Informed Search

CS 486/686: Introduction to Artificial Intelligence
Winter 2016

Outline

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

Last lecture

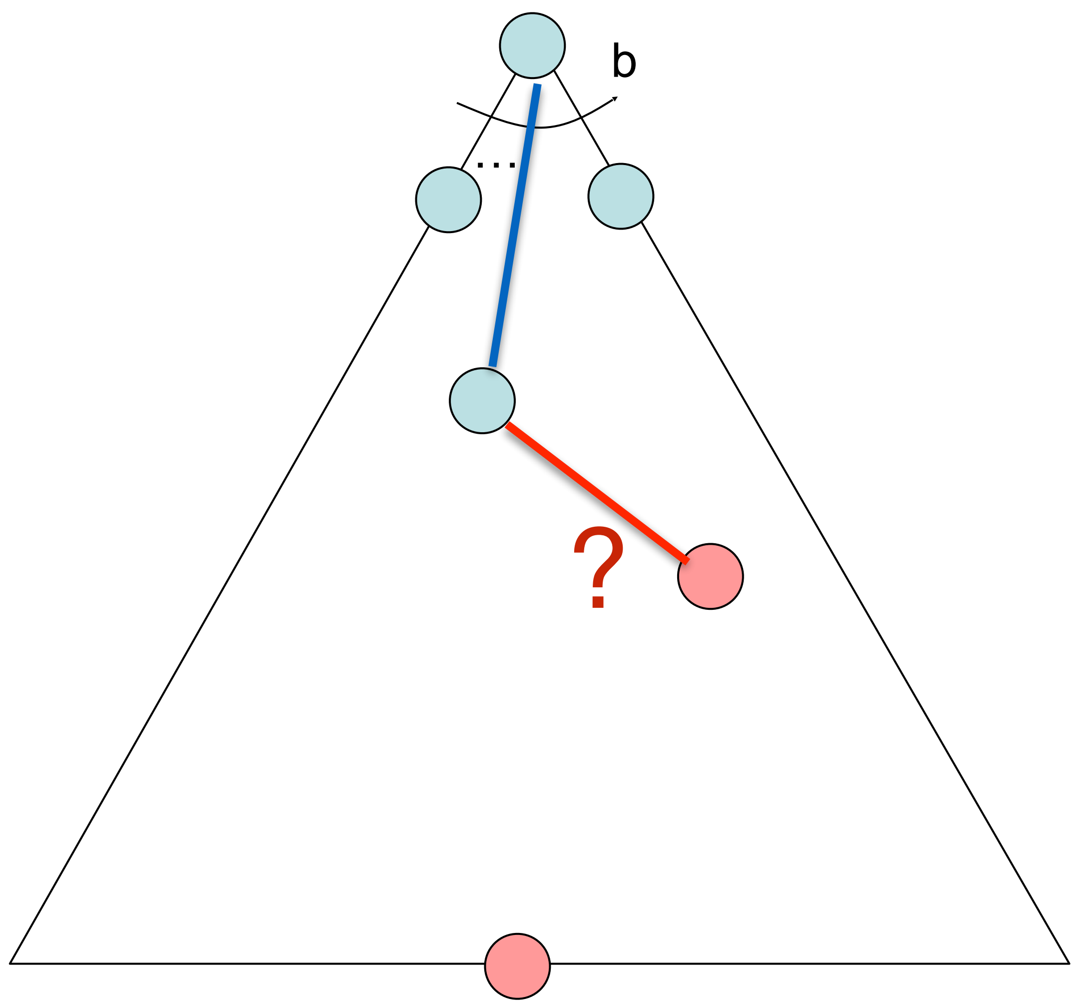
- Uninformed search uses no knowledge about the problem
  - Expands nodes based on “distance” from start node (never looks ahead to goal)
- Pros
  - Very general
- Cons
  - Very expensive
- Non-judgemental
  - Some are complete, some are not

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

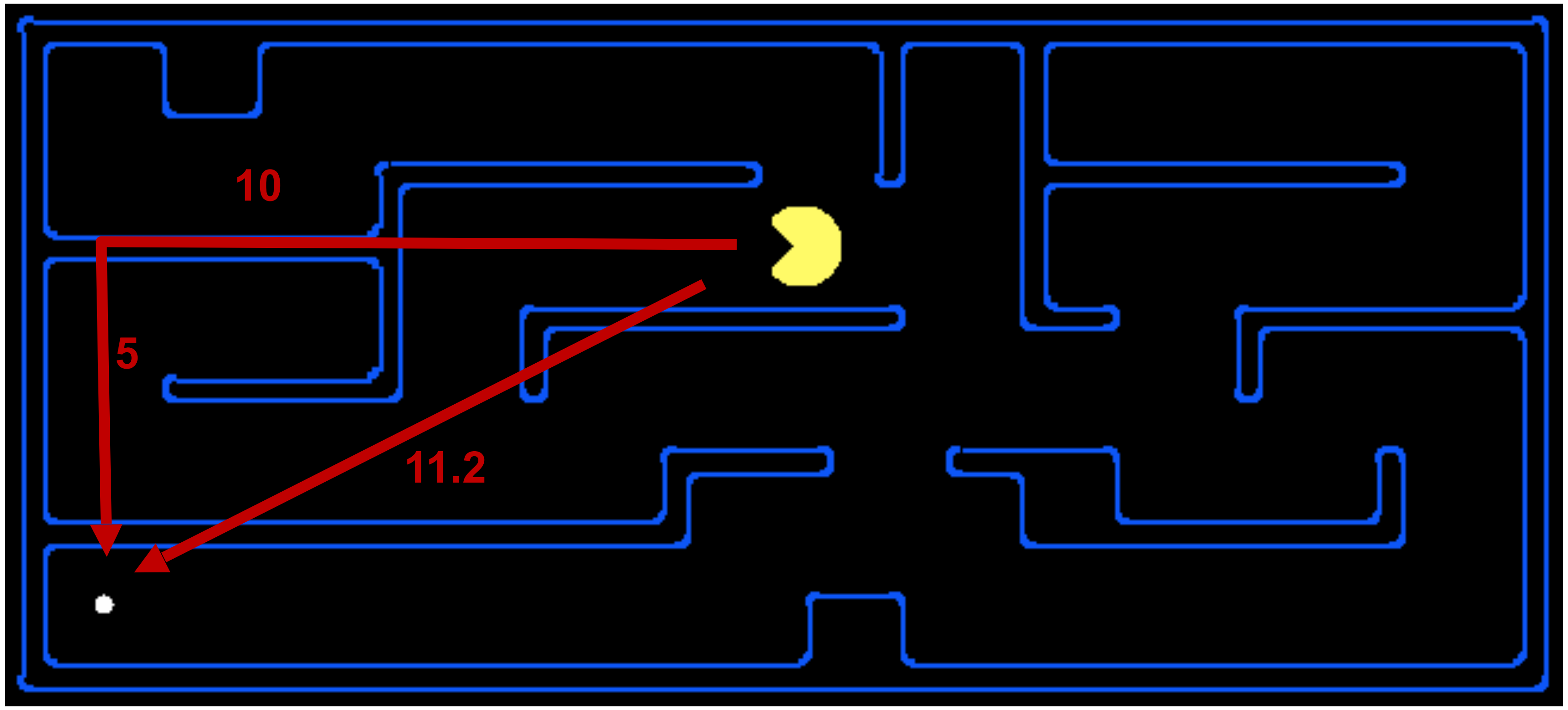
Uninformed vs Informed Search

- Uninformed search expands nodes based on distance from start node, $d(n_{start}, n)$
- Why not expand on distance to goal, $d(n, n_{goal})$?
- What if we do not know $d(n, n_{goal})$ exactly?
  - **Heuristic function, h(n)**



Heuristics

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

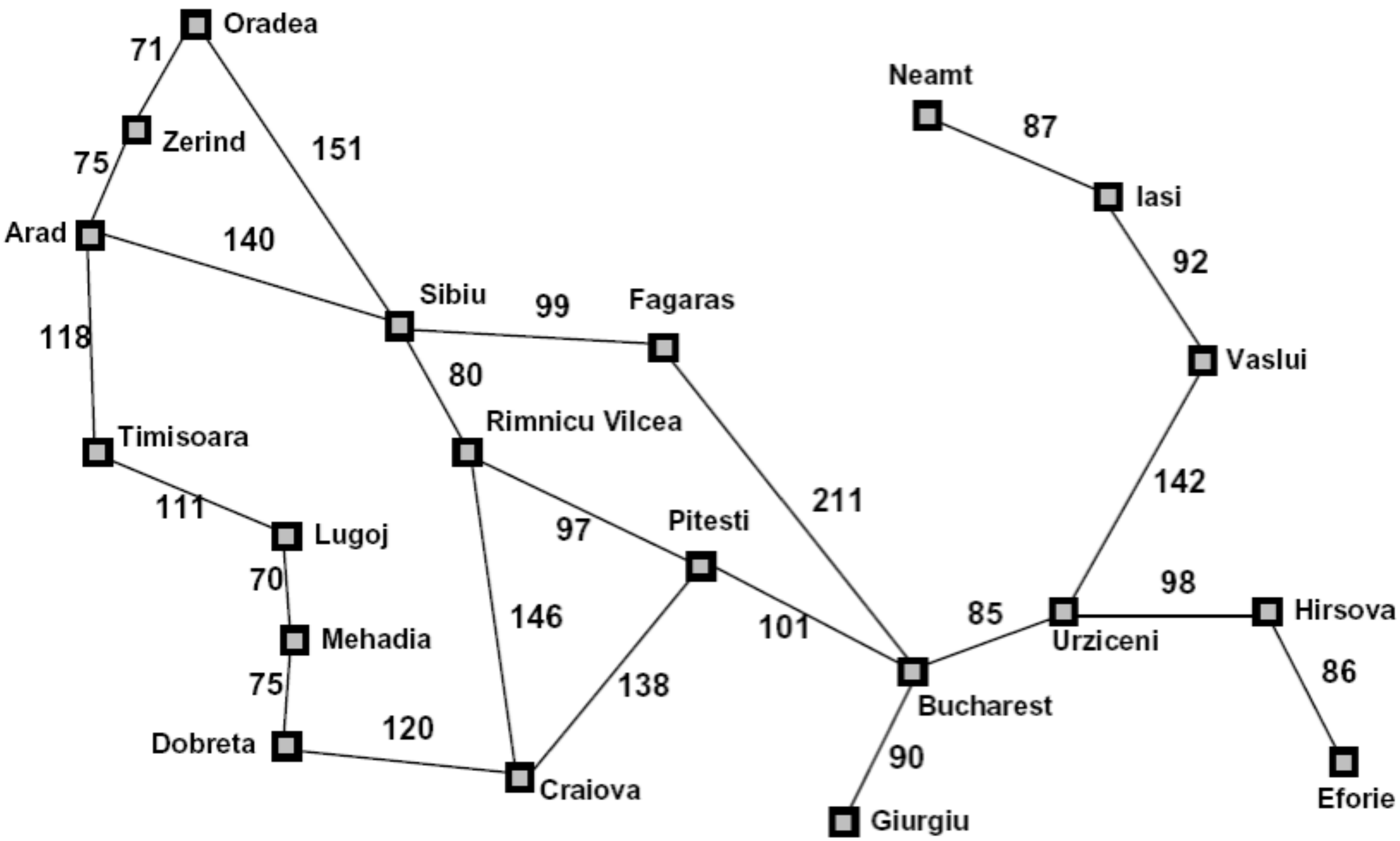


Examples:

- Euclidean distance
- Manhatten distance

Adapted from UC Berkeley's CS188 Course

Example



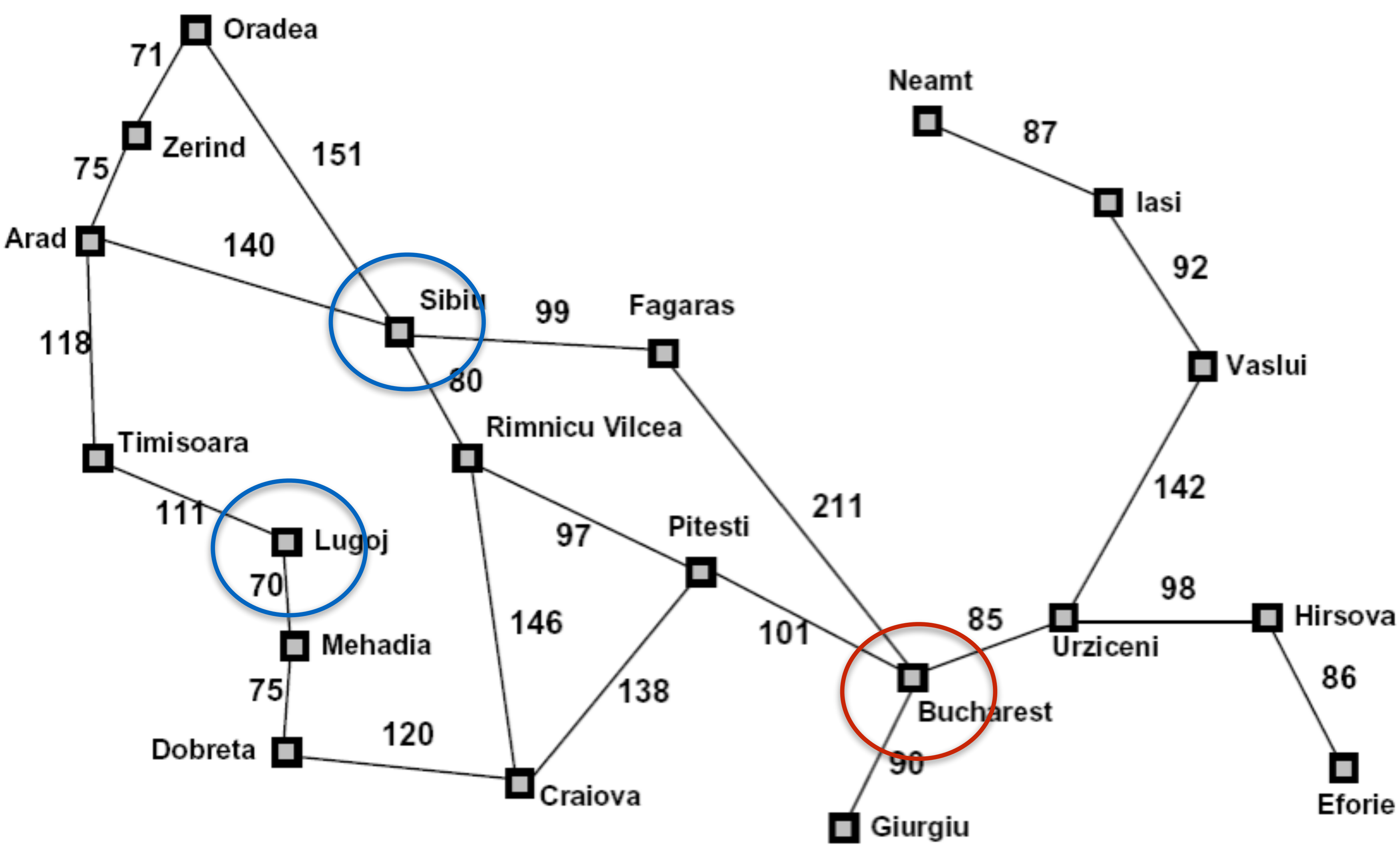
Straight−line distance
to Bucharest

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 $h(n_1)<h(n_2)$ we guess it is cheaper to reach the goal from n_1 than n_2
- We require $h(n_{goal})=0$

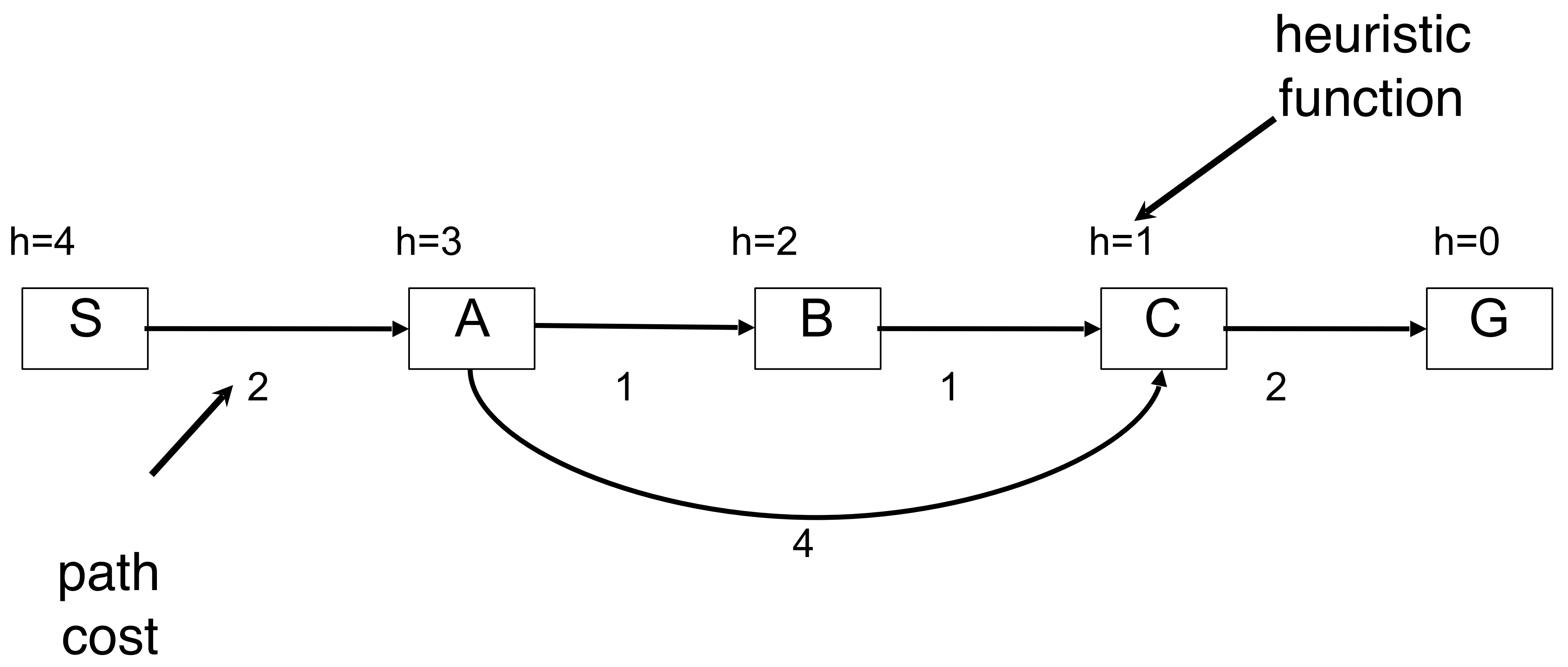


Straight−line distance to Bucharest

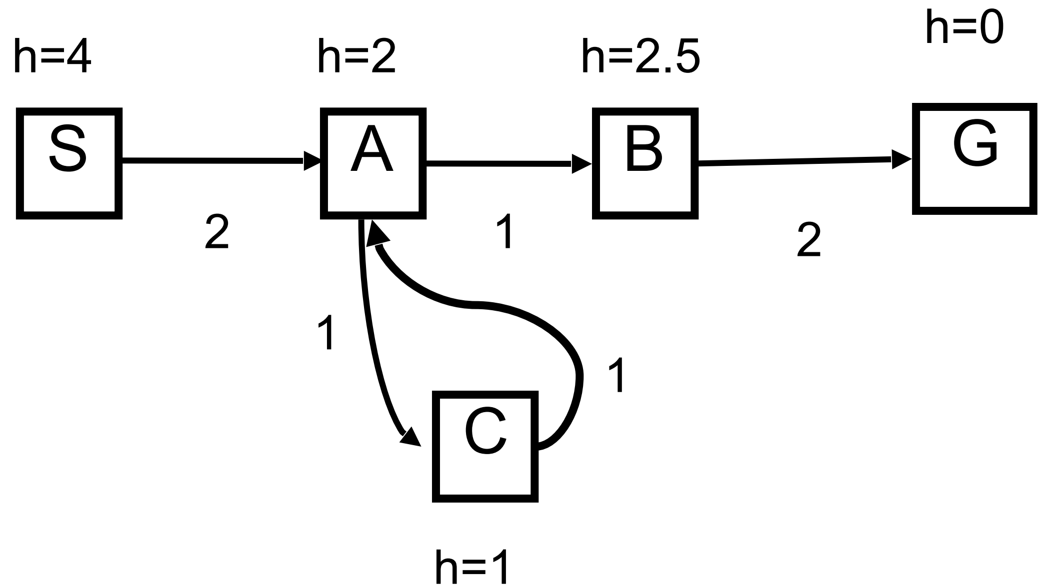
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example: Best First search

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



Example: Best First Search

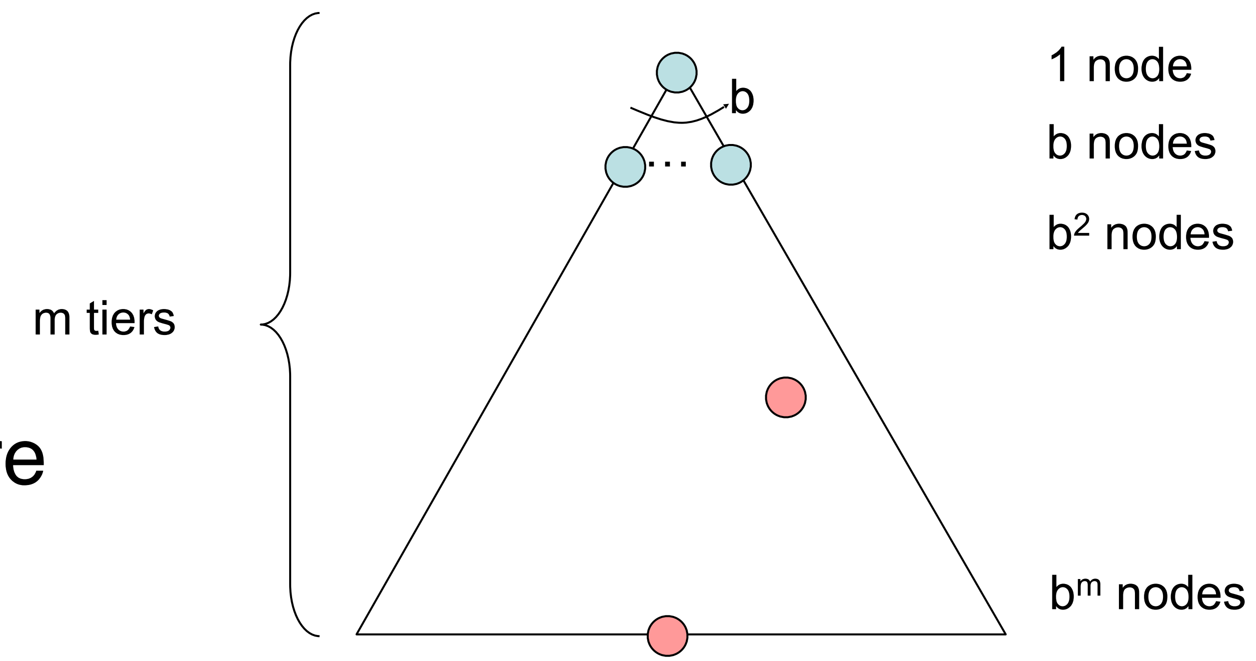


Quiz

- What can we do to make Best-first search simulate Breadth-first search?

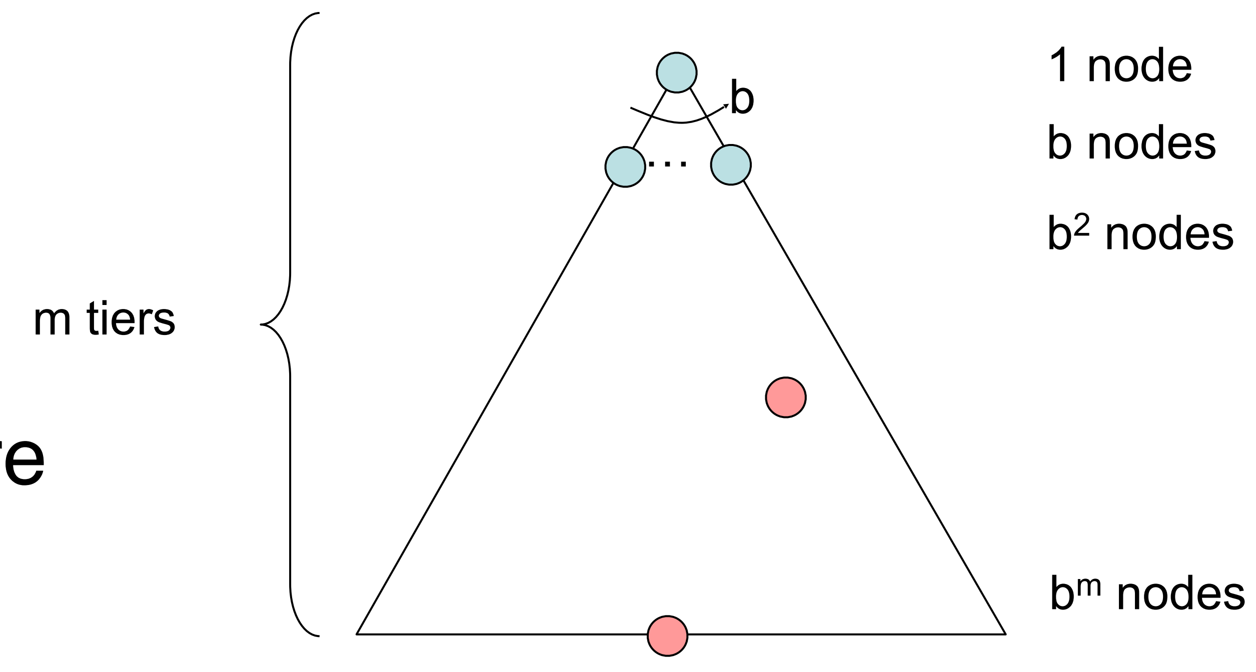
Best First Search Properties

- **Complete?**
  - No
- **Optimal?**
  - No
- **Time complexity**
  - It could process the entire tree until it finds a goal! Therefore $O(b^m)$
- **Space complexity**
  - $O(b^m)$



Best First Search Properties

- **Complete?**
  - No
- **Optimal?**
  - No
- **Time complexity**
  - It could process the entire tree until it finds a goal! Therefore $O(b^m)$
- **Space complexity**
  - $O(b^m)$

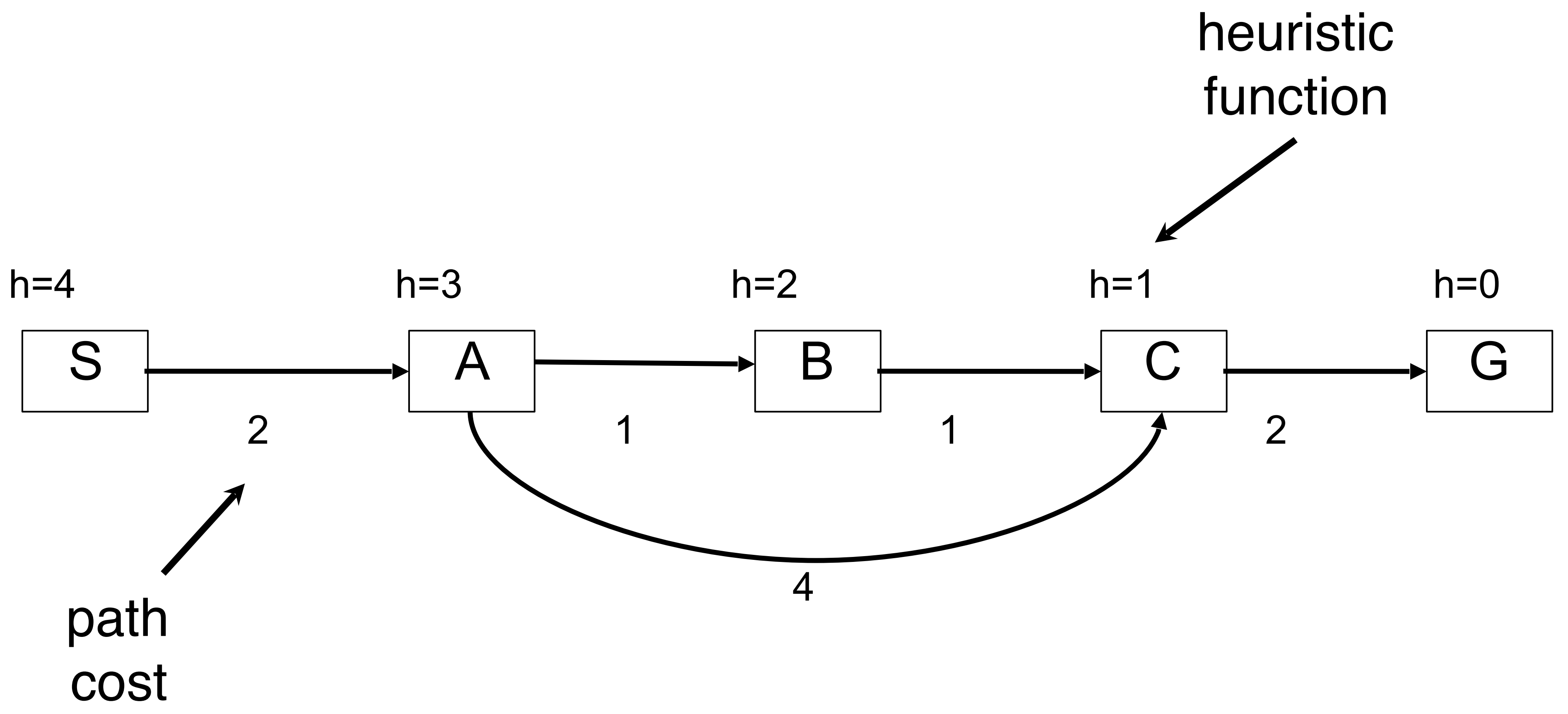


But, if you have a good heuristic, you might do much better

A* Search

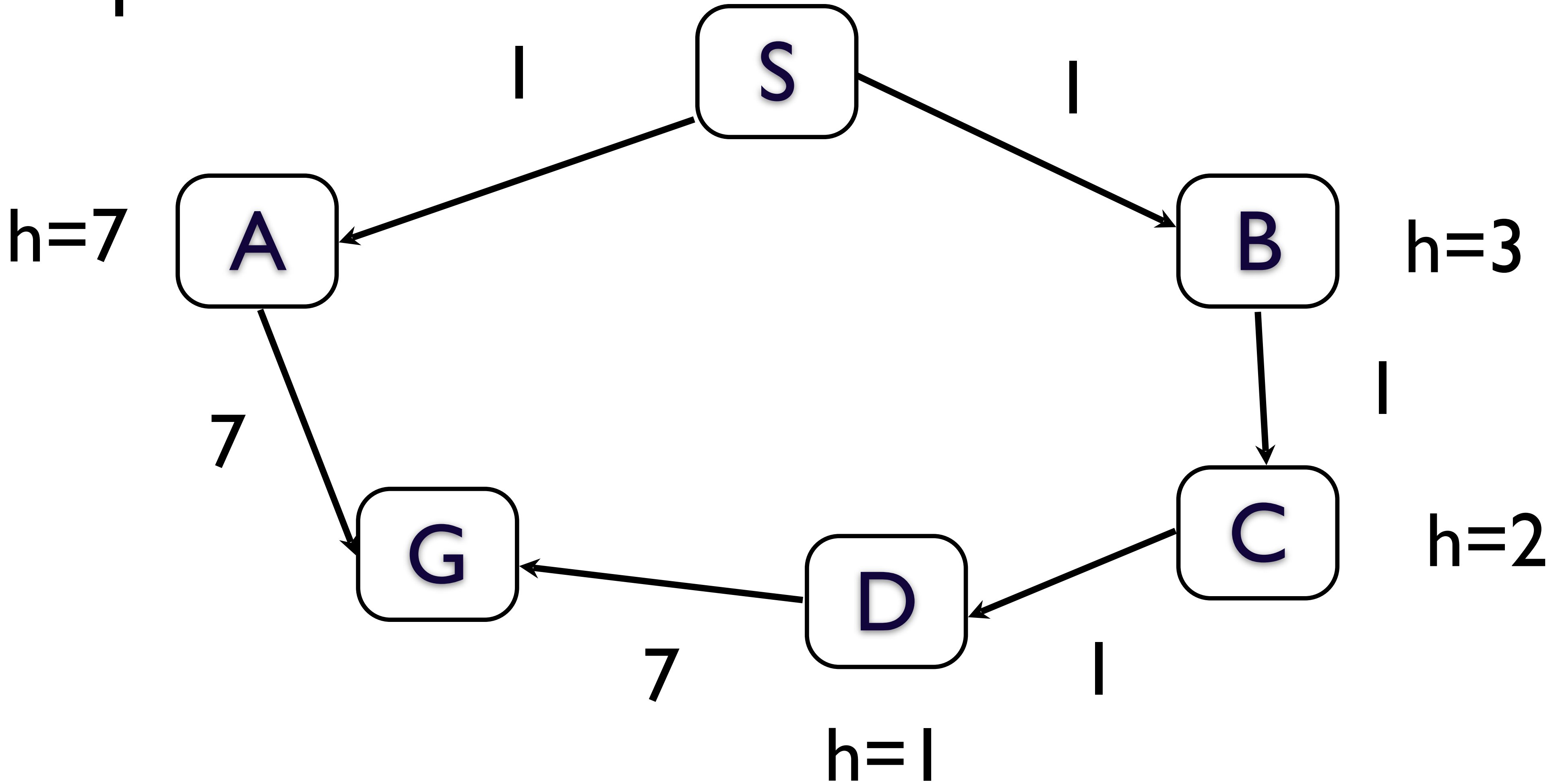
- Observations
  - Best first search ordered nodes for *forward cost to goal,* h(n)
  - Uniform cost search ordered nodes by *backward cost of path so far*, g(n)
- A* search
  - Expand nodes in order f(n)=g(n)+h(n) (*estimate of cost of entire path)*

Example: A* search



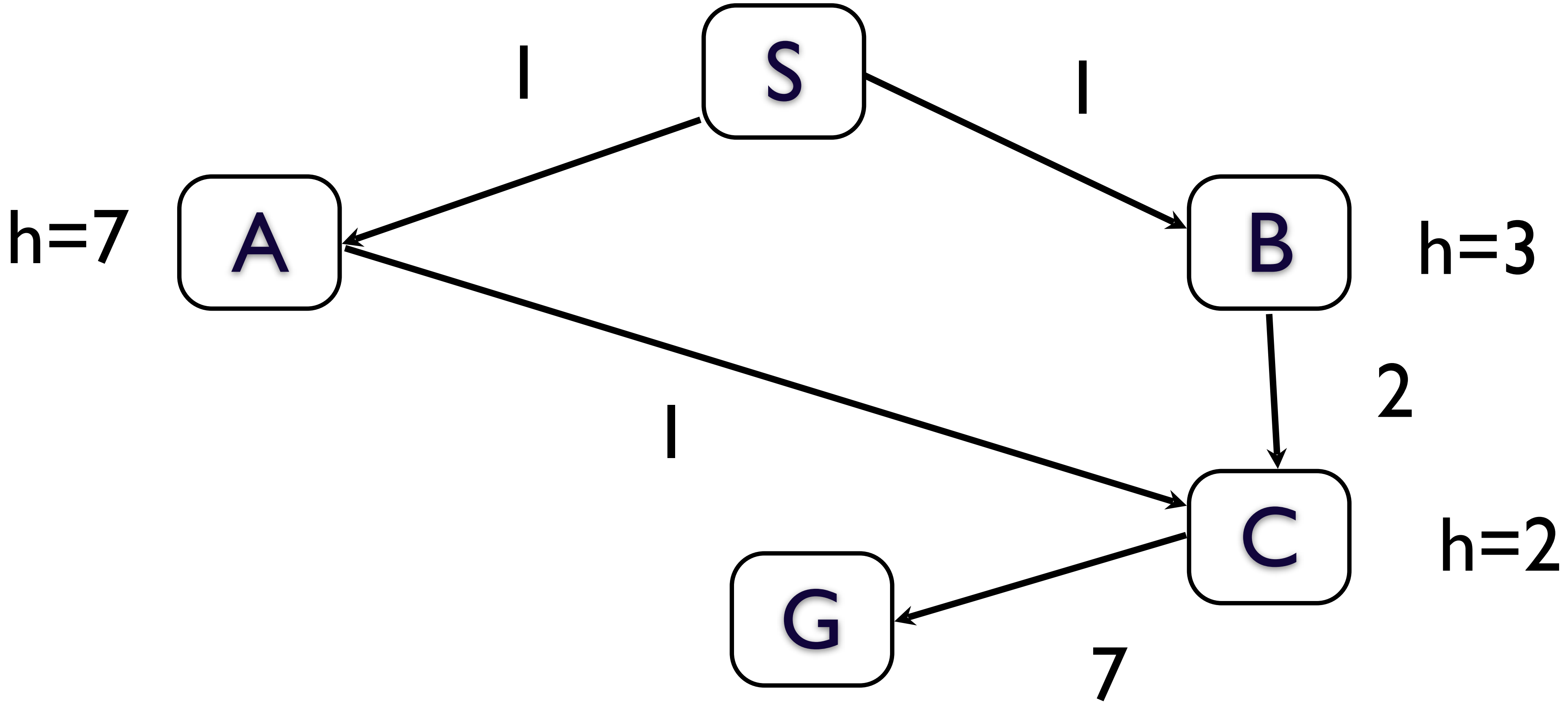
When Should A* Terminate?

- Only when G has been popped from the queue

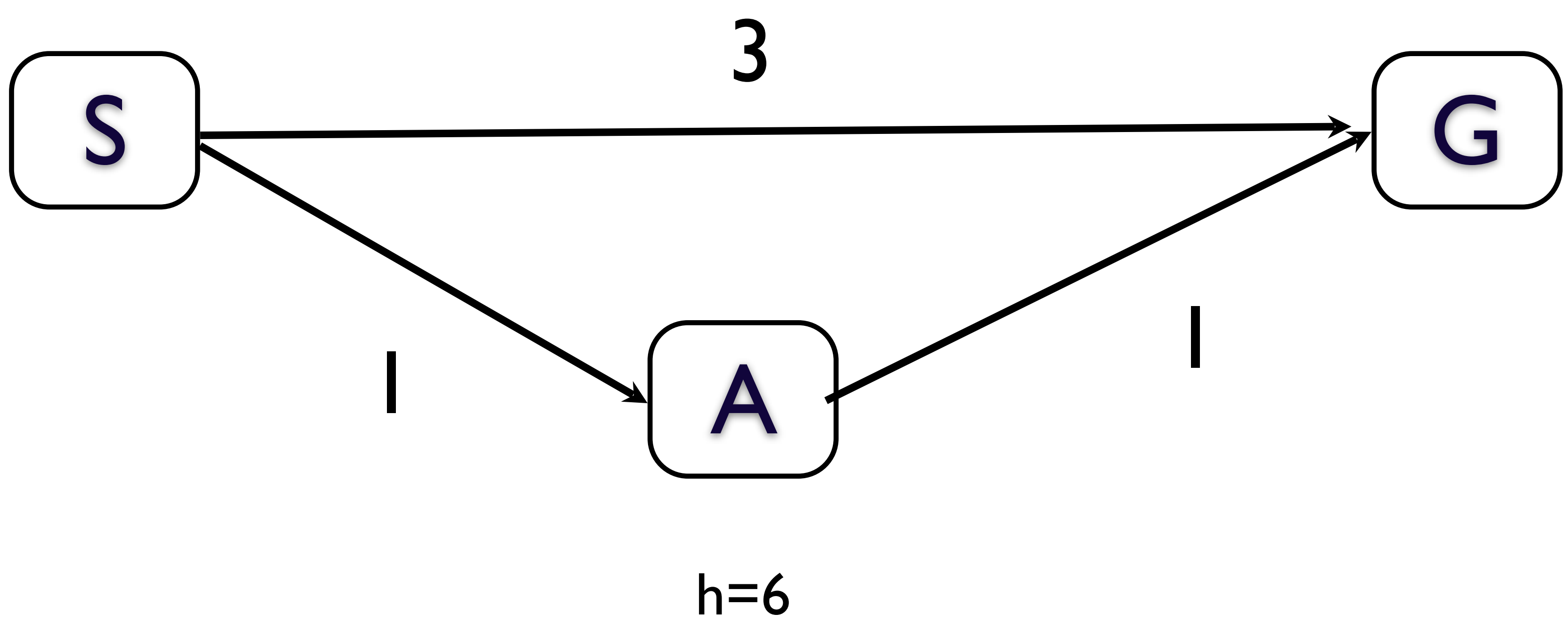


A* and Revisiting States

- What if we revisit a state that was already expanded?



Is A* Optimal?



Admissible Heuristics

A heuristic, h, is ***admissible*** if

$$0 \leq h(n) \leq h^*(n)$$

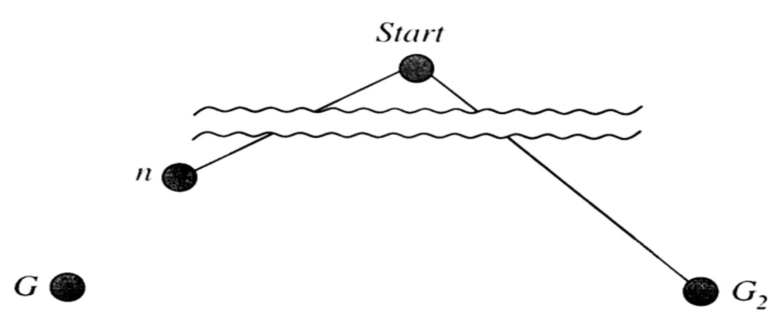
for all n, where $h^*(n)$ is the (true) shortest path from n to any goal state

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

Optimality of A*

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

Proof by contradiction
Let goal G_2 be in the queue. Let n be an unexpanded node on the shortest path to optimal goal G.
Assume that A* chose G_2 to expand. Thus, it must be that $f(n)>f(G_2)$



But
$f(G_2)=g(G_2)$ since $h(G_2)=0$
$>= g(G)$ since G_2 is suboptimal
$>= f(n)$ since h is admissible

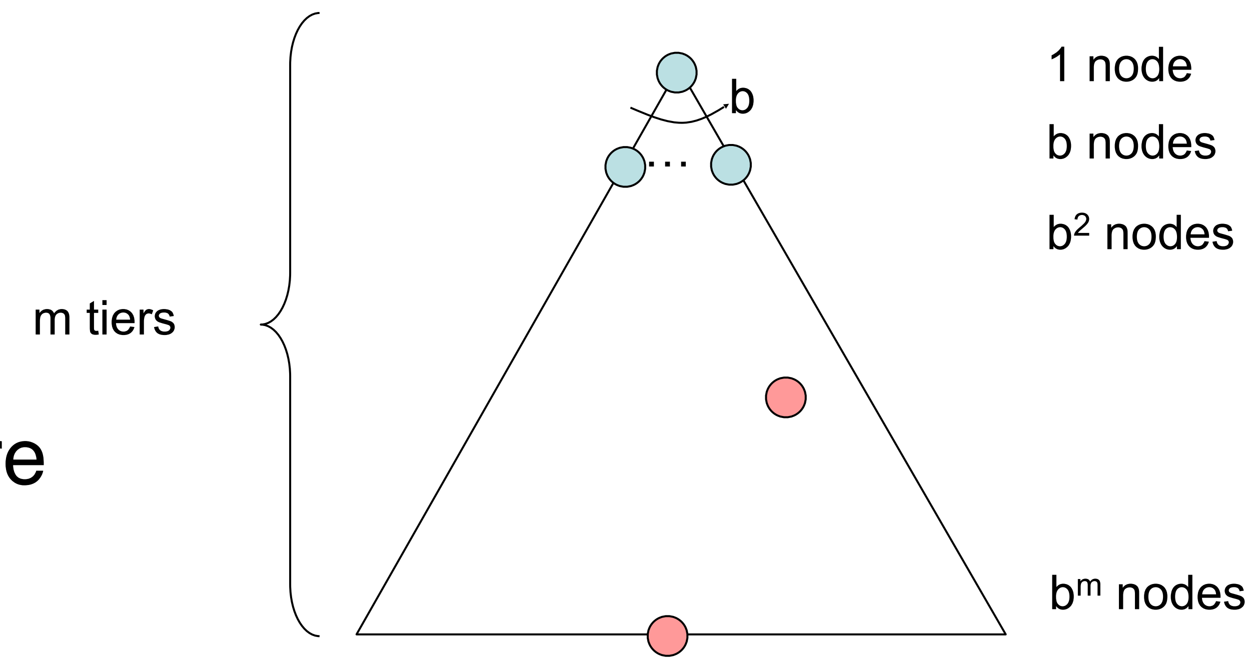
Contradiction. Therefore, A* will never select G_2 for expansion.

Optimality of A*

- For graphs we require consistency
  - $h(n) \leq cost(n,n') + h(n')$
  - Almost any admissible heuristic function will also be consistent
- A* search on graphs with a consistent heuristic is optimal

A* Search Properties

- **Complete?**
  - Yes!
- **Optimal?**
  - Yes!
- **Time complexity**
  - It could process the entire tree until it finds a goal! Therefore $O(b^m)$
- **Space complexity**
  - $O(b^m)$ (keeps all generated nodes in memory)



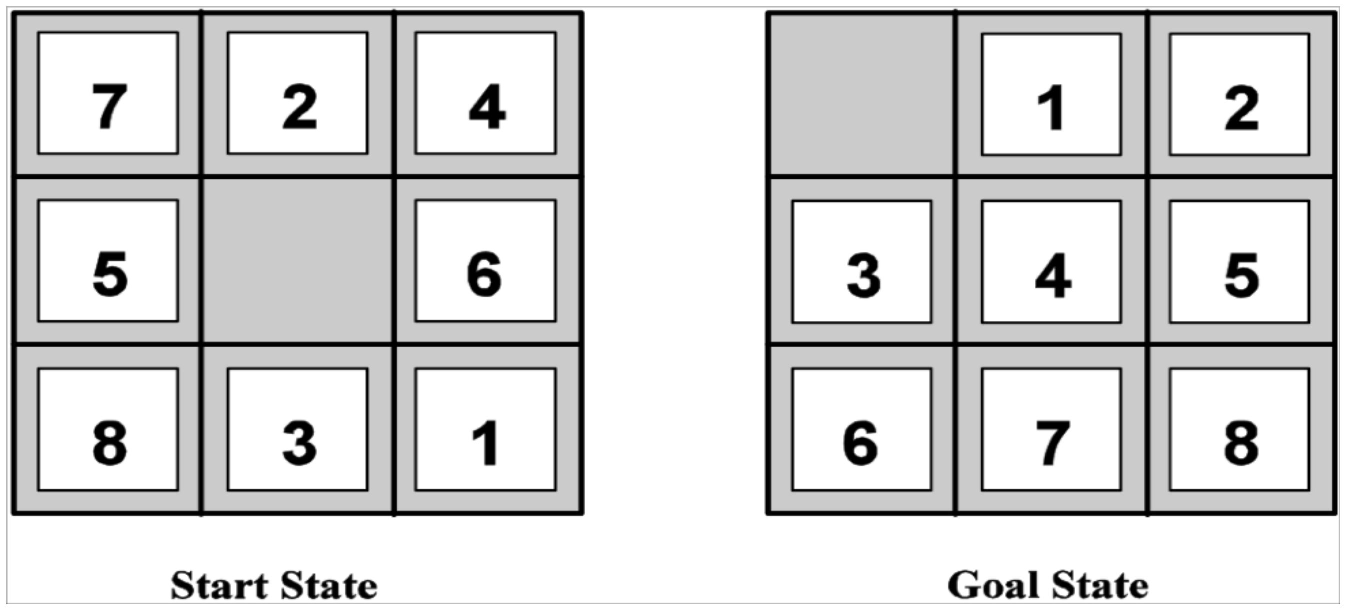
But, if you have a good heuristic, you might do much better

Heuristic Functions

- A good heuristic function can make all the difference!

- How do we get heuristics?

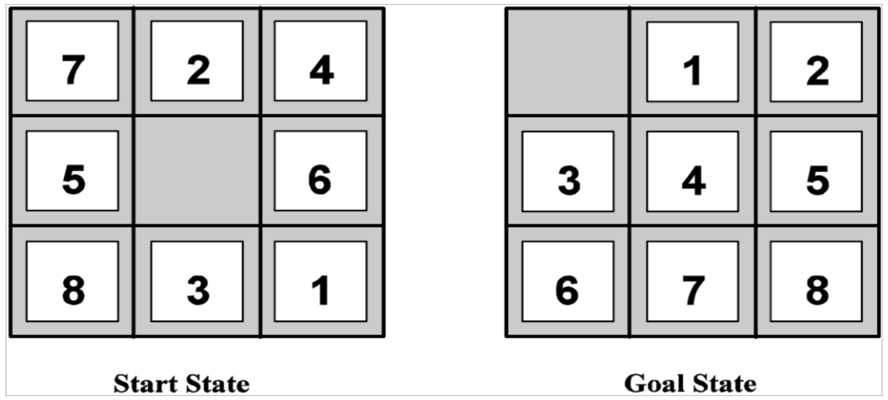
8 Puzzle



- Relax the game
  1. Can move from A to B is A is next to B
  2. Can move from A to B if B is blank
  3. Can move from A to B

8 Puzzle

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



Which is the better heuristic?
(Which one dominates?)

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

Memory-Bounded Heuristic Search

- Iterative Deepening A* (IDA*)
  - Basically depth-first search but using the f-value to decide which order to consider nodes
  - Use f-limit instead of depth limit
    - New f-limit is the smallest f-value of any node that exceeded cutoff on previous iteration
  - Additionally keep track of next limit to consider
  - IDA* has same properties as A* but uses less memory

Memory-Bounded Heuristic Search

- Simplified Memory-Bounded A* (SMA*)
  - Uses all available memory
  - Proceeds like A* but when it runs out of memory it drops the worst leaf node (one with highest f-value)
  - If all leaf nodes have same f-value, drop oldest and expand newest
  - Optimal and complete if depth of shallowest goal node is less than memory size