Informed(Heuristic) Search Strategies

Heuristic

- Estimate based on prior knowledge/belief
  - Include rule of thumb, trial and error or educated-guess
  - May find practical solutions more efficiently
- Terminologies
  - g(n): cost from initial state to the state at node n
  - h(n):estimated cost from state at node n to the closest goal state
    - Depends only in the state at node n
    - h(n)=0, if n is a goal node
  - f(n):evaluation function; cost estimate from the initial state to the closest goal state through the state at node n

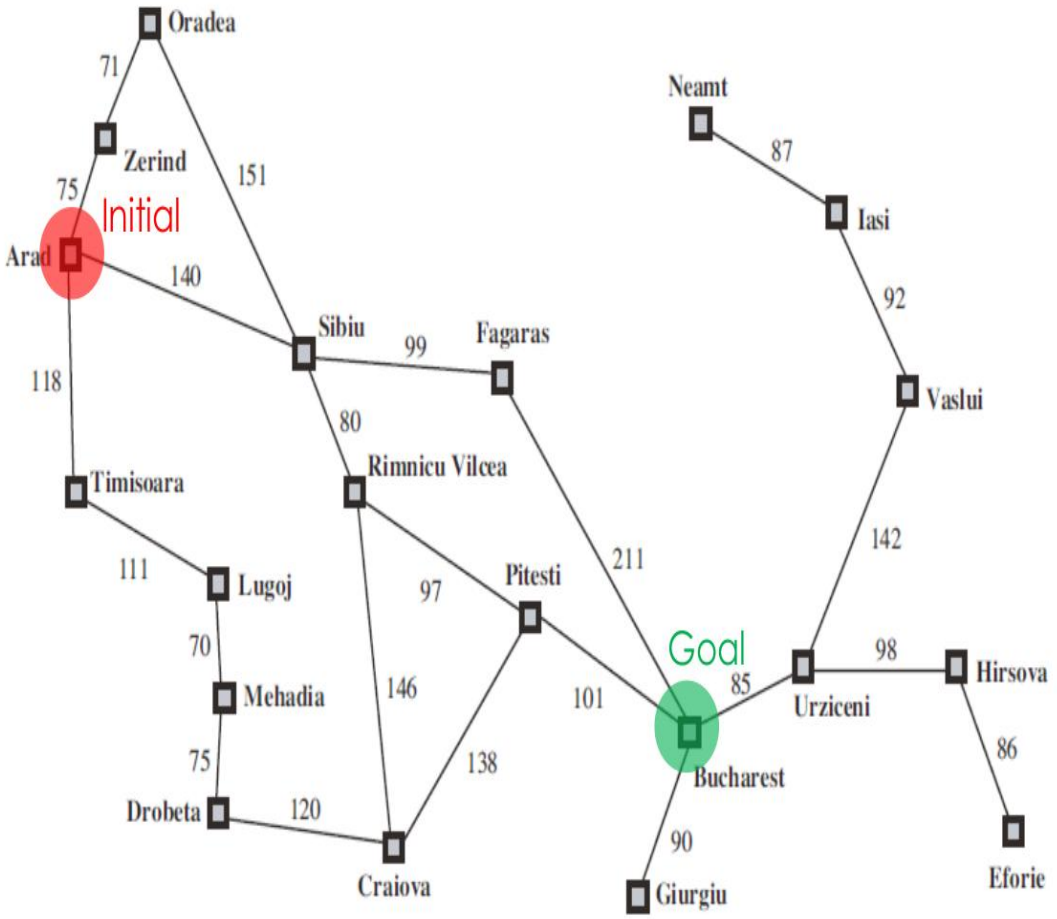
Best-First Search

- Use an evaluation function f(n) for node n
- Always choose the node from fringe that has the lowest f value
- Most of the best-first search algorithm include **a heuristic function** h(n) as a component of the evaluation function f(n)
- h(n) is a problem specific function with the constraint that h(goal-node) = 0

Greedy Best-First Search

- Expand those nodes which deem to be closest to the goal.
- Evaluation function f(n)= h(n)
- Implementation:
  - priority queue based on h(n)

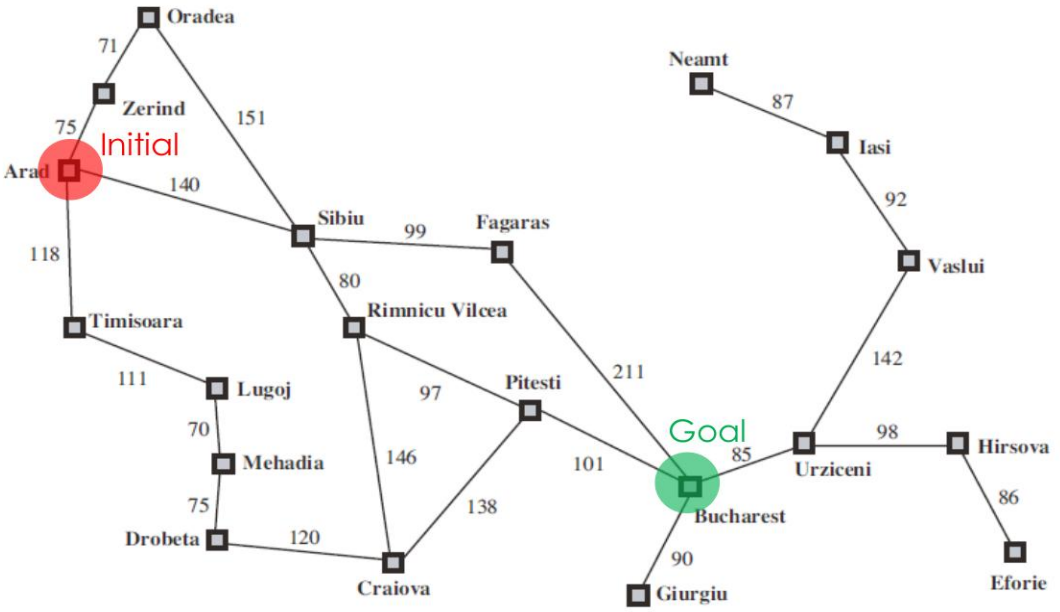
Greedy Best-First Search Example



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)

Greedy Best-First Search Example

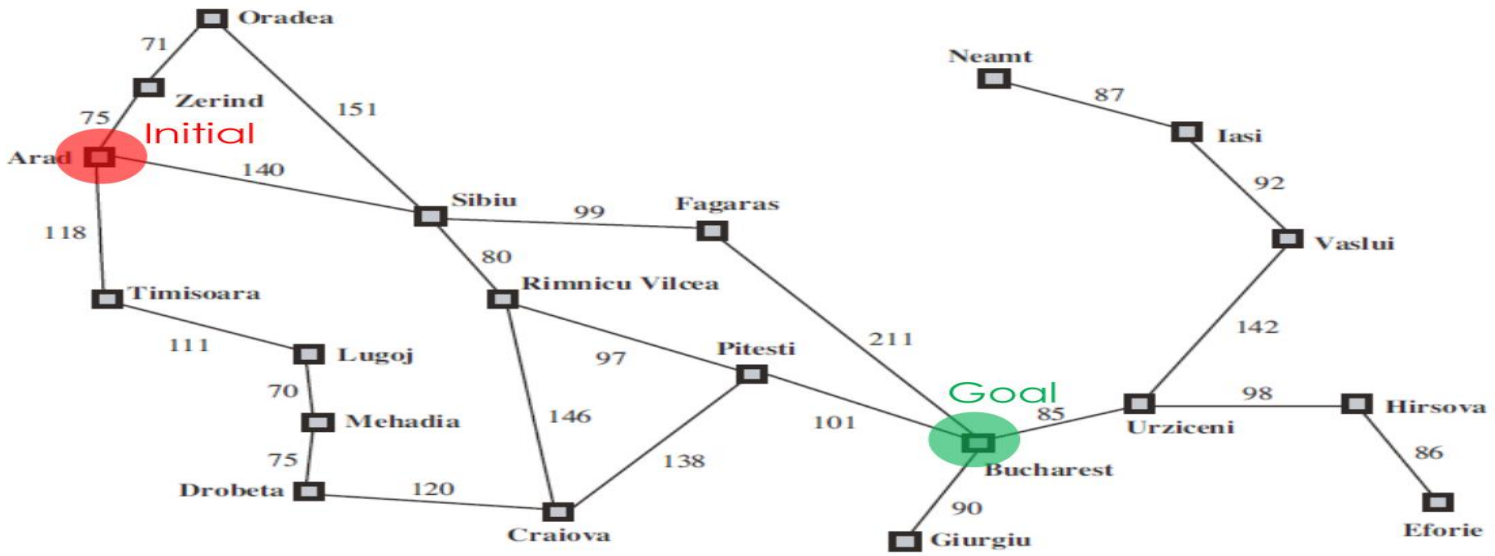


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

- Sequence of the expansion of nodes:
  - Arad->Sibiu-> Fagaras-> Bucharest (Path Cost: 450)
  - Is it a least cost path (optimal solution)?
  - Arad->Sibiu-> Rimnicu Vilcea-> Pitesti ->Bucharest (Path Cost 418)

Greedy Best-First Search Analysis

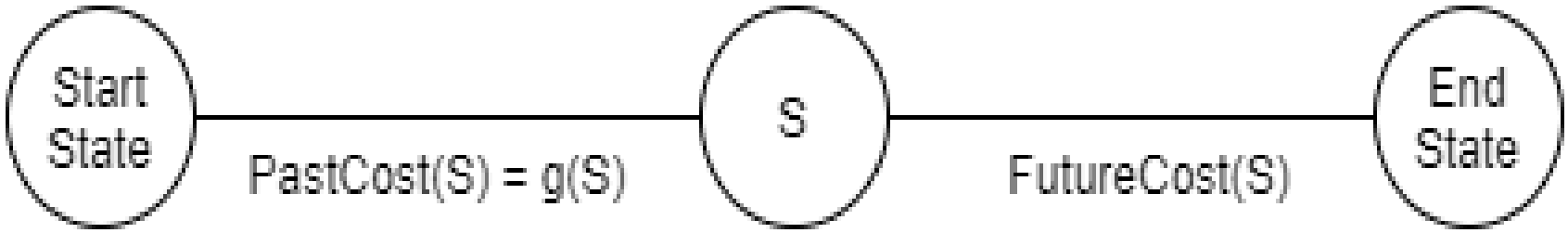


- Completeness: No, can get stuck in loop so can be made complete with repeated state checking
- Optimality: No
- Time Complexity: $O(b^m)$
- Space Complexity: $O(b^m)$

A* Search

- Can Uniform Cost Search (UCS) be improved?
  - Bias UCS such that it favours those states which are closer to goal state
  - Ideal situation:

    explore in order of PastCost(S) + FutureCost(S)



A* Search

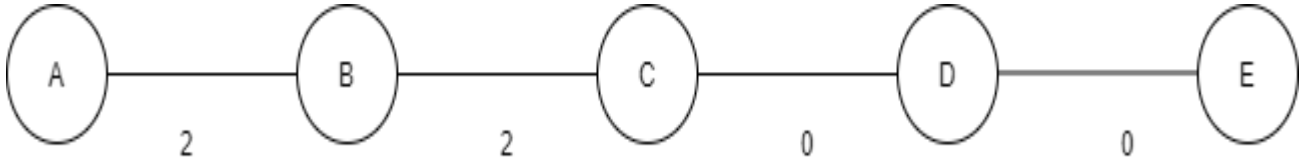
- A*: explore in order of PastCost(s)+h(s) where heuristic h(s) is any estimate of FutureCost(S)
- Runs Uniform Cost Search(UCS) with modified edge costs:

$$cost'(s,a) = cost(s,a) + \underbrace{h(succ(s,a)) - h(s)}_{\text{Penalty}}$$

- Interpretation of penalty:

A* Search

- Penalty measures how much farther from the end state does action a take us
  - If difference is positive => penalize the action a
  - If this difference is negative => favour the action a
- Example:

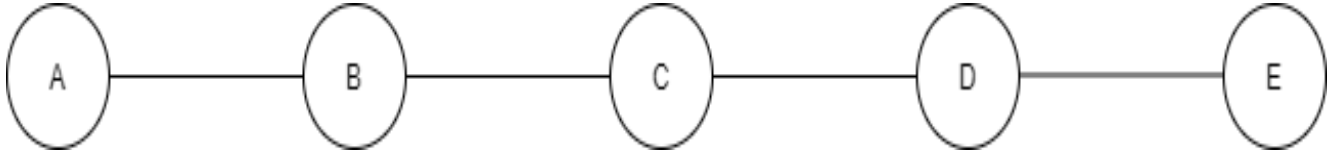


h(s) = 4 3 2 1 0

$$Cost'(B,A)=Cost(B,A)+h(A)-h(B)= 1 + (4-3) = 2$$

A* Search

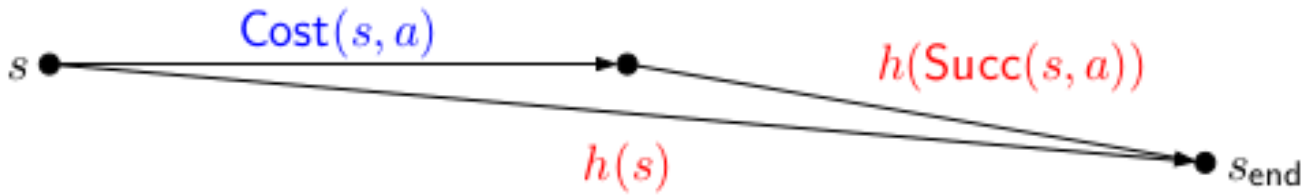
- How many nodes will be explored by UCS?
- How many nodes will be explored by A* Search?
- Does any heuristic work?



h(s) = 8 6 4 2 0

Consistent Heuristics

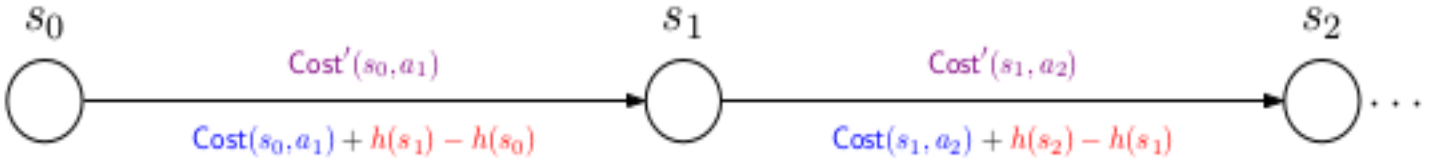
- A heuristic h is consistent if
  - cost'(s,a) = cost(s,a) + h(succ(s,a)) – h(s) ≥ 0
  - $h(s_{goal}) = 0$
- Condition 1: Triangle inequality



- Condition 2: FutureCost(s_{goal}) = 0

Correctness of A*

- If h is consistent, A* returns the minimum cost path. How?
  - Consider any path⌈s_0, a_1, s_1, . . . , a_L, s_L⌉



  - Key identity: $\sum_{i=1}^{L} cost'(s_{i-1}, a_i) = \sum_{i=1}^{L} cost(s_{i-1}, a_i) + h(s_L) - h(s_0)$

modified path cost — original path cost — constant

Correctness of A*

- $h(s_L)=0$ as s_L is the goal state
- $h(s_0)$ is constant, as it doesn't depend on the path at all
- modified path cost = original path cost + constant
- So A*, which is running UCS on the modified edge costs, is equivalent to running UCS on the original edge costs, which minimizes the original path cost
- This is interesting: despite all the edge costs have been modified in A*, but yet the final path cost is the same (up to a constant)
- Therefore A* solves the problem of finding the minimum cost path using modified costs

How A* is More Efficient than UCS?

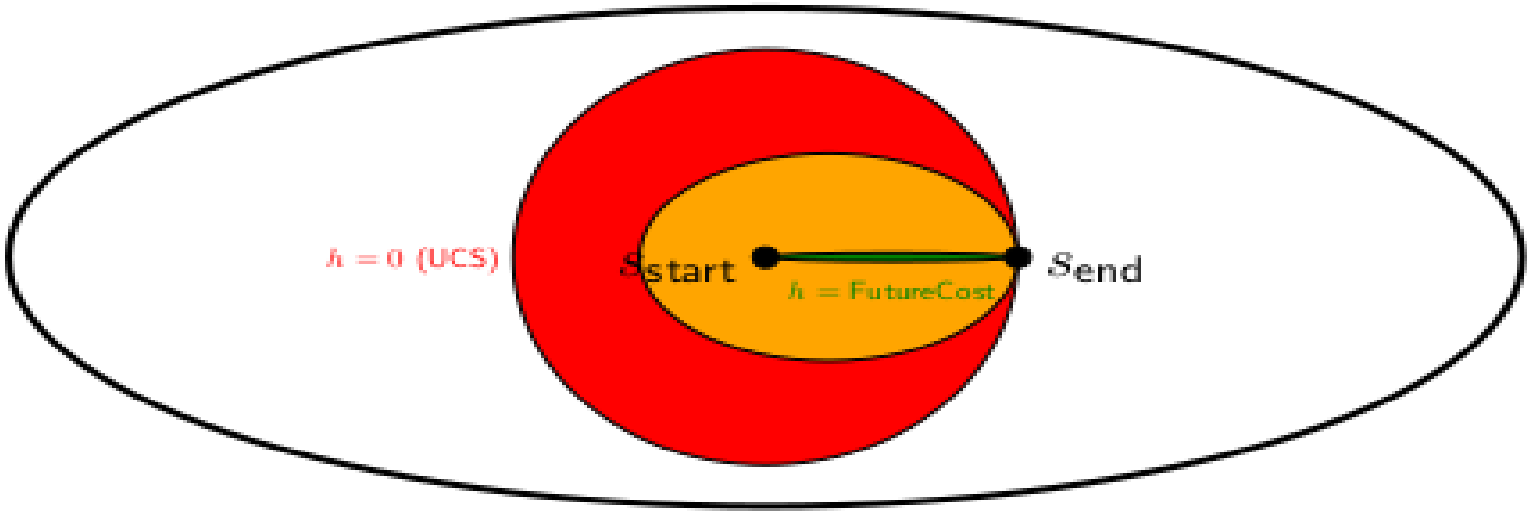
- A* explore all states s satisfying:
  - $\text{PastCost}(s) + h(s) \leq \text{PastCost}(s_{goal})$
  - Efficiency is measured in terms of the number of states explored by the search algorithms
  - UCS will explore every state whose PastCost is less than the PastCost of the goal state in all possible directions
  - A* explores all states for which :

    $\text{PastCost}'(s) = \text{PastCost}(s)+h(s)-h(s_{start})$ is less than $\text{PastCost}'(s_{goal}) = \text{PastCost}(s_{goal})+h(s_{goal})-h(s_{start})$
  - Or equivalently $\text{PastCost}(s)+h(s) \leq \text{PastCost}(s_{goal})$ as $h(s_{goal}) = 0$

How A* is More Efficient than UCS?

- From here, it's clear that h(s) should be as large as possible so that one can push as many states over the PastCost(s_{goal}) threshold
- So one need not have to explore them
- simultaneously one has to maintain the condition that h must be consistent to uphold correctness criteria
- For example, suppose PastCost(s_1) = 1 and h(s_1) = 1 and PastCost(s_{goal}) = 2. Then we would have to explore s_1 ($1 + 1 \leq 2$)
- But if we were able to come up with a better heuristic where h(s_1) = 2, then we wouldn't have to explore s_1($1 + 2>2$)

How A* is More Efficient than UCS?



- each ellipse corresponds to the set of states which are explored by A* with various heuristics
- In general, any heuristic will be between the trivial heuristic h(s) = 0 which corresponds to UCS and the oracle heuristic h(s) = FutureCost(s) which is unattainable

Admissibility of Heuristics

- A heuristic h(s) is admissible if h(s) ≤ FutureCost(s)
  - Admissible heuristics are optimistic as it underestimates the FutureCost
- Show that any consistent heuristic h(s) satisfies h(s) ≤ FutureCost(s) (Proof by induction)
  - Base case, $0 = h(s_{goal}) \leq \text{FutureCost}(s_{goal}) = 0$ by the second condition of consistency
  - Inductive case, let s be a state and a be an optimal action leading to s'=Succ(s, a) that achieves the minimum cost path to the end state
  - i.e. FutureCost(s) = Cost(s, a) + FutureCost(s')
  - As Cost(s, a) ≥ 0, we have that FutureCost(s') ≤ FutureCost(s)

Admissibility of Heuristics

- By the inductive hypothesis, $h(s') \leq \text{FutureCost}(s')$
- Also $h(s) \leq \text{Cost}(s, a) + h(s')$ (Triangle Inequality as h is consistent)
- From the above two inequalities we have:
  $h(s) \leq \text{Cost}(s, a) + h(s') \leq \text{Cost}(s, a) + \text{FutureCost}(s')$
- But $\text{Cost}(s, a) + \text{FutureCost}(s') = \text{FutureCost}(s)$ (as a is an optimal action)
- From above we prove that $h(s) \leq \text{FutureCost}(s)$

References

- Russell & Norvig Text Book
- UCB couse CS188
- Professor Mausam IIT Delhi AI course:
  - http://www.cse.iitd.ac.in/~mausam/

Next Class

- A* analysis, its variants e.g. IDA*
- Examples
- Next topic: Constraint Satisfaction Problems?

Assignments/Quizzes/Projects?

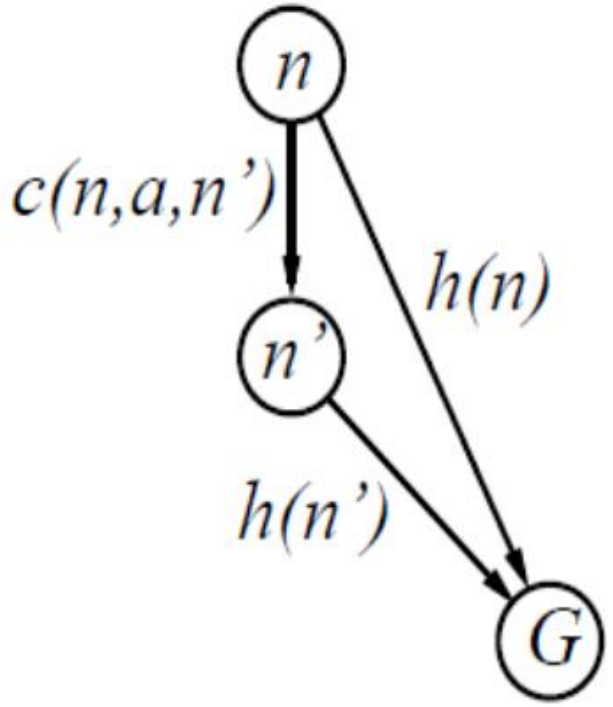
- Take away assignment (weightage: 10%)
  - This/next week
- Project: (weightage: 10%)
  - Some (short) paper implementation
  - 3-4 students in a group
  - Report/Demo submission

A* search Optimality

- Proof by contradiction
  - Let N be the goal node A* outputs and suppose there is another goal node N′. Then g(N)≤ g(N′)
  - Assume to the contrary g(N′) < g(N)
  - When we picked N for expansion, either N′ or an ancestor of N′ which is N′′ must have been on the queue.
  - Since we picked N for expansion f(N) ≤ f(N′′) implies g(N) + h(N) ≤ g(N′′) + h(N′′)
  - As N is a goal node hence h(N) = 0, g(N) ≤ g(N′′) + h(N′′) (1)
  - g(N′)= g(N′′) + dist(N′,N′′) , and h(N′′) ≤ FutureCost(N′′)= dist(N′,N′′) (as N′ is a goal node) (2)
  - So from (2) g(N′′) + h(N′′) ≤ g(N′)
  - From (1) and (2) g(N)≤ g(N′) - contradiction

Monotonic Heuristic

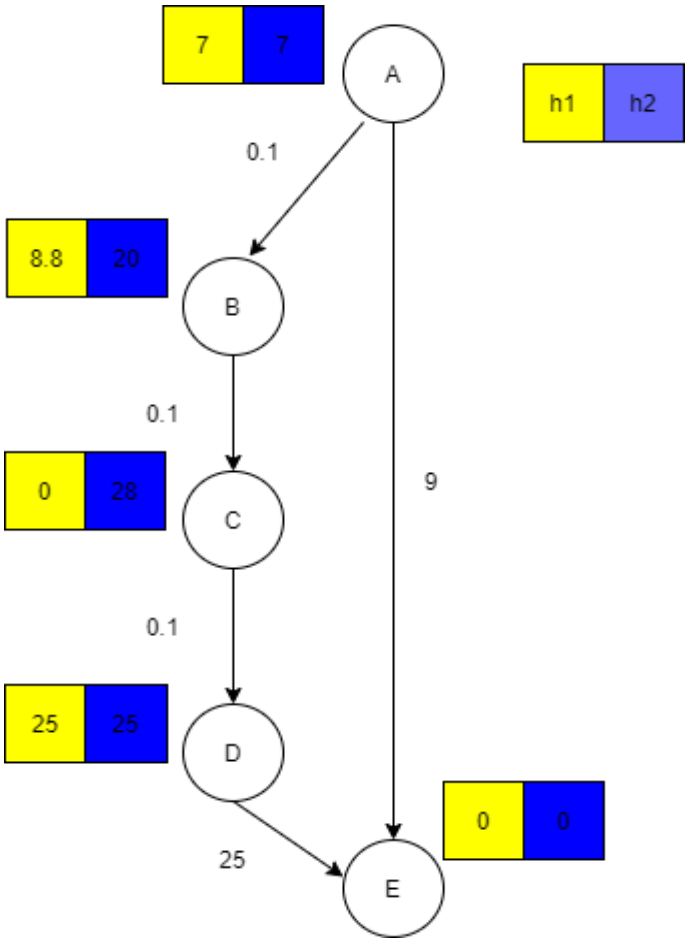
- A heuristic is monotonic if:
  - $h(n) \leq h(n') + c(n,a,n')$
- Consistent Heuristics are monotonic, why?
  - because the estimated final cost of a partial solution, $f(Nj) = g (Nj) + h (Nj)$ is monotonically non-decreasing along the best path to the goal
- Consider the evaluation function f(n')
  - $f(n') = g(n') + h(n') = g(n) + c(n,a,n') + h(n')$
  - $g(n) + c(n,a,n') + h(n') \geq g(n) + h(n) = f(n)$
  - f(n) is monotonic along any path



Triangle Inequality

Monotonicity, Pathmax Correction

- Is yellow $h1$ admissible?
- Is blue $h2$ admissible?
- Does f(C) make sense?
  - f(B)= 0.1 + 8.8 = 8.9
  - f(C)= .2 + 0 = 0.2
- Path cost estimate reduces
  - This is not reasonable since we are reducing the estimate of the actual cost of the path
  - How to make f monotonic along a path?
    - f(n) = max(f(parent), g(n) + h(n))
    - Known as Pathmax correction



A* Search - Analysis

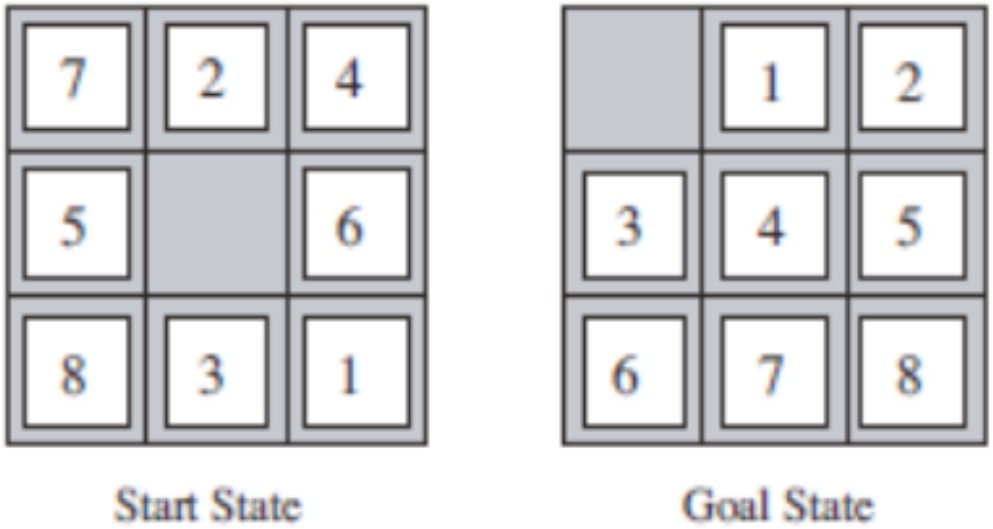
- Completeness:
  - Yes, (branching factor should be finite, i.e. unless there are infinitely many nodes with f ≤ f(G)
- Optimality: Yes
- Space Complexity:
  - exponential as it keeps all nodes in memory
- Time Complexity: exponential in [relative error in terms of h, h *]
  - A* expands all nodes with f < C*
  - A* expands some nodes with f = C*
  - A* expands no nodes with f > C*

Effect of Heuristic Accuracy on Performance

- Total number of nodes generated by A* search is N
- Solution depth is d
- Effective branching factor – b*
  - branching factor of a uniform tree of depth d with n + 1 nodes $N + 1 = 1 + b^* + (b^*)^2 + \cdots + (b^*)^d$
  - What should be the value of b* to perform search efficiently?

Example: 8-puzzle Problem

- 8-puzzle problem:



- Two heuristics:
  - Number of misplaced tiles – $h1$
  - Total Manhattan distance - $h2$

Example: 8-puzzle Problem

- 8-puzzle problem performance:

	Search Cost (nodes generated)			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Figure 3.29 Comparison of the search costs and effective branching factors for the ITERATIVE-DEEPENING-SEARCH and A* algorithms with h_1, h_2. Data are averaged over 100 instances of the 8-puzzle for each of various solution lengths d.

Example: 8-puzzle Problem

- For two heuristics - $h1$ and $h2$
- If $h2$(n) ≥ $h1$(n) , ∀n Then $h2$ dominates $h1$
- Domination : efficiency of search – in terms of number of nodes expanded
  - $h2$ will never expand more nodes than A* using $h1$ (except for some nodes with (f(n) = C*)

Iterative Deepening A* (IDA*) Search

- Essentially IDDS, that uses f as the cost threshold, instead of depth
- Implementation: add child to the queue if f(child) < threshold
- Algorithm:
  - Start with f cutoff equal to the f value of the root node
  - Loop until solution is found
    - Generate and search all nodes whose f values are ≤ the current threshold
    - Use DFS to search the trees in each iteration
    - Keep track of the node which has the smallest f value that is > the current threshold
    - If a goal node is found terminate, else set the threshold to be next highest f value and loop back

IDA* Search Analysis

- Completeness: yes
- Optimality: yes
- Time Complexity: worst case: b^d
  - When all nodes have distinct f values
- Space Complexity: linear - bd