CSPB3202 Artificial Intelligence

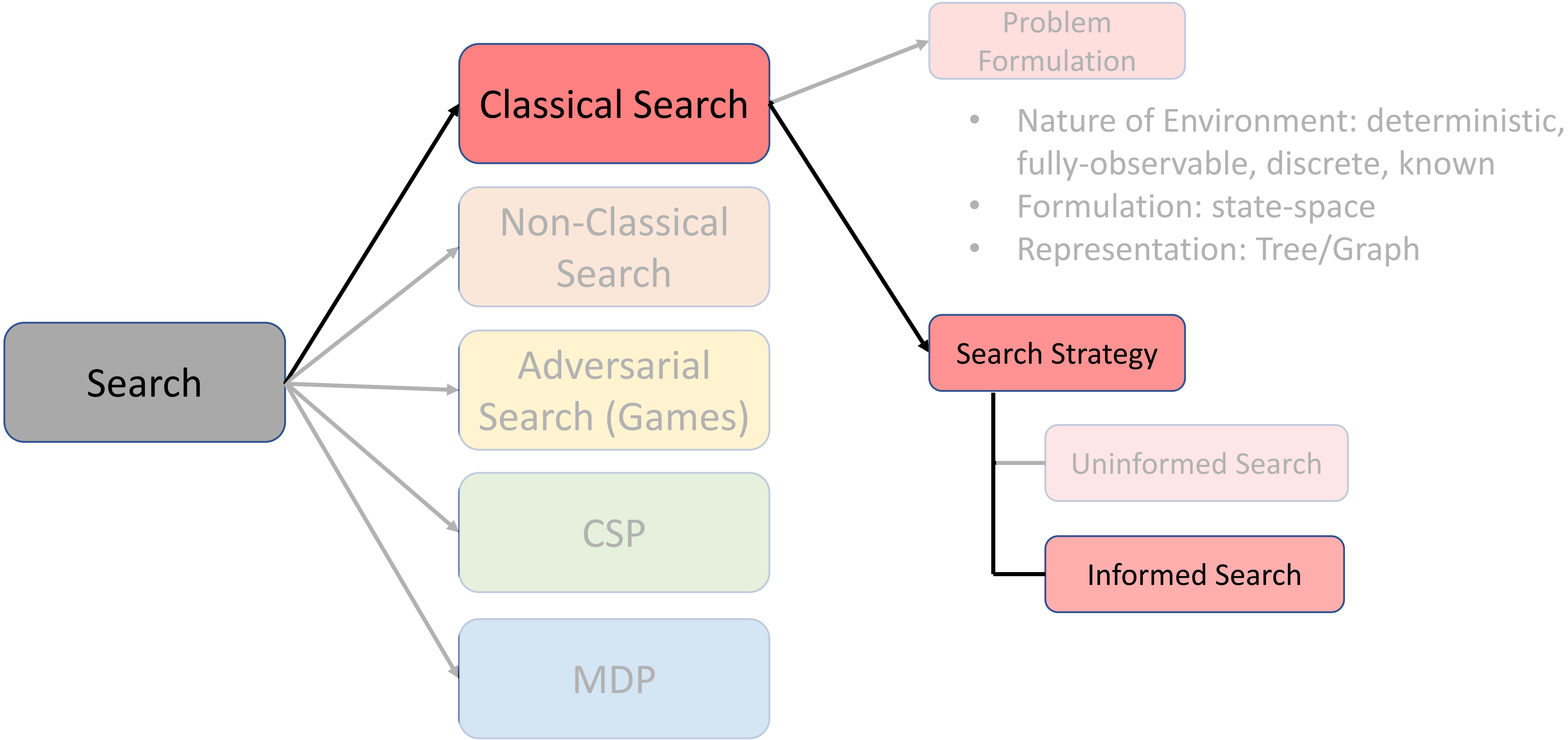
Search

University of Colorado Boulder

Note: Some slides were adapted from the materials from Prof. Pieter Abbeel and Prof. Dan Klein at UC Berkeley, and Russell & Norvig's AI book.

Informed Search

Where we are


Search
Classical Search
Non-Classical Search
Adversarial Search (Games)
CSP
MDP
Problem Formulation
Nature of Environment: deterministic, fully-observable, discrete, known
Formulation: state-space
Representation: Tree/Graph
Search Strategy
Uninformed Search
Informed Search

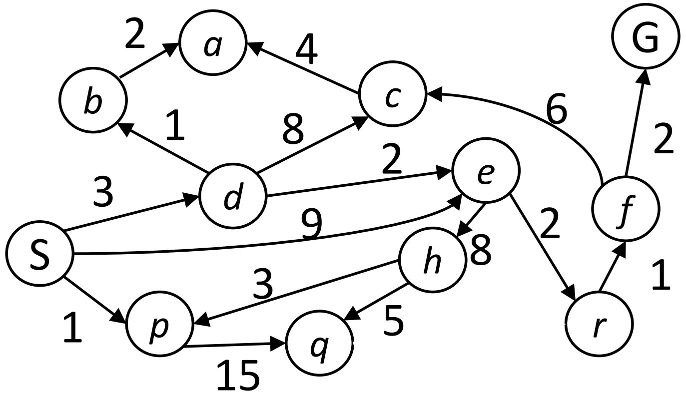
Review- Uninformed Search Summary

	DFS	BFS	Iterative Deepening	UCS
Completeness	Yes only if no cycle	Yes	Yes	Yes (assuming no cost loops)
Optimality	No	Yes if all costs are equal	Yes if all costs are equal	Yes
Time Complexity	$O(b^m)$	$O(b^s)$	$O(b^s)$	$O(b^{C*/\varepsilon})$
Space Complexity	$O(bm)$	$O(b^s)$	$O(bs)$	$O(b^{C*/\varepsilon})$
Advantage	Efficient in space	Relatively efficient in time when $s < m$	Combines advantages of DFS and BFS	Complete and optimal
Disadvantage / Limitation	Not optimal, can go into a rabbit hole	Not memory efficient	Does not consider cost	Cheap cost so far doesn't mean it's a right direction

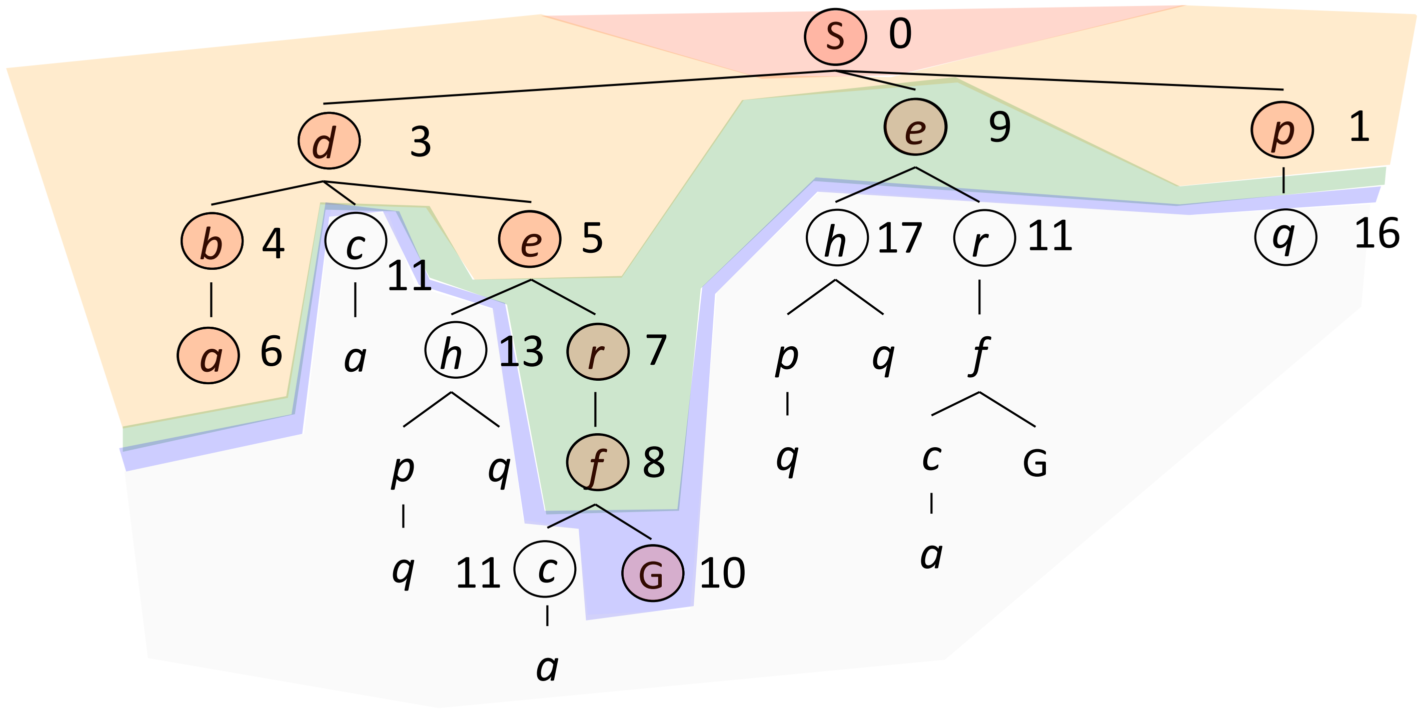
Review- Uniform Cost Search

Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)



Cost contours

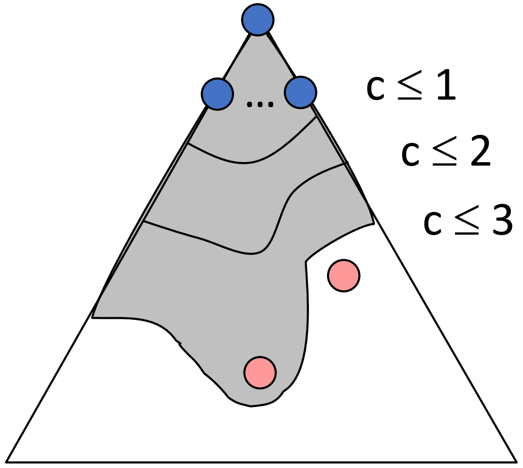


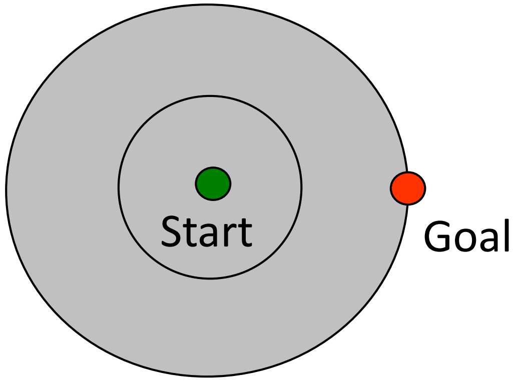
Review- Uniform Cost Issues

- Remember: UCS explores increasing cost contours

- The good: UCS is complete and optimal!

- The bad:
  - Explores options in every “direction”
  - No information about goal location

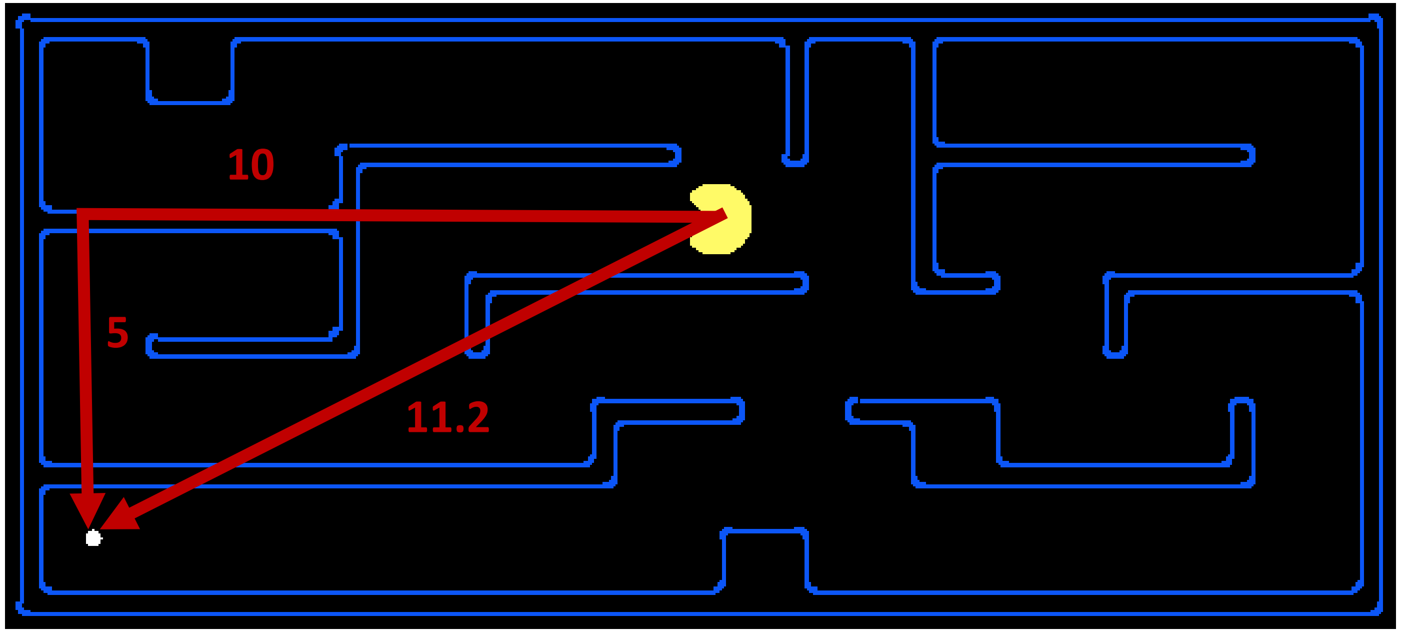


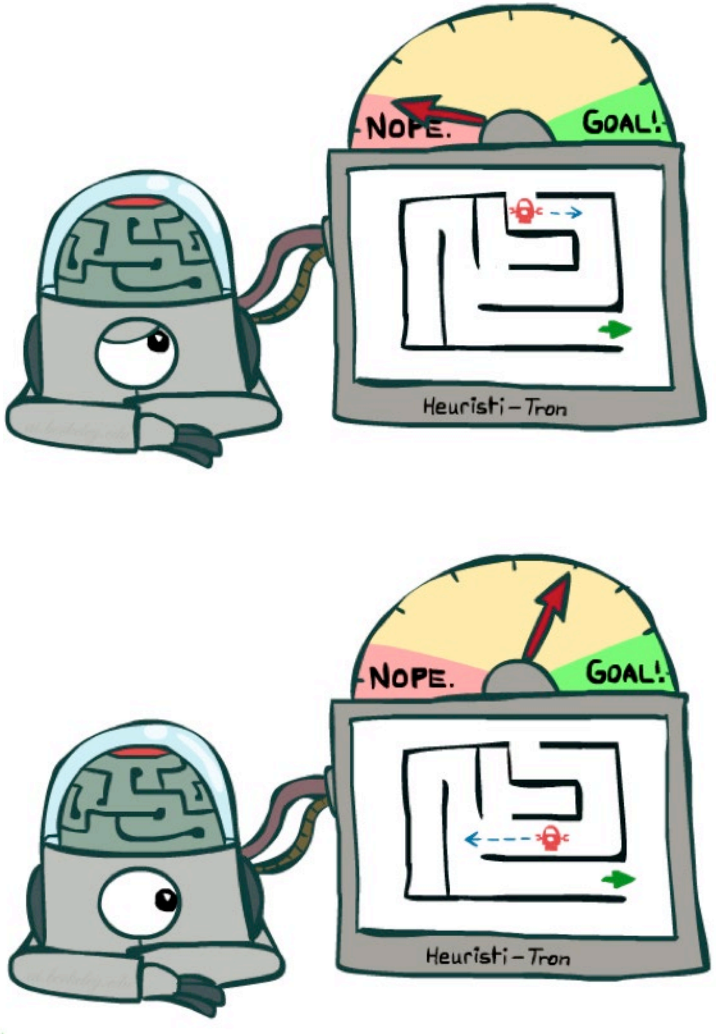


Informed Search

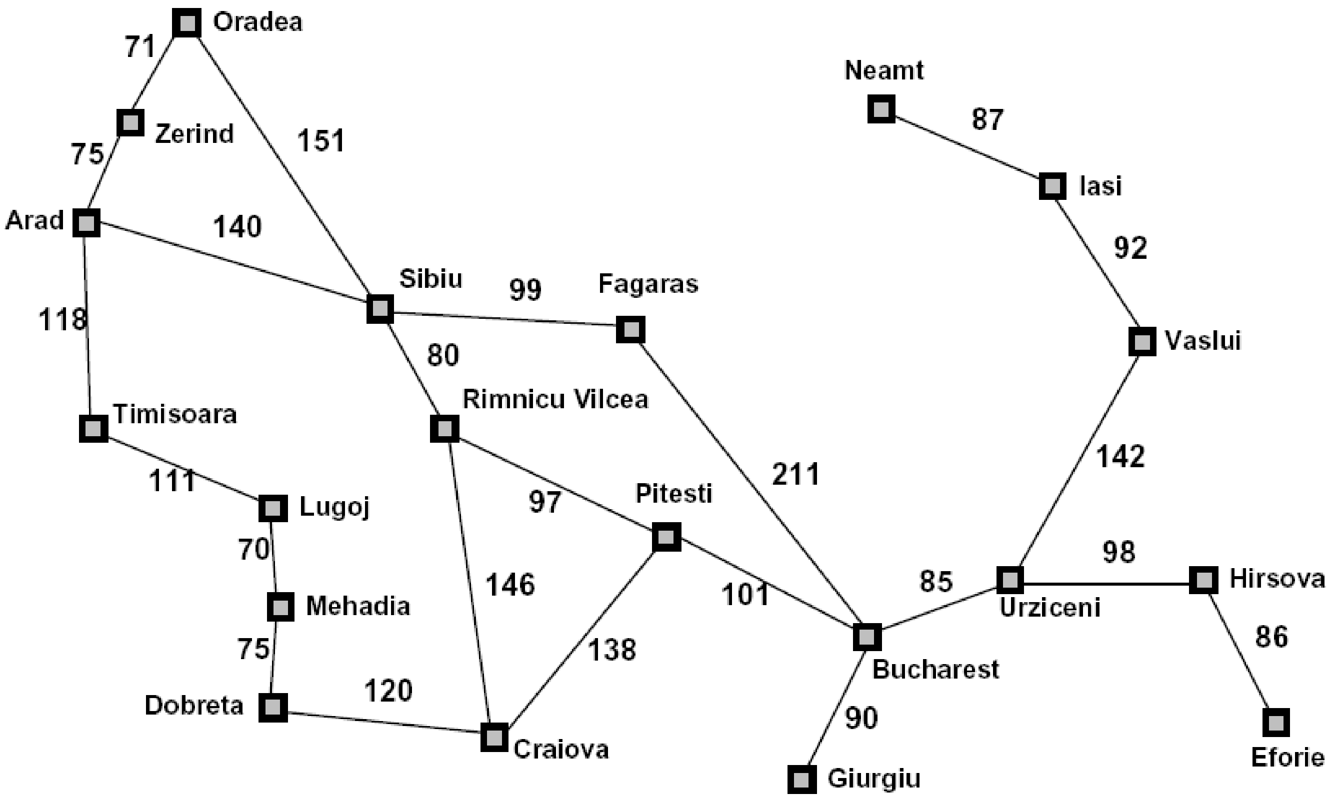
Search Heuristics

- A heuristic is:
  - A function that *estimates* how close a state is to a goal
  - Designed for a particular search problem
  - Examples: Manhattan distance, Euclidean distance for pathing





Example: Heuristic Function



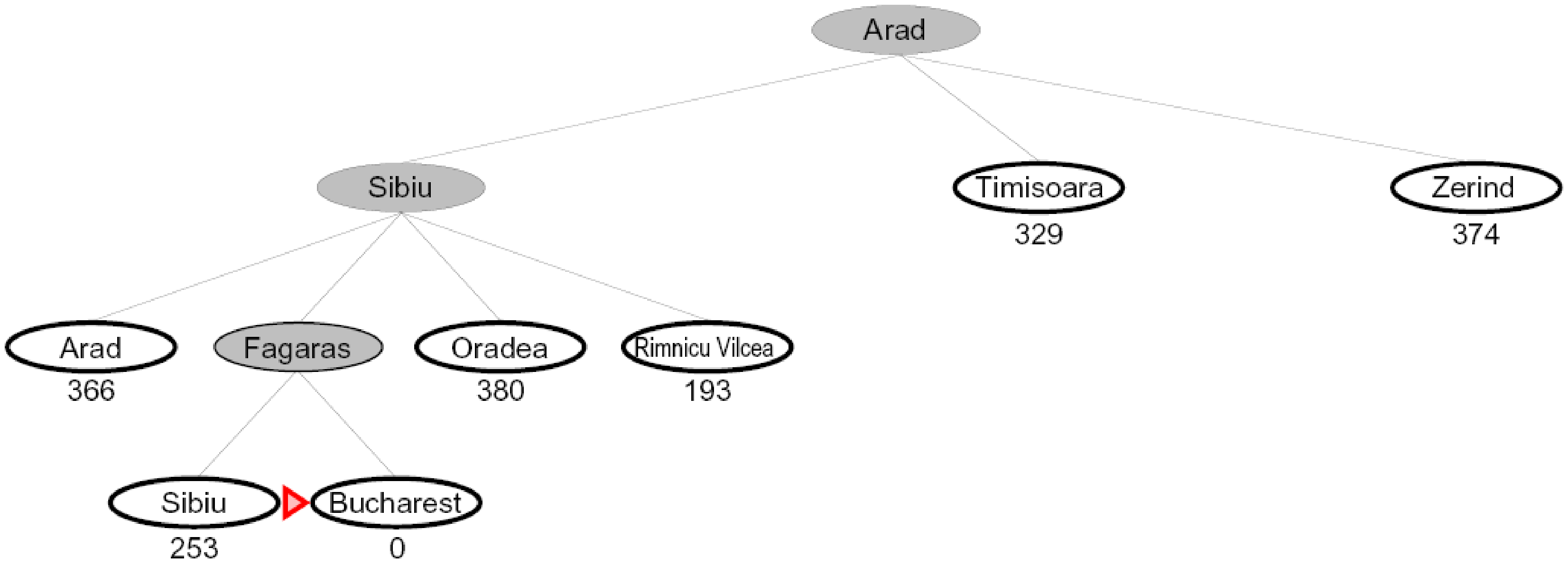
Straight−line distance to Bucharest

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)

Greedy Search

- Expand the node that seems closest…

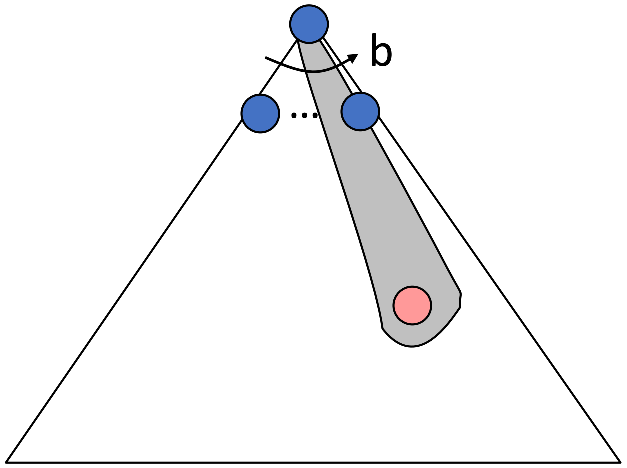


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

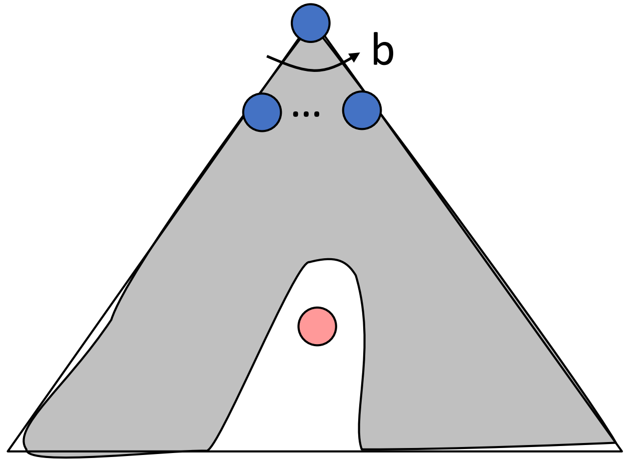
- What can go wrong?

Greedy Search

- Strategy: expand a node that you think is closest to a goal state
  - Heuristic: estimate of distance to nearest goal for each state



- A common case:
  - Best-first takes you straight to the (wrong) goal

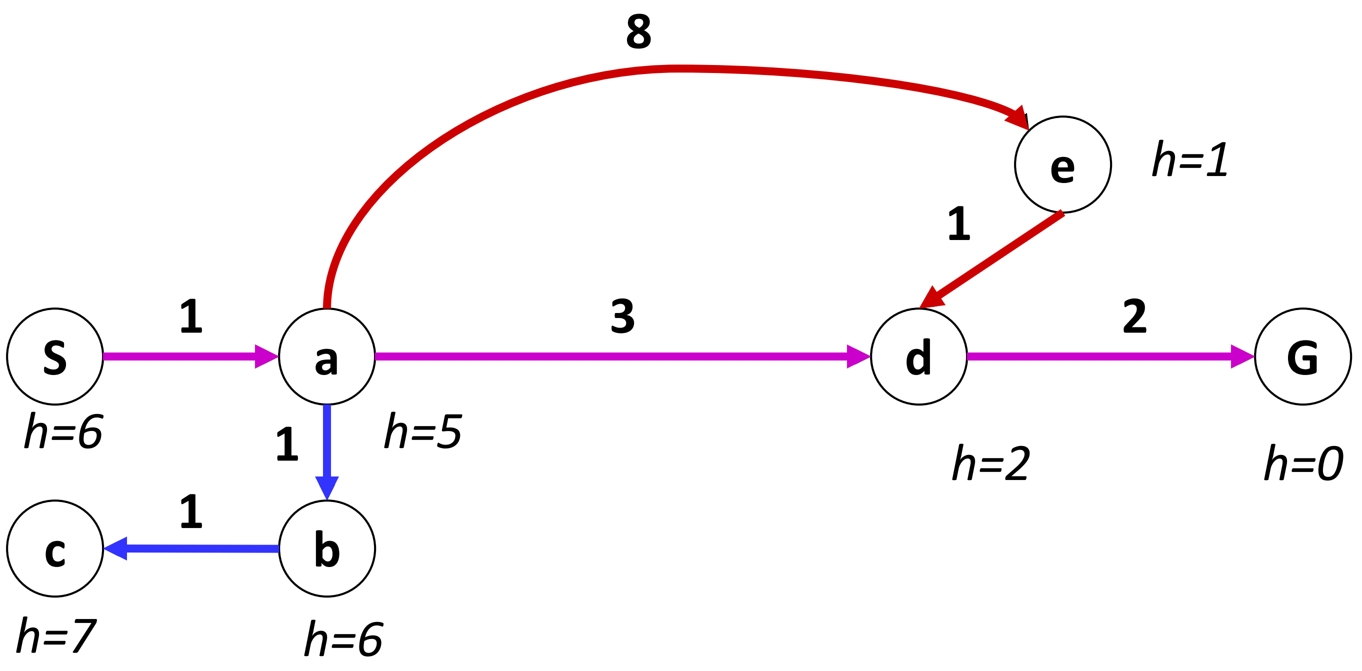


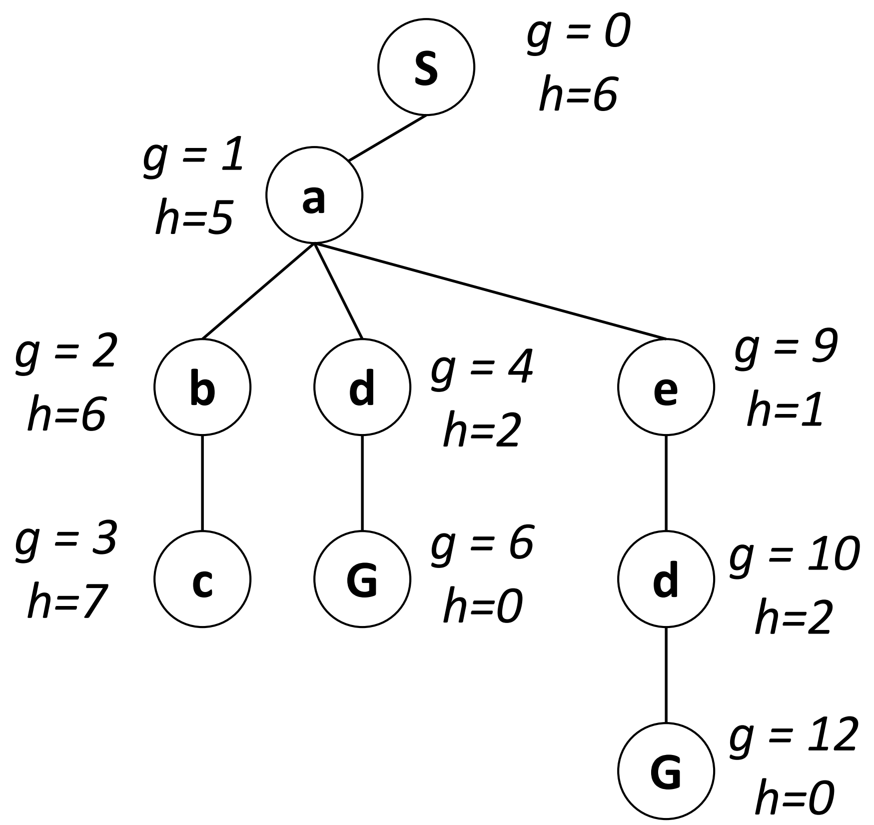
- Worst-case: like badly guided DFS

Need another strategy?

Combining UCS and Greedy

- Uniform-cost orders by path cost, or *backward cost* g(n)
- Greedy orders by goal proximity, or *forward cost* h(n)

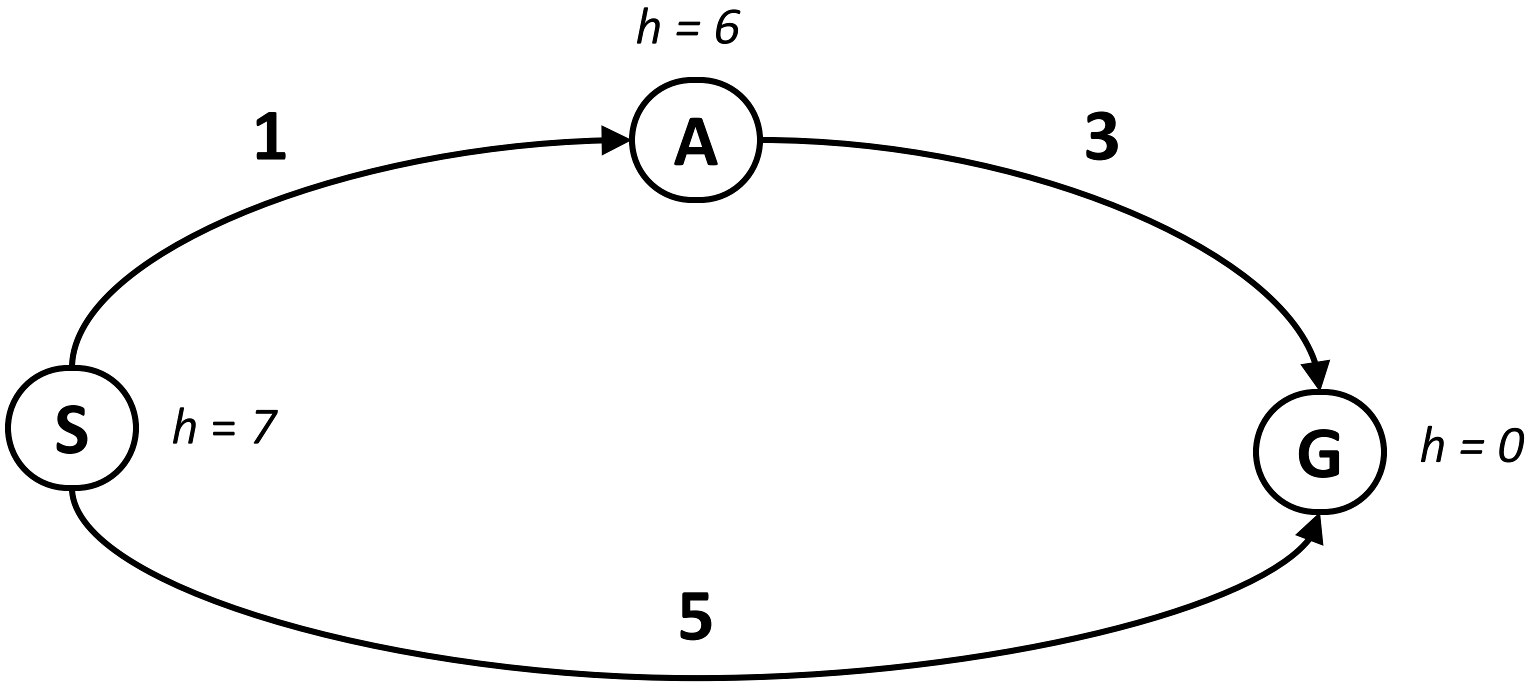




- A* Search orders by the sum: f(n) = g(n) + h(n)

Example: Teg Grenager

Is A* Optimal?



- What went wrong?
- Actual bad goal cost < estimated good goal cost
- We need estimates to be less than actual costs!

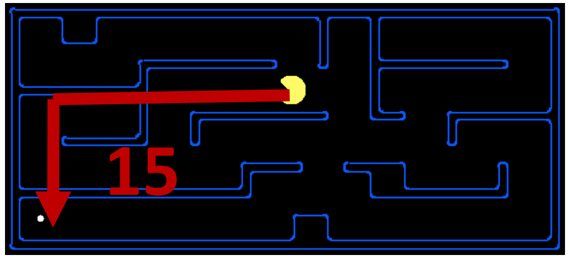
Admissible Heuristics

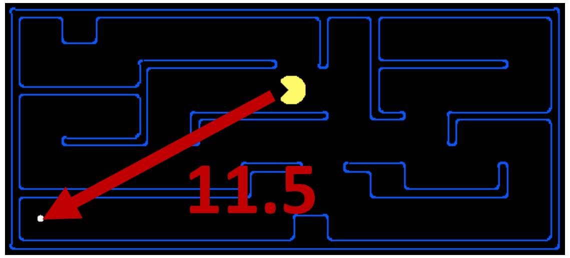
- A heuristic h is *admissible* (optimistic) if:

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

  where $h^*(n)$ is the true cost to a nearest goal

- Examples:





0.0

- Coming up with admissible heuristics is most of what's involved in using A* in practice.

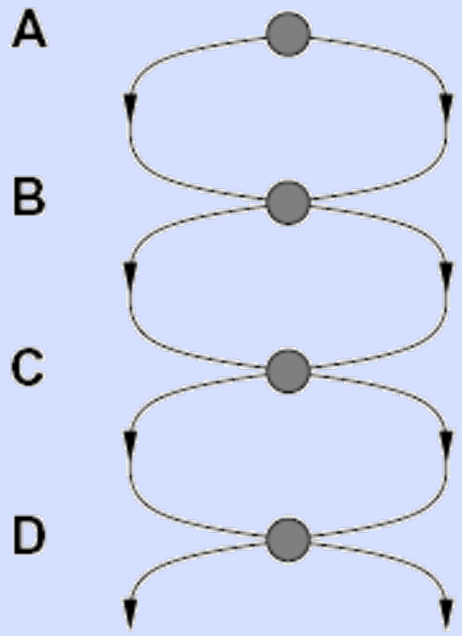
Admissibility and Optimality

Is A* optimal if the heuristic is admissible?

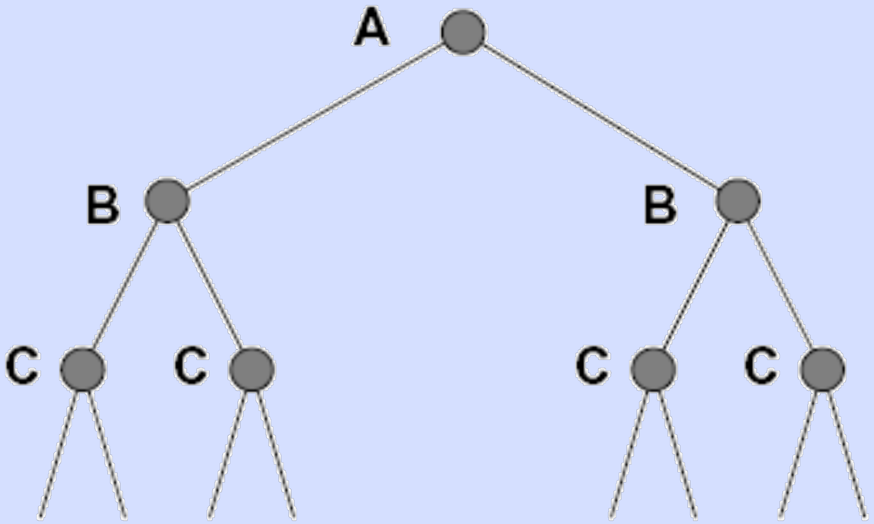
What's wrong with Tree Search?

- Failure to detect repeated states can cause exponentially more work.

State Graph

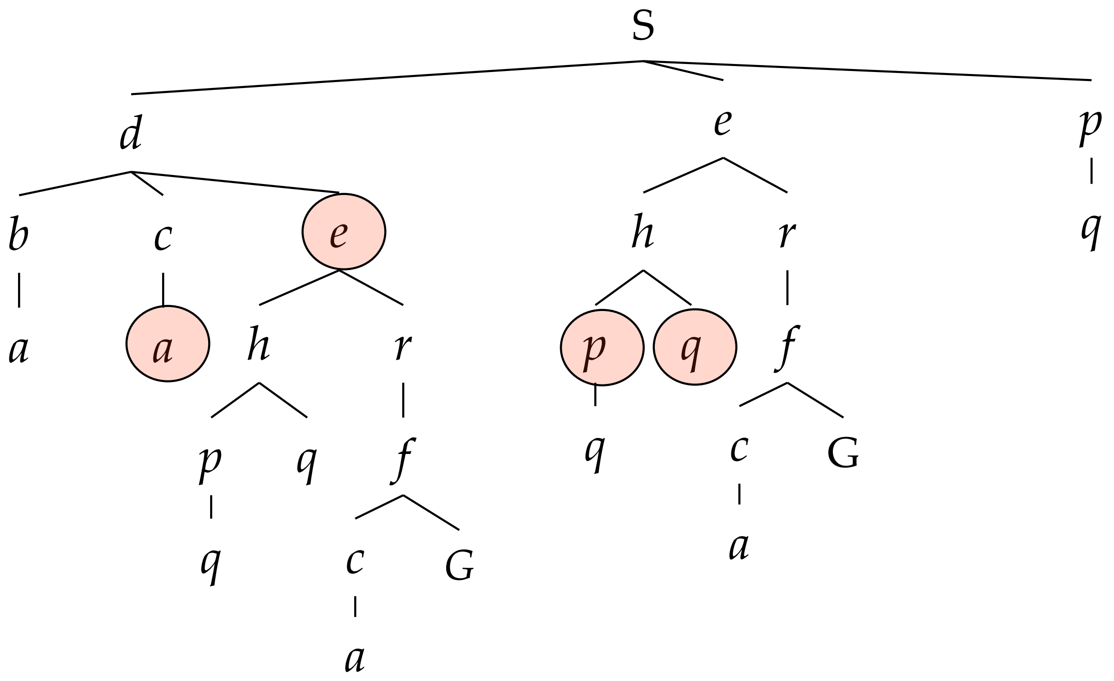


Search Tree



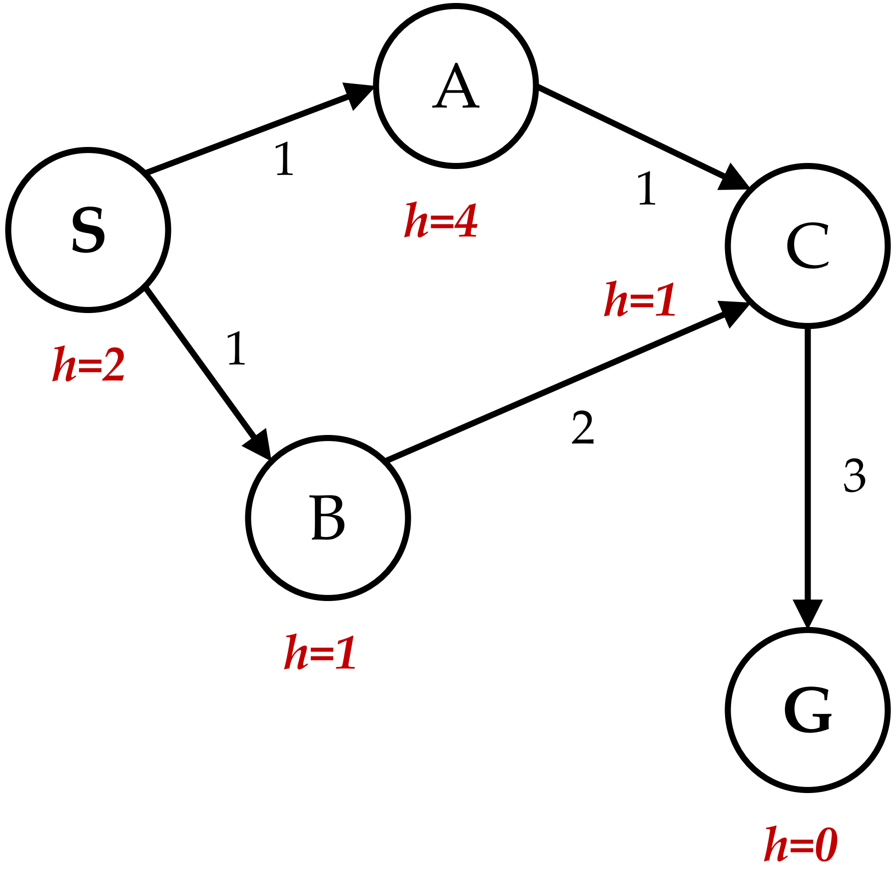
Graph Search

- In BFS, for example, we shouldn't bother expanding the circled nodes (why?)

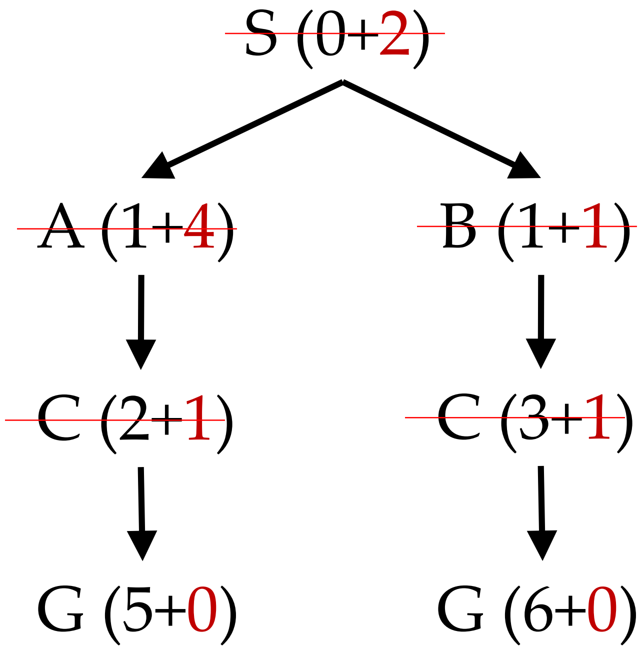


A* Graph Search Gone Wrong?

State space graph

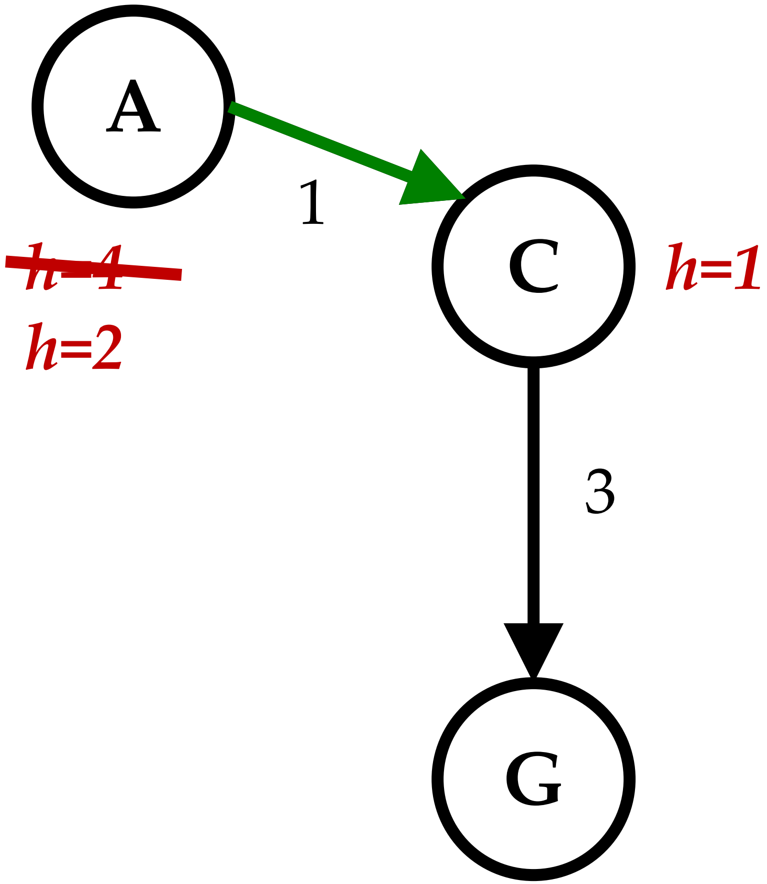


Search tree



Closed Set:S B C A

Consistency of Heuristics



- Admissibility: heuristic cost ≤ actual cost to goal

  h(A) ≤ actual cost from A to G

- Consistency: heuristic "arc" cost ≤ actual cost for each arc

  h(A) – h(C) ≤ cost(A to C)

- Consequences of consistency:
  - The f value along a path never decreases

    h(A) ≤ cost(A to C) + h(C)

  - A* graph search is optimal

A*: Summary

- A* uses both backward costs and (estimates of) forward costs
- A* is optimal with admissible / consistent heuristics
- Heuristic design is key: often use relaxed problems