Artificial Intelligence

Lec 3 - Informed Search

Pratik Mazumder

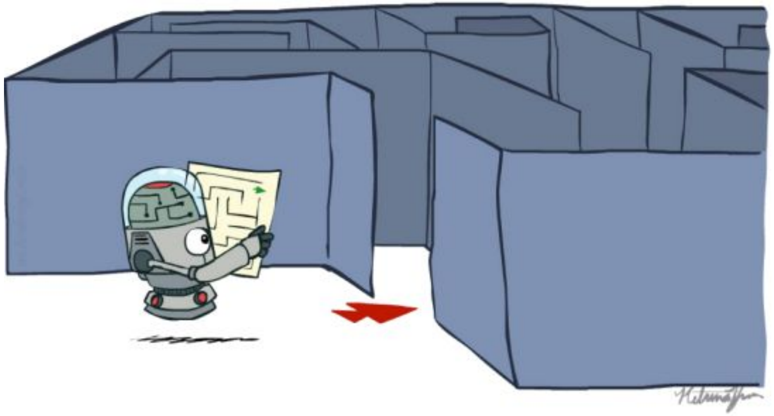
Content taken from various sources including
CS188 Intro to AI at UC Berkeley

Recap: Search

- **Search problem:**
  - States (configurations of the world)
  - Actions and costs
  - Successor function (world dynamics)
  - Start state and goal test

- **Search tree:**
  - Nodes: represent plans for reaching states
  - Plans have costs (sum of action costs)

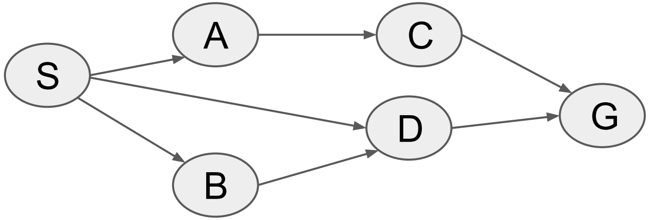
- **Search algorithm:**
  - Systematically builds a search tree
  - Chooses an ordering of the fringe (unexplored nodes)
  - Optimal: finds least-cost plans



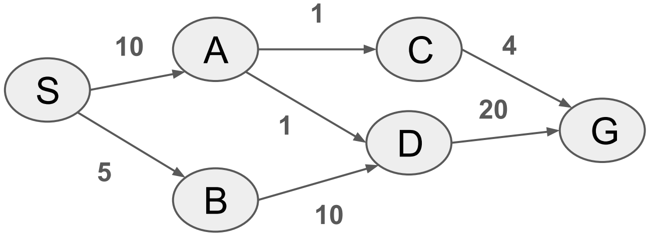
Practice

Which chooses the shorter plan? BFS, DFS, UCS

Ques1



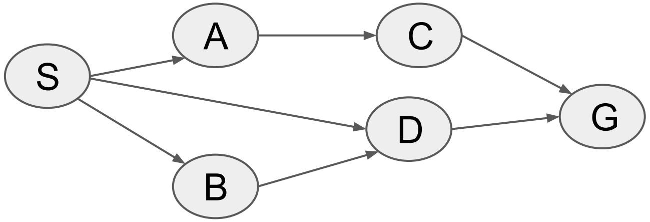
Ques2

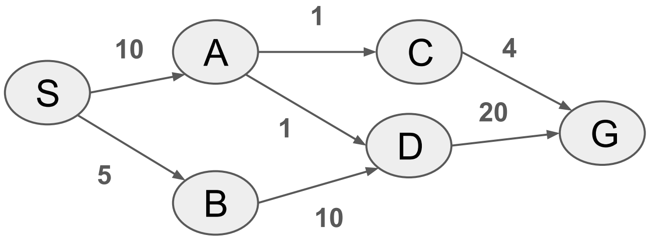


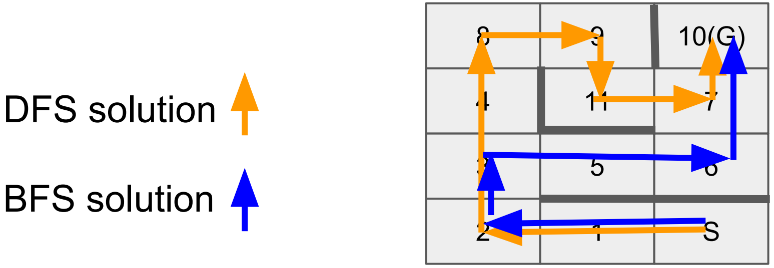
Ques3

Practice

Which chooses the shorter plan? BFS, DFS, UCS

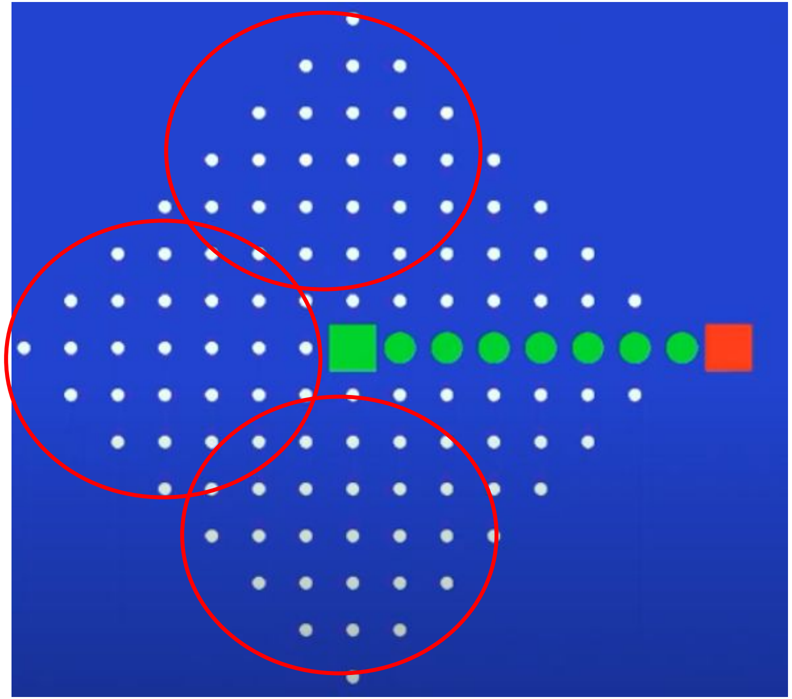






Uninformed Search

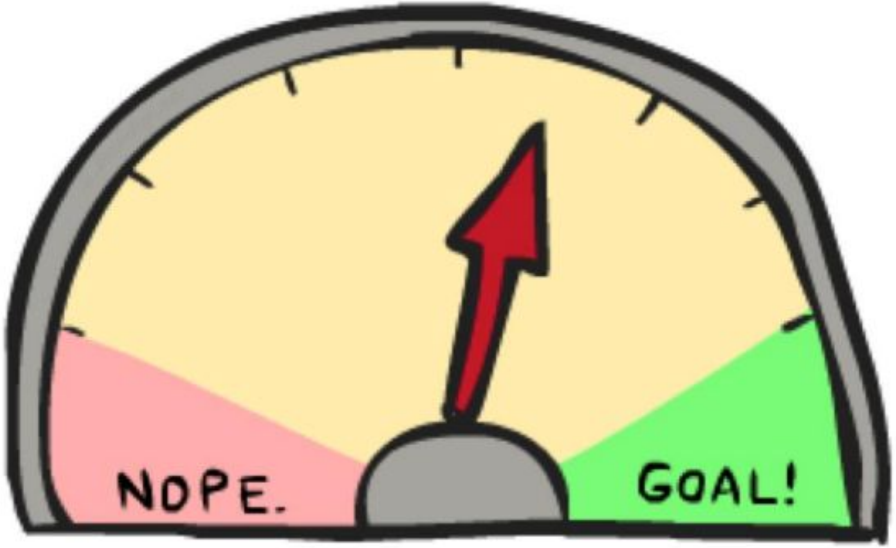
- BFS and UCS give complete plans.
- BFS gives optimal plans in terms of no. of actions.
- UCS gives optimal plans in terms of the cost involved.



The bad aspect:

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

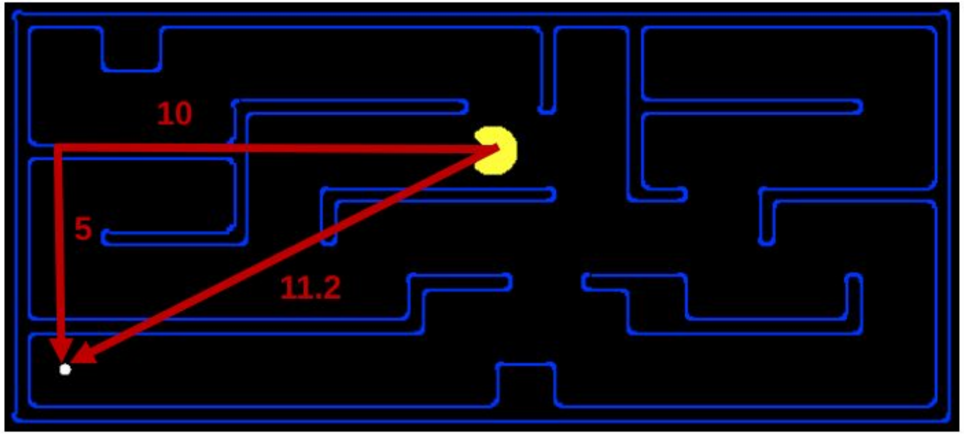
Informed Search

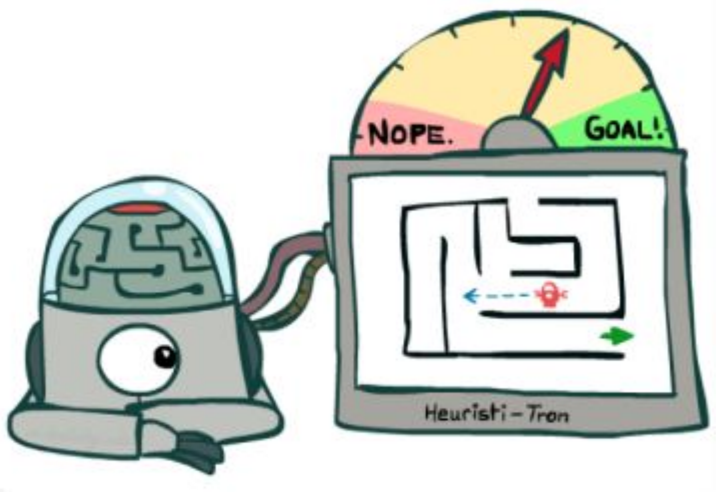


Search Heuristics

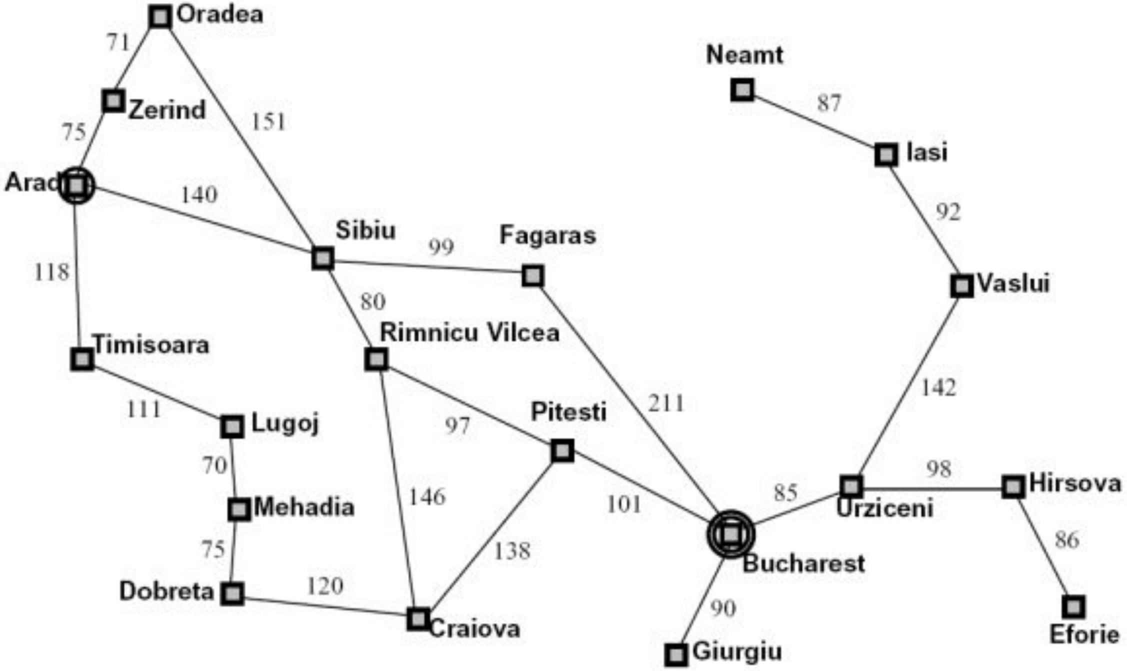
A heuristic is:

- 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
- Guesses - may not be the actual distance





Example: Heuristic Function



Straight−line distance to Bucharest

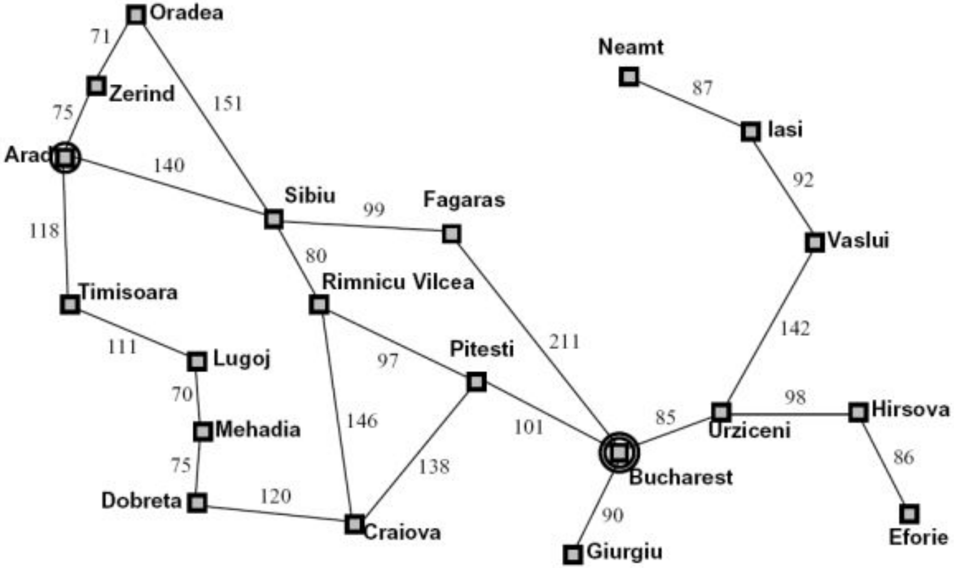
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

Greedy Search

Greedy Search

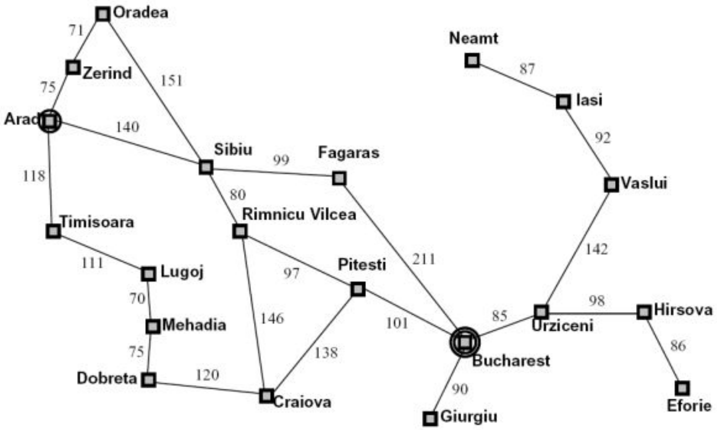
Strategy: Expand the **node that seems closest to a goal state**



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



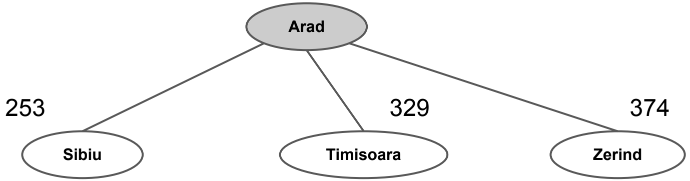
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

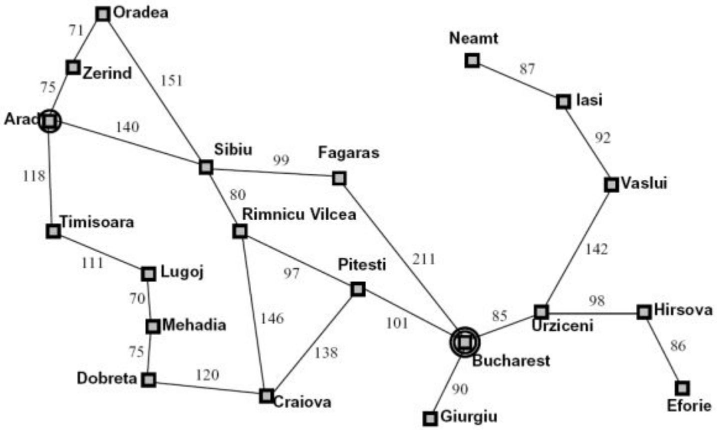
h(x)

<u>Fringe</u>

Arad, 366

Greedy Search





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)

Fringe

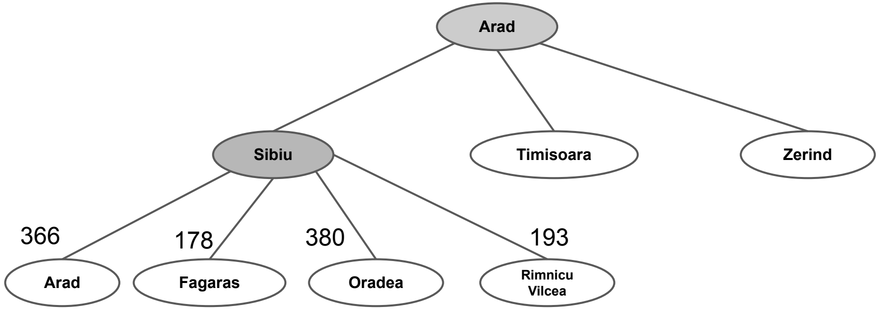
~~Arad, 366~~

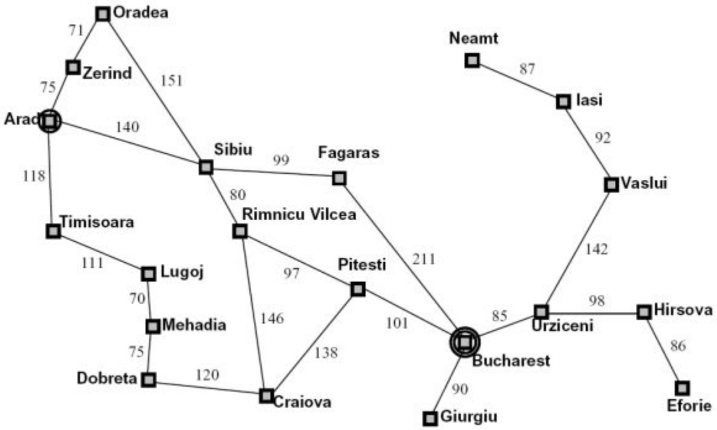
Arad→Sibiu,253

Arad→Timisoara,329

Arad→Zerind,374

Greedy Search





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)

Fringe

~~Arad, 366~~

~~Arad→Sibiu,253~~

Arad→Timisoara,329

Arad→Zerind,374

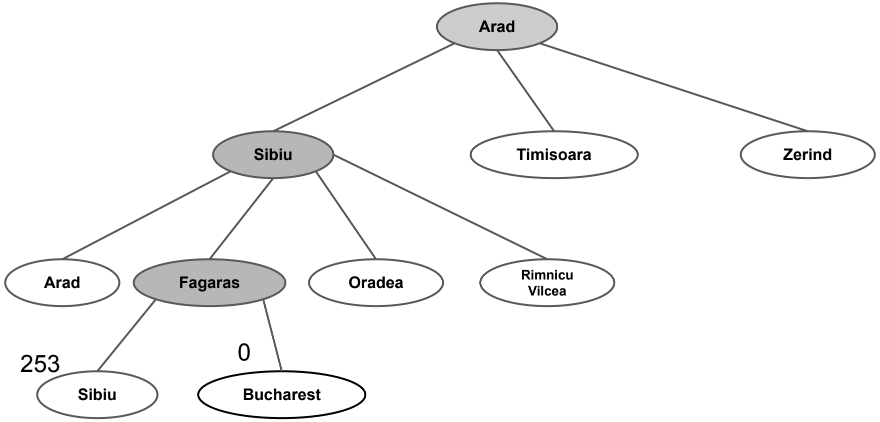
Arad→Sibiu→Arad,366

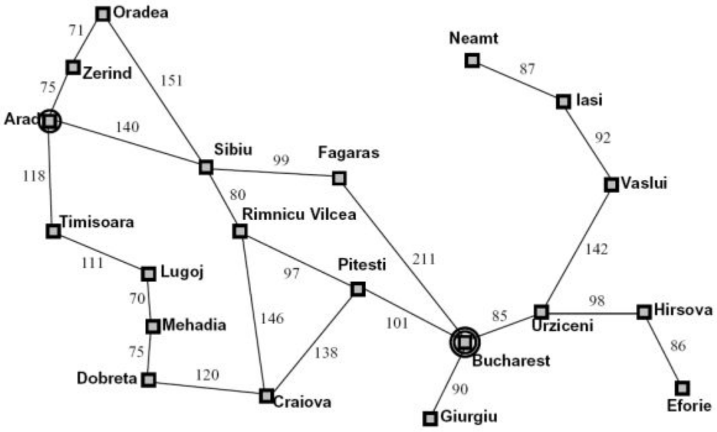
Arad→Sibiu→Fagaras,178

Arad→Sibiu→Oradea,380

Arad→Sibiu→Rimnicu Vilcea,193

Greedy Search





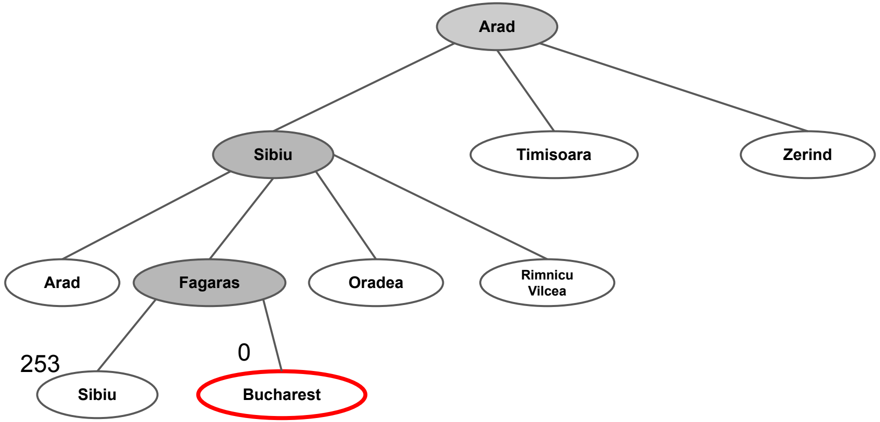
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

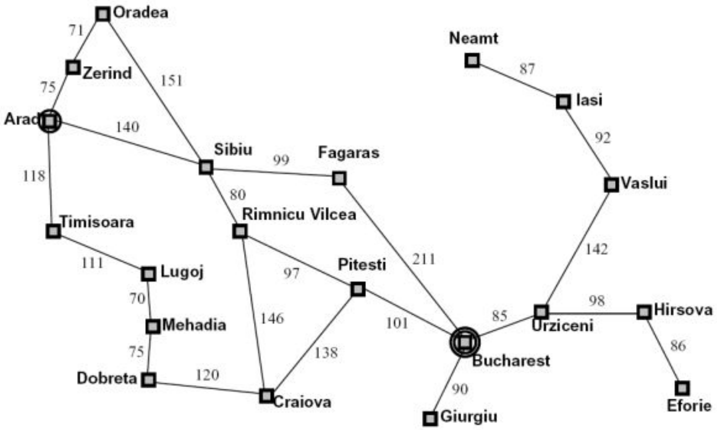
h(x)

Fringe

~~Arad, 366~~
~~Arad→Sibiu,253~~
Arad→Timisoara,329
Arad→Zerind,374
Arad→Sibiu→Arad,366
~~Arad→Sibiu→Fagaras,178~~
Arad→Sibiu→Oradea,380
Arad→Sibiu→Rimnicu Vilcea,193
Arad→Sibiu→Fagaras→Sibiu,253
Arad→Sibiu→Fagaras→Bucharest,0

Greedy Search





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)

Fringe

~~Arad, 366~~

~~Arad→Sibiu,253~~

Arad→Timisoara,329

Arad→Zerind,374

Arad→Sibiu→Arad,366

~~Arad→Sibiu→Fagaras,178~~

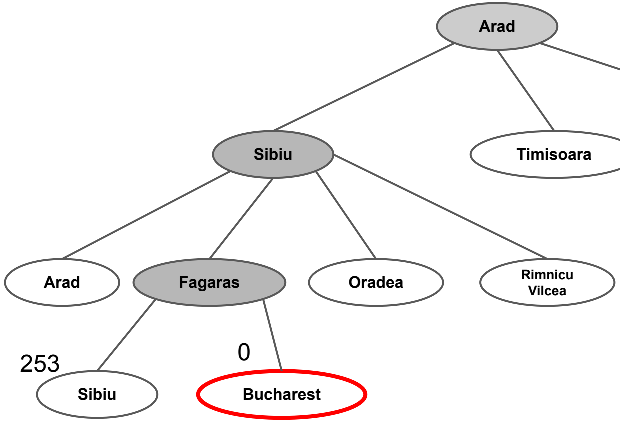
Arad→Sibiu→Oradea,380

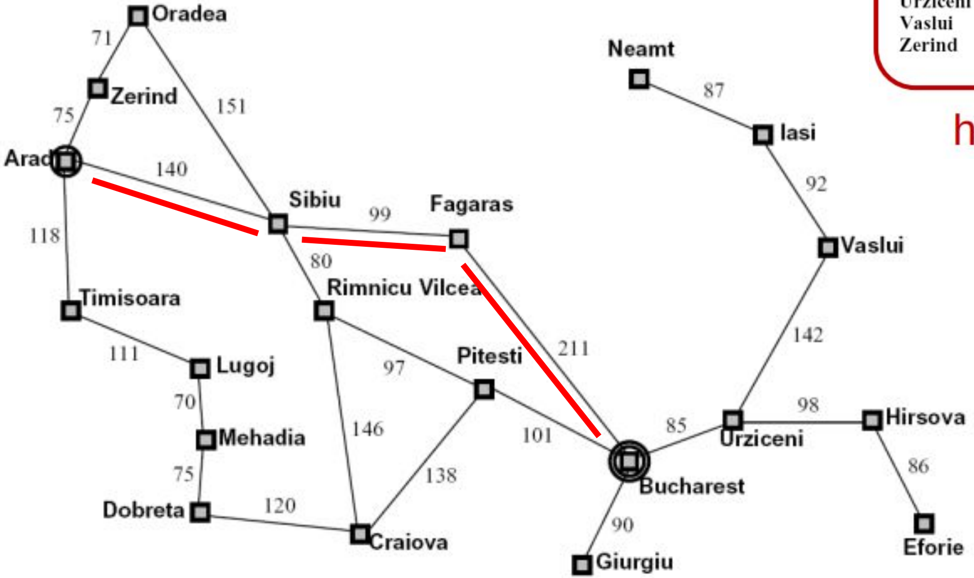
Arad→Sibiu→Rimnicu Vilcea,193

Arad→Sibiu→Fagaras→Sibiu,253

solution Arad→Sibiu→Fagaras→Bucharest,0

Greedy Search

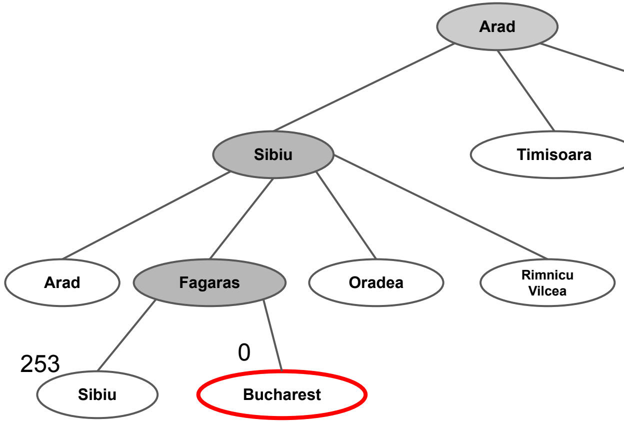


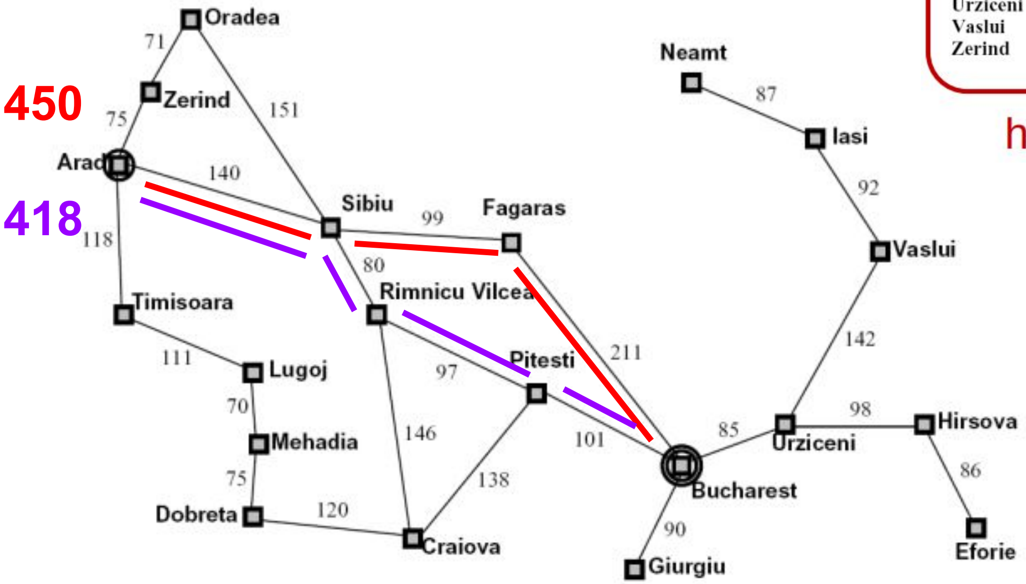


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

Greedy Search





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)

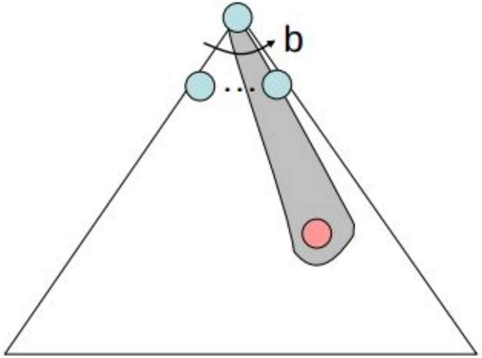
Greedy Search

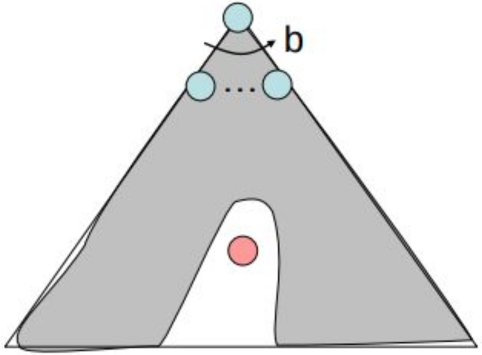
Strategy: expand a node that **you think** is closest to a goal state

Heuristic: **estimate** of **distance** to **nearest goal** for each state

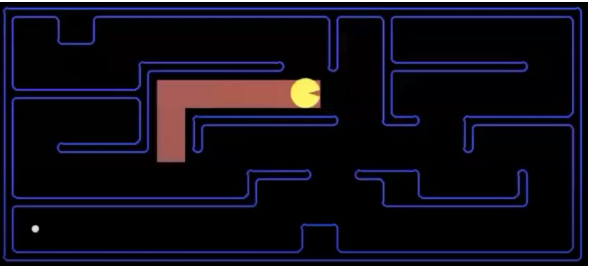
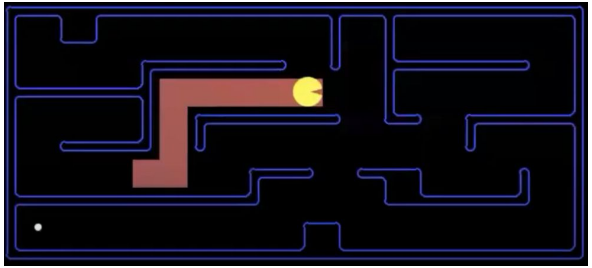
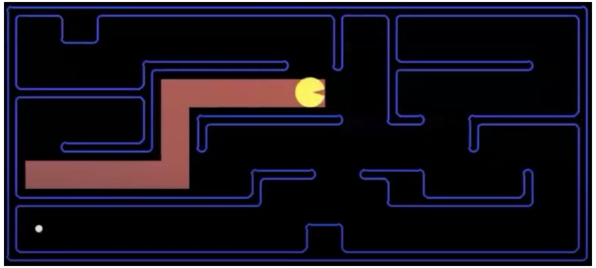
A common case: Best-first takes you straight to the (wrong) goal - wrong in terms of non optimal

Worst-case: like a badly-guided DFS

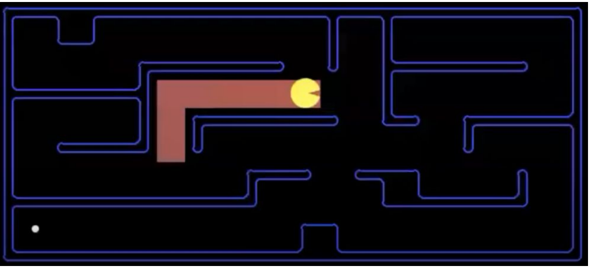


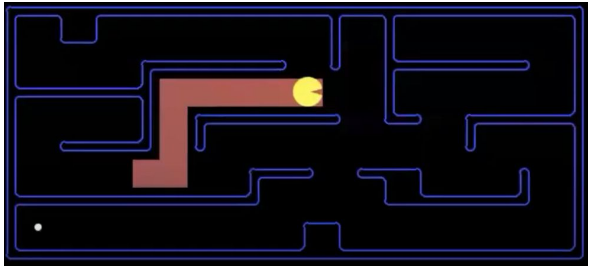


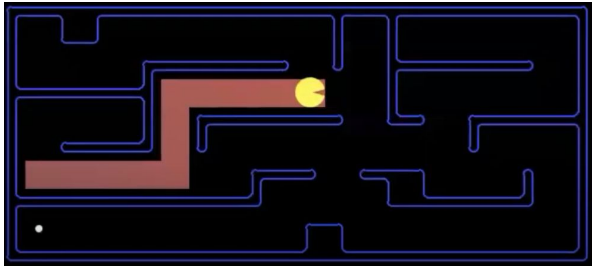
Greedy Search

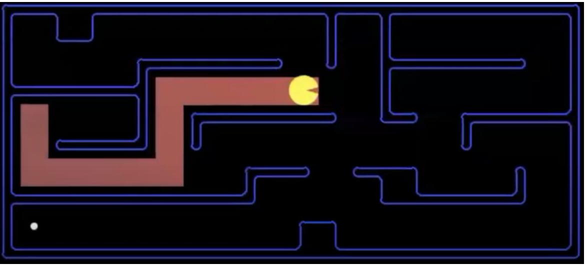
  

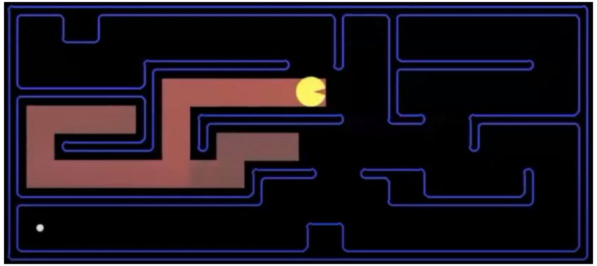
Greedy Search

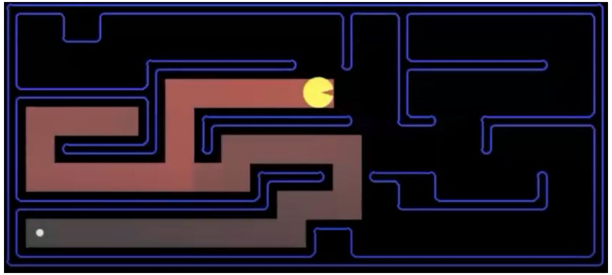




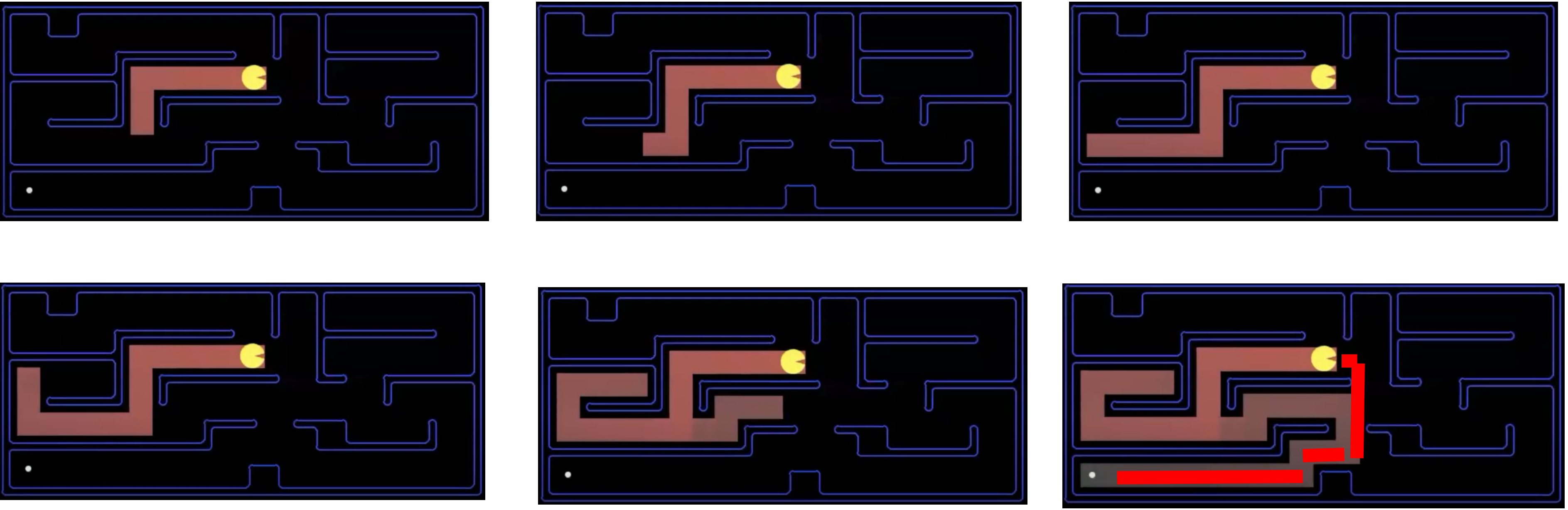








Greedy Search



A* Search

A* Search

UCS

Greedy

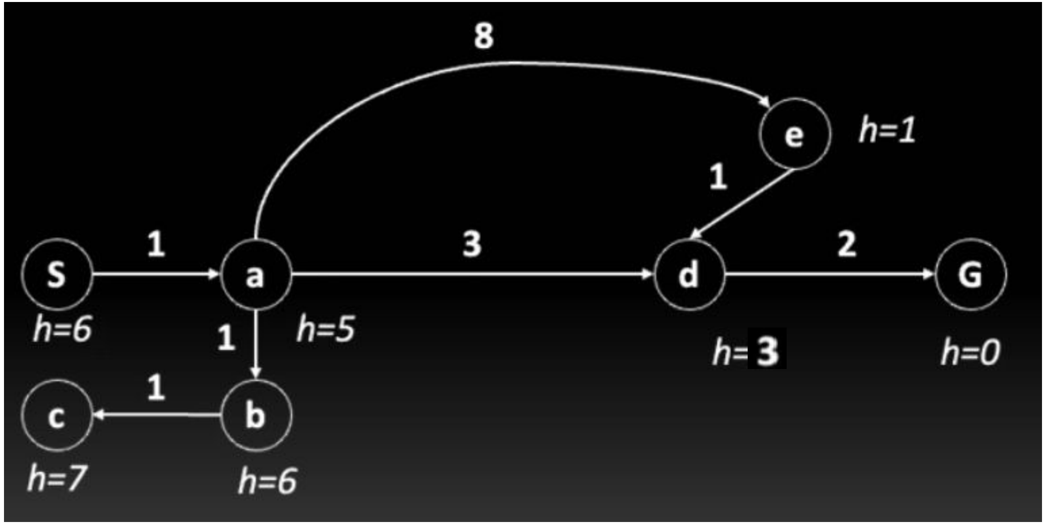
A* Search

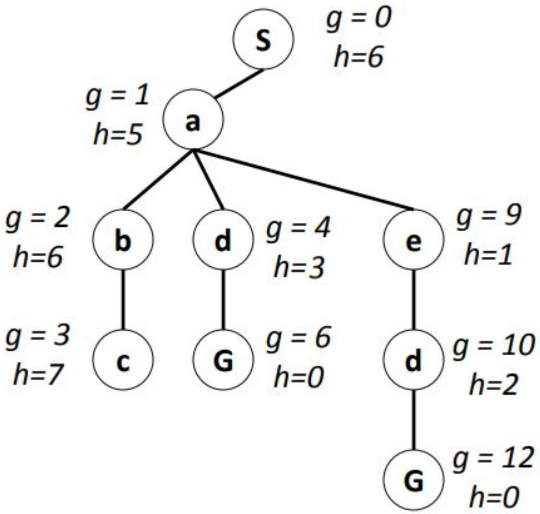
UCS

Greedy

A*

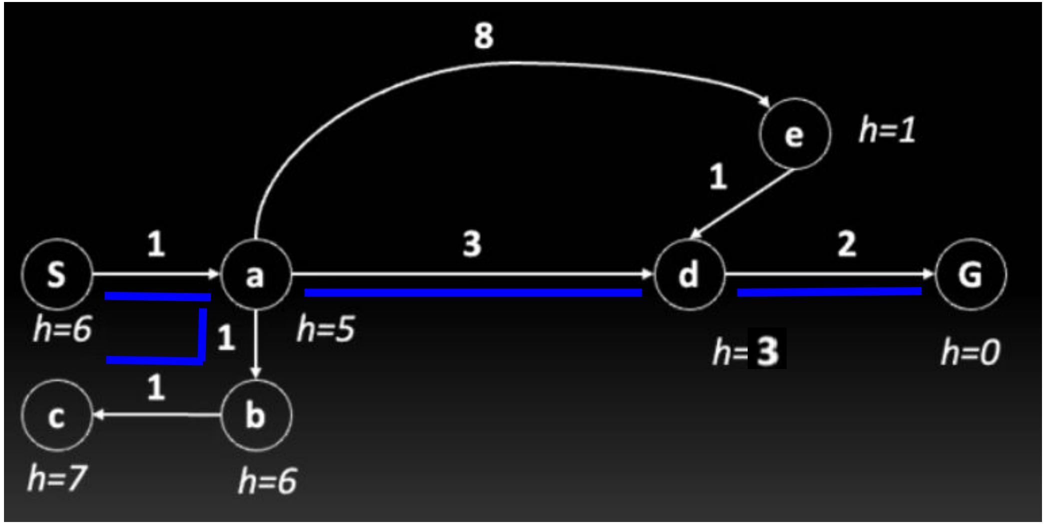
Combining UCS and Greedy

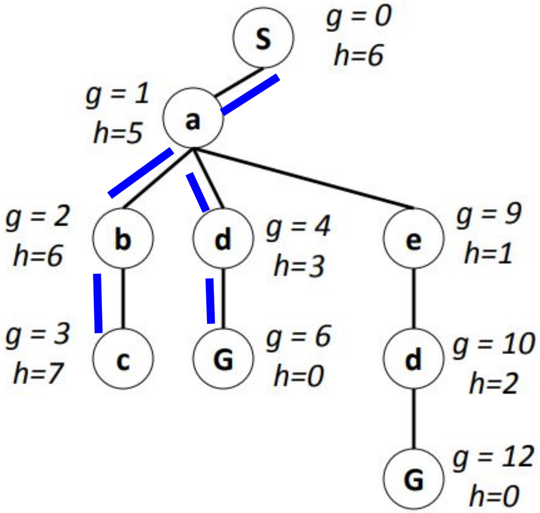




Combining UCS and Greedy

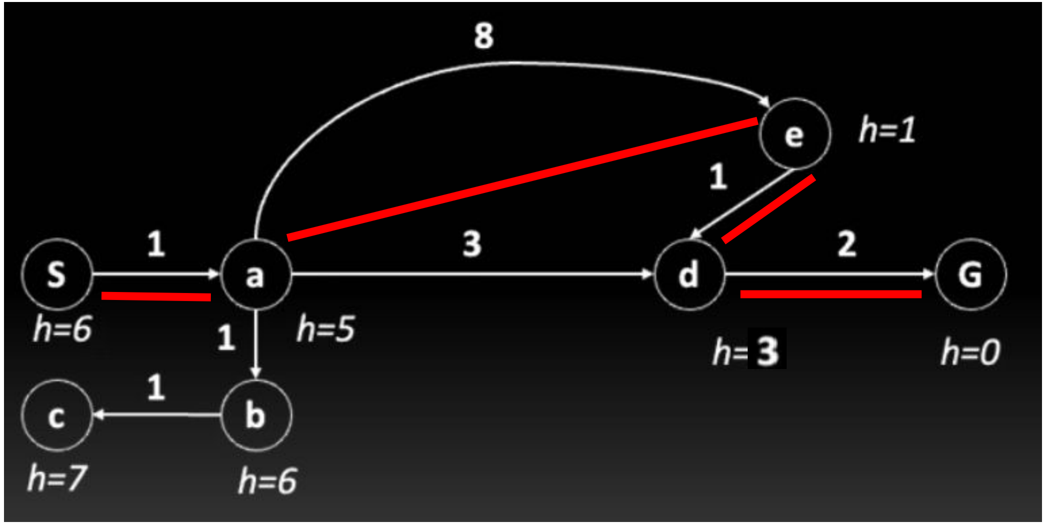
- Uniform-cost orders by path cost, or backward cost g(n)

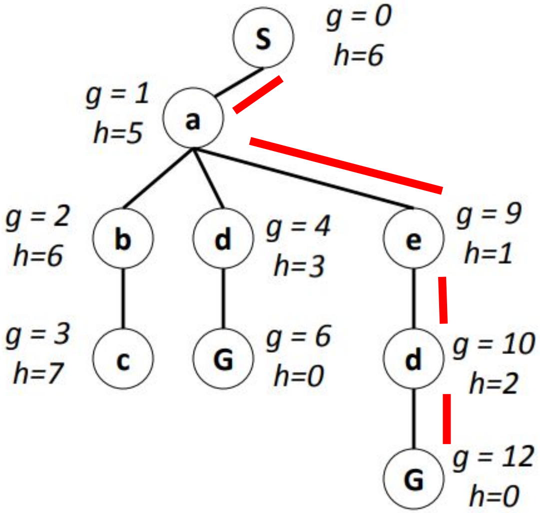




Combining UCS and Greedy

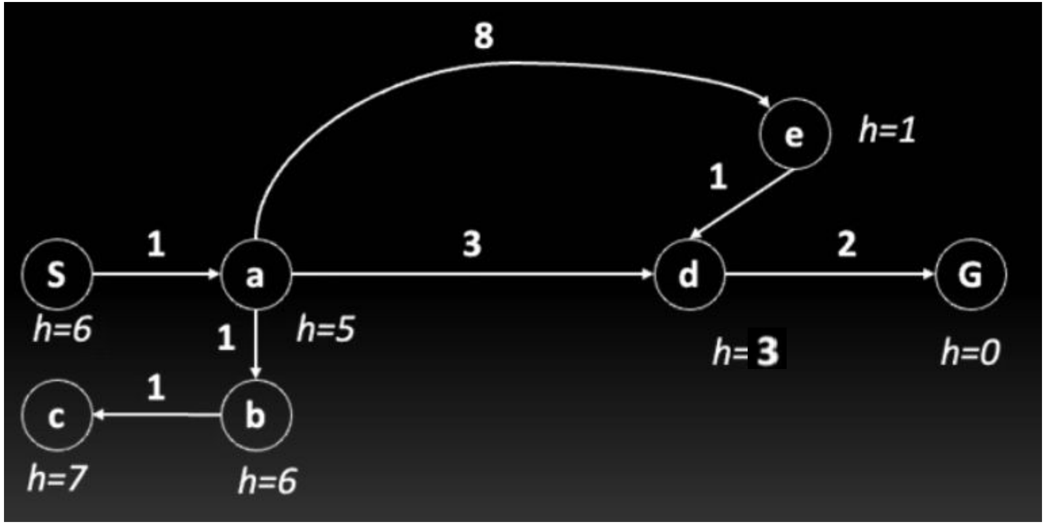
- Uniform-cost orders by path cost, or backward cost g(n)
- Greedy orders by goal proximity, or forward cost h(n)

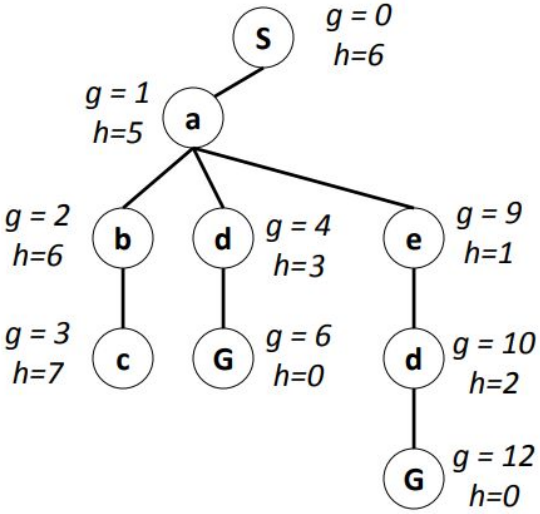




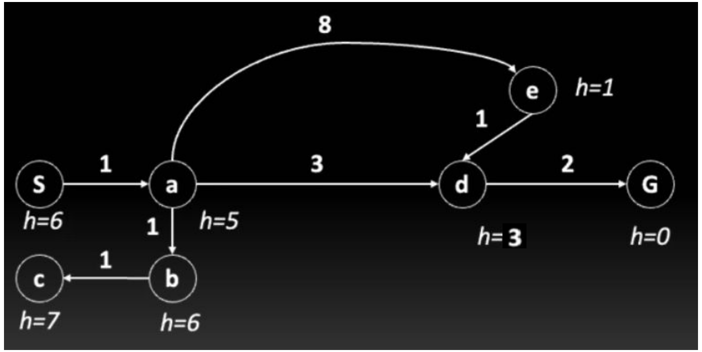
Combining UCS and Greedy

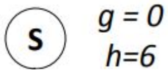
- Uniform-cost orders by path cost, or backward cost g(n)
- Greedy orders by goal proximity, or forward cost h(n)
- A* Search orders by the sum: f(n) = g(n) + h(n)





A* : Combining UCS and Greedy

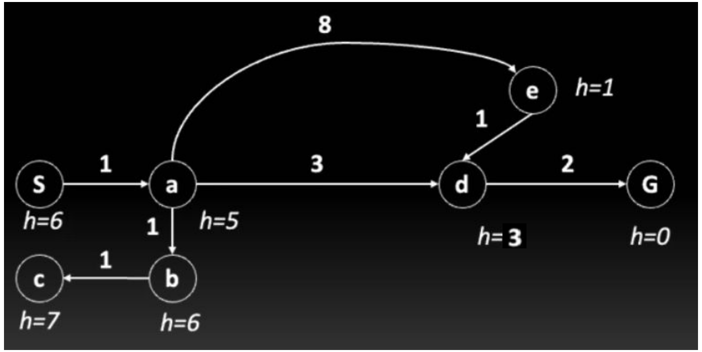


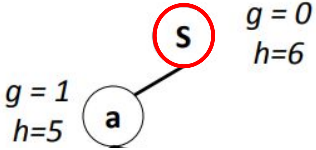


Fringe

S,f=6

A* : Combining UCS and Greedy



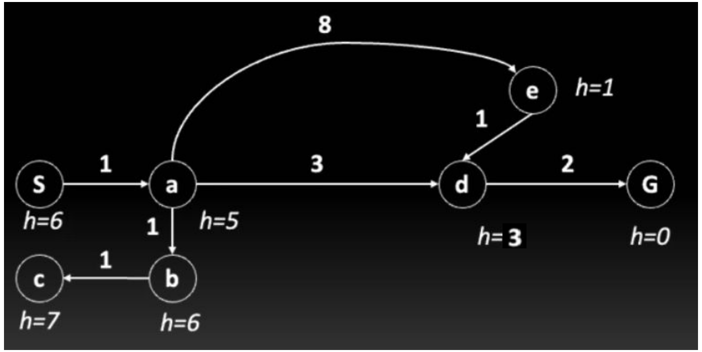


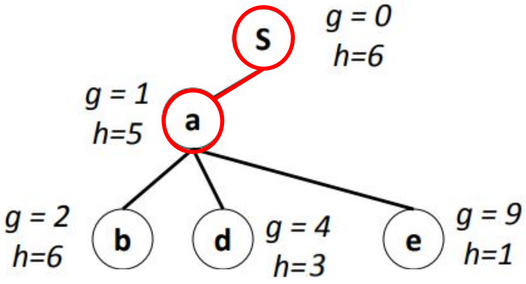
Fringe

~~S,f=6~~

S→a,f=6

A* : Combining UCS and Greedy





<u>Fringe</u>

~~S,f=6~~

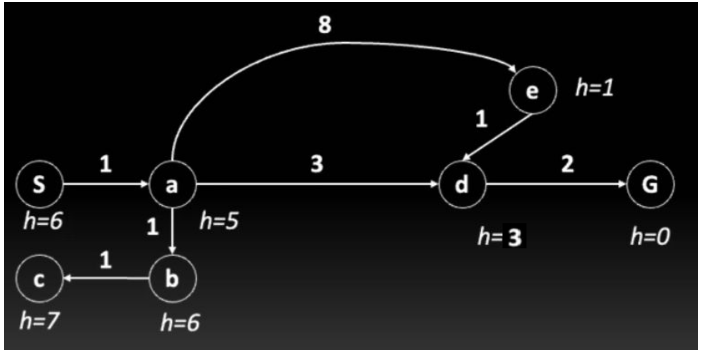
~~S→a,f=6~~

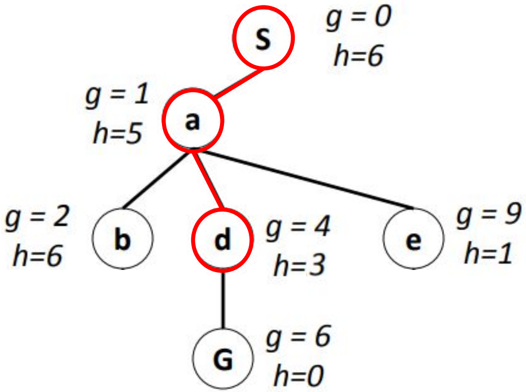
S→a→b,f=8

S→a→d,f=7

S→a→e,f=10

A* : Combining UCS and Greedy





Fringe

~~S,f=6~~

~~S→a,f=6~~

S→a→b,f=8

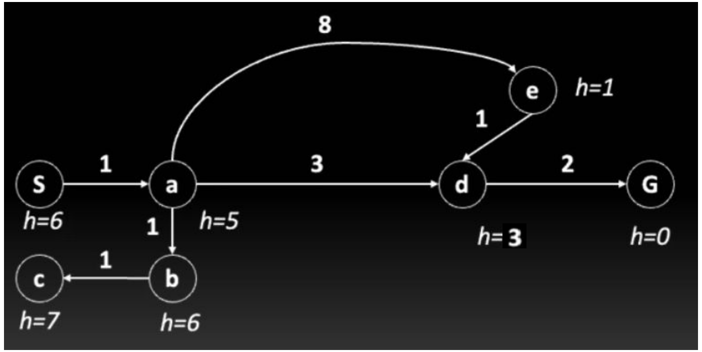
~~S→a→d,f=7~~

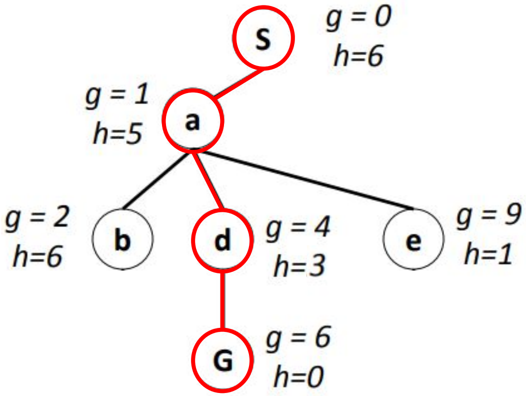
S→a→e,f=10

S→a→b→c,f=10

S→a→d→G,f=6

A* : Combining UCS and Greedy





<u>Fringe</u>

~~S,f=6~~

~~S→a,f=6~~

S→a→b,f=8

~~S→a→d,f=7~~

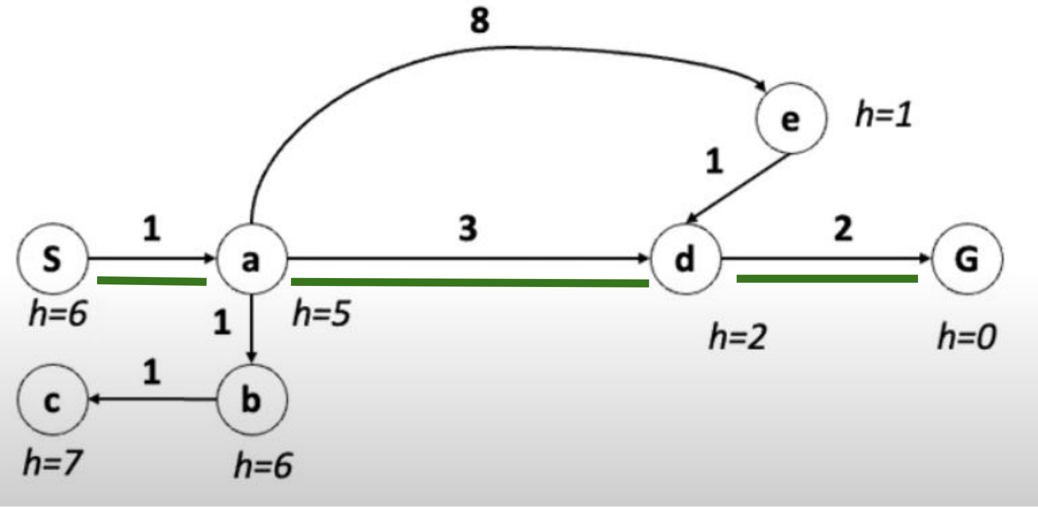
S→a→e,f=10

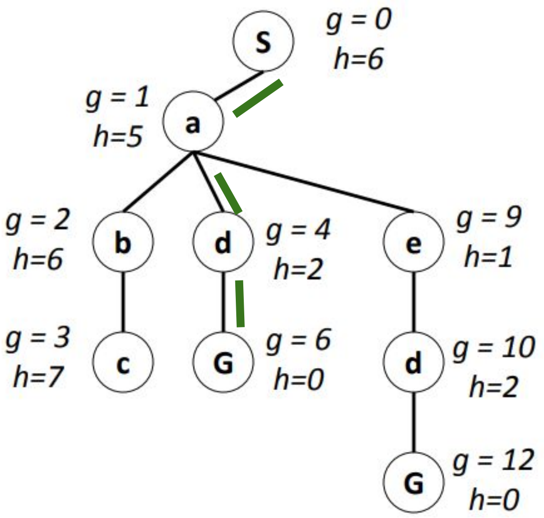
S→a→b→c,f=10

solution S→a→d→G,f=6

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)



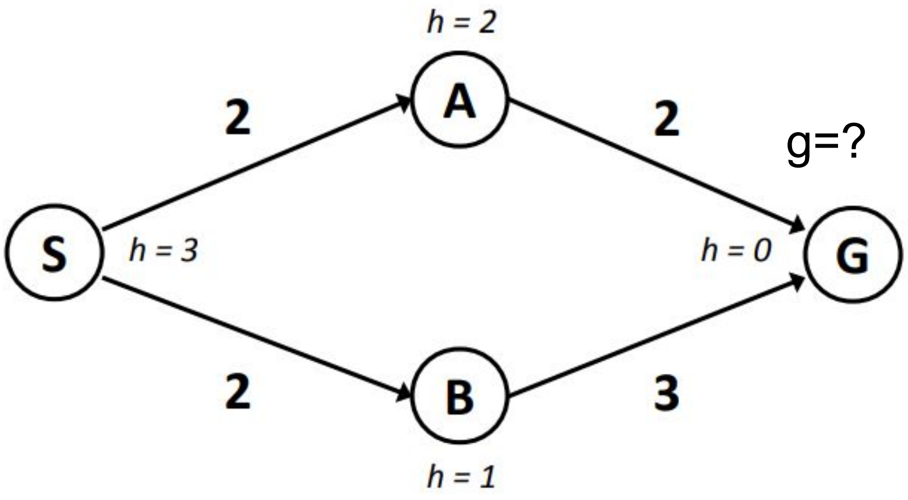


When should A* terminate?

Should we stop when we enqueue/insert a goal state in the fringe?

No, only stop when we dequeue/remove a goal from the fringe

- Insert S [f=3]
- Expand S
  - Insert S->B [f=3]
  - Insert S->A [f=4]
  - Fringe contains S->B[f=3],S->A[f=4]
- Remove S
- Expand S->B
  - Insert S->B->G [f=5]
  - Fringe contains S->A[f=4], S->B->G[f=5]
- Remove S->B
- Expand S->A
  - Insert S->A->G[f=4]
  - Fringe contains S->B->G[f=5], S->A->G[f=4]

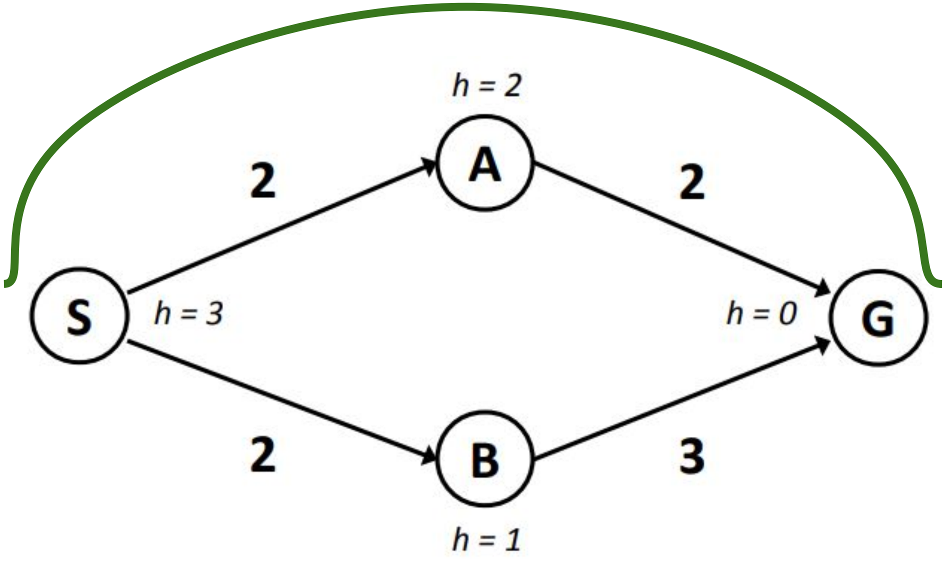


When should A* terminate?

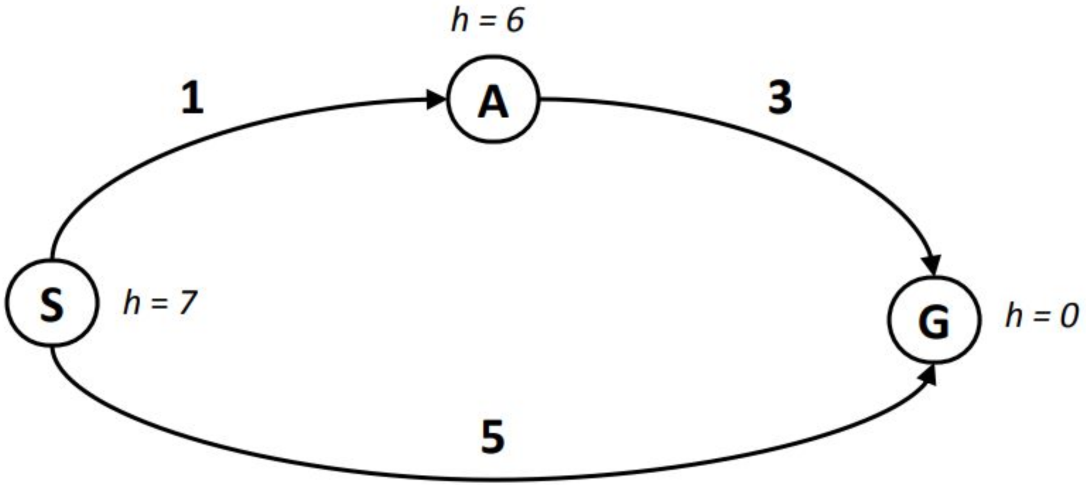
Should we stop when we enqueue/insert a goal state in the fringe?

No, only stop when we dequeue/remove a goal from the fringe

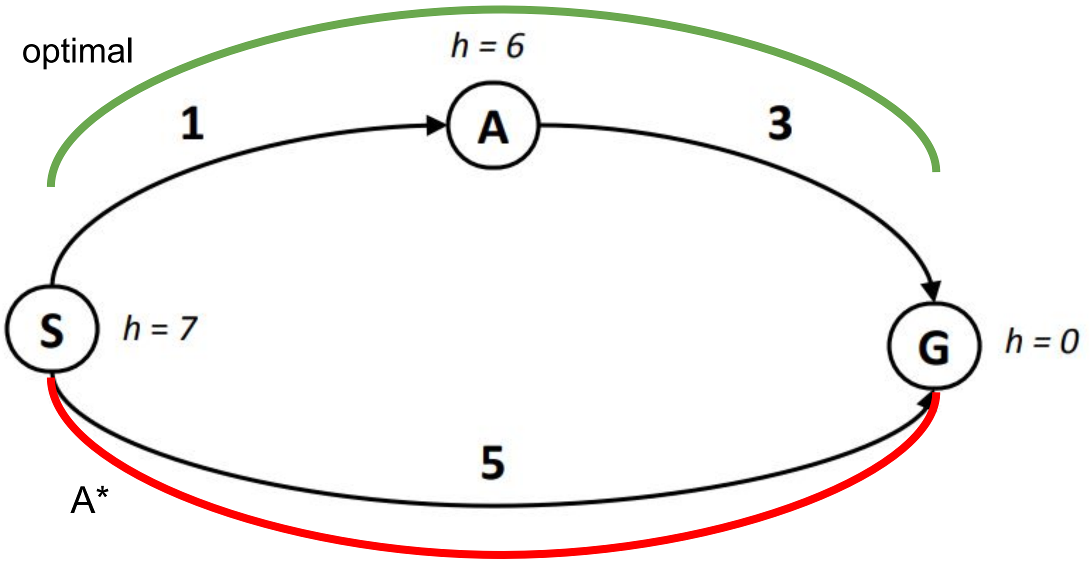
- Insert S [f=3]
- Expand S
    - Insert S->B [f=3]
    - Insert S->A [f=4]
    - Fringe contains S->B[f=3],S->A[f=4]
- Remove S
- Expand S->B
    - Insert S->B->G [f=5]
    - Fringe contains S->A[f=4], S->B->G[f=5]
- Remove S->B
- Expand S->A
    - Insert S->A->G[f=4]
    - Fringe contains S->B->G[f=5], S->A->G[f=4]



Is A* Optimal?



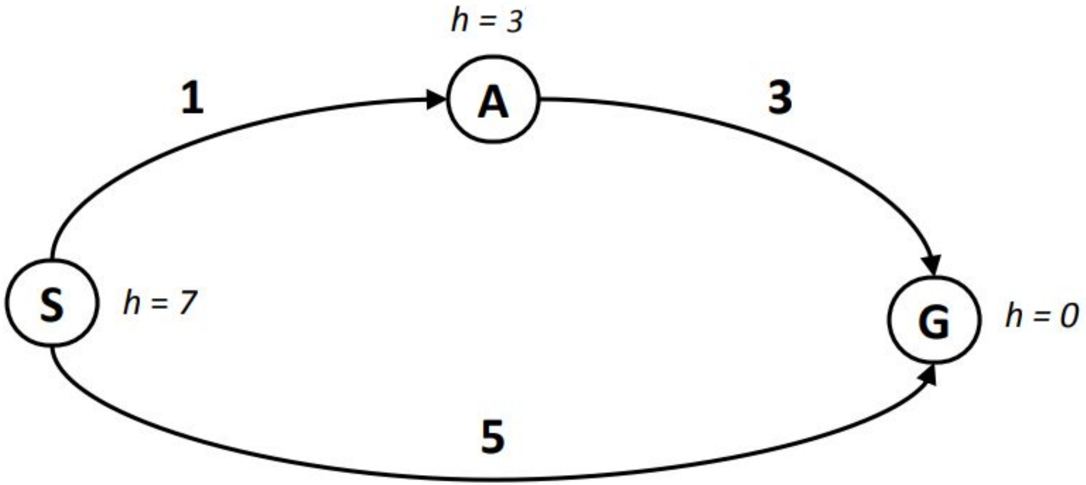
Is A* Optimal?



Actual bad goal cost < estimated good goal cost

We need estimates to be less than actual costs!

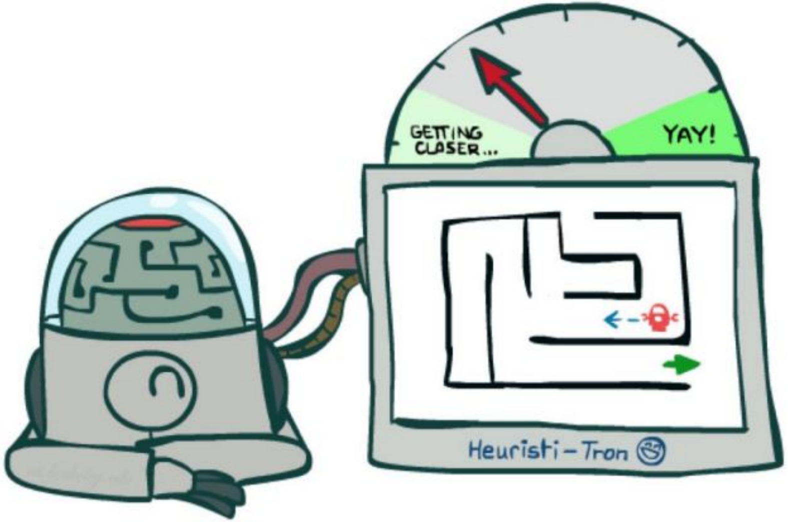
Is A* Optimal?



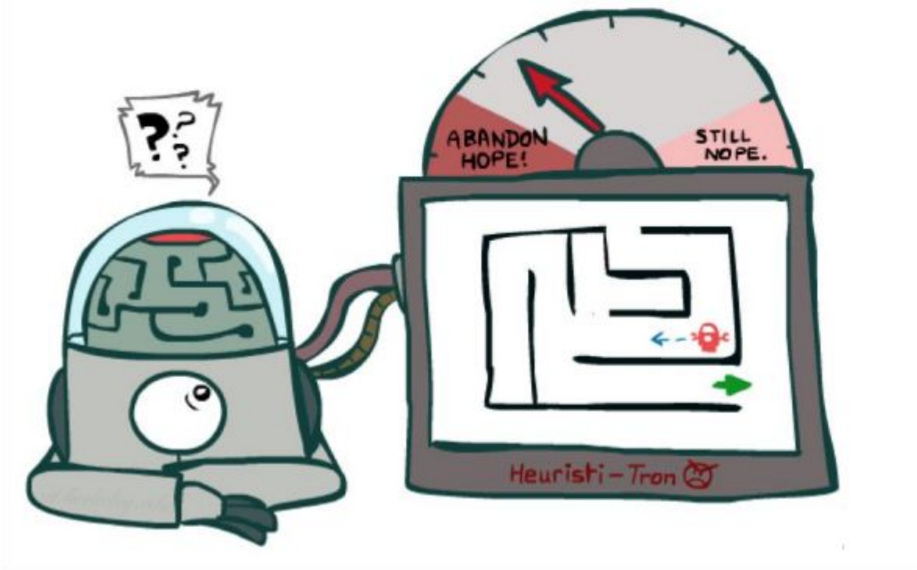
Actual bad goal cost < estimated good goal cost

We need estimates to be less than actual costs!

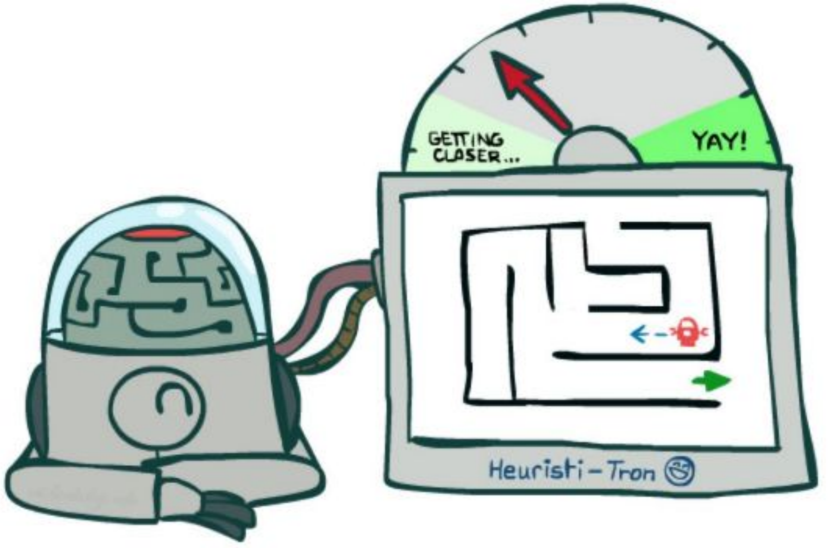
Admissible Heuristics



Idea: Admissibility



Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the fringe



Admissible (optimistic) heuristics slow down bad plans but never outweigh true 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

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

- Needs to decided per problem

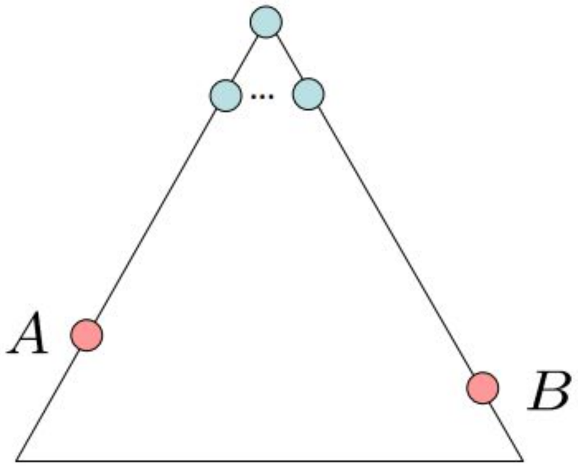
Optimality of A* Tree Search

Assume:

- A is an optimal goal node
- B is a suboptimal goal node
- h is admissible

Claim:

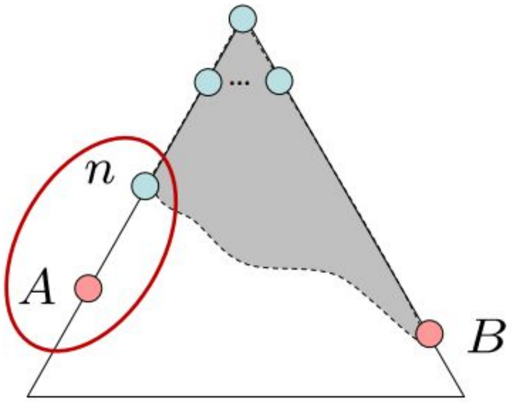
- A will exit the fringe before B



Optimality of A* Tree Search

Proof:

- Imagine B is on the fringe
- Some ancestor *n* of A is on the fringe, too (maybe A!)
- Claim: *n* will be expanded before B
  1. f(n) is less or equal to f(A)



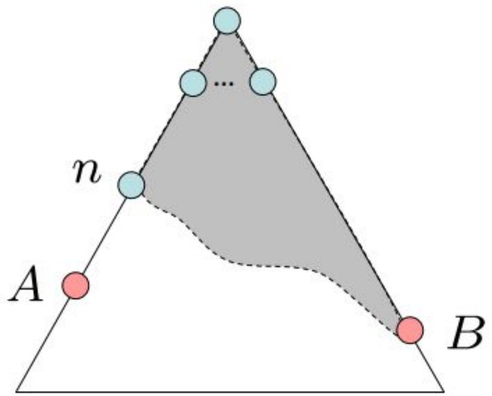
$$f(n) = g(n) + h(n)$$ Definition of f-cost

$$f(n) \leq g(A)$$ Admissibility of h

$$g(A) = f(A)$$ h = 0 at a goal

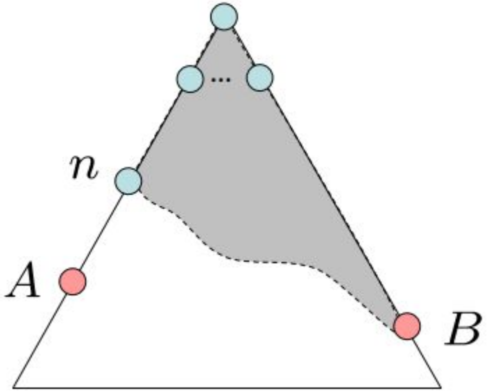
Optimality of A* Tree Search

1. f(n) is less than or equal to f(A)
   - Definition of f-cost says:
     f(n) = g(n) + h(n) = (path cost to n) + (est. cost of n to A)
     f(A) = g(A) + h(A) = (path cost to A) + (est. cost of A to A)



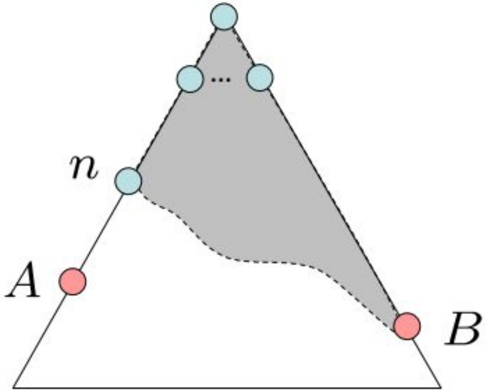
Optimality of A* Tree Search

1. f(n) is less than or equal to f(A)
   - Definition of f-cost says:
     f(n) = g(n) + h(n) = (path cost to n) + (est. cost of n to A)
     f(A) = g(A) + h(A) = (path cost to A) + (est. cost of A to A)
   - The admissible heuristic must underestimate the true cost
     h(A) = (est. cost of A to A) = 0



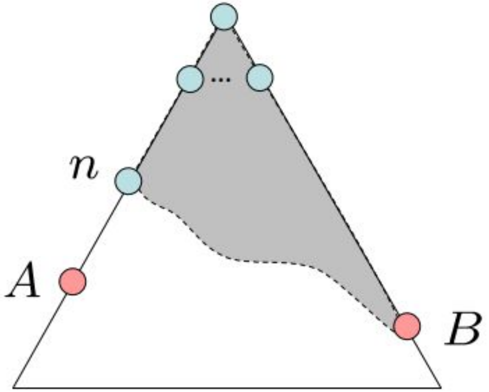
Optimality of A* Tree Search

1. f(n) is less than or equal to f(A)
   - Definition of f-cost says:
     f(n) = g(n) + h(n) = (path cost to n) + (est. cost of n to A)
     f(A) = g(A) + h(A) = (path cost to A) + (est. cost of A to A)
   - The admissible heuristic must underestimate the true cost
     h(A) = (est. cost of A to A) = 0
   - So now, we have to compare:
     f(n) = g(n) + h(n) = (path cost to n) + (est. cost of n to A)
     f(A) = g(A) = (path cost to A)



Optimality of A* Tree Search

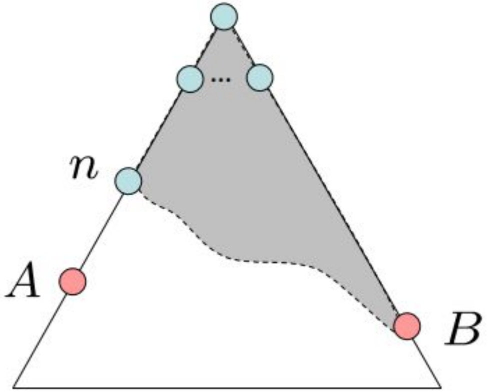
1. f(n) is less than or equal to f(A)
   - Definition of f-cost says:
     f(n) = g(n) + h(n) = (path cost to n) + (est. cost of n to A)
     f(A) = g(A) + h(A) = (path cost to A) + (est. cost of A to A)
   - The admissible heuristic must underestimate the true cost
     h(A) = (est. cost of A to A) = 0
   - So now, we have to compare:
     f(n) = g(n) + h(n) = (path cost to n) + (est. cost of n to A)
     f(A) = g(A) = (path cost to A)
   - h(n) must be an underestimate of the true cost from n to A
     (path cost to n) + (est. cost of n to A) ≤ (path cost to A)



Since, path cost to A = path cost to n + path cost from n to A

Optimality of A* Tree Search

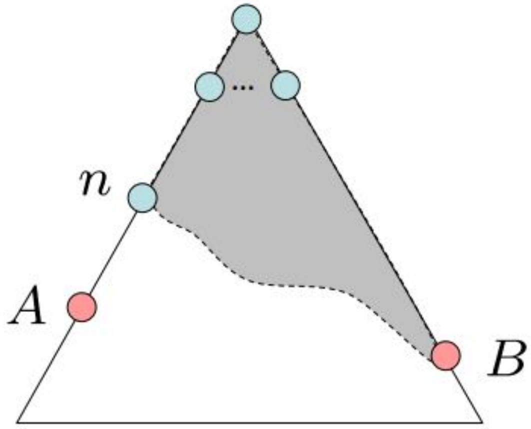
1. f(n) is less than or equal to f(A)
   - Definition of f-cost says:
     f(n) = g(n) + h(n) = (path cost to n) + (est. cost of n to A)
     f(A) = g(A) + h(A) = (path cost to A) + (est. cost of A to A)
   - The admissible heuristic must underestimate the true cost
     h(A) = (est. cost of A to A) = 0
   - So now, we have to compare:
     f(n) = g(n) + h(n) = (path cost to n) + (est. cost of n to A)
     f(A) = g(A) = (path cost to A)
   - h(n) must be an underestimate of the true cost from n to A
     (path cost to n) + (est. cost of n to A) ≤ (path cost to A)
     g(n) + h(n) ≤ g(A)
     f(n) ≤ f(A)



Optimality of A* Tree Search

Proof:

- Imagine B is on the fringe
- Some ancestor *n* of A is on the fringe, too (maybe A!)
- Claim: *n* will be expanded before B
  1. f(n) is less or equal to f(A)
  2. f(A) is less than f(B)



$g(A) < g(B)$	B is suboptimal
$f(A) < f(B)$	h = 0 at a goal

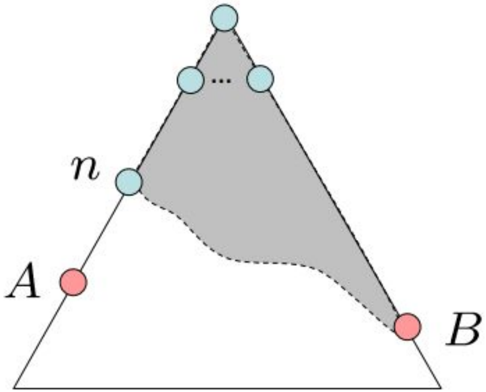
Optimality of A* Tree Search

2. f(A) is less than f(B)

   - We know that:

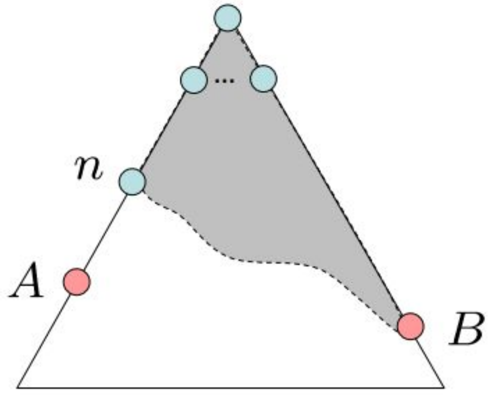
     f(A) = g(A) + h(A) = (path cost to A) + (est. cost of A to A)

     f(B) = g(B) + h(B) = (path cost to B) + (est. cost of B to B)



Optimality of A* Tree Search

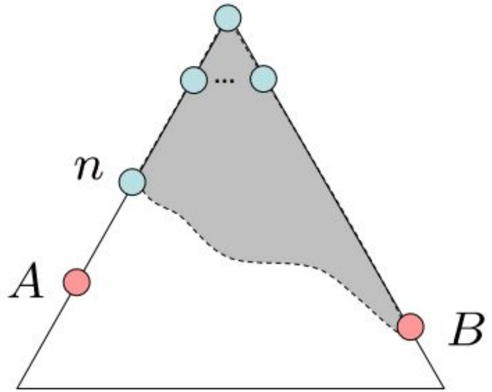
2. f(A) is less than f(B)
   - We know that:
     f(A) = g(A) + h(A) = (path cost to A) + (est. cost of A to A)
     f(B) = g(B) + h(B) = (path cost to B) + (est. cost of B to B)
   - The heuristic must underestimate the true cost:
     h(A) = h(B) = 0
   - So now, we have to compare:
     f(A) = g(A) = (path cost to A)
     f(B) = g(B) = (path cost to B)



Optimality of A* Tree Search

2. f(A) is less than f(B)

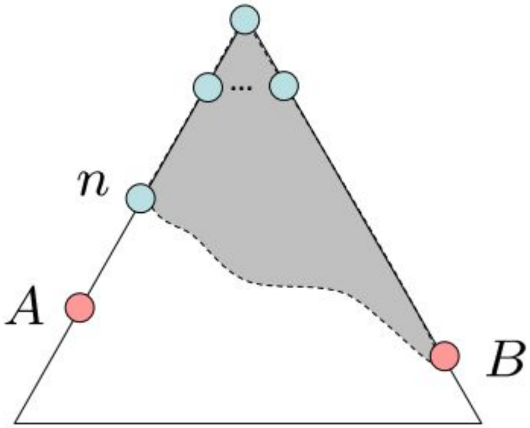
- We know that:
  f(A) = g(A) + h(A) = (path cost to A) + (est. cost of A to A)
  f(B) = g(B) + h(B) = (path cost to B) + (est. cost of B to B)
- The heuristic must underestimate the true cost:
  h(A) = h(B) = 0
- So now, we have to compare:
  f(A) = g(A) = (path cost to A)
  f(B) = g(B) = (path cost to B)
- We assumed that B is suboptimal! So
  (path cost to A) < (path cost to B)
  g(A) < g(B)
  f(A) < f(B)



Optimality of A* Tree Search

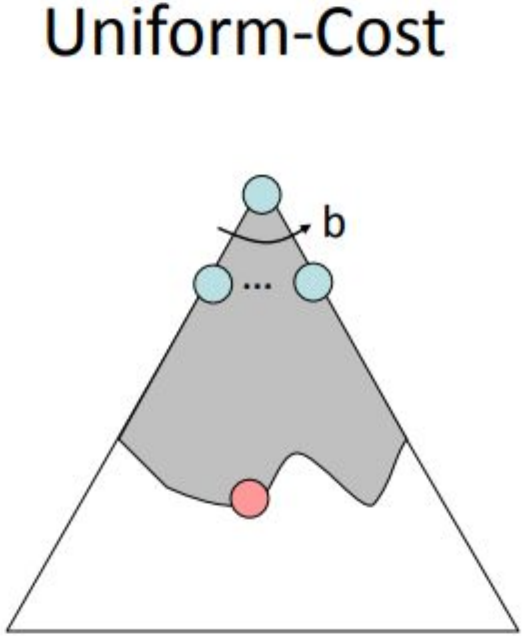
Proof:

- Imagine B is on the fringe
- Some ancestor *n* of A is on the fringe, too (maybe A!)
- Claim: *n* will be expanded before B
  1. f(n) is less or equal to f(A)
  2. f(A) is less than f(B)
  3. *n* expands before B
- All ancestors of A expand before B
- A expands before B
- A* search is optimal

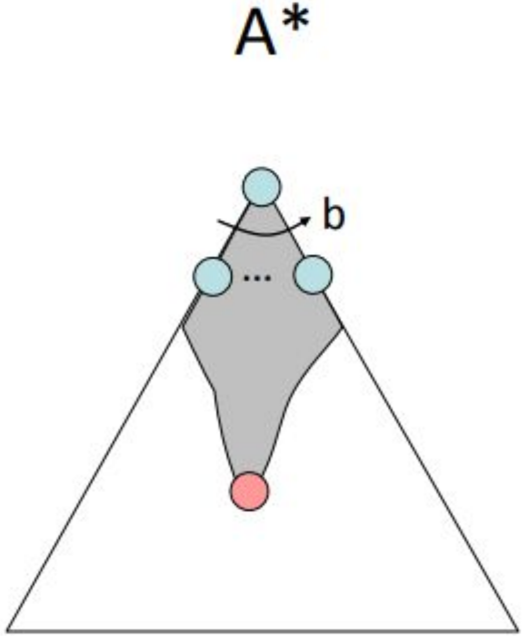


$$f(n) \leq f(A) < f(B)$$

Properties of A*



Contour-wise



Still Contour-wise but contour defined by f

Practice

Is the heuristics function admissible?
Find the solution using A*