Heuristic Search

Best-First Search

- use an evaluation function f(n) for each node
  - estimates "desirability"
  - expand most desirable node in fringe
- enqueueing function maintains fringe in order of f(n) — smallest (lowest cost) first
- two approaches: Greedy and A*

Romania

- map of roads between cities with distances (as used in uninformed search)
- straight-line distances to Bucharest from each city (as the crow flies)
  - Arad 366, Bucharest 0, Craiova 160, etc...

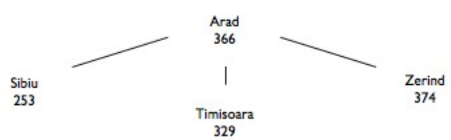
Greedy Best-First Search

- we introduce h(n): a heuristic function that estimates the cost from n to goal
- evaluation function f(n) = h(n),
- h(n) = straight line distance from state(n) to Bucharest
- greedy best-first search expands the node that appears to be closest to the goal

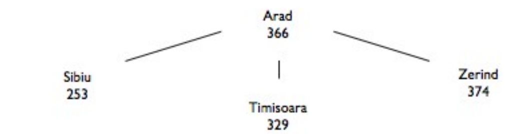
Greedy Best-First Search

Arad
366

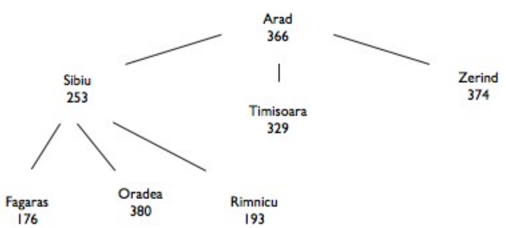
Greedy Best-First Search



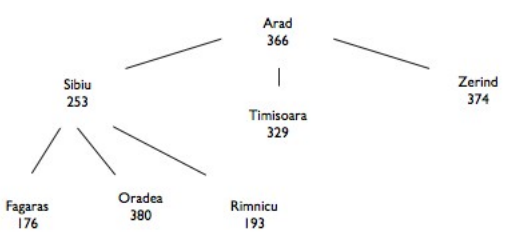
Greedy Best-First Search



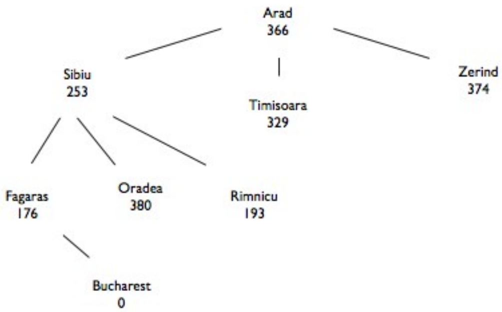
Greedy Best-First Search



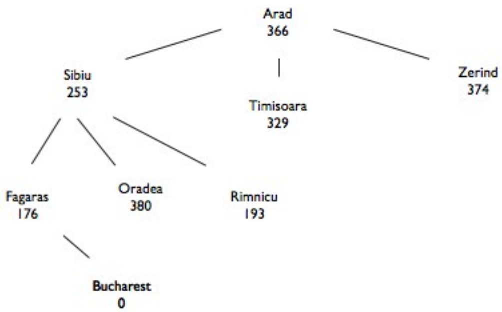
Greedy Best-First Search



Greedy Best-First Search



Greedy Best-First Search



Properties of Greedy Search

- complete? no (if tree search) – can get stuck in loops; yes if repeated nodes are eliminated (graph search)
- time? $O(b^m)$, but a good heuristic dramatically improves performance
- space? $O(b^m)$, keeps all nodes in memory
- optimal? no. greedy search is like heuristic depth-first

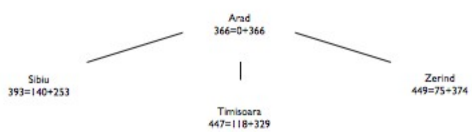
A* Search

- avoid expanding paths that are already expensive
- f(n) = g(n) + h(n)
- g(n) = path cost from initial to n
- h(n) = estimated cost from n to goal
- f(n) = estimated cost from initial to goal through n

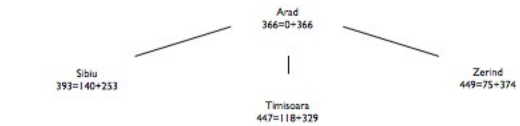
A* Search

Arad
366=0+366

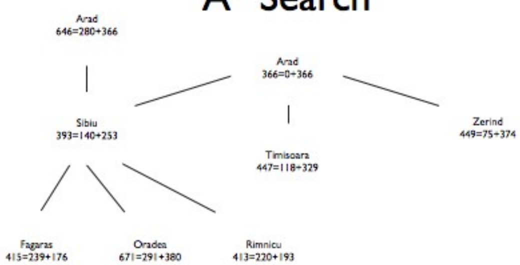
A* Search



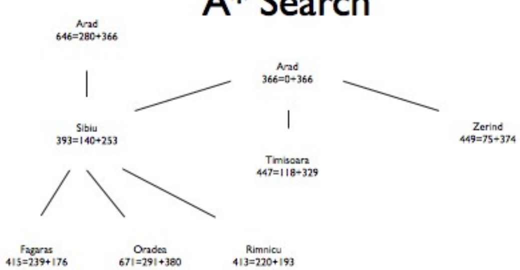
A* Search



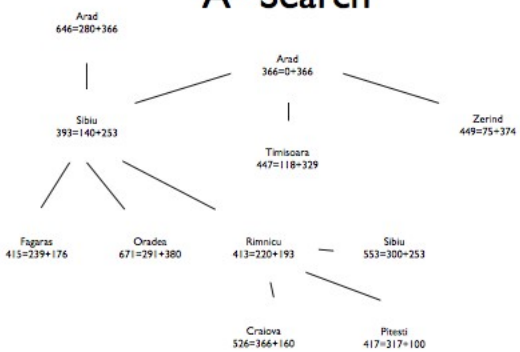
A* Search



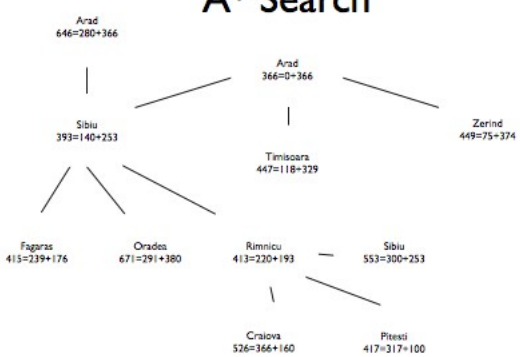
A* Search



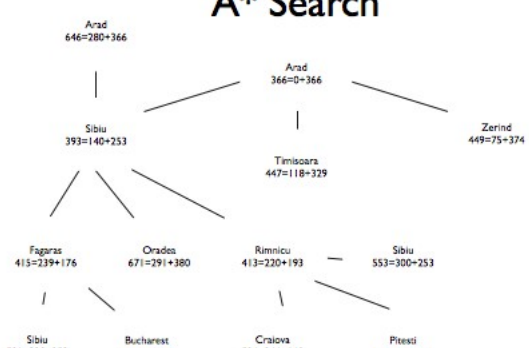
A* Search



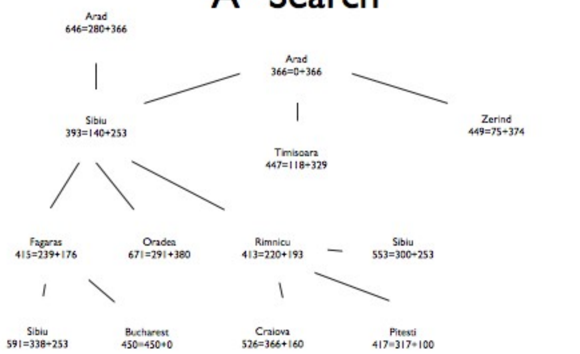
A* Search



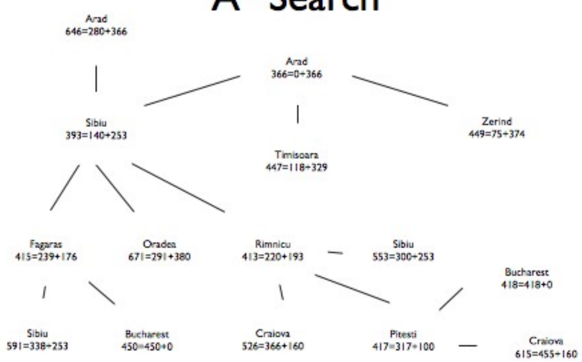
A* Search



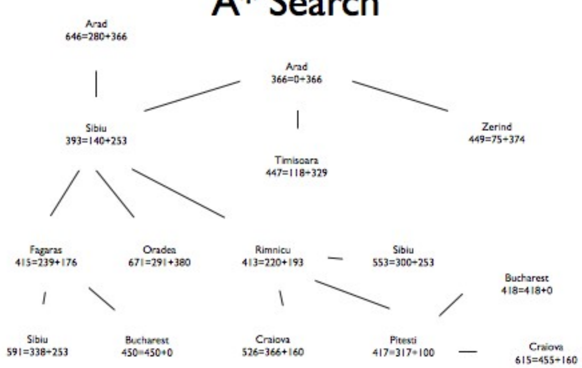
A* Search



A* Search



A* Search

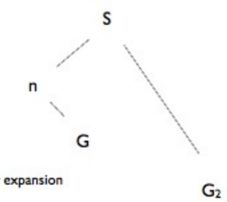


Admissible Heuristics

- a heuristic h(n) is admissible if for every node n, h(n) <= h*(n), where h*(n) is the true cost to reach the goal from n
- an admissible heuristic thus never overestimates the cost to reach the goal -- that is, it must be optimistic
- for example, straight line distance is admissible
- theorem: if h(n) is admissible, A* using tree-search is optimal

Proof of Optimality of A*

- suppose some suboptimal goal G_2 has been generated (in the fringe). Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G
- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $g(G_2) > g(G)$ since G_2 is suboptimal
- $f(G) = g(G)$ since $h(G) = 0$
- $f(G_2) > f(G)$ from above
- $h(n) <= h^*(n)$ since h is admissible
- $g(n) + h(n) <= g(n) + h^*(n)$
- $f(n) <= f(G)$
- Hence $f(G_2) > f(n)$ so A* will never select G_2 for expansion



A* Tree vs Graph Search

- A* with admissible h is optimal for tree search
- not so for graph search – A* may discards repeated states even if cheaper routes to them (i.e. g(n)) are found
- fix in two ways
  - modify graph search to check and replace repeated state nodes with cheaper alternatives
  - leave graph search as is, but insist on consistent h(n)

Consistent Heuristics

- a heuristic is consistent if for every node n, every successor n' of n generated by action a, h(n) <= c(n,a,n') + h(n')
- consistent heuristics satisfy triangularity
- difficult to concoct an admissible yet inconsistent heuristic
- if h is consistent, f(n')
  = g(n') + h(n')
  = g(n) + c(n,a,n') + h(n')
  >= g(n) + h(n)
  = f(n)
- that is, f(n) is non-decreasing along any path
- theorem: if h(n) is consistent, A* using graph-search is optimal

Properties of A*

- complete? yes (unless there are infinitely many nodes with f <= f(G))
- time? exponential
- space? keeps all nodes in memory
- optimal? yes

Admissible Heuristics

- for example, the 8-puzzle
  - $h_1(n)$ = number of misplaced tiles
  - $h_2(n)$ = total manhattan distance

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

$h_1(S)$ =

$h_2(S)$ =

Admissible Heuristics

- for example, the 8-puzzle
  - $h_1(n)$ = number of misplaced tiles
  - $h_2(n)$ = total manhattan distance

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

$h_1(S) = 8$

$h_2(S) =$

Admissible Heuristics

- for example, the 8-puzzle
  - $h_1(n)$ = number of misplaced tiles
  - $h_2(n)$ = total manhattan distance

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

$h_1(S) = 8$

$h_2(S) = 3+1+2+2+2+3+3+2 = 18$

Dominance

- for admissible heuristics h_1 and h_2,
  h_2 dominates h_1 if $h_2(n) >= h_1(n)$ for all n
- typical time complexities (number of expanded nodes) for 8-puzzle
  - d = 12
    IDS = 3,644,035
    A*(h_1) = 227
    A*(h_2) = 73
  - d = 24
    IDS = too many
    A*(h_1) = 39,135
    A*(h_2) = 1,641

Relaxation

- finding heuristics systematically by relaxing a problem
- a problem with fewer restrictions on actions is a relaxed problem
- the cost of an optimal solution to the relaxed problem is an admissible heuristic for the original problem
- for 8-puzzle, allowing tiles to move anywhere generates h_1 and allowing tiles to move to any adjacent square generates h_2
- for Romania problem, straight line distance is a relaxation generating its heuristic