Introduction to Artificial Intelligence

Informed Search

CS 486/686

University of Waterloo

Outline

- Using knowledge
  - Heuristics
- Best-first search
  - Greedy best-first search
  - A* search
  - Variations of A*
- Back to heuristics

Last lecture

- Uninformed search uses no knowledge about the problem
  - Expands nodes based on “distance” from start node (never looks ahead to goal)
- Pros
  - Very general
- Cons
  - Very expensive
- Non-judgmental
  - Some are complete, some are not

Informed Search

We often have additional knowledge about the problem

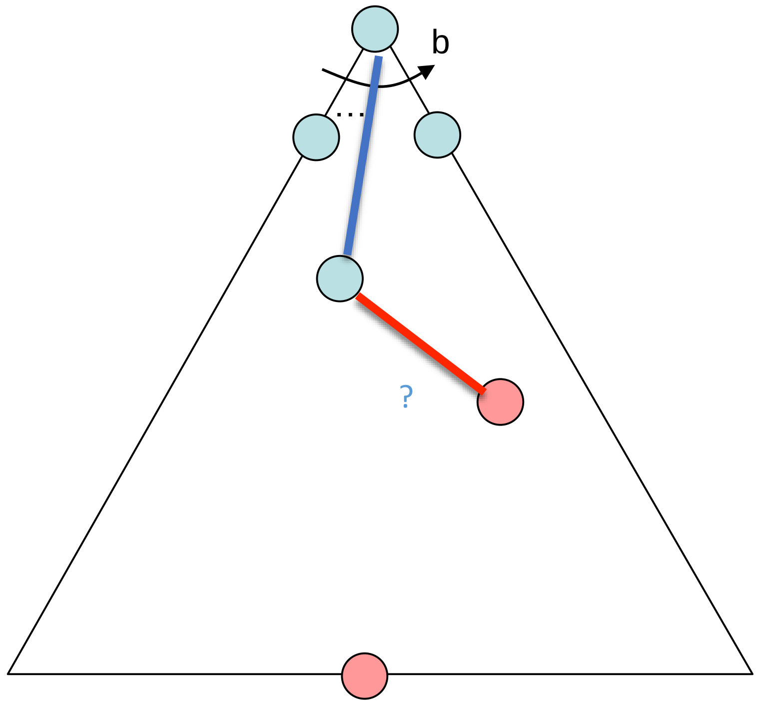
- Knowledge is often merit of a node (value of a node)
  - Example: Romania travel problem?

Different notions of merit

- Cost of solution
- Minimizing computation

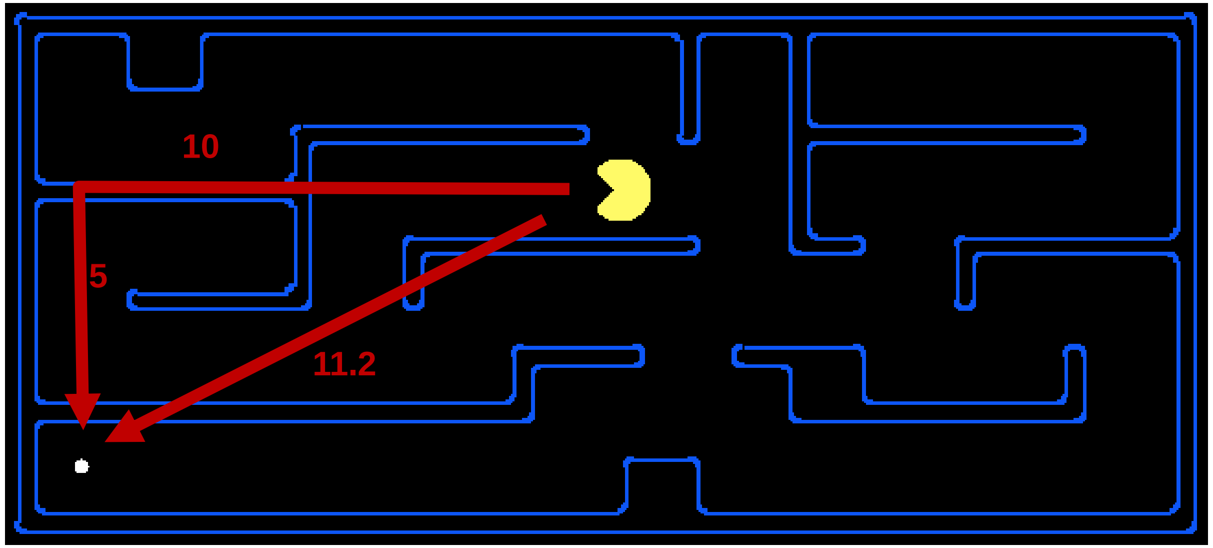
Uninformed vs Informed Search

- Uninformed search expands nodes based on distance from start node, d(nstart, n)
- Why not expand on distance to goal, d(n,goal)?



Heuristics

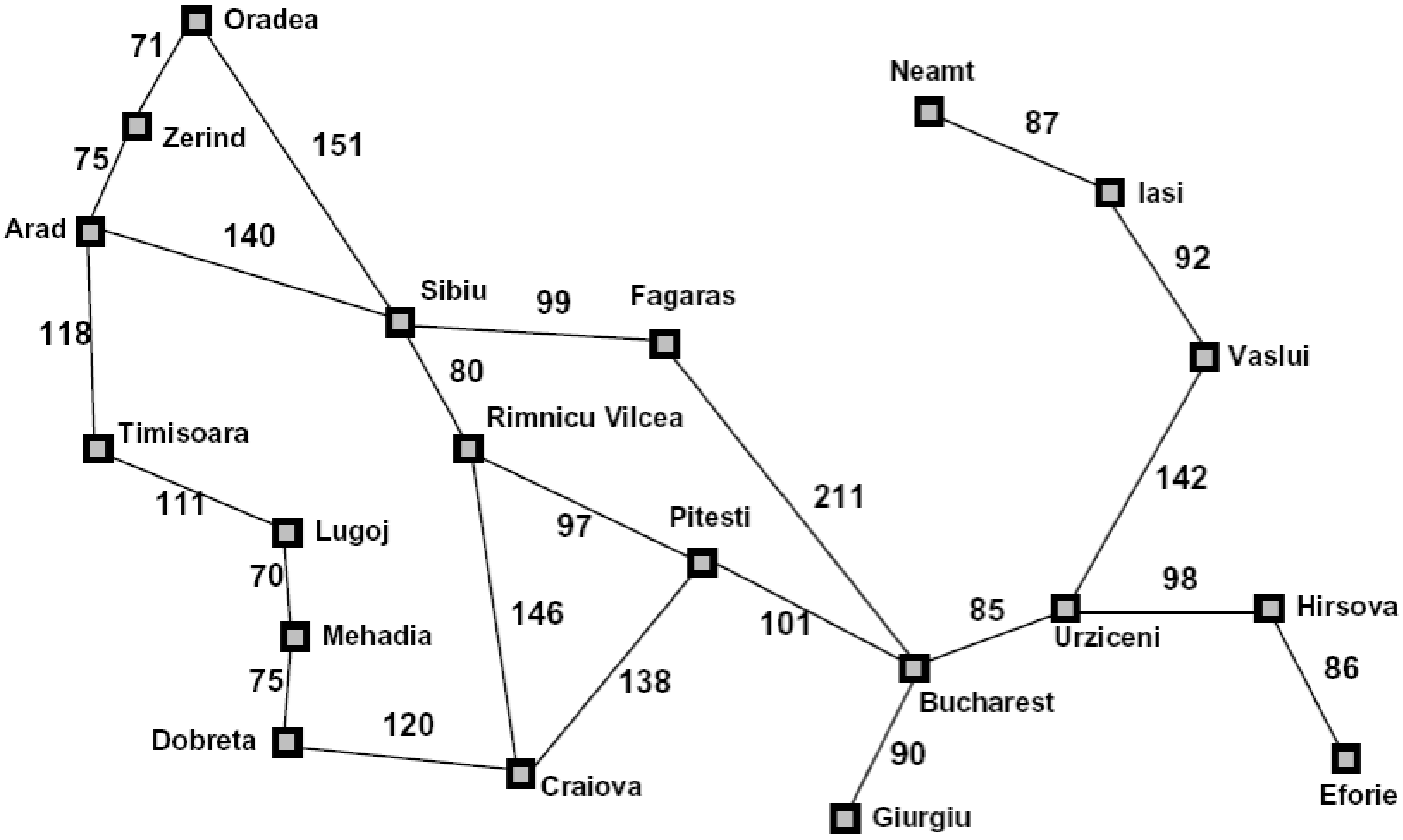
A heuristic is a function that **estimates** the cost of reaching a goal from a given state



Examples:
- Euclidean distance
- Manhatten distance

Adapted from UC Berkeley's CS188 Course

Example

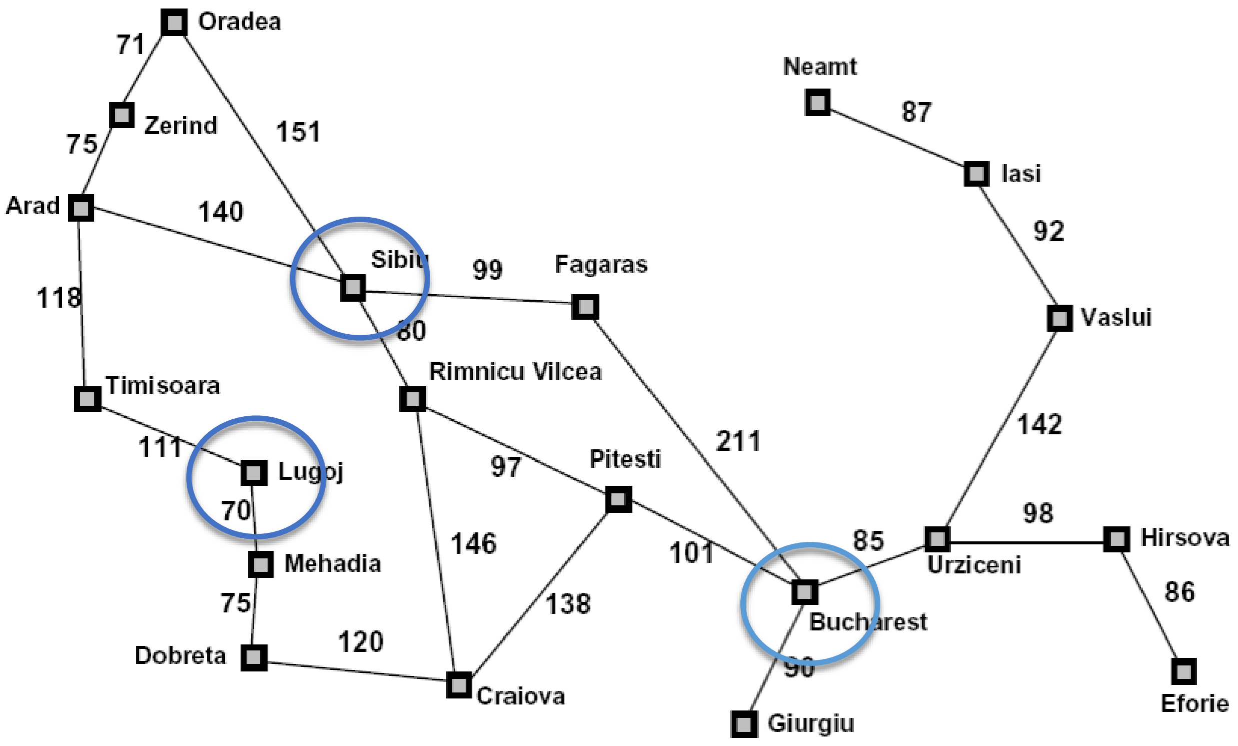


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

Heuristics: Structure

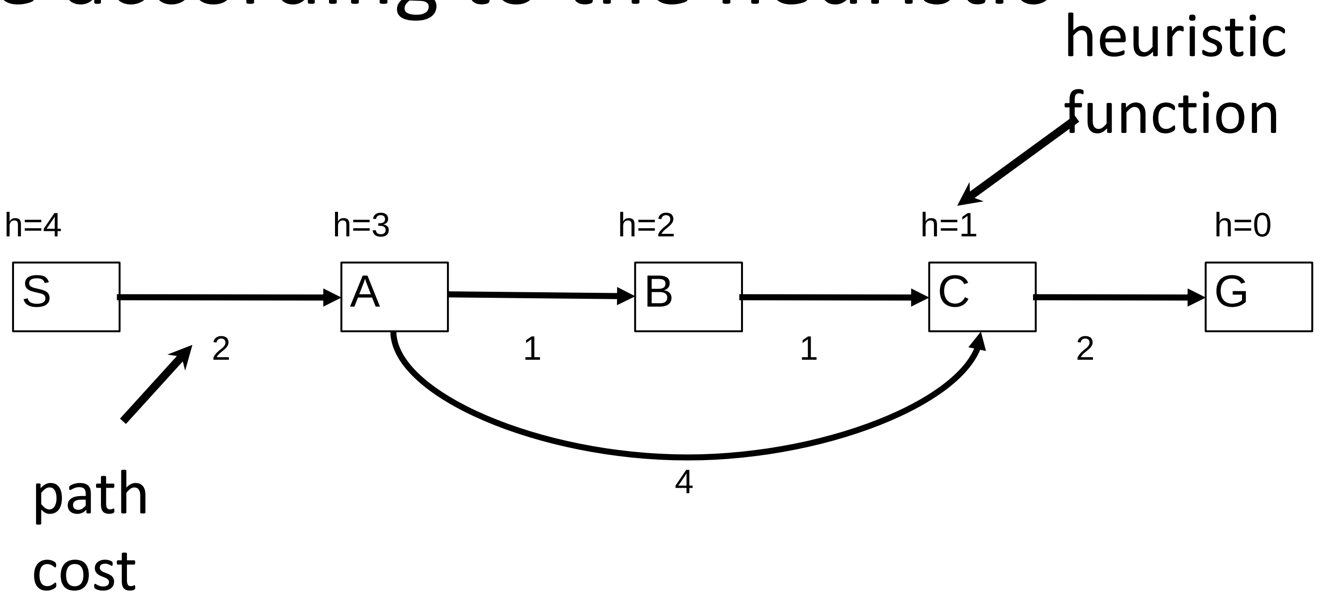
- If h(n1)<h(n2) we guess it is cheaper to reach the goal from n1 than n2
- We require h(n,goal)=0



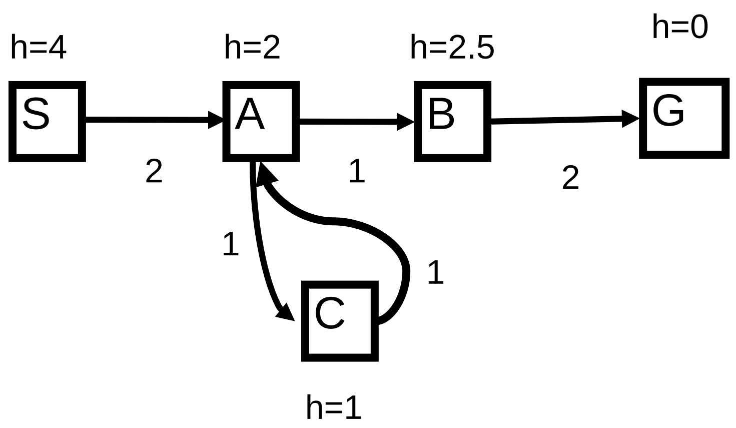
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example: Best First search

Search strategy: Expand the most promising node according to the heuristic

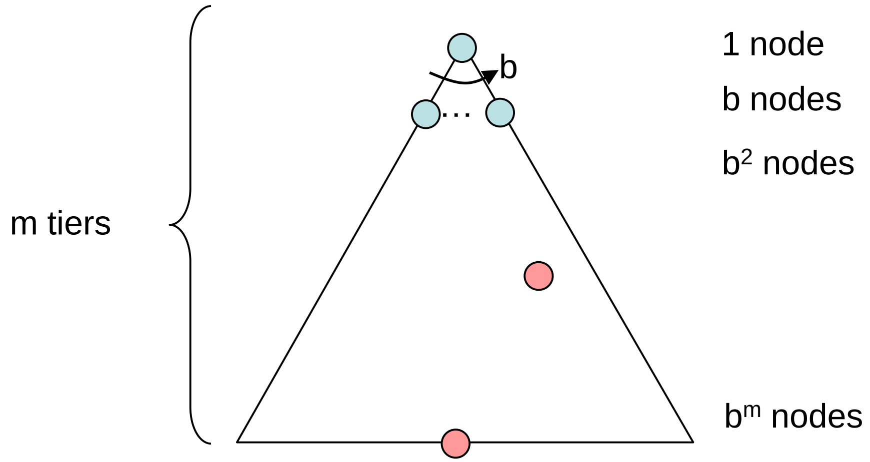


Example: Best First Search



Best First Search Properties

- Complete?
- Optimal?
- Time complexity
- Space complexity



A* Search

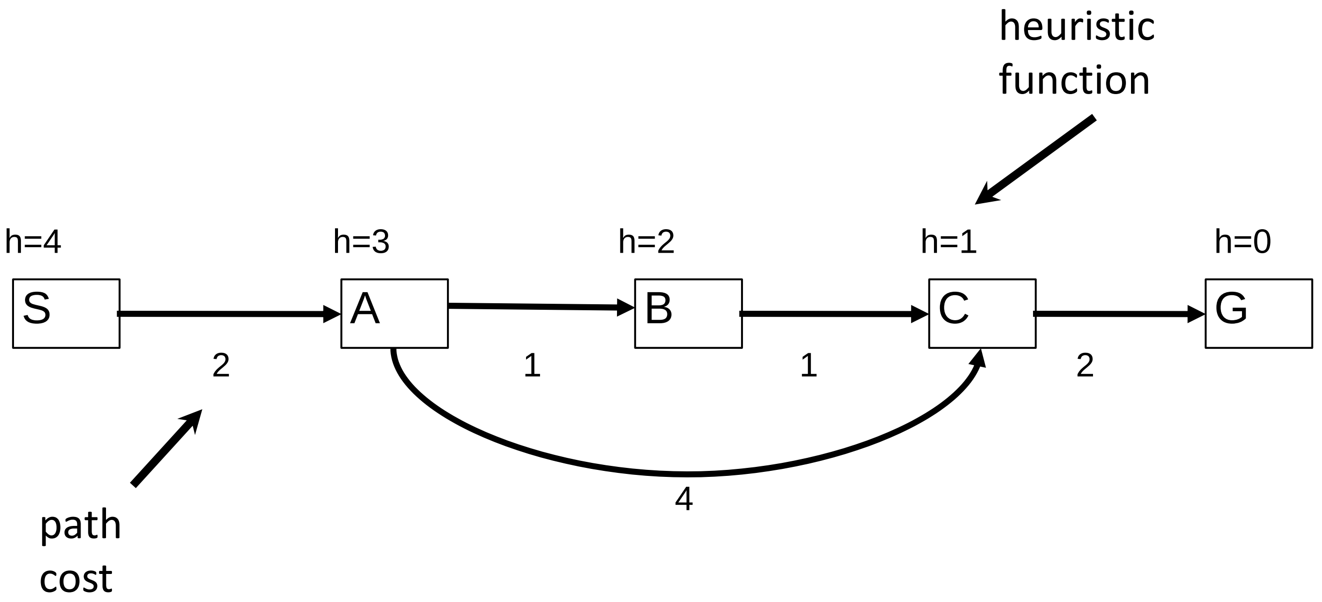
Observations

- Best first search ordered nodes by forward cost to goal, h(n)
- Uniform cost search ordered nodes by backward cost of path so far, g(n)

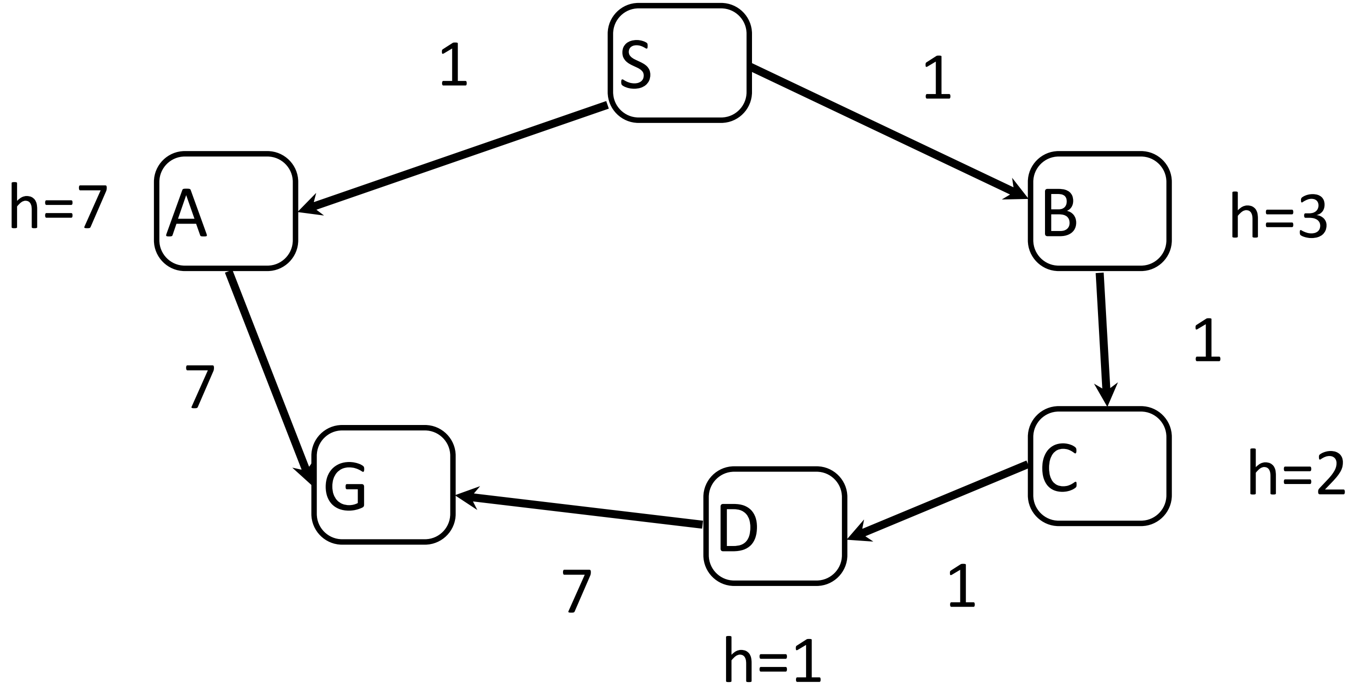
A* search

- Expand nodes in order $f(n)=g(n)+h(n)$

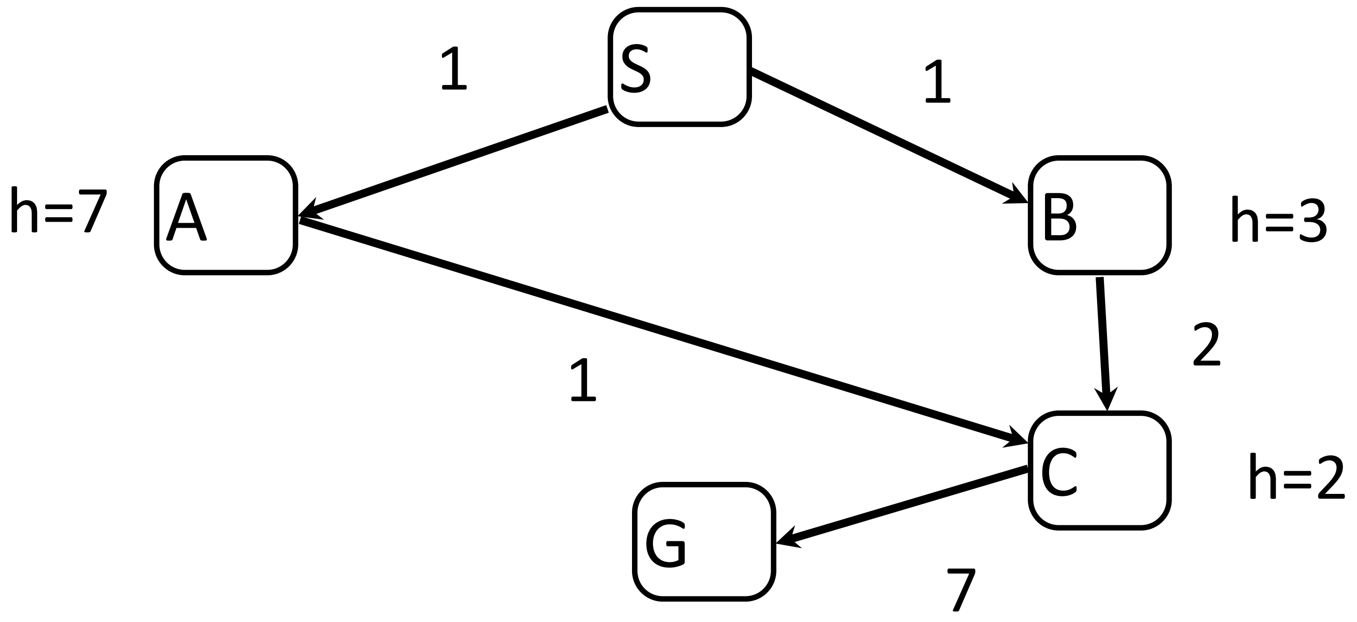
Example: A* search



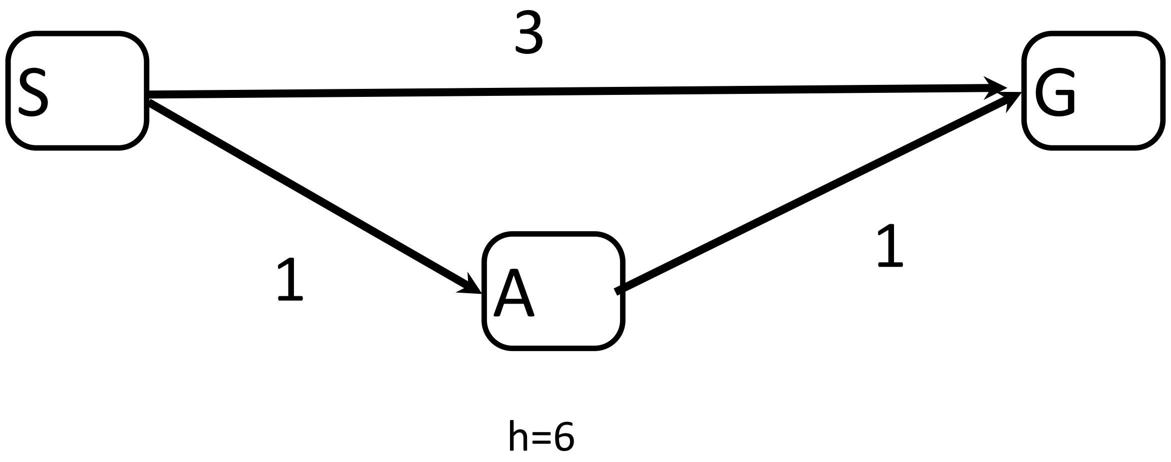
When Should A* Terminate?



A* and Revisiting States



Is A* Optimal?



1

Admissible Heuristics

A heuristic is admissible if $0 \leq h(n) \leq h^*(n)$ for all n, where $h^*(n)$ is the true shortest path cost from n to any goal state.

Admissible heuristics are optimistic. Note that $(h)=0$ is admissible.

Optimality of A*

If the heuristic is admissible then A* with tree search is optimal.

If we have a graph, then we require a stronger property for the heuristic function.

A heuristic is consistent if $h(n) \leq cost(n,n') + h'(n)$

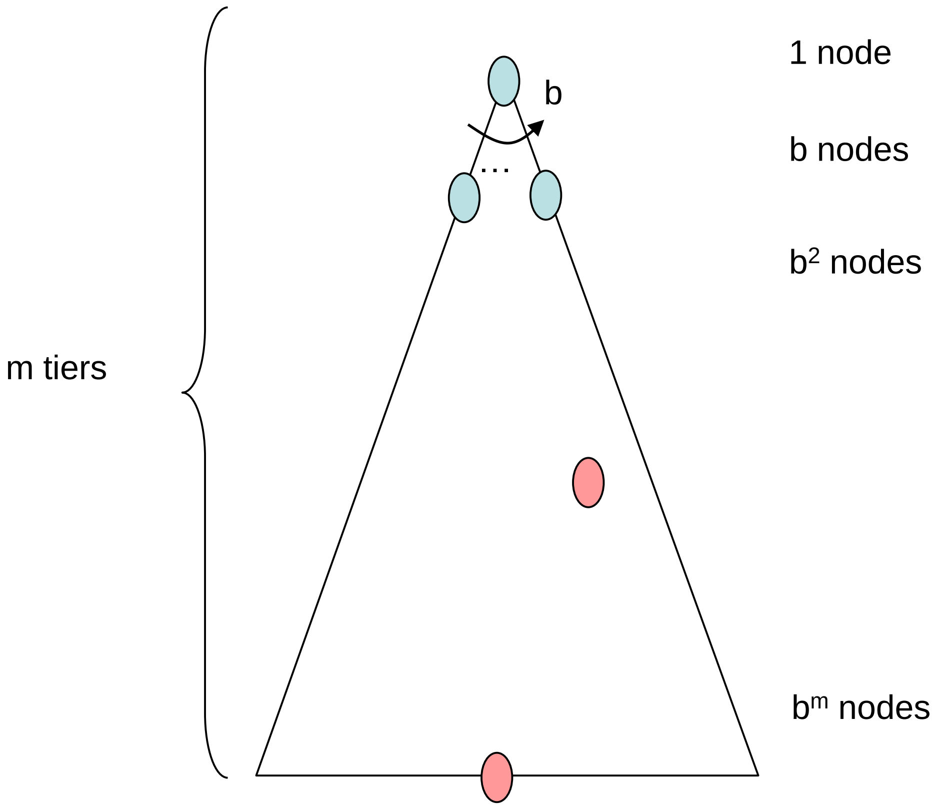
Almost any admissible heuristic will also be consistent.

A* is Optimally Efficient

Among all optimal algorithms that start at the same start node and use the same heuristic, A* expands the fewest nodes

A* Search Properties

- Complete?
- Optimal?
- Time complexity
- Space complexity



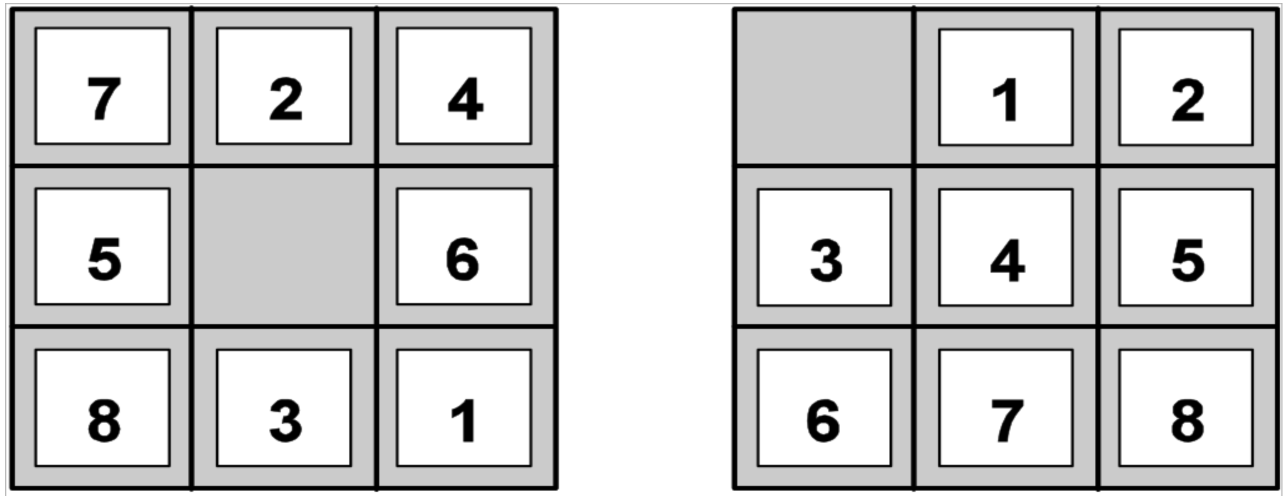
Heuristic Functions

A good heuristic function can make all the difference!

How do we get heuristics?

8 Puzzle

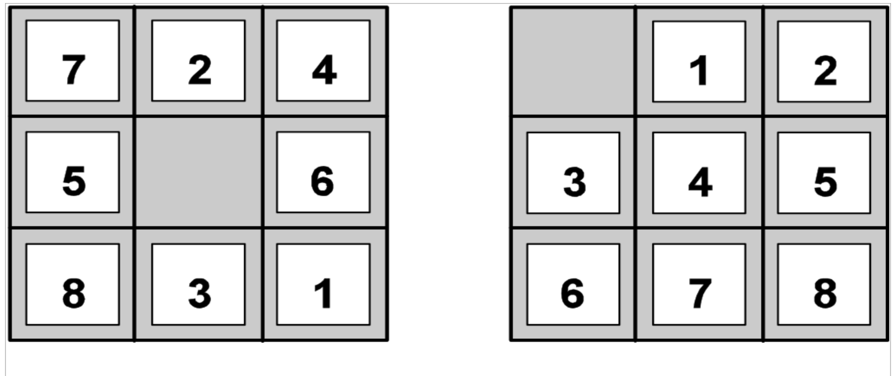
- Relax the game
  - Can move from A to B is A is next to B
  - Can move from A to B if B is blank
  - Can move from A to B



Start State Goal State

8 Puzzle

Dominating heuristic: Given heuristics h1(n) and h2(n), h2(n) dominates h1(n) if
$\forall n \; h2(n) \geq h1(n) \; and \; \exists n \; h2(n) > h1(n)$



Start State   Goal State

Theorem: If h2(n) dominates h1(n), A* using h2(n) will never expand more nodes that A* using h1(n).

- Can move from A to B: (Misplaced Tile Heuristic, h1)
- Can move from A to B if B is next to A:(Manhatten Distance Heuristic, h2)

8 Puzzle and Heuristics

Depth	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6
4	112	13	12
8	6384	39	25
12	3644035	227	73
24	-	39135	1641

Designing Heuristics

- Relax the problem
- Precompute solution costs of subproblems and storing them in a pattern database
- Learning from experience with the problem class
- …

Often there is a **tradeoff** between accuracy of your heuristic (and thus the amount of search) and the amount of computation you must do to compute it

Summary

- What you should know
  - Thoroughly understand A*
  - Be able to trace simple examples of A* execution
  - Understand admissibility of heuristics
  - Completeness, optimality

Some Things to Think About

- What is the relationship between A* search and Dijkstra's algorithm?

- A* search can be very memory intensive. Can you think of some variants of A* search that might reduce the memory overhead?