ANNOUNCEMENTS

Projects Ideas – Next week

INFORMED / HEURISTIC SEARCH

WITH MATERIAL DRAWN FROM

RUSSELL & NORVIG, THE WEB, ROB PLATT CS4100/CS5100, BERKELEY CS188

OUTILINE

- Informed Search
  - Heuristics
  - Use of heuristics
    - In Greedy Search
    - In A* Search
- Graph Search vs Tree Search
  - Impact on heuristics

REMARK: HEURISTICS AT THE BIRTH OF AI

"Heuristic Search Hypothesis. The solutions to problems are represented as symbol structures. A physical symbol system exercises its intelligence in problem solving by search-that is, by generating and progressively modifying symbol structures until it produces a solution structure."
--- Alan Newell

"A physical symbol system has the **necessary and sufficient** means for general intelligent action."
― Allen Newell and Herbert A. Simon

necessary → human intelligence possesses the features of a physical symbol system

sufficient → Machines can be intelligent

RECAP: SEARCH

TREE SEARCH

VERSUS

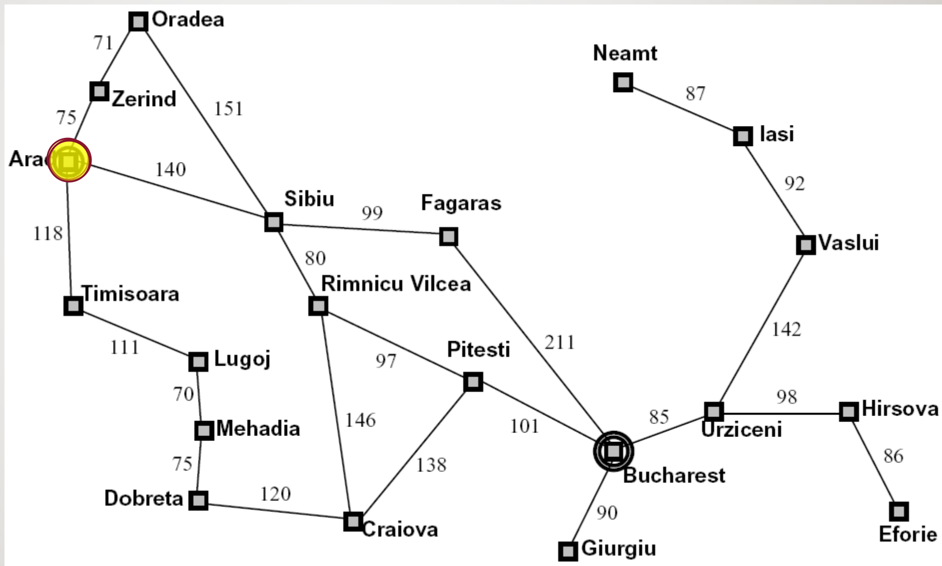
GRAPH SEARCH

```
function TREE-SEARCH(problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  loop do
     if the frontier is empty then return failure
     choose a leaf node and remove it from the frontier
     if the node contains a goal state then return the corresponding solution
     expand the chosen node, adding the resulting nodes to the frontier
```

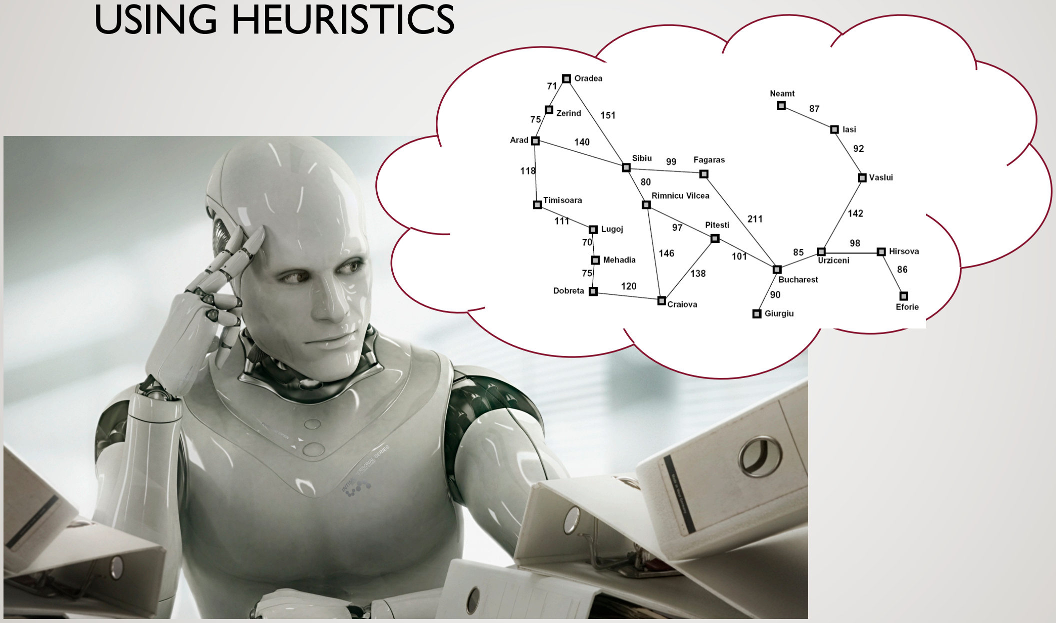
```
function GRAPH-SEARCH(problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  initialize the explored set to be empty
  loop do
     if the frontier is empty then return failure
     choose a leaf node and remove it from the frontier
     if the node contains a goal state then return the corresponding solution
     add the node to the explored set
     expand the chosen node, adding the resulting nodes to the frontier
       only if not in the frontier or explored set
```

UNIFORM COST SEARCH

- UCS is complete and optimal
- But explores in every direction
- No info on goal location



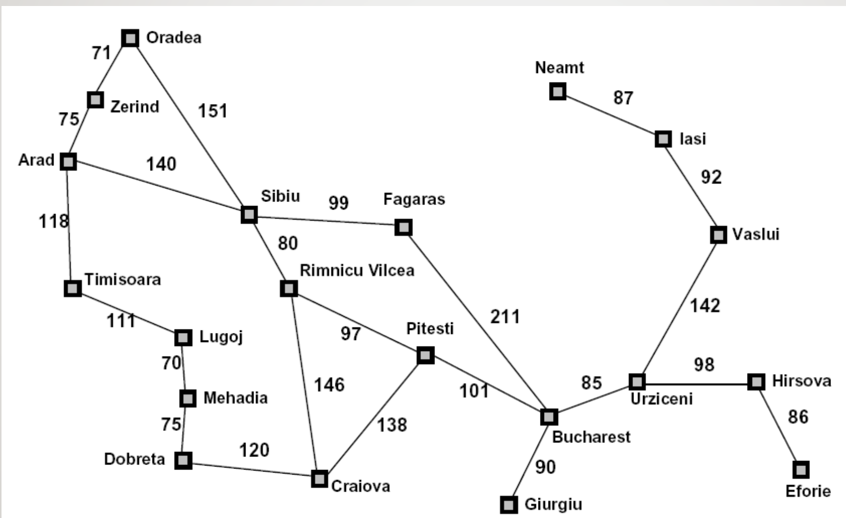
USING HEURISTICS



SEARCH HEURISTICS

- A heuristic is:
  - A function that *estimates* how close a state is to a goal
  - Designed for a particular search problem
    - IE – it encapsulates **domain knowledge**

STRAIGHT PATH HEURISTIC FOR PATH PLANNING



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

GREEDY SEARCH

Greedy Search

Essentially a guess about how far the state is from the goal

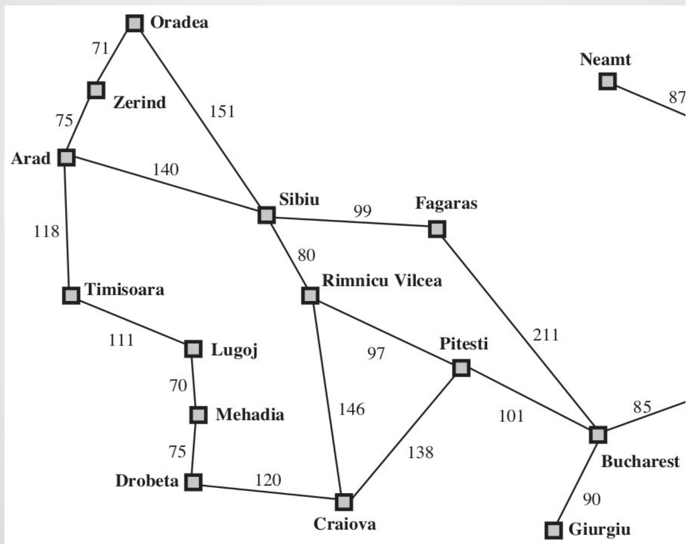
When the search expands a state,
- Calculate a heuristic measure *h(s)* of the distance to the goal for each state that is being added to the frontier

When choosing a state from the frontier to expand
- Expand the state with the lowest heuristic value

Example: For the Rumania example one could use the straight line distance to Bucharest

EXAMPLE: HEURISTIC FUNCTION

- Expand the node whose straight line dx to goal is lowest
- What can go wrong?

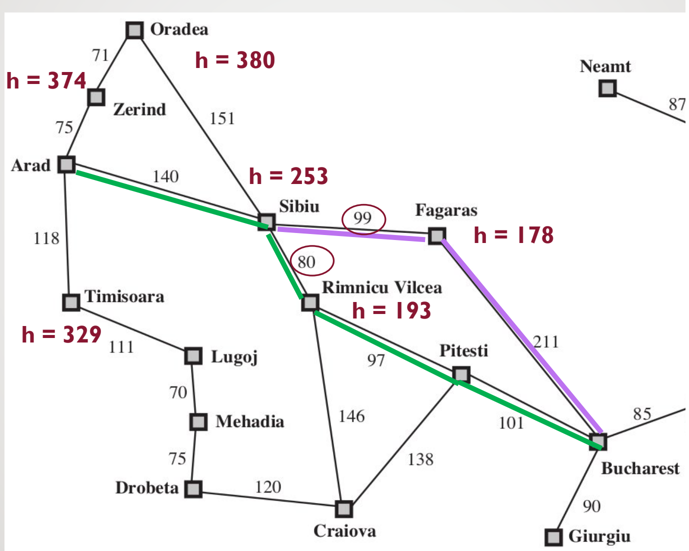


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example: Greedy Search

Path: A-S-F-B

is not optimal path!



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

GREEDY SEARCH

- The Good: takes into account where you want to go unlike UCS
  - expand a node that you think is closest to a goal state
  - Heuristic: estimate of distance to nearest goal for each state on the frontier
- The Bad:
  - Not taking into account how costly the path is to that state
  - Even worse the heuristic can be bad

Greedy vs UCS

Greedy Search:

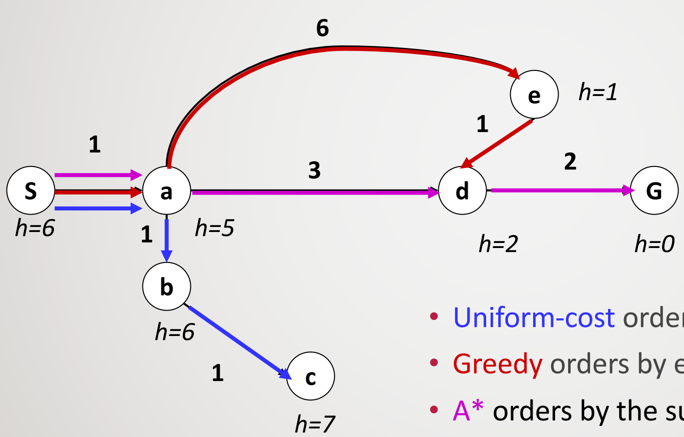
- Not optimal
- Not complete
- But, it can be very fast

UCS:

- Optimal
- Complete
- Usually very slow

A* SEARCH

A* COMBINES UCS AND GREEDY



- Uniform-cost orders by path cost from Start: g(n)
- Greedy orders by estimated cost to goal: h(n)
- A* orders by the sum: f(n) = g(n) + h(n)

Modified from: Teg Grenager

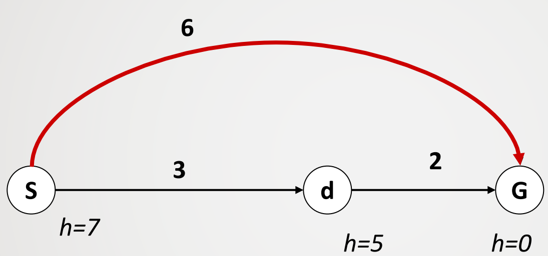
WHEN SHOULD A* TERMINATE?

When a goal is put on the queue (frontier)?

OR

When a goal is taken off the queue (to be expanded)?

A* WHEN IS IT OPTIMAL



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

- Actual non-optimal cost to the goal < estimated good path cost
- Need estimates less than actual costs

When is A* optimal?

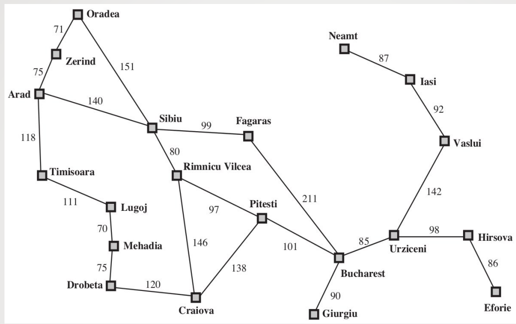
It depends on whether we are using the tree search or the graph search version of the algorithm.

Recall:
- in tree search, we do not track the explored set
- in graph search, we do

When is A* optimal?

- A* using tree search is optimal if h is **admissible**
  - h is admissible if h(n) is always less than or equal to the actual cost to go from state n to the goal
- A* graph search is optimal if h is **consistent**
  - h(n) does not overestimate actual cost of each arc

Admissible?



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

Stright-line distances to Bucharest

h(s) = straight-line distance to goal state (Bucharest)

Is this heuristic admissible???

BASIC IDEA OF ADMISSIBLE HEURISTICS

- Inadmissible heuristics break optimality because good plans get left on the frontier
- Admissible heuristics slow down expansion of bad plans but never override true costs

ADMISSIBLE HEURISTICS: DEFINITION

- A heuristic h(n) is admissible if it underestimates the true cost to the goal and is non-negative
  - $0 \le h(n) \le c(n)$, where c(n) is cost to the goal
  - $h(n) = 0$ if n is the goal

- Keep in mind: $f(n) = g(n) + h(n)$
  - The estimated cost of the path to the goal from n, f(n), is the sum of the cost from the start to n, g(n), and estimated cost to get to the goal from n, h(n).

- Finding (good) admissible heuristics is the critical part of using A*

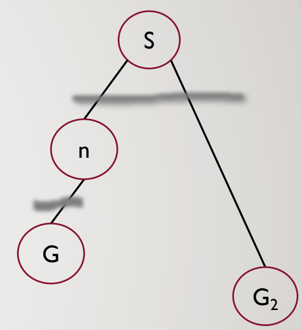
OPTIMALITY OF A* TREE SEARCH

OPTIMALITY OF A* TREE SEARCH

- Suppose suboptimal goal G_2 in the frontier
- Let n be an unexpanded node on a lowest cost path to optimal G (possibly n=G)
- Recall f(n) = g(n) + h(n)

1. $f(G_2)$ $= g(G_2)$ since $h(G_2) = 0$
2. $f(G_2)$ $> g(G)$ since G_2 is suboptimal
3. $g(G)$ $\geq f(n)$ since h admissible & n on lowest cost path
4. $f(G_2)$ $> f(n)$ combining 2 and 3

Since $f(G_2) > f(n)$, A* will expand any n on the optimal path before G_2 and therefore never select G_2 for expansion



RECALL: When is A* optimal?

- A* using tree search is optimal if h is **admissible**
  - h is admissible if h(n) is always less than or equal to the actual cost to get to the an underestimate of the cost to go from state n in the search
- A* graph search is optimal if h is consistent
  - h(n) is equal or less than actual cost of each arc

TREE SEARCH

VERSUS

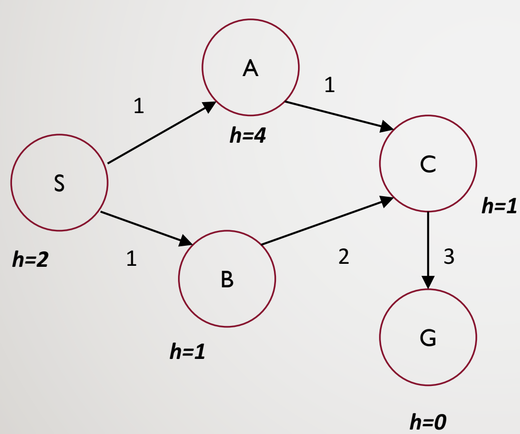
GRAPH SEARCH

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

QUESTION: A* TREE SEARCH VS GRAPH SEARCH

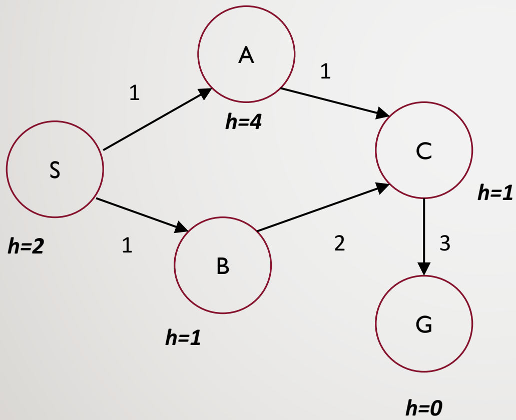
State space graph

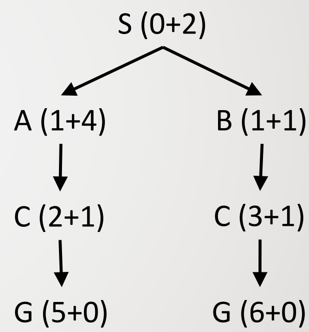


- Is this heuristic admissible?
- What order does Tree Search expand nodes?
  - Does it find the optimal solution?
- What order does Graph Search expand nodes?
  - Does it find the optimal solution?

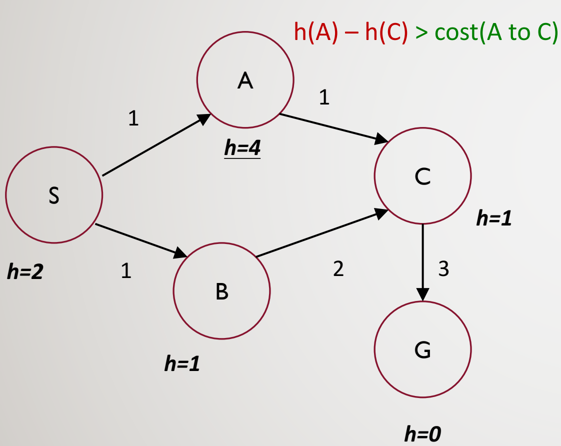
QUESTION: A* TREE SEARCH VS GRAPH SEARCH

State space graph





CONSISTENCY OF HEURISTICS



Estimated heuristic costs ≤ actual costs

- 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)

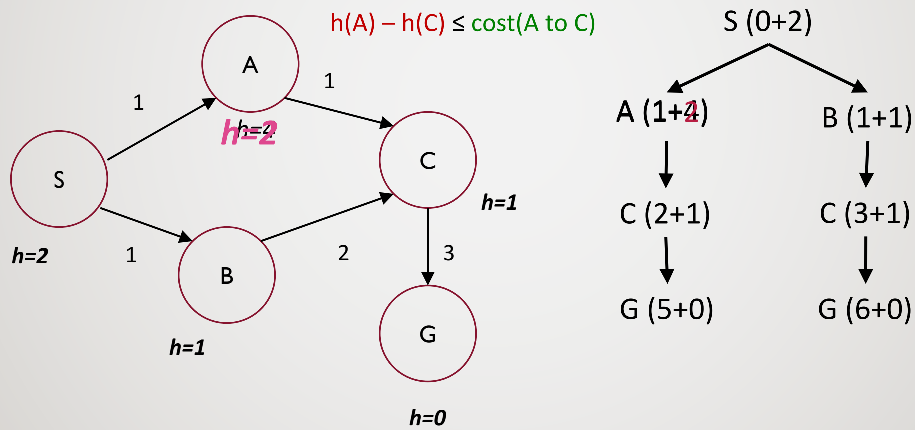
- Consequences of consistency:
  - The f value along a path never decreases

    h(A) ≤ cost(A to C) + h(C)

  - A* graph search is optimal

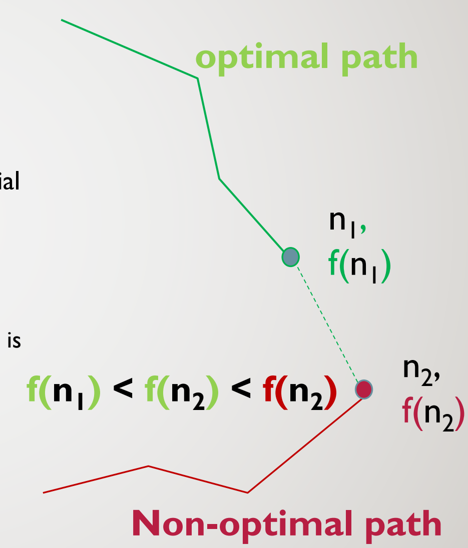
LETS FIX IT FOR GRAPH SEARCH

h(A) – h(C) ≤ cost(A to C)



CONSISTENT HEURISTICS

- Definition: h(n) ≤ c(n,a,n') +h(n')
  - For every node n and successor n' of n generated by any action a
- For consistent heuristics the estimated final cost of a partial solution is monotonically non-decreasing along the best path to the goal
- Implication in graph search
  - Once a node is expanded, the path by which it was reached is the lowest possible
- Can make an admissible heuristic into a consistent one by the pathmax eqn
  - h'(n') = max(h(n'),h(n) – c(n,n'))
  - Where n' is any descendant of n



PROPERTIES OF A*

UCS VS A* SEARCH

- Uniform-cost expands equally in all "directions"
- A* expands mainly toward the goal, but maintains alternatives in frontiers just in case

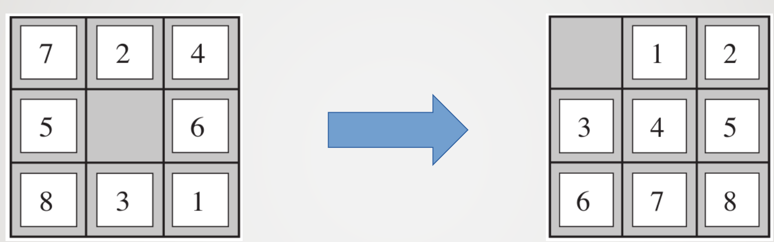
A* APPLICATIONS

A* APPLICATIONS

- Video games
- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition
- …

CREATING ADMISSIBLE HEURISTICS

Admissibility: Example



Start state

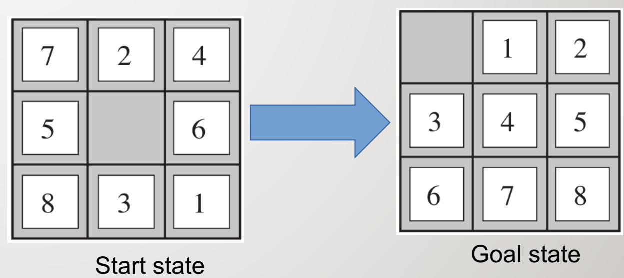
Goal state

h(s) = ?

Can you think of an admissible heuristic for this problem?

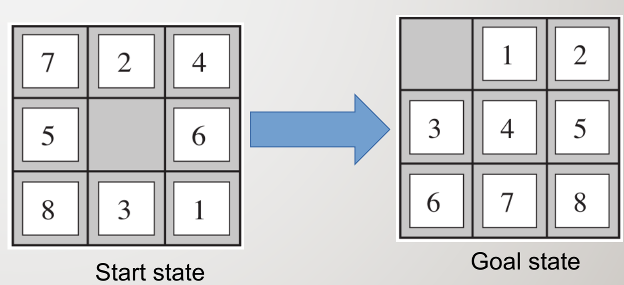
RELAXED PROBLEM APPROACH: 8 PUZZLE I

- Heuristic: Total Number of tiles misplaced
- Why is it admissible?
- h(start) = ???
- This is a *relaxed-problem* heuristic



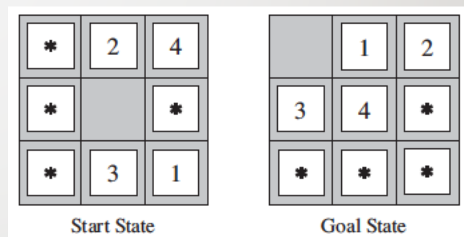
RELAXED PROBLEM APPROACH: 8 PUZZLE II

- What if any tile could slide any direction at any time, ignoring other tiles?
- Total *Manhattan* distance
- Why is it admissible?
- h(start) =



SUBPROBLEM APPROACH: 8 PUZZLE II

- Use exact solution cost for s subproblem
- Why is it admissible?
- Can store in pattern databases
- Construct by searching back from the goal
- Disjoint databases



COMBINING HEURISTICS

- Max of admissible heuristics is admissible
  - $h(n) = \max(h_a(n), h_b(n))$

USING ACTUAL COSTS

- How about using the *actual cost* as a heuristic?
  - Would it be admissible?
  - What's wrong with it?

- With A*: a trade-off between quality of estimate and work per node
  - As heuristics get closer to the true cost, you will expand fewer nodes but usually do more work per node to compute the heuristic itself

COMMENTS: HEURISTICS AND A*

- At one extreme, if h(n) is 0, then only g(n) plays a role, and A* turns into UCS
- The lower h(n) is, with respect to the real cost the more node A* expands, making it slower.
- If h(n) is exactly equal to the cost of moving from n to the goal, then A* will only follow the best path and never expand anything else, making it very fast.
- Non-admissible heuristics:
  - If h(n) is sometimes greater than the cost of moving from n to the goal, then A* is not guaranteed to find a shortest path, but it can run faster.
  - At the other extreme, if h(n) is very high relative to g(n), then only h(n) plays a role, and A* turns into Greedy Search.

SUMMARY: OPTIMALITY & HEURISTICS

- Tree search:
  - A* is optimal if heuristic is admissible
  - UCS is a special case (h = 0)
- Graph search:
  - A* optimal if heuristic is consistent
  - UCS optimal (since h = 0 is consistent)
- Consistency implies admissibility
- Heuristic design is key: often use relaxed problems