CS 182 Lecture 3: Informed Search and Local Search

Professors: Ariel Procaccia and **Stephanie Gil**

Email: sgil@seas.harvard.edu

Prof. Gil Office hours: Wednesdays 2:30-3:30p

Last Time:

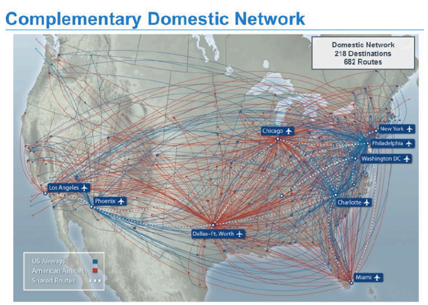
- Problem representation
  - States
  - Actions
  - Successor functions

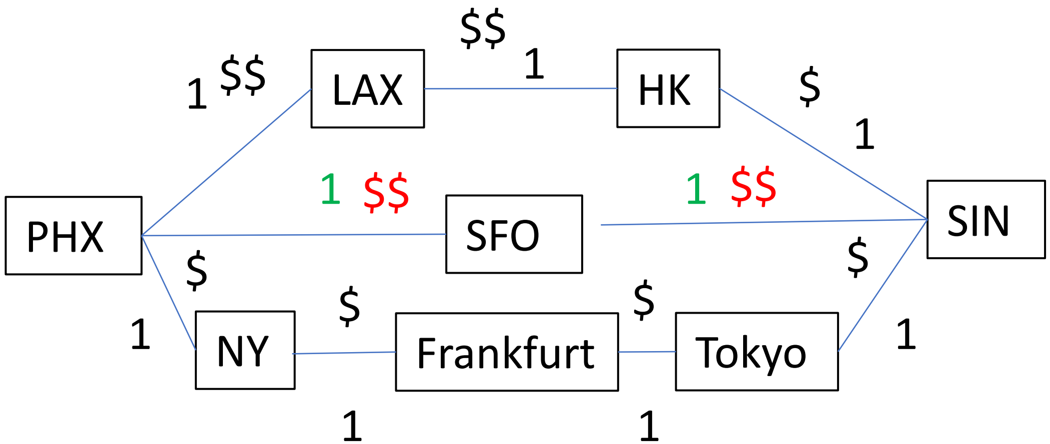
- Uninformed search
  - DFS
  - BFS
  - And variants like IDS

- Today: reference readings Russell and Norvig ch 3.5-3.7, 4.1

Last Time: Breadth-first Search

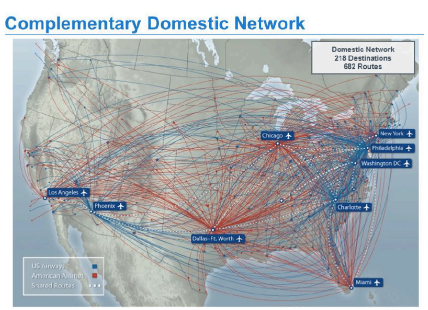
- Is BFS optimal?
- Airline route example…

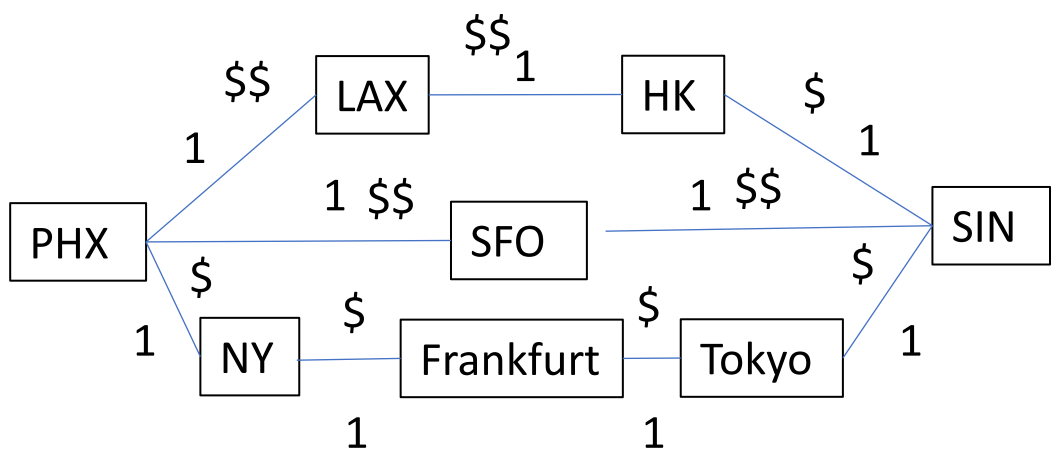




Uniform-cost Search

- Idea: change the order of node expansion
- <u>Uniform-cost search</u> – expand the node n with the lowest path cost $g(n)$
- Which path is returned by uniform-cost search on the airline example?





Uniform-cost Search Performance

- Is uniform-cost search **optimal**?
- Is uniform-cost search **complete**?
  - Where can this go wrong?
  - Idea: require every action to exceed some $\epsilon > 0$
- Time and space complexity?
  - Not only a function of b, d, m in this case! Depends on the cost of the optimal solution C*

Uninformed vs. Informed Search

- Uninformed methods – Only generate successors and distinguish goal from non-goal states
- Informed methods – Use *strategies* that know whether one non-goal states is more promising than another
  - How? Usually by using some more information about the problem!

Informed Search

- Evaluation function f(n)
  - Example: distance to the goal
  - Implemented as a priority queue that maintains the fringe in ascending order of f-values

- Heuristic function h(n)
  - Estimated cost of cheapest path from node n to a goal node

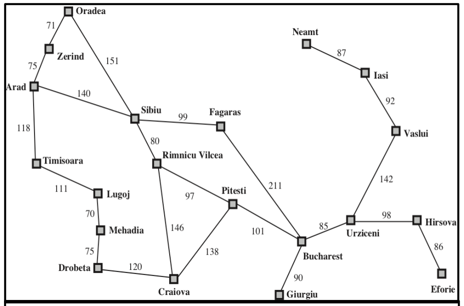


Fig 3.2 Russell and Norvig

What is a good candidate for h(n) in this case?

Heuristic Functions

Characteristics of h(n):

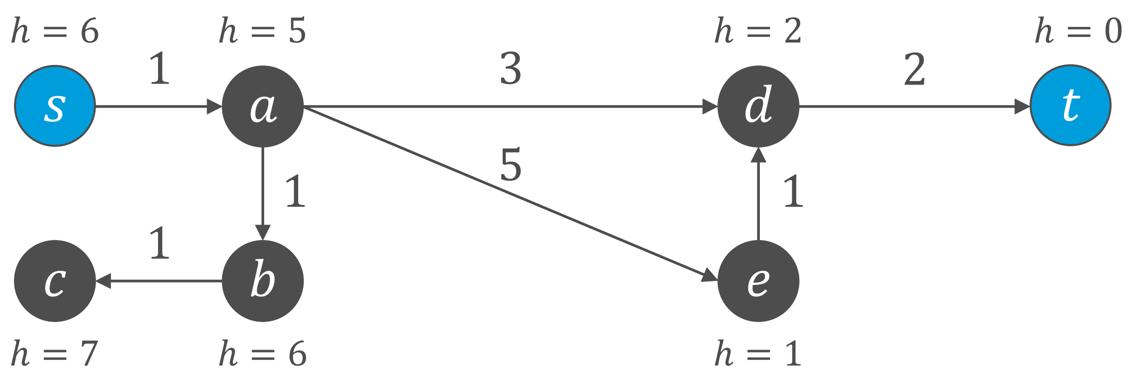
1) Most common form in which additional knowledge of the problem is imparted to the search algorithm

2) Should underestimate the cost to the goal (admissible heuristics – more on this soon)

3) If n is the goal node then h(n)=0

Greedy Best-first Search

- Strategy: Expand the node closest to the goal. Uses f(n)=h(n).
- **Q1:** Which path is returned by greedy search?



Greedy Best-first Search

- Example: perform greedy best-first search to get from Arad to Bucharest

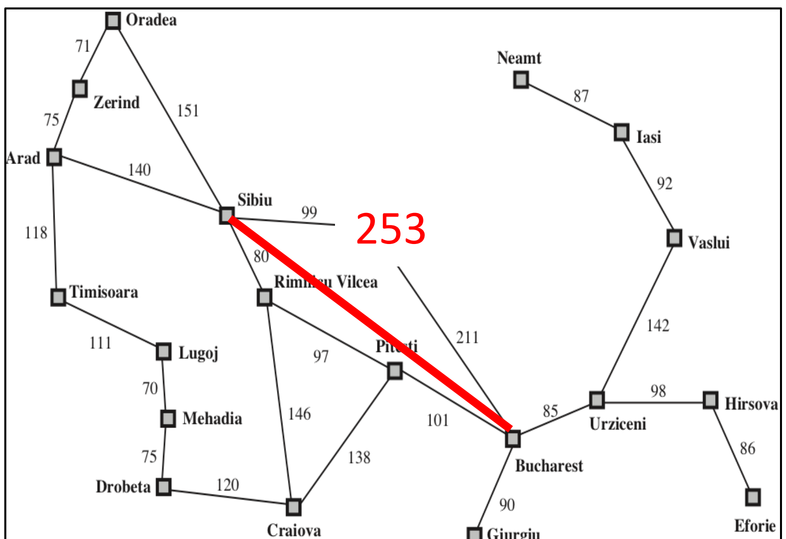


Fig 3.2 Russell and Norvig

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Greedy Best-first Search (cont.)

- Example: perform greedy best-first search to get from Arad to Bucharest

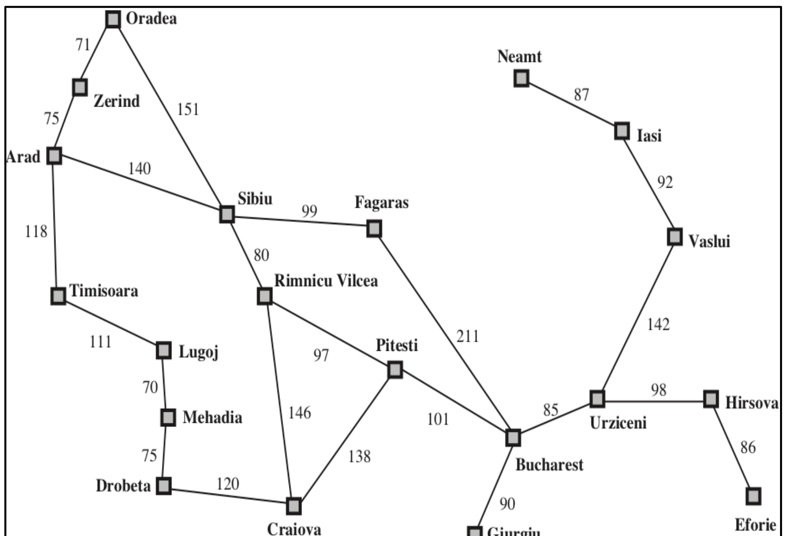


Fig 3.2 Russell and Norvig

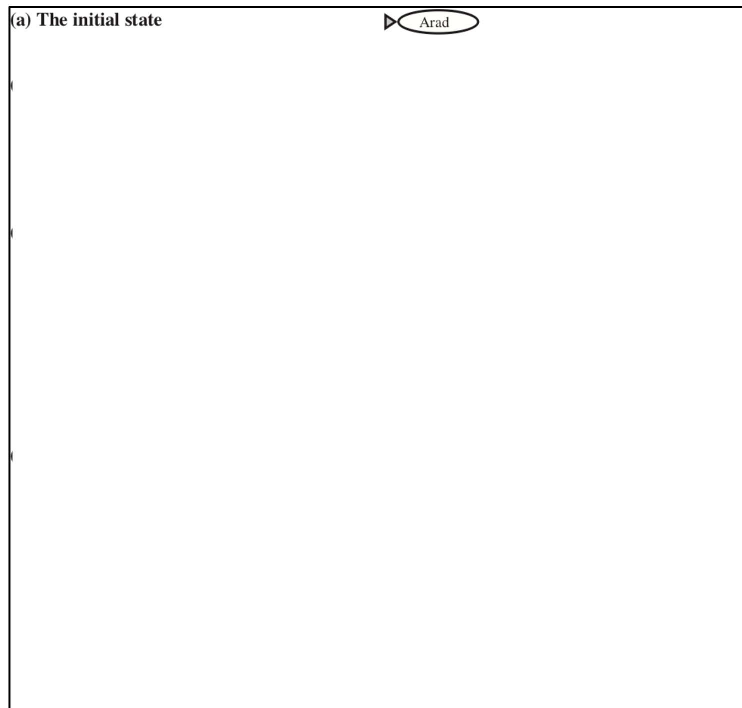


Fig 4.2 Russell and Norvig 2nd edition

Is the solution optimal?

Characteristics of Greedy Search

- Not optimal
- Incomplete (like DFS it can start down an infinite path and never return to try others)
- Worse case time and space complexity is $O(b^m)$, where m is the maximum depth of the search space
- A good heuristic can reduce complexity substantially

A* Search

- Uses a different evaluation function

$$f(n) = g(n) + h(n)$$

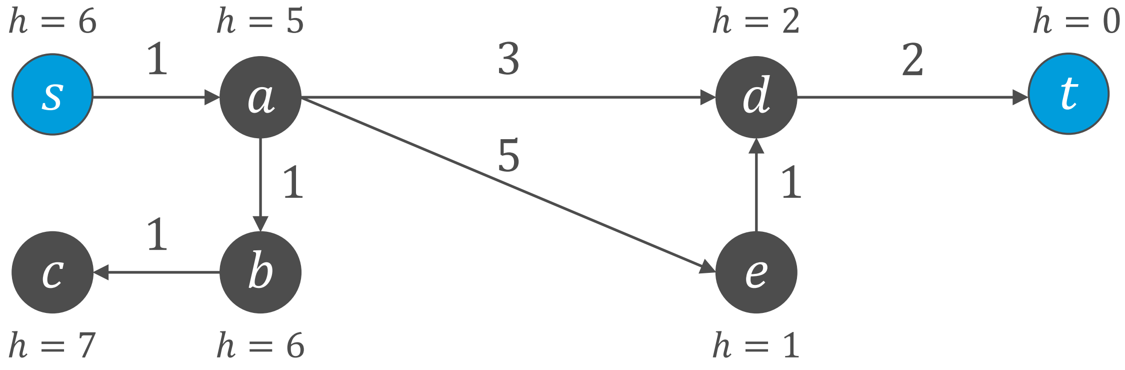
- f(n): the estimated cost of the cheapest solution through n

➢ Provided that the heuristic function h(n) satisfies certain conditions, A* search is both complete and optimal

A* Poll

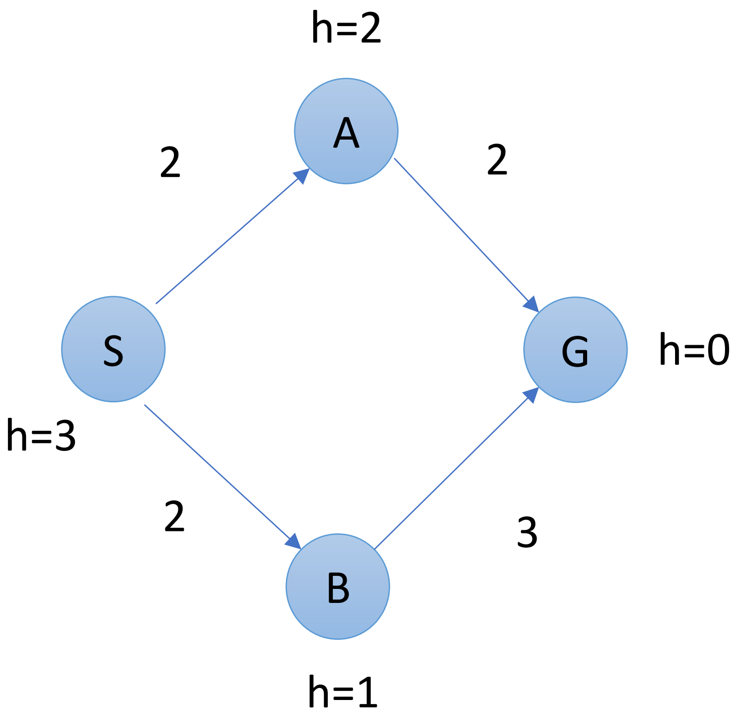
- Strategy: Expand using lowest cost f(n)=h(n)+g(n)

- **Q2** (polls-everywhere poll): Which node is expanded fourth?



A* Termination

- Rule: expand nodes in order of lowest cost f(n) = g(n) + h(n)
- **Q3:** Should we stop when we enqueue a goal node?



Fringe	f(n)	Pop

Admissible Heuristics

- Admissible heuristic – h(n) never overestimates the cost to reach the goal

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Greedy search: Arad -> Sibiu -> Fagaras -> Bucharest

A*: Arad -> Sibiu -> Fagaras (f =140+99+176=415)

-> RV (f=140+80+193=413)

**RV indeed is the optimal choice*

- Since g(n) is the exact cost to reach n, an immediate consequence is that f(n) never overestimates the true cost of a solution through n
  - What does this mean for missing an optimal solution?

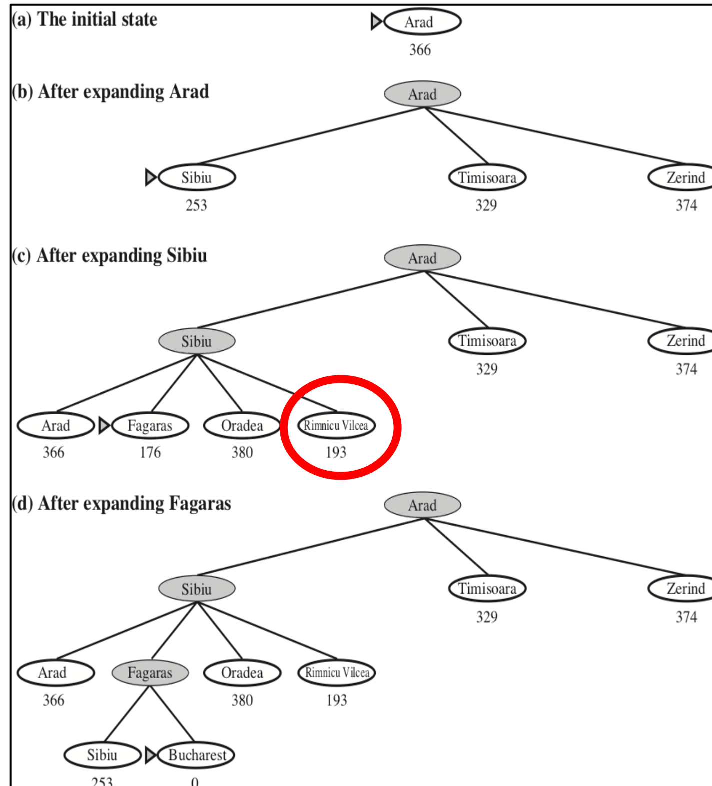


Fig 4.2 **Greedy Search** Russell and Norvig 2nd edition

Admissibility and Optimality

- A heuristic h is *admissible* if for all nodes n,

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

where h* is the cost of the optimal path to a goal from n

- Theorem: A* tree search with an admissible heuristic returns an optimal solution

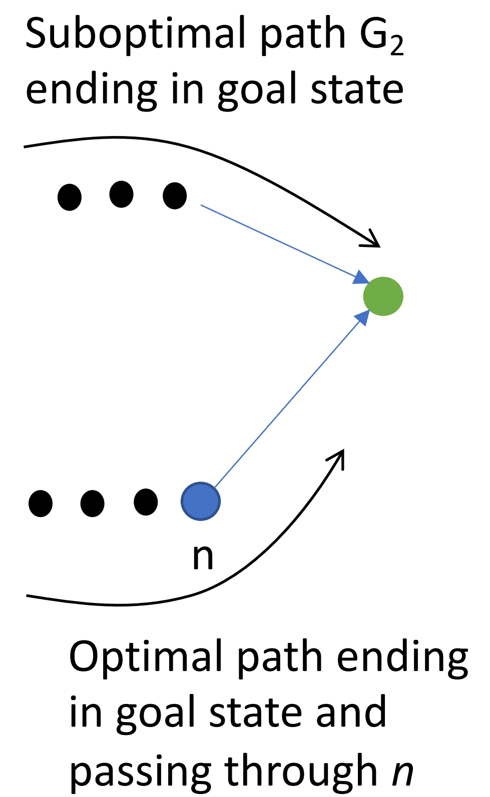
Proof of Theorem

1) Let G_2 be a *suboptimal* goal node (path) that appears on the fringe, let the cost of the optimal solution be C^*

2) Then $f(G_2) = g(G_2) + h(G_2) = g(G_2) > C^*$

3) Consider a fringe node n on an optimal solution path. If h(n) does not overestimate the cost to complete the path we have that

$$f(n) = g(n) + h(n)$$



Design of Heuristics: 8 Puzzle

- What is a state for this game?
- 181,440 distinct states are reachable
- We need an admissible heuristic.

Hint: relax the problem

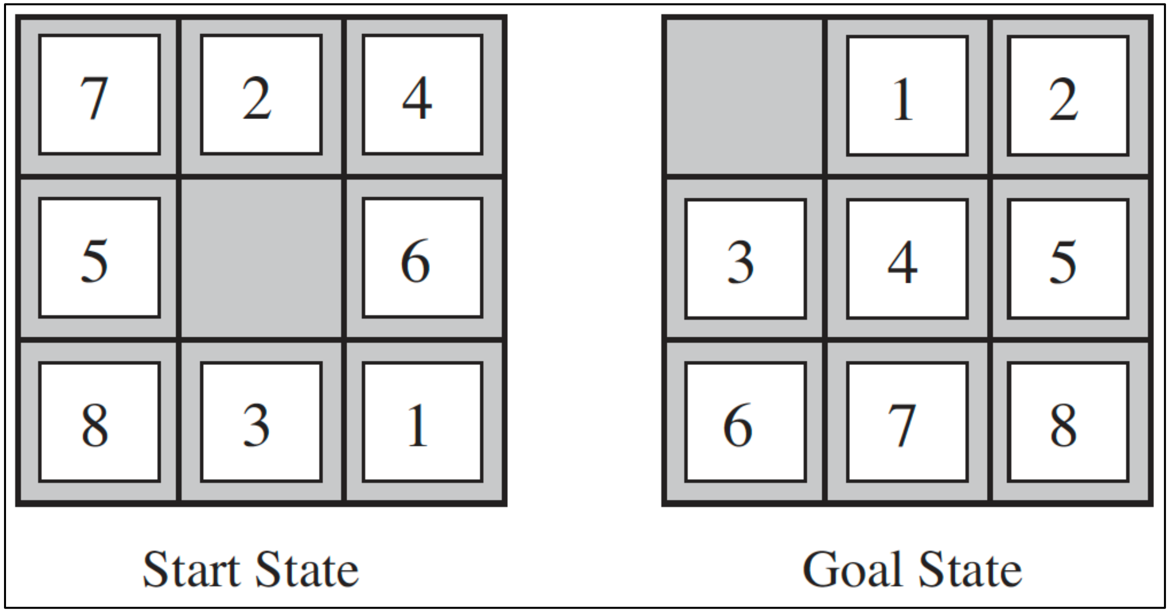


Figure 3.28 Russell and Norvig text

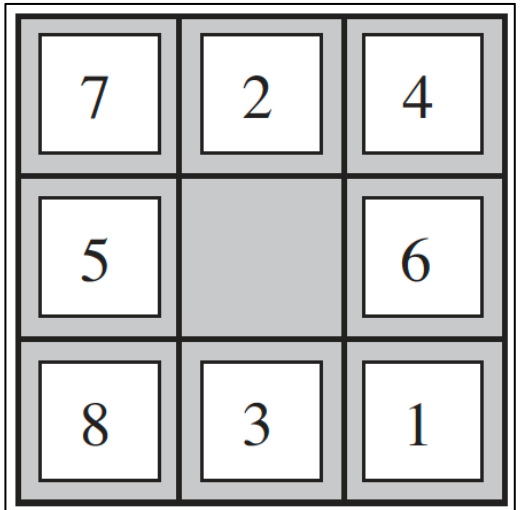
Two possible heuristics:

- h_1= the number of misplaced tiles
- h_2= sum of the distances of tiles from their goal positions (Manhattan distance)
- True solution = 26

Design of Heuristics: 8 Puzzle

- **Two possible heuristics:**
- h_1= the number of misplaced tiles
- h_2= sum of the distances of tiles from their goal positions (Manhattan distance)

What is the ideal (best case) branching factor for a search algorithm?



Start State

Figure 3.28 Russell and Norvig text

	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

Figure 3.29 Russell and Norvig text

Characteristics of A*

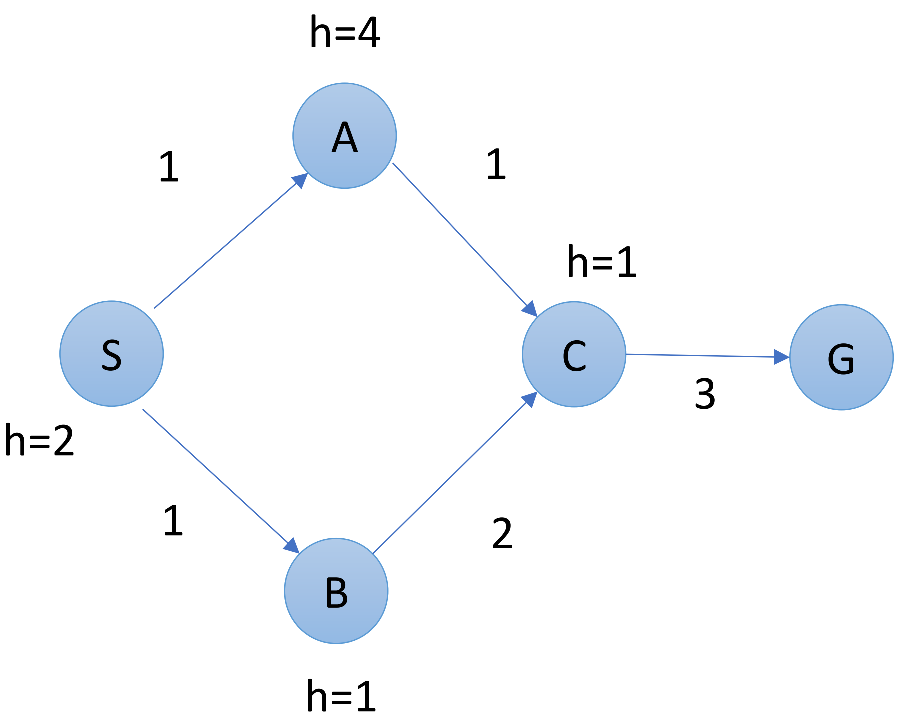
A* is complete (finds a solution if one exists) and optimal (finds the optimal path to the goal) if:

- The branching factor is finite
- Arc costs are >0
- h(n) is admissible

- But A* is expensive in memory $O(b^d)$ (like BFS)

A* using Graph Search

- Rule: expand nodes in order of lowest f(n) cost, do not expand the same node twice (use a "closed set")
- **Q4:** Is A* using a graph search implementation optimal?



Fringe　　Closed Set　　Pop

Optimality of Graph Search

- The graph-search algorithm always discards the newly discovered path, even if it is shorter than the first path discovered

- New idea: before discarding a path, check if newly discovered path to a node is better than the originally discovered path. If yes, revise depths and path cost of node's descendants.
  - How to tell if a new path is better?...

A* and Graph Search

- The previous proof does not hold for graph search. Why?

- Two ways around this
  - Extend graph search to discard the more expensive of any two paths found to the same node.
  - Ensure that the optimal path to any repeated state is always the first one followed.

- Important concept, *consistency*

Consistent Heuristics

- A heuristic h(n) is consistent if for every node n and every successor n’ of n generated by action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n’ plus the estimated cost of reaching the goal from n’

$$h(n) \leq c(n, a, n') + h(n')$$

- Example: Suppose that n’ is a successor of n

n’=RV

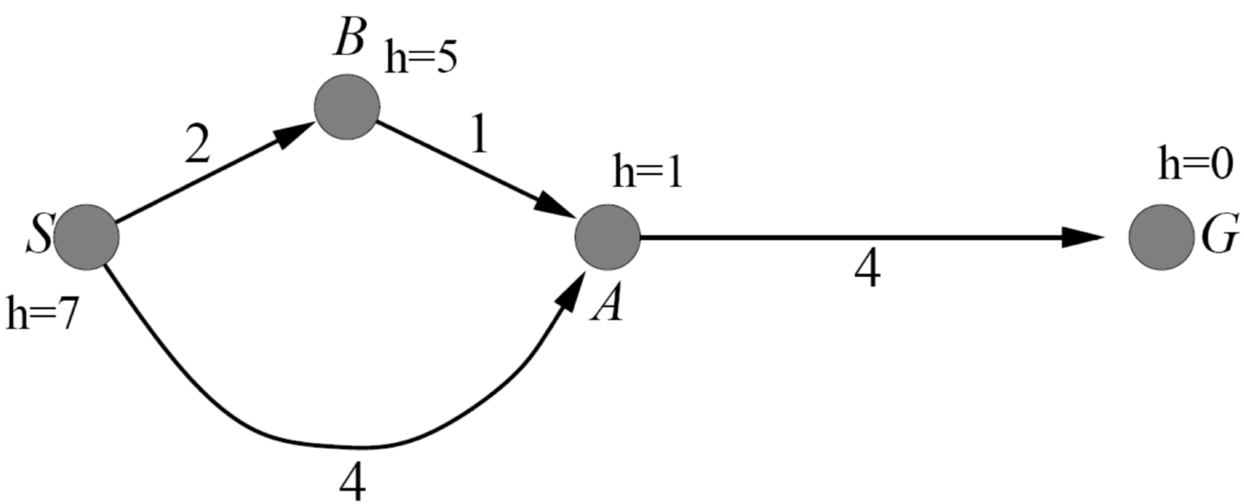
n = Sibiu

Exercise at home: Show that f(n) along any path is nondecreasing such that f(n’)>= f(n) if h(n) is consistent

Consistent vs Admissible Heuristic

- **Q5:** Is this heuristic admissible? Is it consistent?

$h(n) \leq c(n, a, n') + h(n')$



Consistency in the Heuristic (cont.)

- This means that the first goal node selected *for expansion* must be an optimal solution (since all later nodes will be at least as expensive)
  - This allows us to get around book-keeping!
  - Note that we can arrive at the goal node via a suboptimal path but this path won't be *expanded* (e.g. Bucharest example)

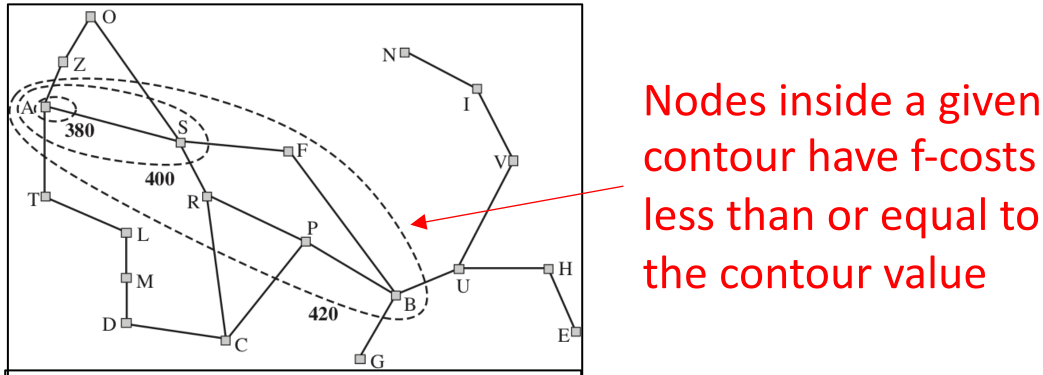


Fig 3.25 Russell and Norvig

Optimality

- Tree Search:
  - A* is optimal if heuristic is admissible
- Graph Search:
  - A* is optimal if heuristic is consistent
- Consistency implies admissibility

Local Search

- Uses a single current state (rather than keeping track of multiple paths) and generally move only to neighbors of that state.

- Typically, the paths followed by the search are not retained

- When to use:

1) Use very little memory (often use a constant amount)
2) Can often find reasonable solutions in a large or infinite (i.e. continuous) state space for which previous systematic algorithms are unsuitable

An Objective Function

- These algorithms aim to find the best state according to an objective function (often replaces the "goal test" and "path cost" of previous search methods)
- The "goodness" of a state is described by a function

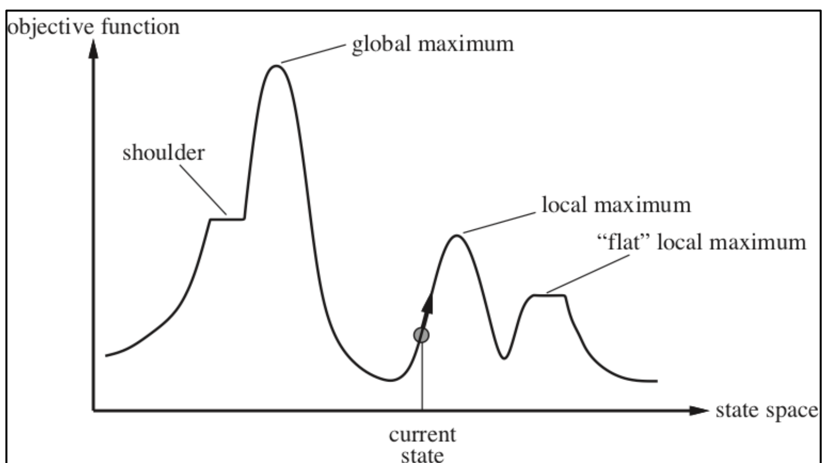
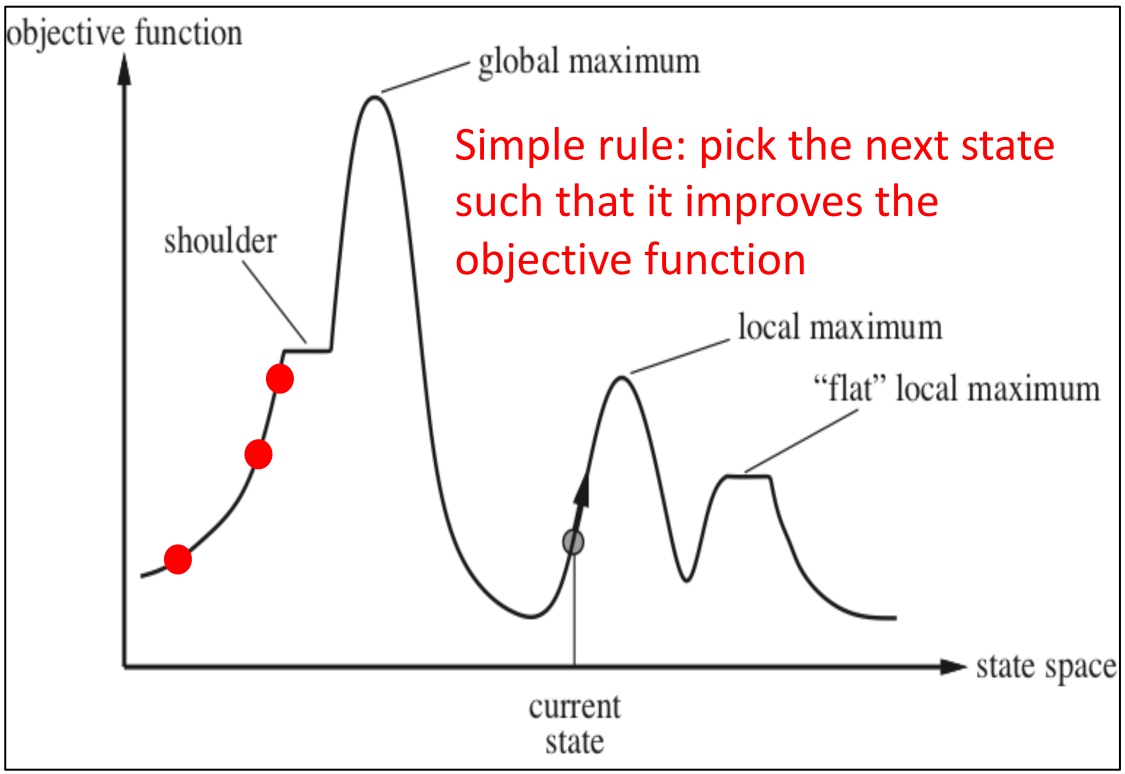


Fig 4.1Russell and Norvig

Hill Climbing



- Pitfalls
  - Local maxima – may not every find solution
  - Non-smooth function
- Stochastic variants
  - Can escape local maxima
  - But still no guarantee of finding global maximum

Next Time…

- Constraint Satisfaction Problems (Russell and Norvig Ch. 6)