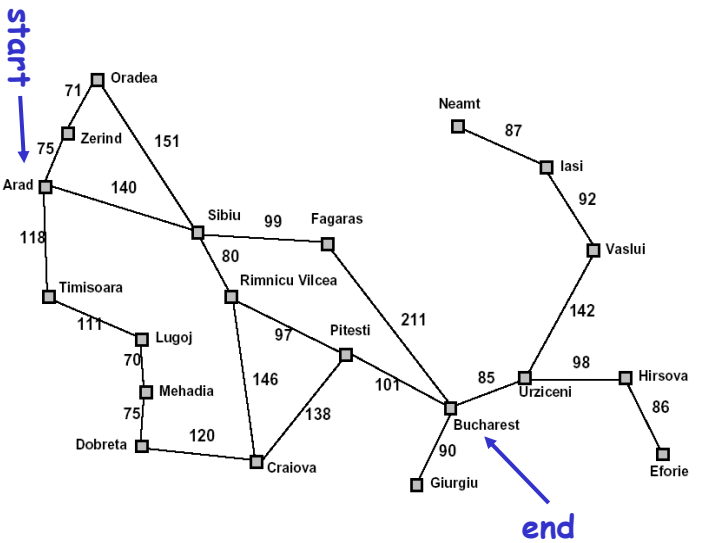
- Use a *heuristic* evaluation function *f(n)* = *h(n)* = estimate of cost from *n* to *goal*



- E.g., $h_{SLD}(n)$ = straight-line distance from *n* to destination
- Greedy best-first search expands the node that appears to be closest to goal

10

Example: Lost in Romania

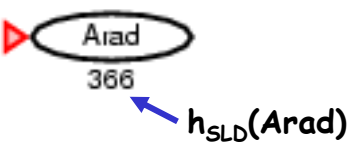


h(n)= SLD to Bucharest

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	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

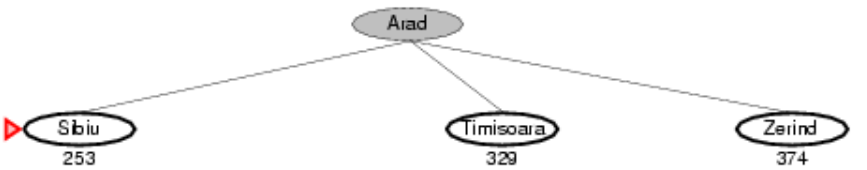
11

Example: Greedily Searching for Bucharest



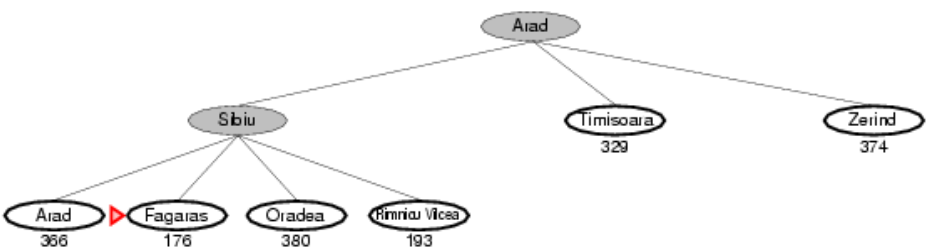
12

Example: Greedily Searching for Bucharest



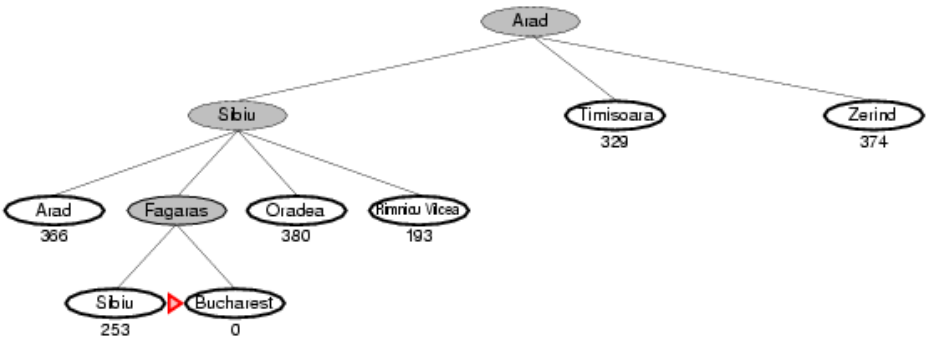
13

Example: Greedily Searching for Bucharest

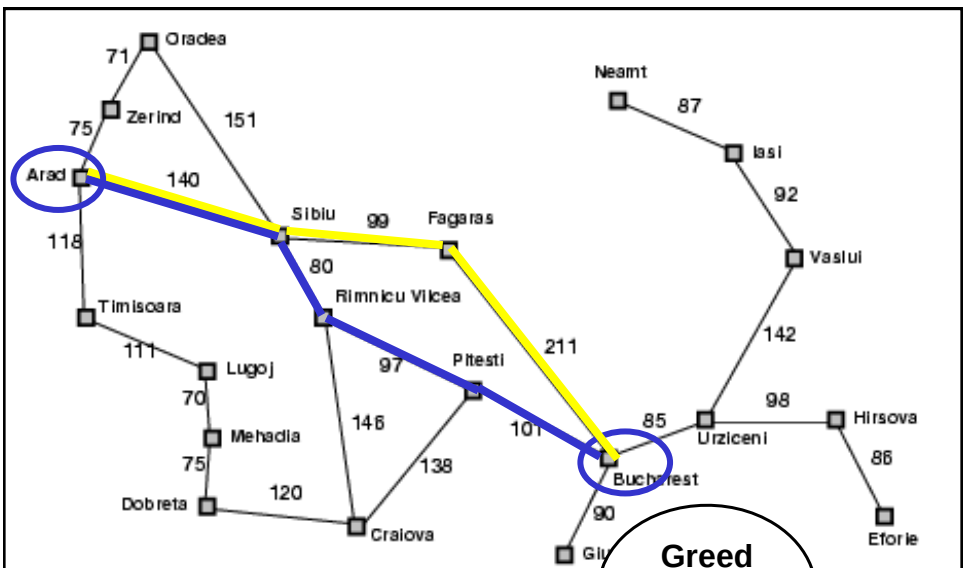


14

Example: Greedily Searching for Bucharest



15



Not optimal!
Yellow = greedy SLD-based search
Blue = optimal (418 versus 450)

Properties of Greedy Best-First Search

- Complete? No – can get stuck in loops (unless we keep an "explored" set)
- Time? $O(b^m)$, but a good heuristic can give dramatic improvement
- Space? $O(b^m)$ (nodes in priority queue + explored set)
- Optimal? No, as our example illustrated

17

A* Search

(Hart, Nilsson & Rafael 1968)

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

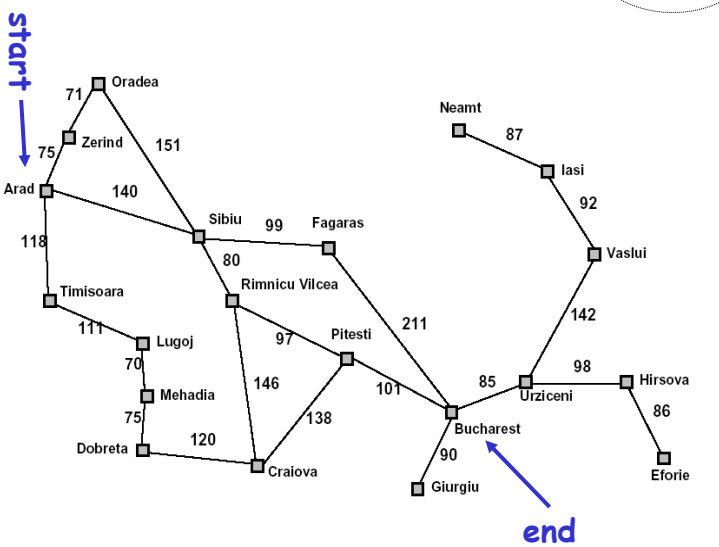
g(n) = sum of edge costs from start to n
+ heuristic function h(n) = estimate of lowest cost path from n to goal

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

Underestimates cost of any solution which can be reached from node

18

Back in Romania Again

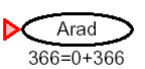


h(n)= SLD to Bucharest

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	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

19

A* Example

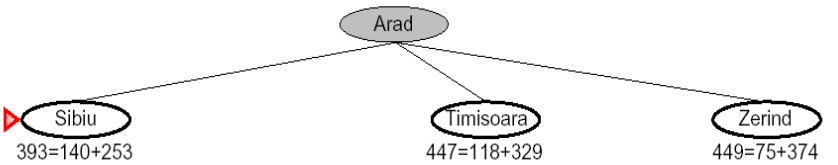


366=0+366

f(n)=g(n)+h(n)

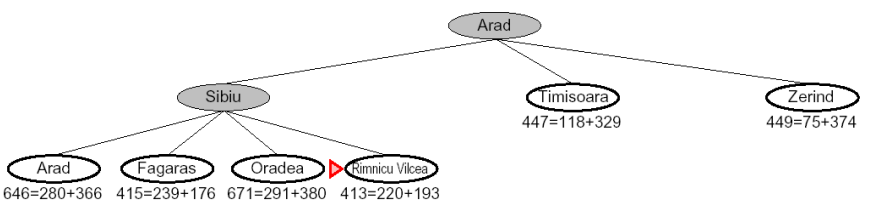
20

A* Example



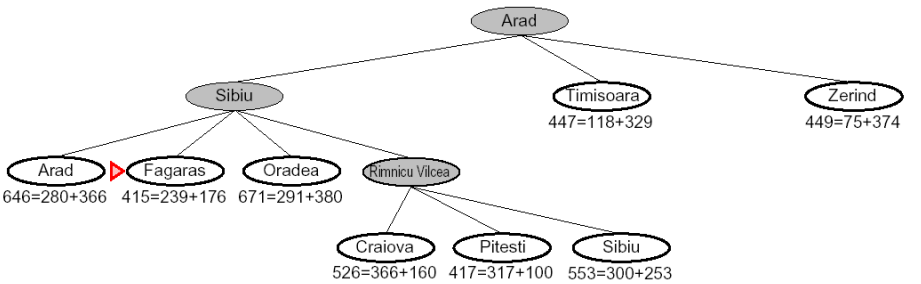
21

A* Example



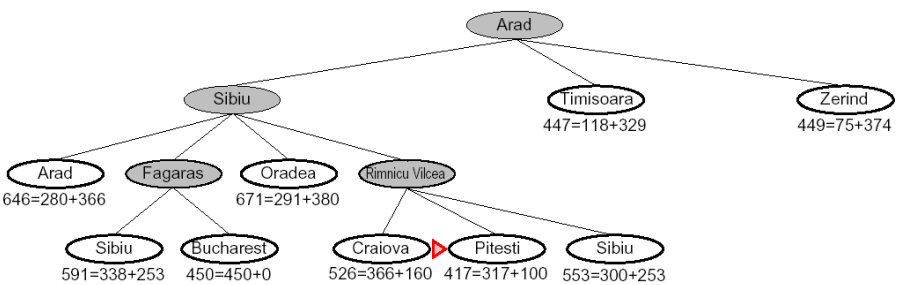
22

A* Example



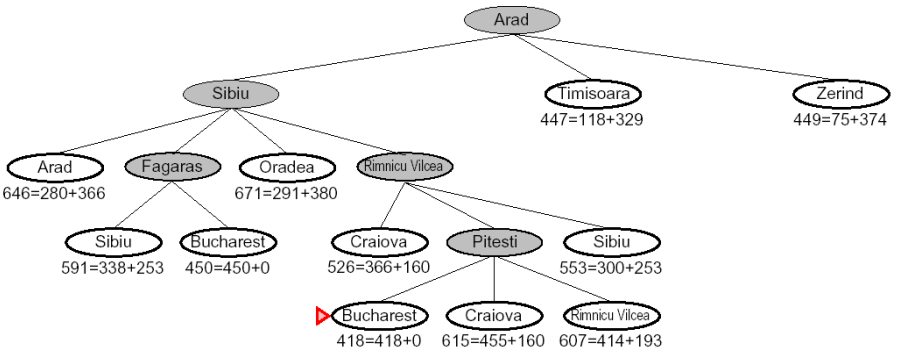
23

A* Example



24

A* Example



25

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**

26

Admissible Heuristics

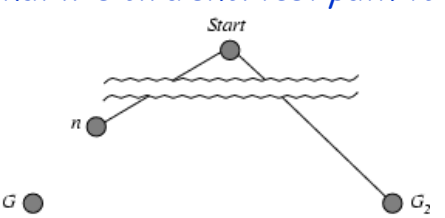
- Is the Straight Line Distance heuristic $h_{SLD}(n)$ *admissible?*
- Yes, it never overestimates the actual road distance

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

27

Optimality of A* (proof)

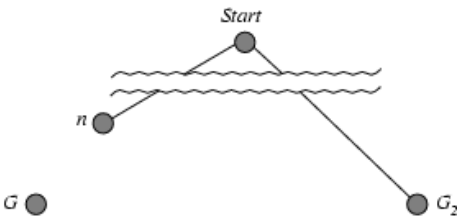
Suppose some suboptimal goal G_2 has been generated and is in the frontier. Let *n* be an unexpanded node in the frontier such that *n is on a shortest path to an optimal goal G.*



$f(G_2) = g(G_2)$		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

28

Optimality of A* (cont.)

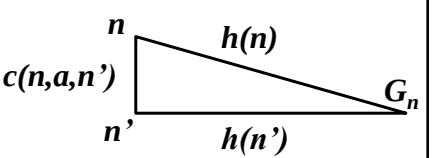


$f(G) < f(G_2)$ from prev slide
$h(n) \leq h^*(n)$ since h is admissible
$g(n) + h(n) \leq g(n) + h^*(n) = f(G)$
$f(n) \leq f(G) < f(G_2)$

Hence $f(n) < f(G_2) \Rightarrow A^*$ will never select G_2 for expansion.

Optimality of A* for Graph Search

- A heuristic *h(n)* is consistent if
  for every node *n* and every successor *n'* generated by an action *a*,
  $h(n) \leq c(n,a,n') + h(n')$
  (general triangle inequality)



- Theorem: If *h(n)* is consistent, A* using GRAPH-SEARCH is optimal.
  (see text for proof)

- Most admissible heuristics turn out to be consistent too
  E.g. SLD is a consistent heuristic for the route problem (prove it!)

Properties of A*

- Complete? Yes (unless there are infinitely many nodes with $f \leq f(G)$)
- Time? Exponential worst case but may be faster in many cases
- Space? Exponential: Keeps all generated nodes in memory (exponential # of nodes)
- Optimal? Yes

31

Okay, enough theory… time to wake up!

32

Next Time

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

33