

Heuristic Search

Introduction to Artificial Intelligence

Dr. Robin Burke



Review

- We can turn (certain classes of) problems
 - into state spaces
- We can use search to find solutions
 - DFS
 - BFS
 - IDS
- But what about operator cost?

Example: path planning

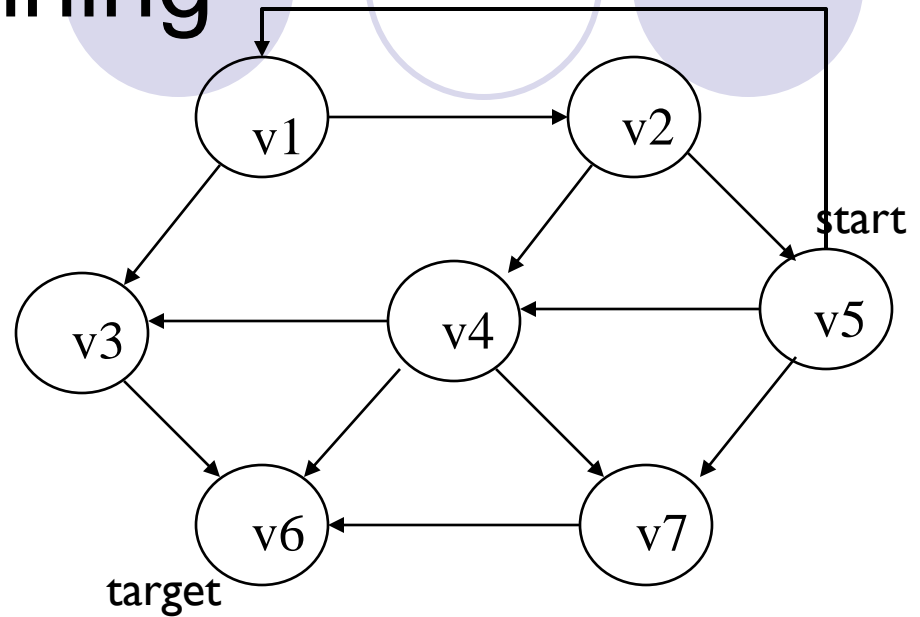
- states = positions

- initial state

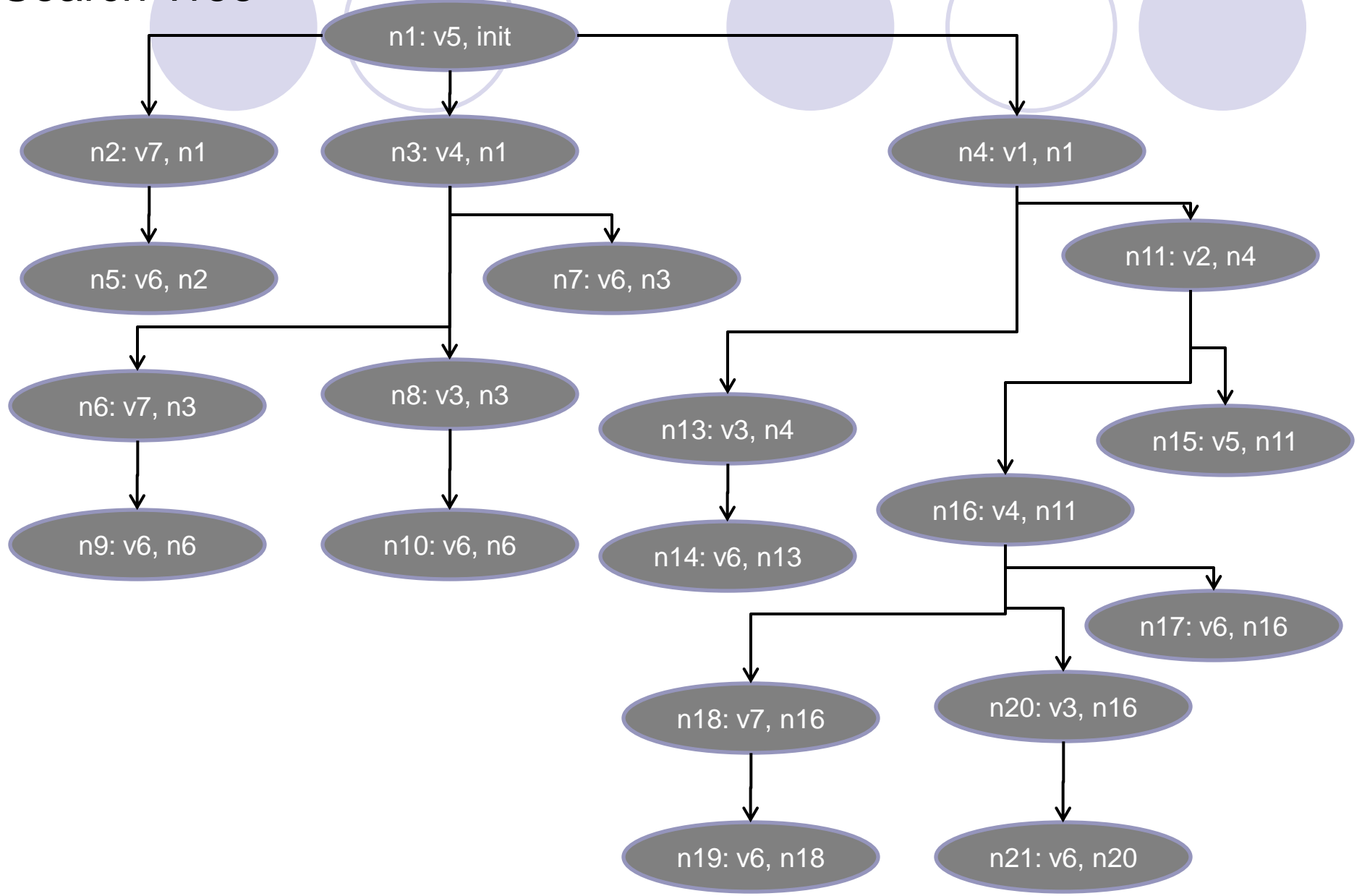
 - v5

- goal state

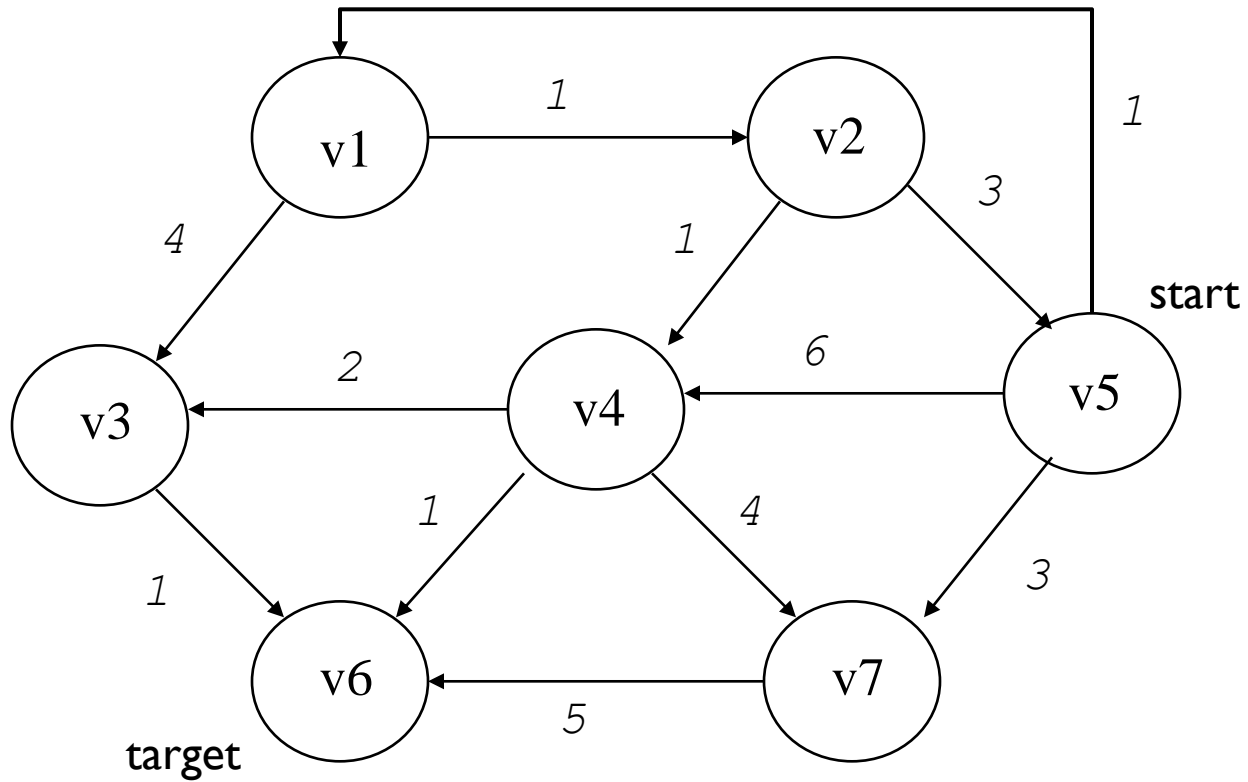
 - v6



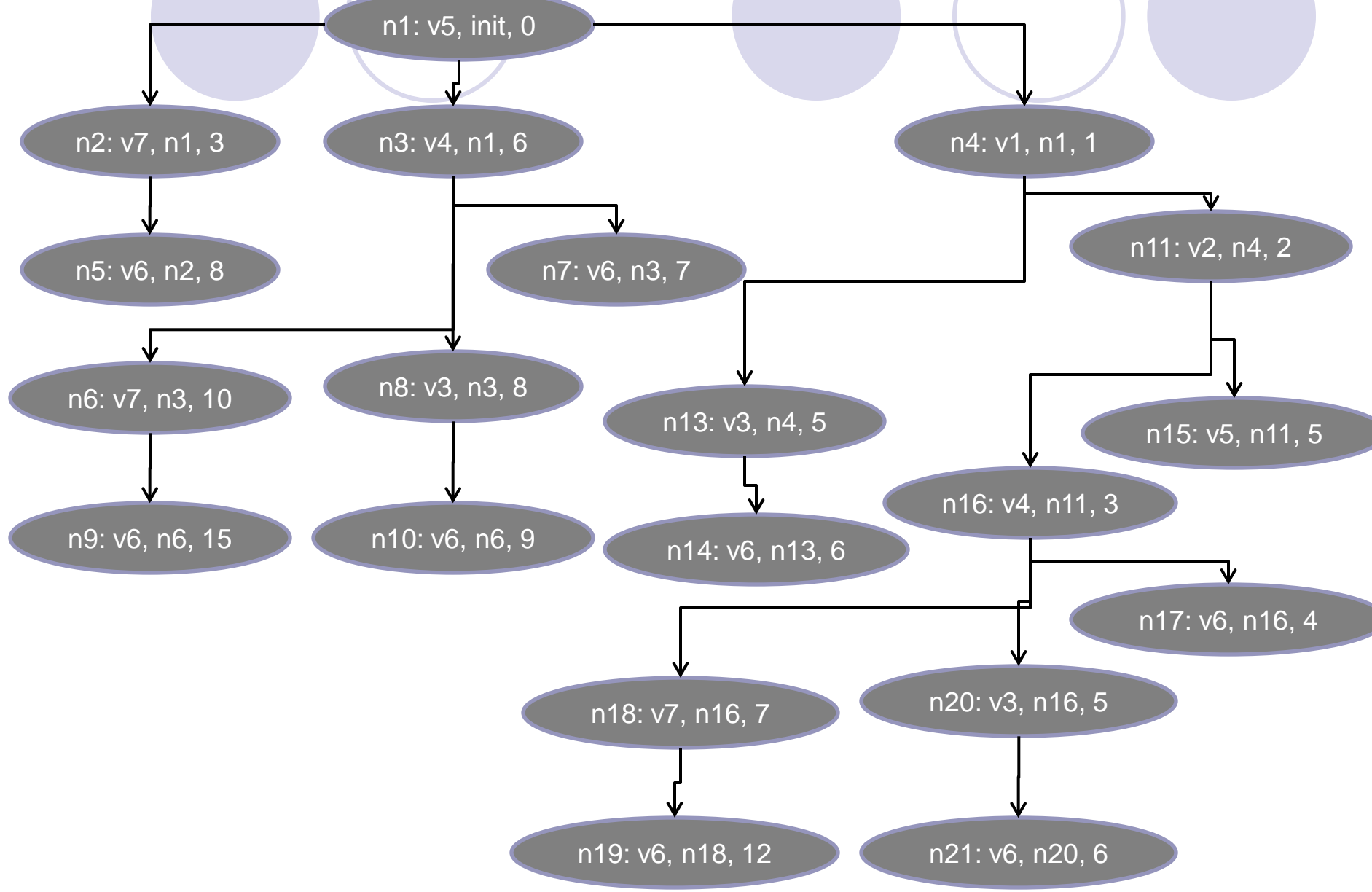
Search Tree



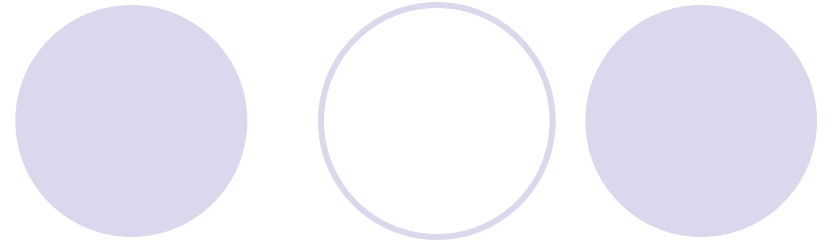
Graph with costs



Search Tree



Possible solutions



- Costs

- 4, 6, 6, 7, 9, 12, 15

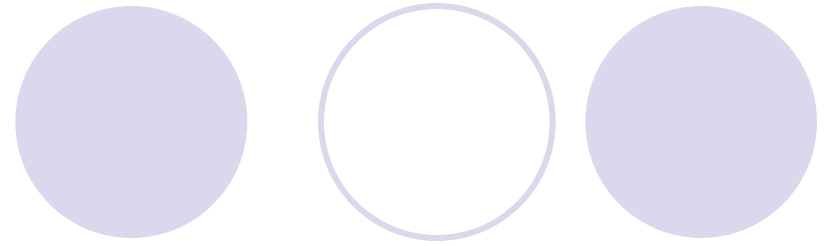
- worst is almost 4x best

- shortest path is not lowest-cost

Uniform-cost search

- Simple idea
 - use the least cost option
- Don't want
 - to use the least cost operation at a given node
 - why not?
- Concentrate on lowest-cost path so far
 - Dijkstra's algorithm

Search algorithm

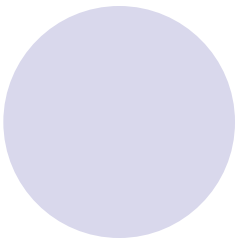
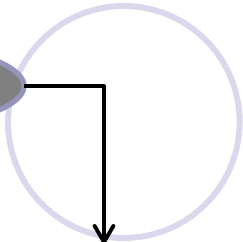
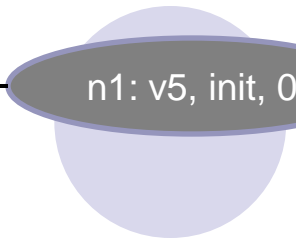
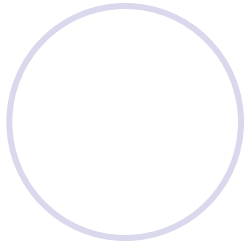
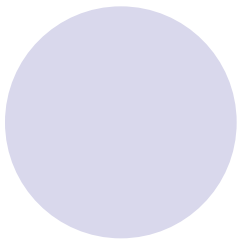


- Given
 - a set of nodes N
 - a successor function f that
 - takes a node n
 - returns all of the nodes S reachable from n in a single action
- Algorithm
 - sort N by path cost, select cheapest path
 - $s = \text{state}(n)$
 - $p = \text{path}(n)$
 - $S = f(s)$
 - for all $s', a \in S$
 - if s is solution, done
 - if s is illegal, discard
 - if on closed list, ignore this path, must be costlier
 - else
 - create new node $n_a = \langle s', (n,a) \rangle$
 - $N = N - n + n_a$
 - repeat

Basic idea



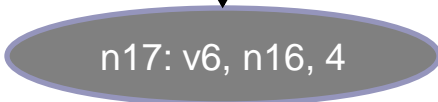
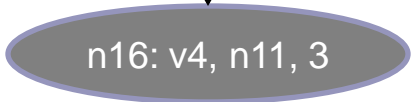
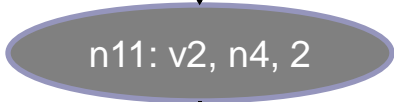
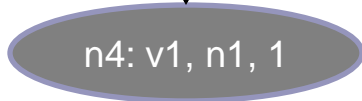
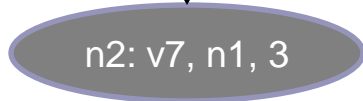
- Don't consider any paths of cost k
 - until you've considered paths of cost $< k$
- Implementation
 - need a priority queue
 - path cost = inverse priority
 - nodes with lowest path cost come first
 - possible data structure
 - heap



open

closed

n5: v6, n2, 8
n18: v7, n16, 7
n3: v4, n1, 6
n20: v3, n16, 5
n15: v5, n11, 5
n13: v3, n4, 5
n17: v6, n16, 4
n16: v4, n11, 3
n2: v7, n1, 3
n11: v2, n4, 2
n4: v1, n1, 1
n1: v5,init, 0



path: v5, v1, v2, v4, v6

Properties of Uniform-Cost Search

- Complete?

- Yes

- Time?

- $O(b^{(1+C/e)})$

- *where C is the cost of the best path*

- *e is the minimum action cost*

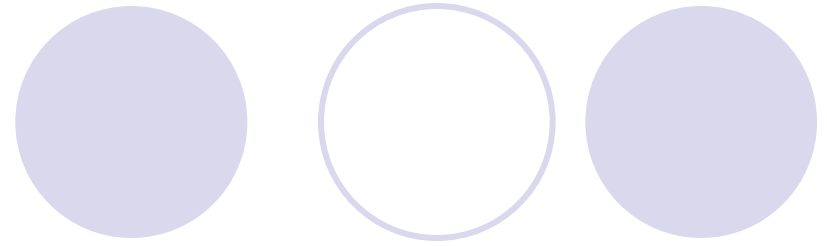
- Space?

- same

- Optimal?

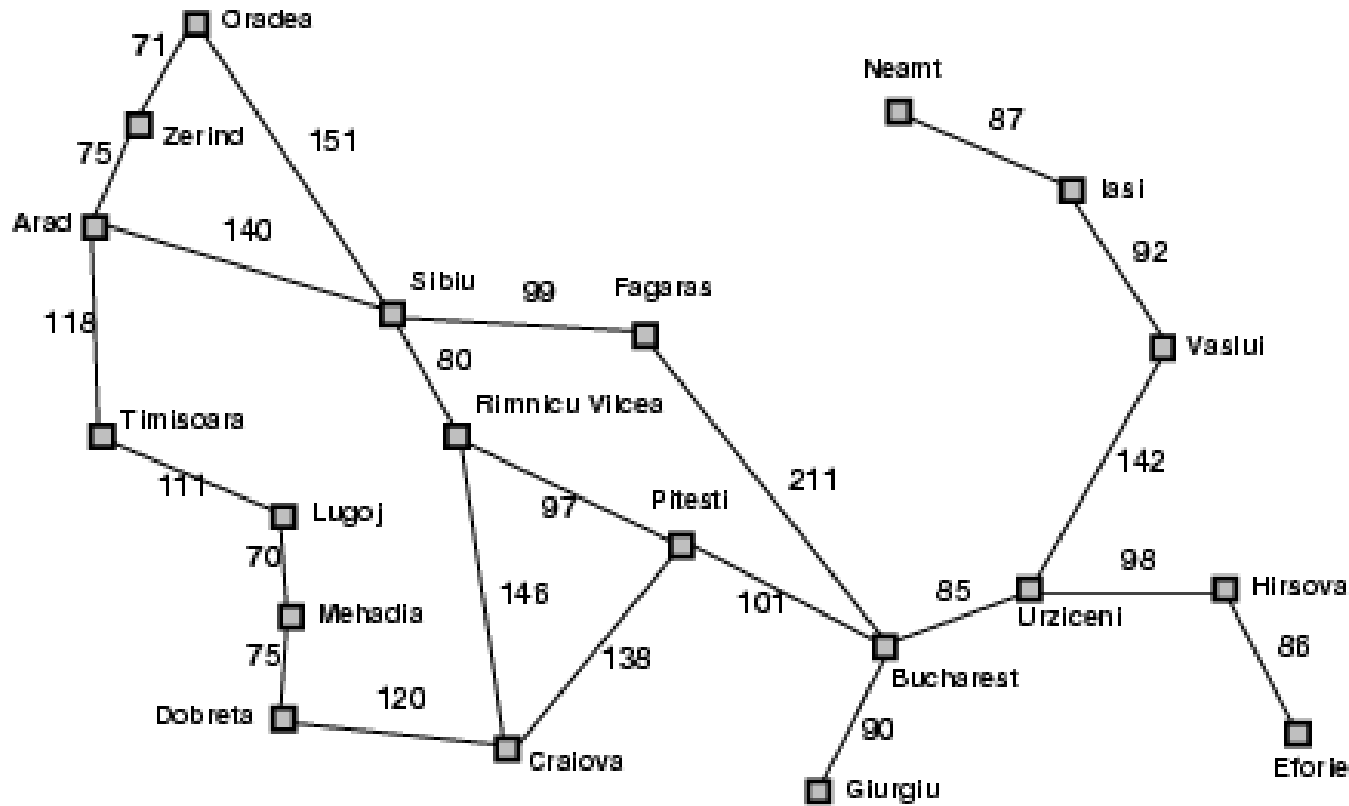
- Yes

Heuristic search



- What if we can measure our distance to a solution?
- We don't have to guess about the "right" direction
 - perhaps not perfect
- Example
 - Straight-line distance on a map

Example



Straight-line distance
to Bucharest

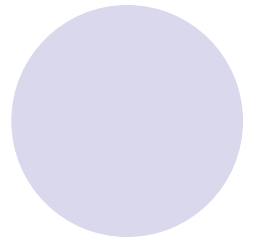
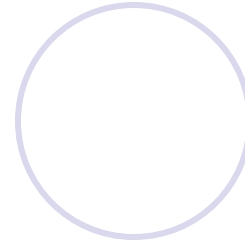
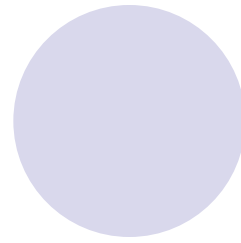
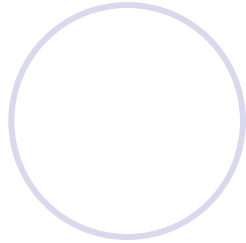
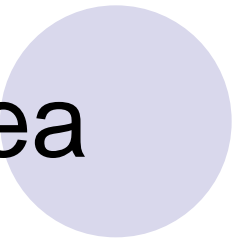
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Heuristic



- Suggests paths that are likely to lead in the right direction
 - unlike uniform-cost algorithm
- Example
 - start in Arad (366)
 - cheapest edge to Zerind
 - but Zerind is actually farther (374)
 - better choice Sibiu (253)
 - even though the edge is longer

Idea



- Greedy best-first search
 - maximize the heuristic at each step
- Same priority queue as before
 - but prioritize by heuristic
 - the closer the better

Greedy best-first search example

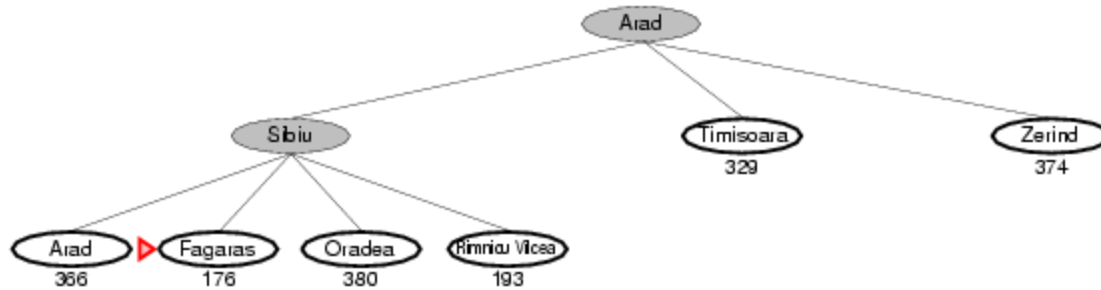


Arad
366

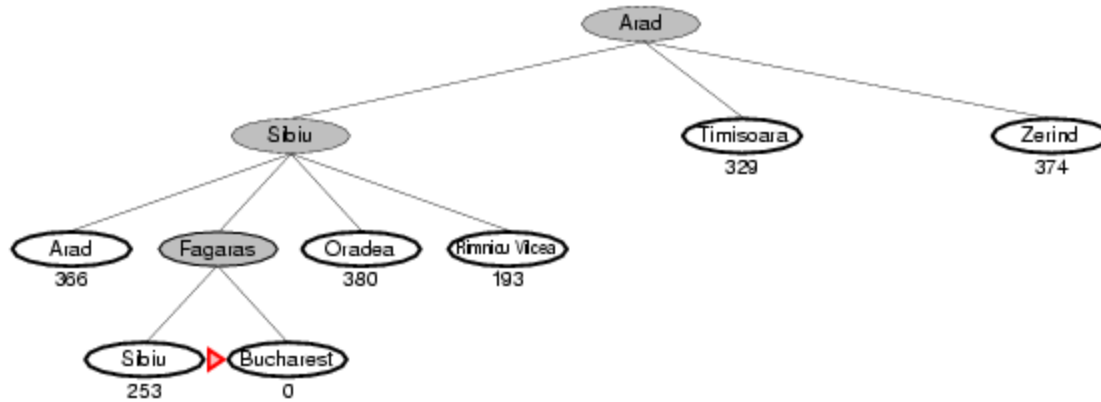
Greedy best-first search example



Greedy best-first search example



Greedy best-first search example



Properties of greedy best-first search

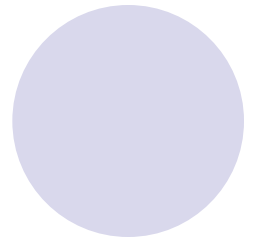
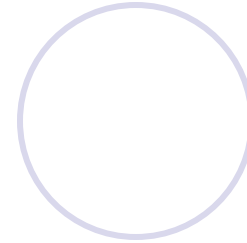
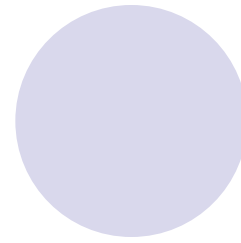
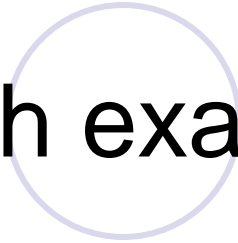
- Complete?
 - No – can get stuck in loops
 - Mehadia -> Dobreta -> Mehadia ...
 - if road to Craiova missing
- Time?
 - $O(b^m)$,
 - but a good heuristic can give dramatic improvement
- Space?
 - $O(b^m)$ -- keeps all nodes in memory
- Optimal?
 - No
 - There is a shorter path to Bucharest
 - via Fagaras = 450
 - via Rimnicu Vilcea = 418

A* search



- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$
 - $g(n)$ = cost so far to reach n
 - $h(n)$ = estimated cost from n to goal
 - $f(n)$ = estimated total cost of path through n to goal

A* search example



▶ A*ad
366=0+366

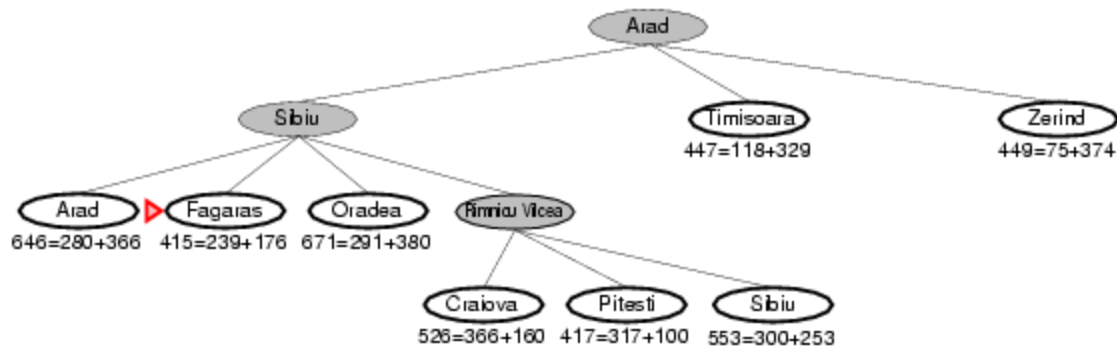
A* search example



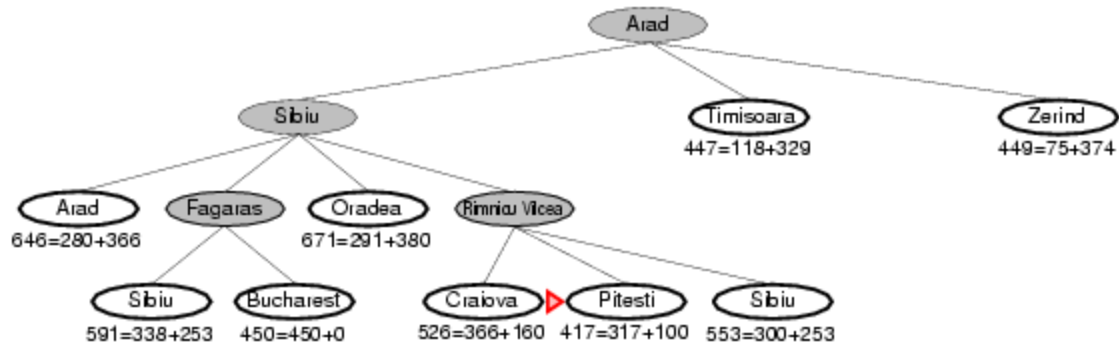
A* search example



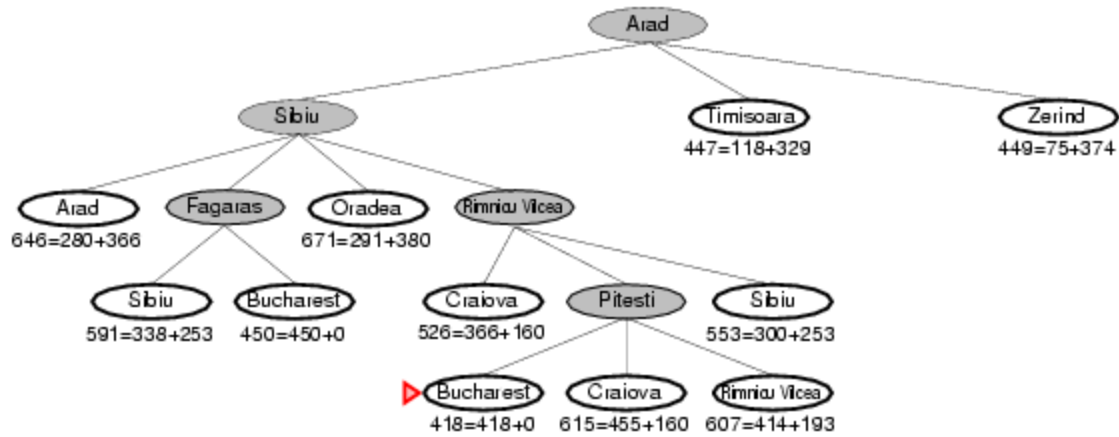
A* search example



A* search example



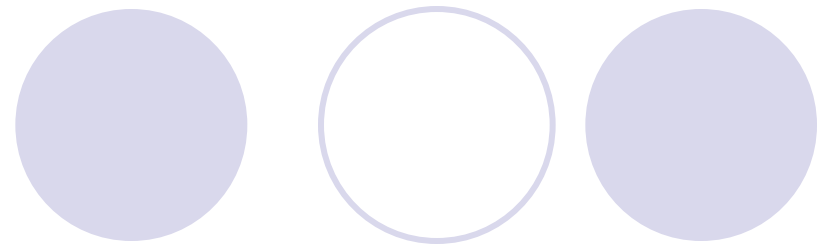
A* search example



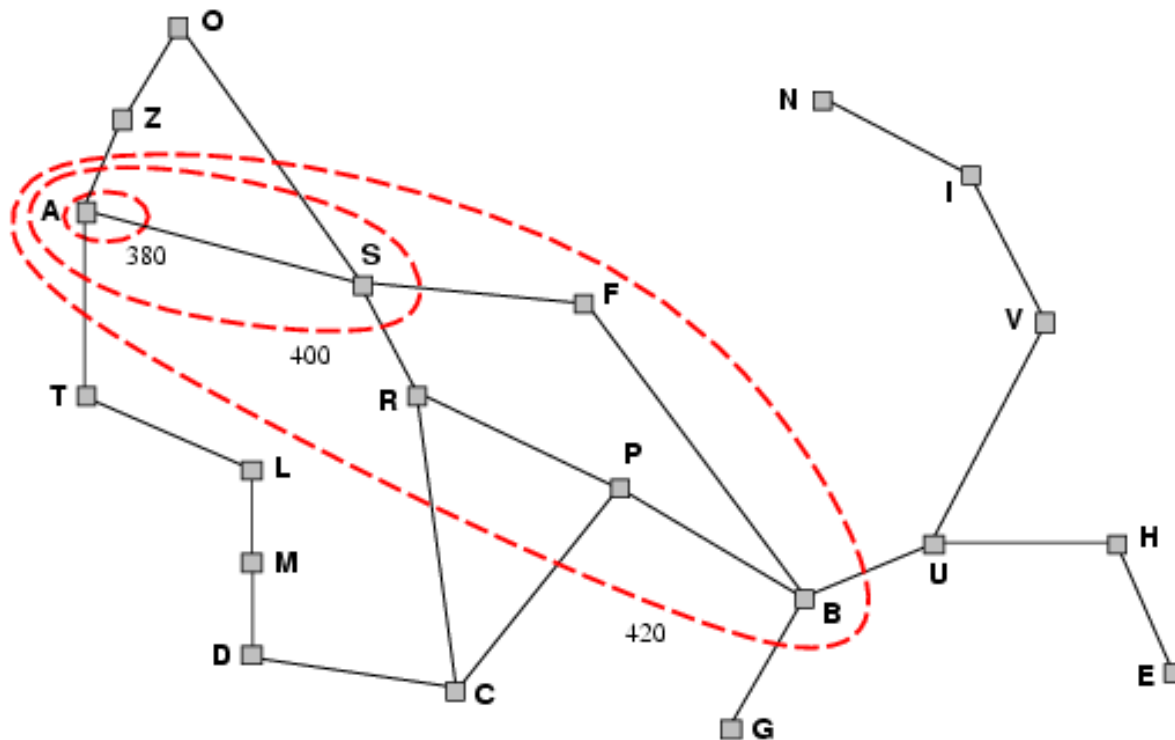
Admissible heuristics

- 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**
- Example: $h_{SLD}(n)$
 - (never overestimates the actual road distance)
- If $h(n)$ is admissible, A^* is optimal
 - (see book for proof)

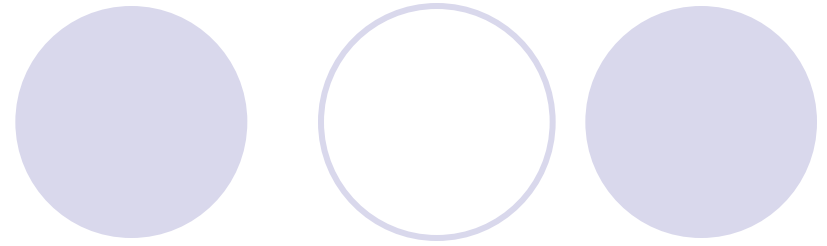
Optimality of A*



- A* expands nodes in order of increasing f value
- Gradually adds " f -contours" of nodes
- Contour i has all nodes with $f=f_i$, where $f_i < f_{i+1}$



Properties of A*



- Complete?

- Yes

- unless there are infinitely many nodes with $f \leq f(G)$

- Time?

- Exponential

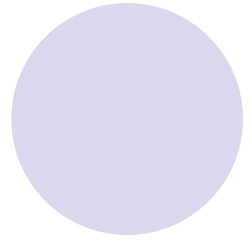
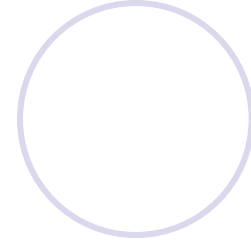
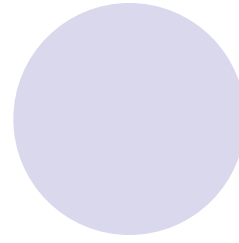
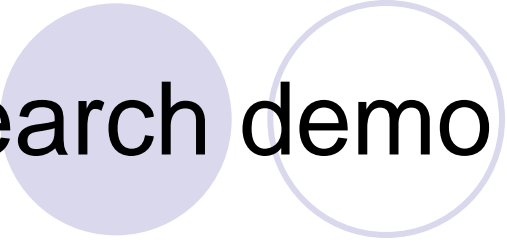
- Space?

- Keeps all nodes in memory

- Optimal?

- Yes

Search demo



Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance
(i.e., no. of squares from desired location of each tile)

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

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

Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance
(i.e., no. of squares from desired location of each tile)

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

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

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

- Typical search costs (average number of nodes expanded):
- $d=12$ 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
-

Relaxed problems



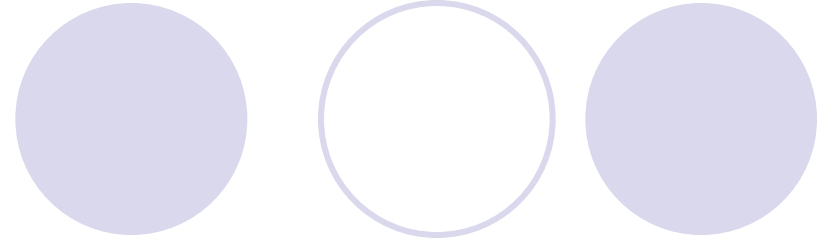
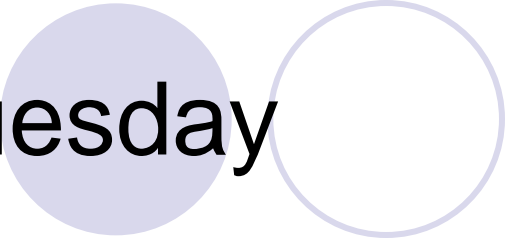
- A problem with fewer restrictions on the actions is called a **relaxed problem**
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
- If the rules of the 8-puzzle are relaxed so that a tile can move **anywhere**, then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to **any adjacent square**, then $h_2(n)$ gives the shortest solution

Practical



- Programming assignment using AIMA search code
 - link on course page
- Download and compile
- Take a look at search demos for eight-puzzle, etc.

Tuesday



- Reading Ch. 4.3-4.6