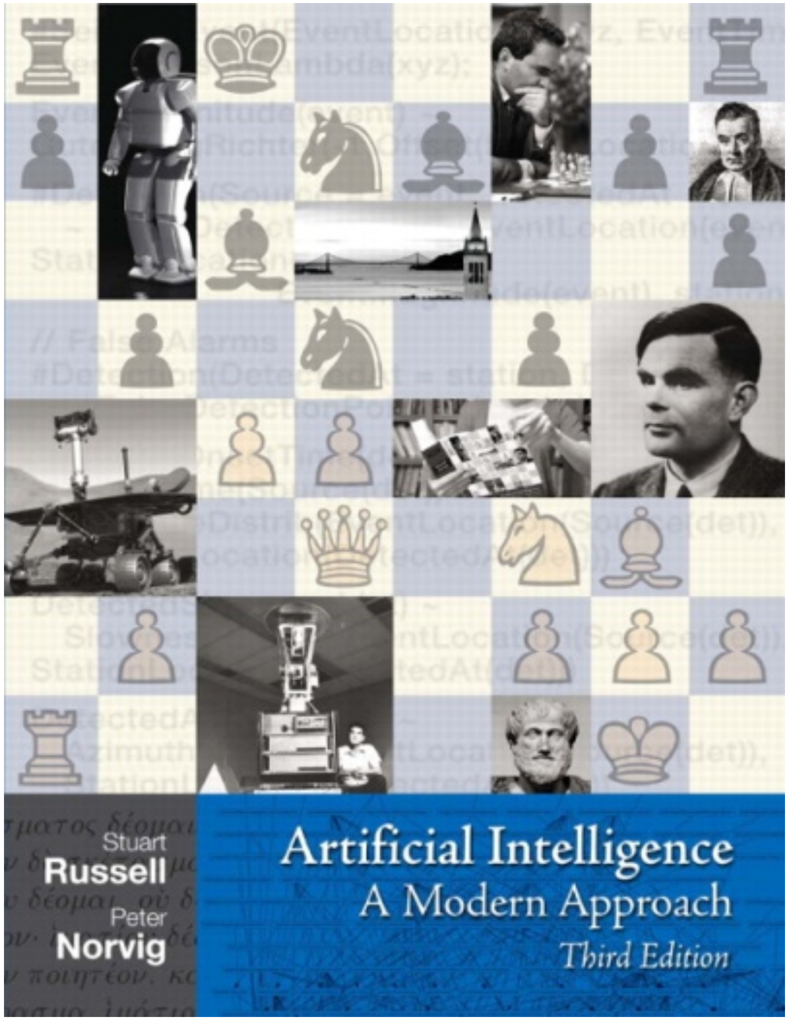
Informed Search

Read AIMA 3.1-3.6.
Some materials will not be covered in lecture, but will be on the exam.



Reminder – HW 2 has been released

- **HW2 has been released. It is due on Tuesday. It covers Uninformed Search and A* Search.**
- **I recommend that you start early.**

- **Friendly reminder about the late day policy: the homework is due by 11:59pm. 1 late day = anywhere between 1 second to 24 hours late. Please don't submit at the last minute.**

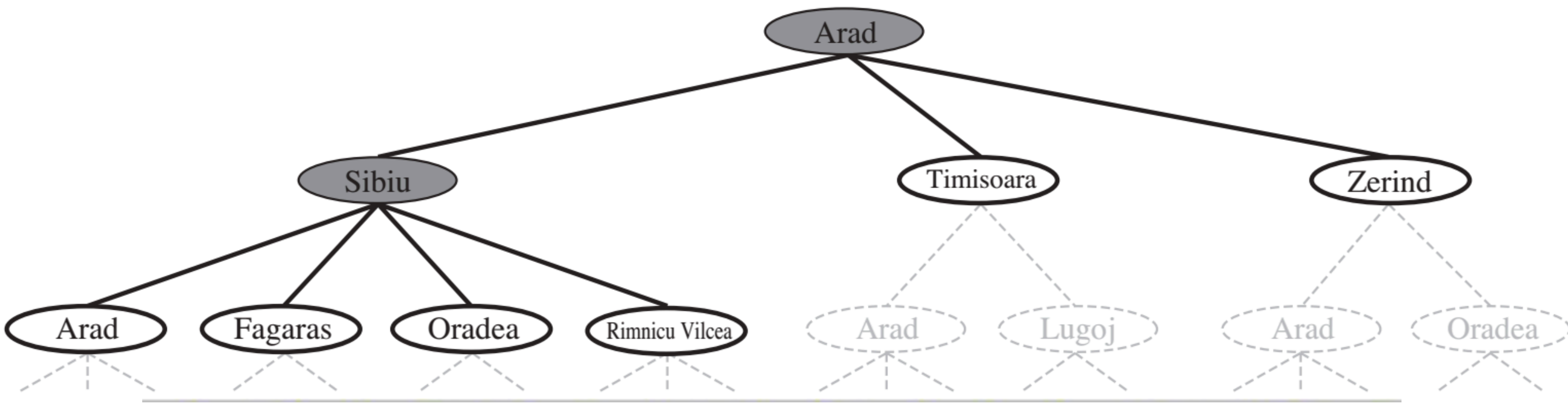
Review: Search problem definition

1. *States*: a set S
2. An *initial state* $s_i \in S$
3. *Actions: a set* A
   - $\forall s$ $Actions(s)$ = *the set of actions that can be executed in* s, *that are applicable* in s.
4. *Transition Model:* $\forall s \forall a \in Actions(s)$ $Result(s, a) \rightarrow s_r$
   - s_r *is called a successor of* s
   - $\{s_i\} \cup Successors(s_i)^*$ = *state space*
5. *Path cost (Performance Measure):* Must be additive
   - e.g. sum of distances, number of actions executed, …
   - $c(x,a,y)$ is the step cost, assumed ≥ 0
     - (where action a goes from state x to state y)
6. *Goal test:* $Goal(s)$
   - Can be implicit, e.g. $checkmate(s)$
   - s is a *goal state* if $Goal(s)$ is *true*

Review: Useful Concepts

- ***State space***: **the set of all states reachable from the initial state by *any* sequence of actions**
  - *When several operators can apply to each state, this gets large very quickly*
  - *Might be a proper subset of the set of configurations*
- ***Path***: **a sequence of actions leading from one state s_j to another state s_k**
- ***Frontier:*** **those states that are available for *expanding* (for applying legal actions to)**
- ***Solution***: **a path from the initial state s_i to a state s_g that satisfies the goal test**

Review: Tree search



function **TREE-SEARCH(*problem, strategy*)** return **a solution or failure**

Initialize frontier to the *initial state* of the *problem*

do

if **the frontier is empty** then return ***failure***

choose leaf node for expansion according to strategy & remove from frontier

if **node contains goal state** then return ***solution***

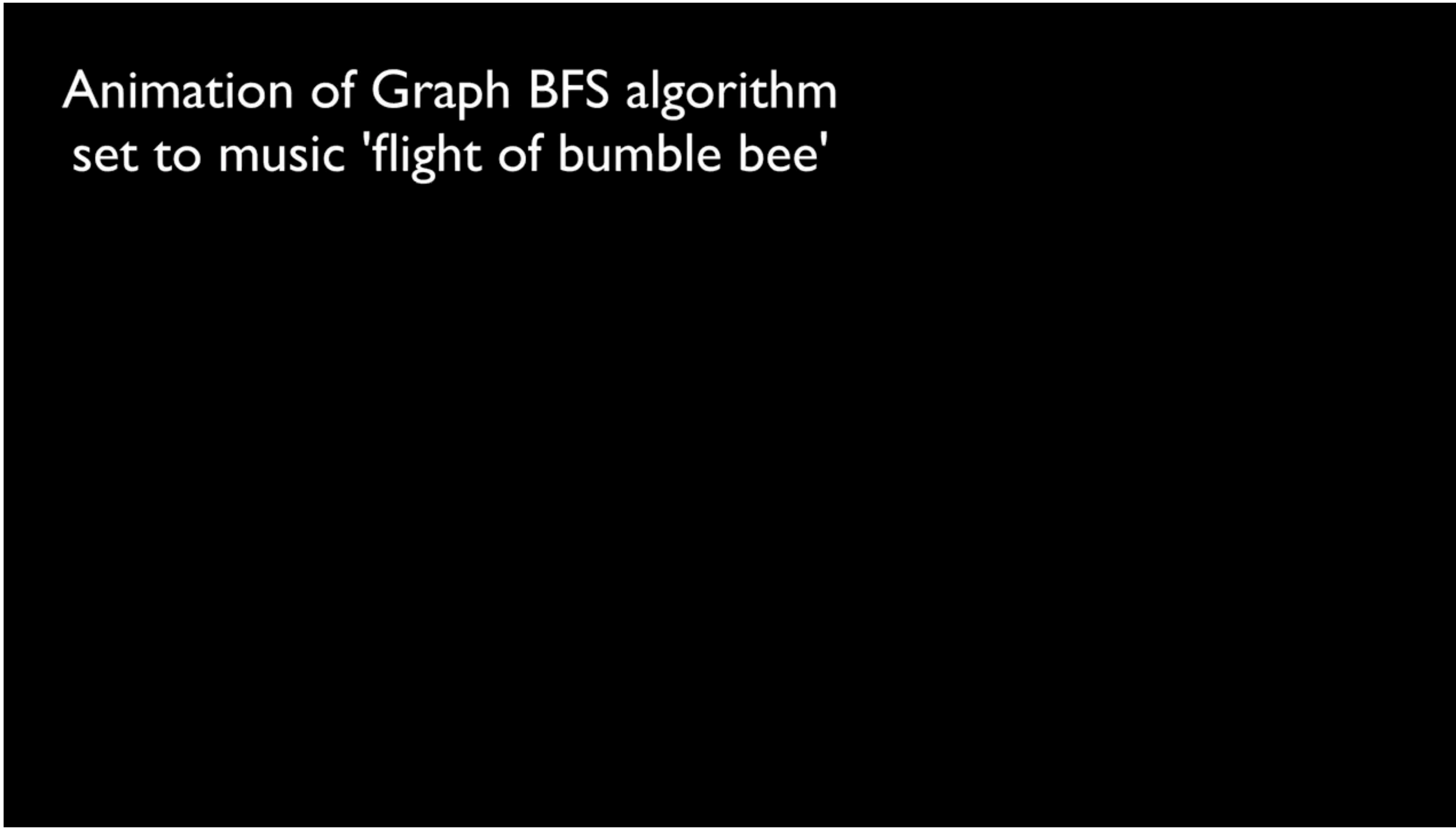
else **expand the node and add resulting nodes to the frontier**

Determines search *process!!*

Review: Search Strategies

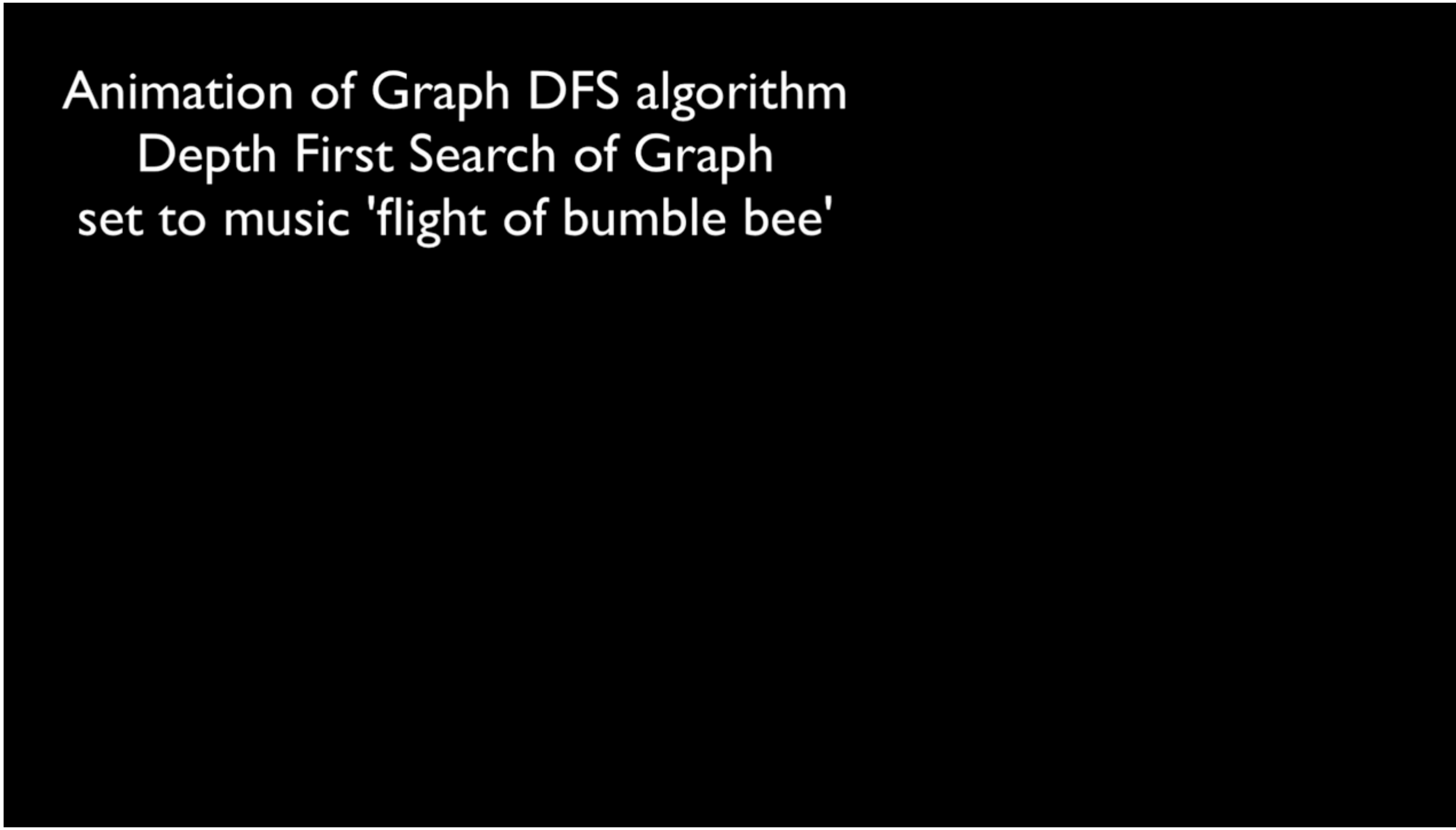
- ***Strategy* = order of tree expansion**
  - Implemented by different queue structures (LIFO, FIFO, priority)
- **Dimensions for evaluation**
  - *Completeness-* always find the solution?
  - *Optimality* - finds a least cost solution (lowest path cost) first?
  - *Time complexity* - # of nodes generated *(worst case)*
  - ***Space complexity* - # of nodes simultaneously in memory** *(worst case)*
- **Time/space complexity variables**
  - *b, maximum branching factor* of search tree
  - *d, depth* of the shallowest goal node
  - *m*, maximum length of any path in the state space (potentially ∞)

Breadth first search



https://youtu.be/x-VTfcmrLEQ

Depth first search



https://youtu.be/NUgMa5coCoE

Review: Breadth-first search

- **Idea:**
  - Expand *shallowest* unexpanded node

- **Implementation:**
  - *frontier* is FIFO (First-In-First-Out) Queue:
    - —Put successors at the *end* of *frontier* successor list.

Image credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Review: Depth-first search

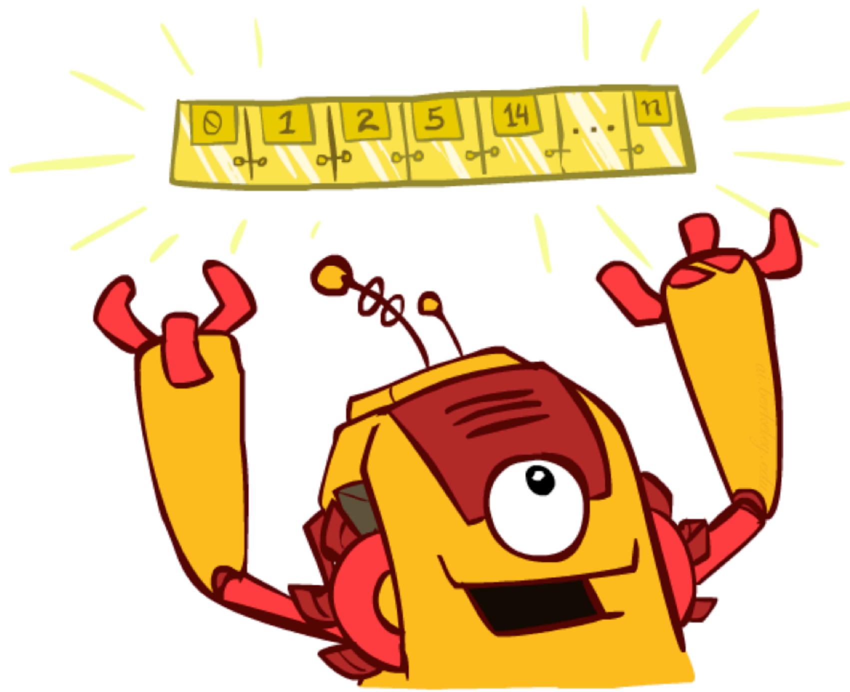
- **Idea:**
  - Expand *deepest* unexpanded node

- **Implementation:**
  - *frontier* is LIFO (Last-In-First-Out) Queue:
    - —Put successors at the *front* of *frontier* successor list.

Image credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Fringe Strategies with One Queue

- **These search algorithms are the same except for fringe strategies**
  - DFS strategy = LIFO stack
  - BSF strategy = FIFO queue
  - Conceptually, all fringes are priority queues (i.e. collections of nodes with attached priorities)
  - You can even code one implementation that takes a variable queuing object

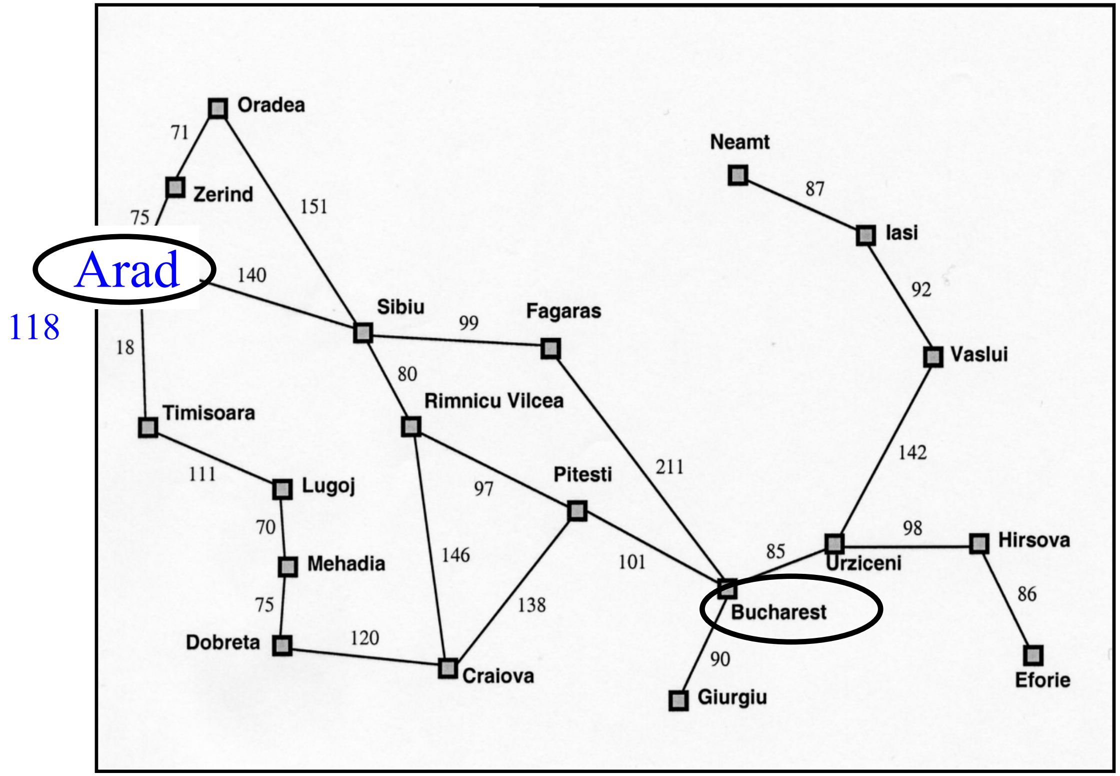


Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

“Uniform Cost” Search

“In computer science, *uniform-cost* search (UCS) is a tree search algorithm used for traversing or searching a *weighted* tree, tree structure, or graph.” - Wikipedia

Motivation: Romanian Map Problem



- **All our search methods so far assume**
  step-cost = 1
- ***This is only true for some problems***

g(N): the ***path cost*** function

- **Our assumption so far: All moves equal in cost**
  - Cost = # of nodes in path-1
  - *g(N)* = *depth(N)* in the search tree

- **More general: Assigning a (potentially) unique cost to each step**
  - N_0, N_1, N_2, N_3 = nodes visited on path *p* from N_0 to N_3
  - *C(i,j)*: Cost of going from N_i to N_j
  - If N_0 the root of the search tree,

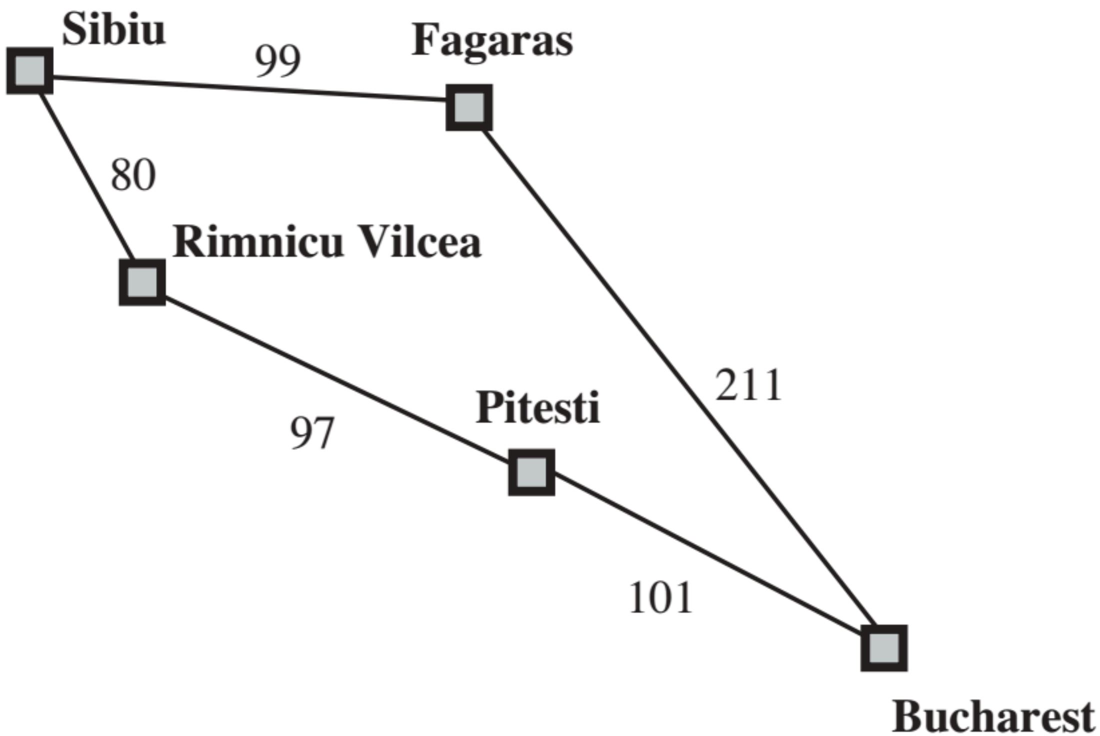
$$g(N3)=C(0,1)+C(1,2)+C(2,3)$$

Uniform-cost search (UCS)

- **Extension of BF-search:**
  - **Expand node with *lowest path cost***
- **Implementation:**
  ***frontier* = priority queue ordered by g(n)**
- **Subtle but significant difference from BFS:**
  - Tests if a node is a goal state when it is selected for expansion, not when it is added to the frontier.
  - Updates a node on the frontier if a better path to the same state is found.
  - So always enqueues a node *before checking whether it is a goal.*

WHY???

When should we check for goal state?



Uniform Cost Search

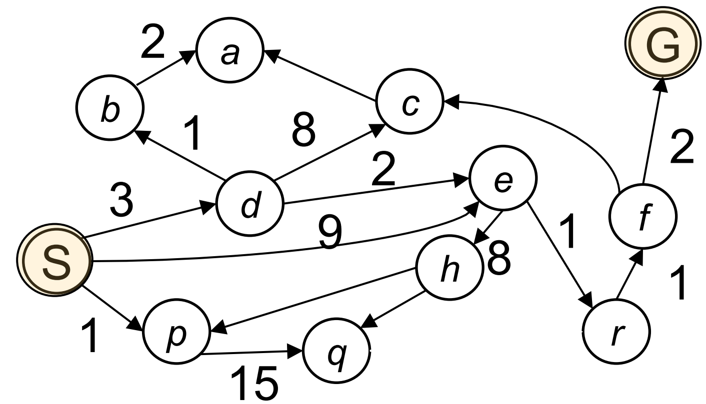
Slide from Stanford CS 221 (from slide by Dan Klein (UCB) and many others)

Expand cheapest node first:

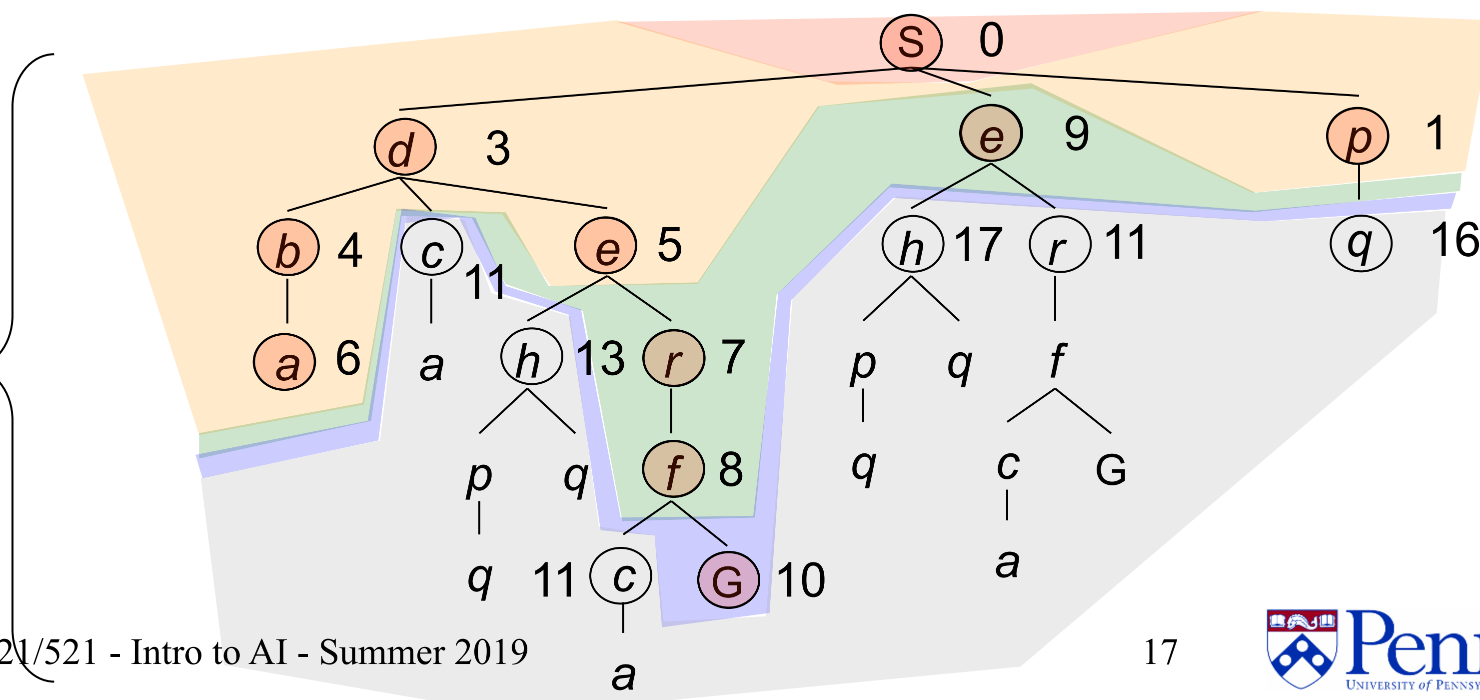
Frontier is a priority queue

No longer ply at a time, but follows cost contours

Therefore: Must be optimal



Cost contours



Complexity of UCS

- **Complete!**
- **Optimal!**
  - if the cost of each step exceeds some positive bound ε.
- **Time complexity:** $O(b^{C^*/\varepsilon + 1})$
- **Space complexity:** $O(b^{C^*/\varepsilon + 1})$

where C is the cost of an optimal solution, and* ε is min(*C(i,j)*)

(if all step costs are equal, this becomes $O(b^{d+1})$

NOTE: Dijkstra's algorithm just UCS without goal

Summary of algorithms (for notes)

Criterion	Breadth-First	Uniform-cost	Depth-First	Depth-limited	Iterative deepening	Bidirectional search
Complete?	**YES**	**YES**	**NO**	**NO**	**YES**	**YES**
Time	b^d	$b^{(C^*/e)+1}$	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	$b^{(C^*/e)+1}$	bm	bl	bd	$b^{d/2}$
Optimal?	**YES**	**YES**	**NO**	**NO**	**YES**	**YES**

Assumes b is finite

Outline for today's lecture

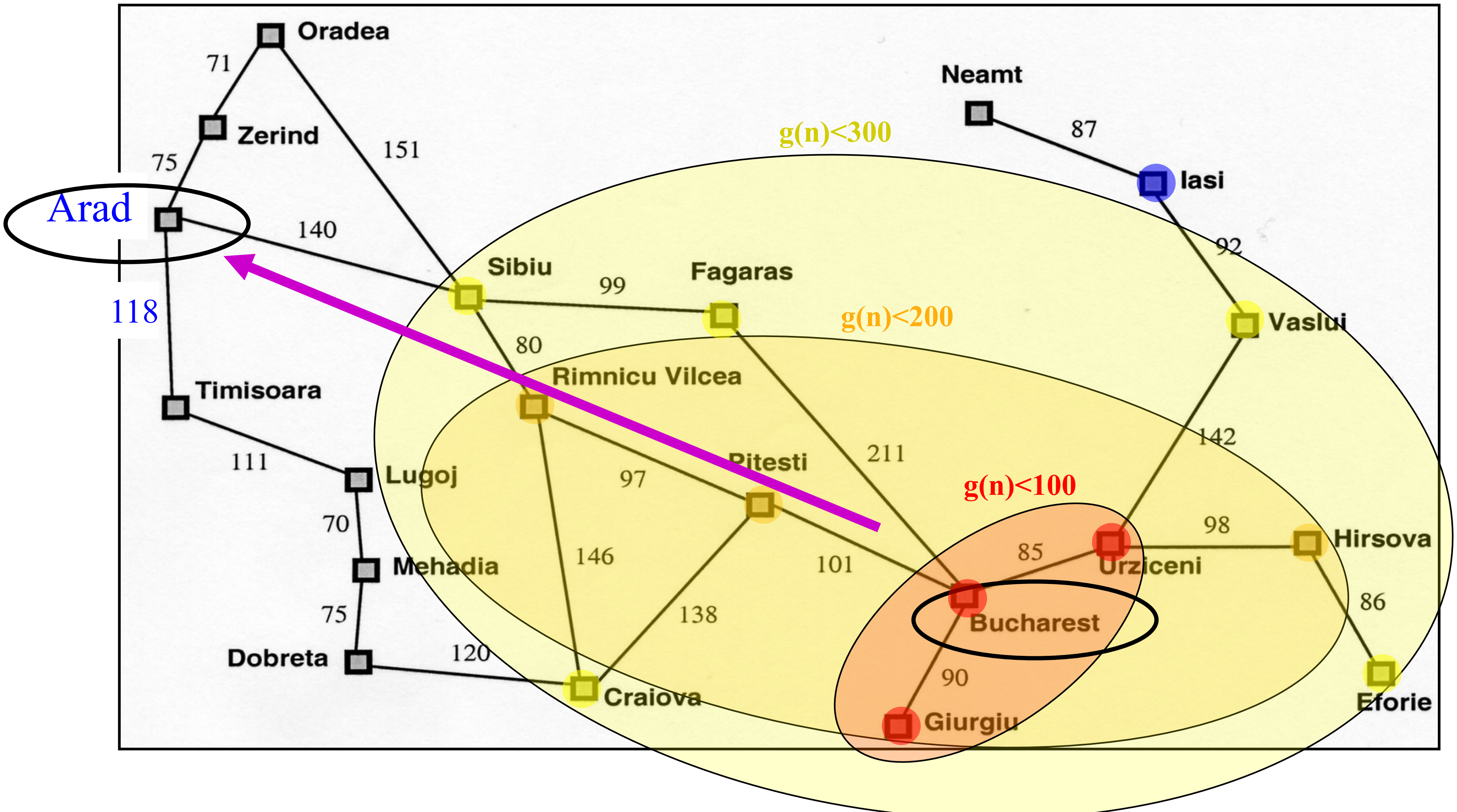
Uninformed Search

- **Briefly: Bidirectional Search**
- **"Uniform Cost" Search (UCS)**

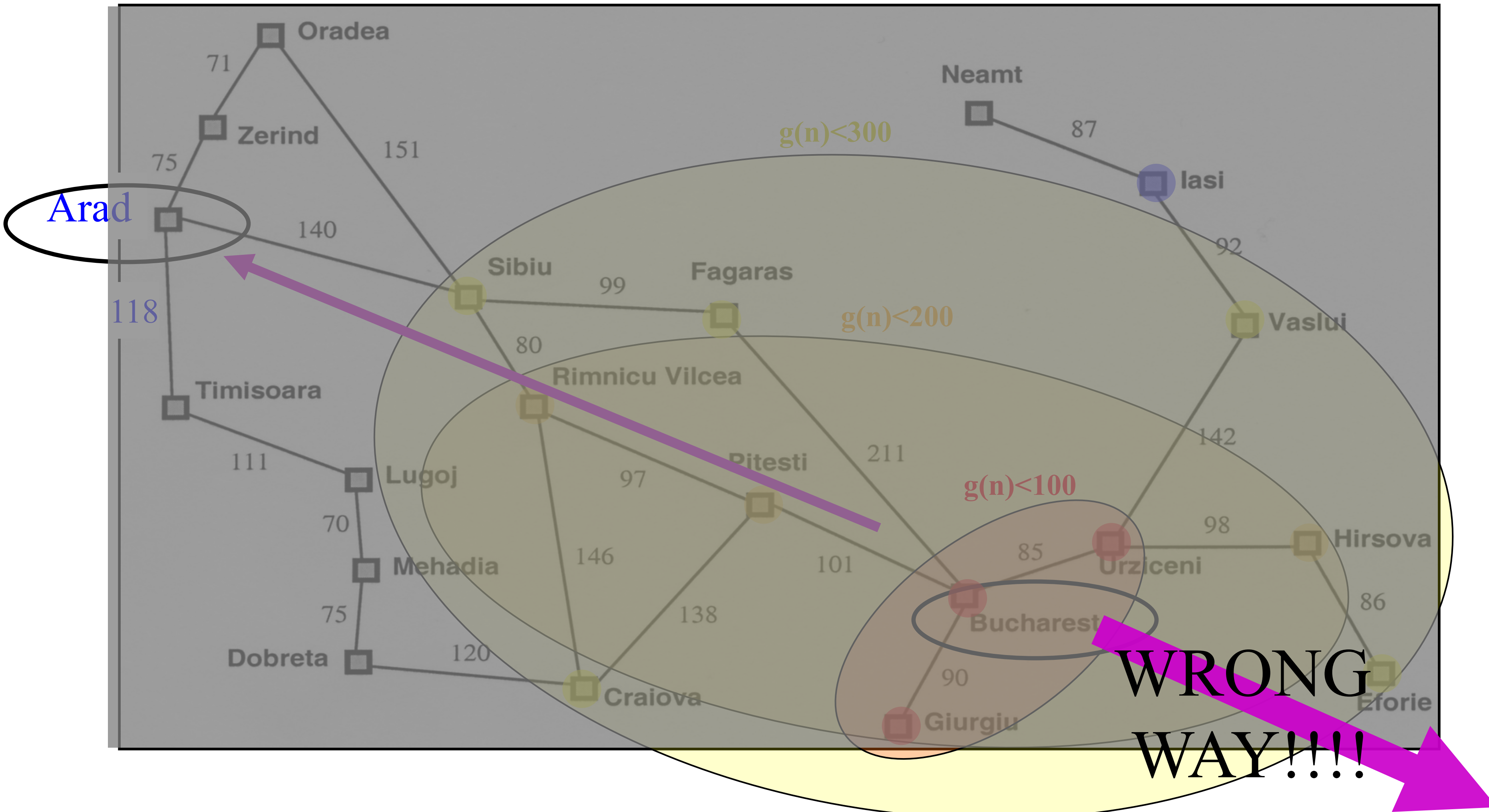
Informed Search

- ***Introduction to Informed search***
  - Heuristics
- **1st attempt: Greedy Best-first search**

Is Uniform Cost Search the best we can do?
Consider finding a route from Bucharest to Arad..



Is Uniform Cost Search the best we can do?
Consider finding a route from Bucharest to Arad..



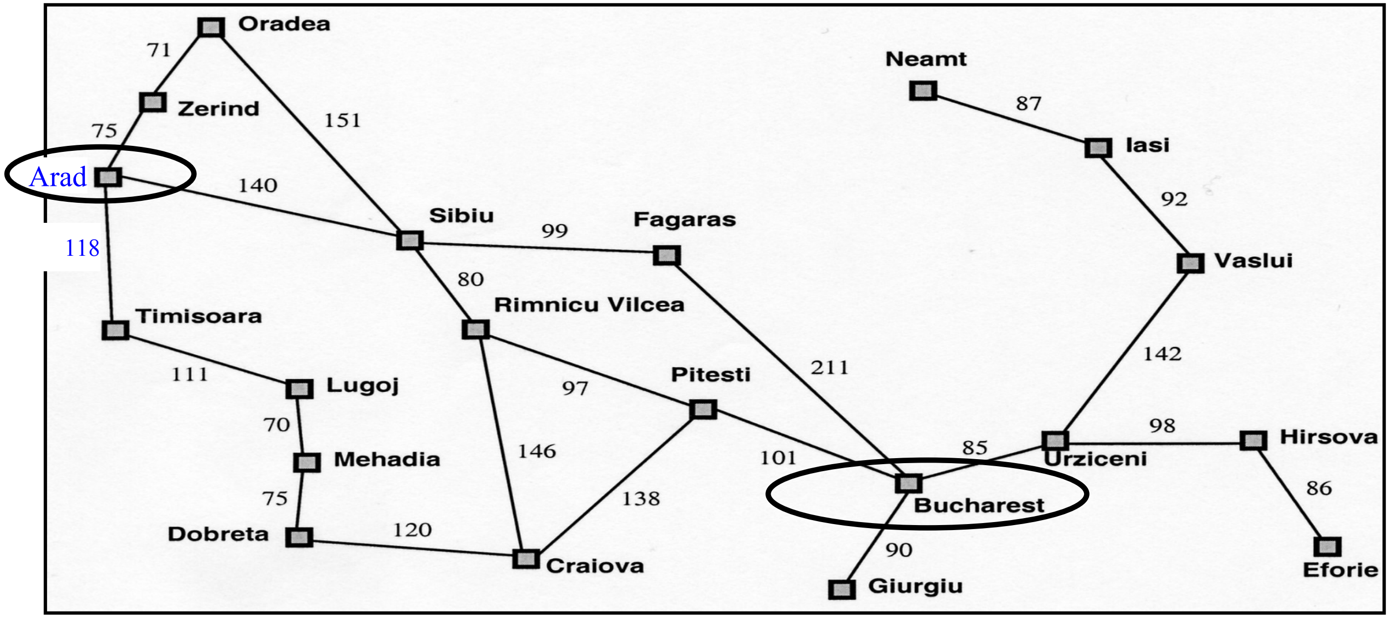
A Better Idea…

- **Node expansion based on *an estimate* which *includes distance to the goal***

- **General approach of informed search:**
  - *Best-first search*: node selected for expansion based on an *evaluation function* $f(n)$
    - —$f(n)$ includes *estimate* of distance to goal ***(new idea!)***

- **Implementation: Sort frontier queue by this new $f(n)$.**
  - Special cases: greedy search, ***A* search***

Simple, useful estimate *heuristic*: *straight-line distances*



Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	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

Heuristic (estimate) functions

Heureka! ---Archimedes

[dictionary] *"A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood."*

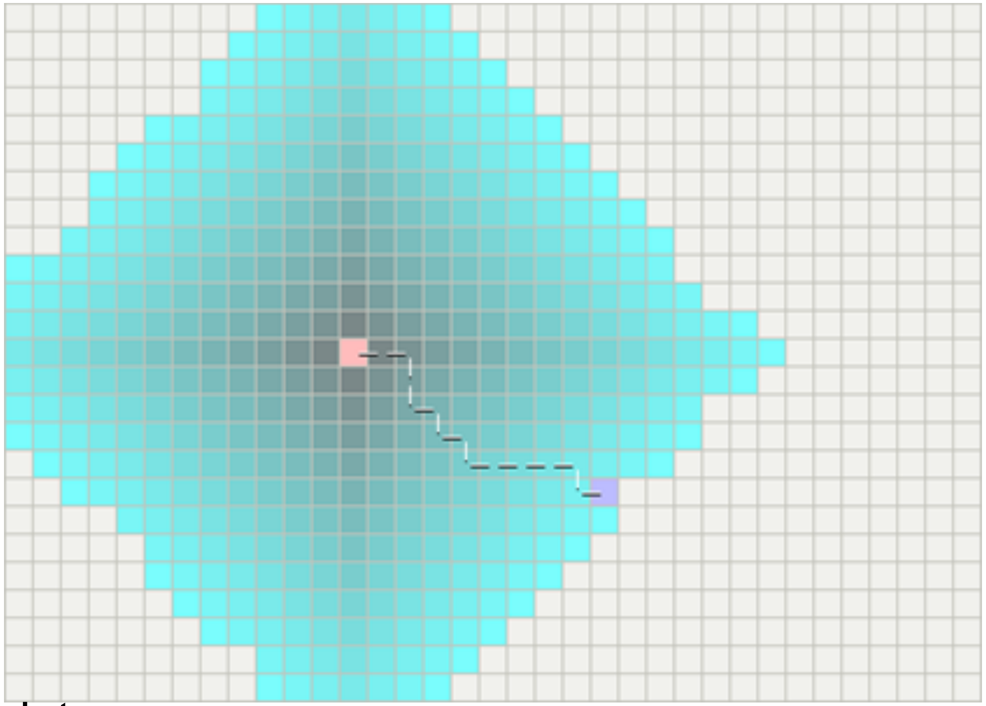
Heuristic knowledge **is useful, but not necessarily correct.**

Heuristic algorithms **use heuristic knowledge to solve a problem.**

A ***heuristic function*** $h(n)$ **takes a state** n **and returns an** ***estimate*** **of the distance from** n **to the goal.**

(graphic: http://hyperbolegames.com/2014/10/20/eureka-moments/)

Breadth First for Games, Robots, …



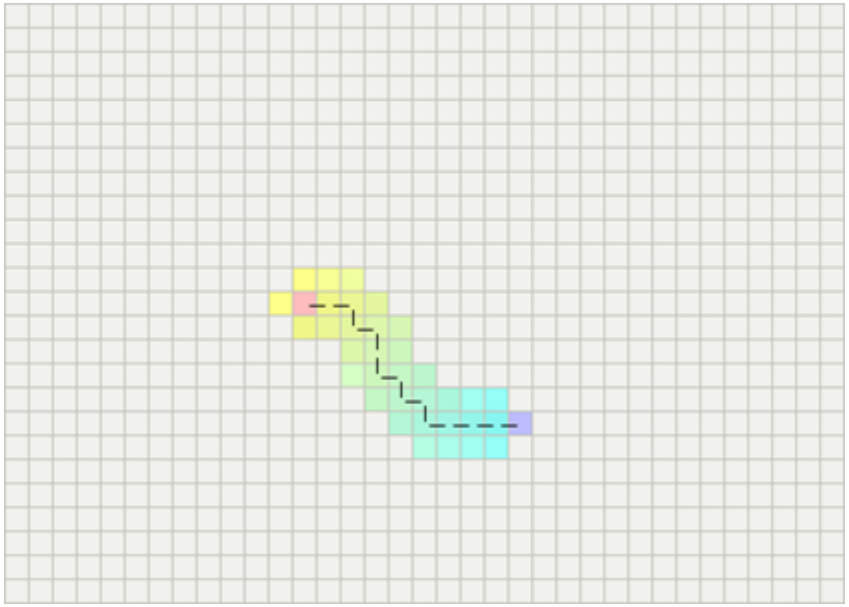
- **Pink: Starting Point**
- **Blue: Goal**
- **Teal: Scanned squares**
  - Darker: Closer to starting point…

Graphics from

http://theory.stanford.edu/~amitp/GameProgramming/

(A great site for practical AI & game Programming

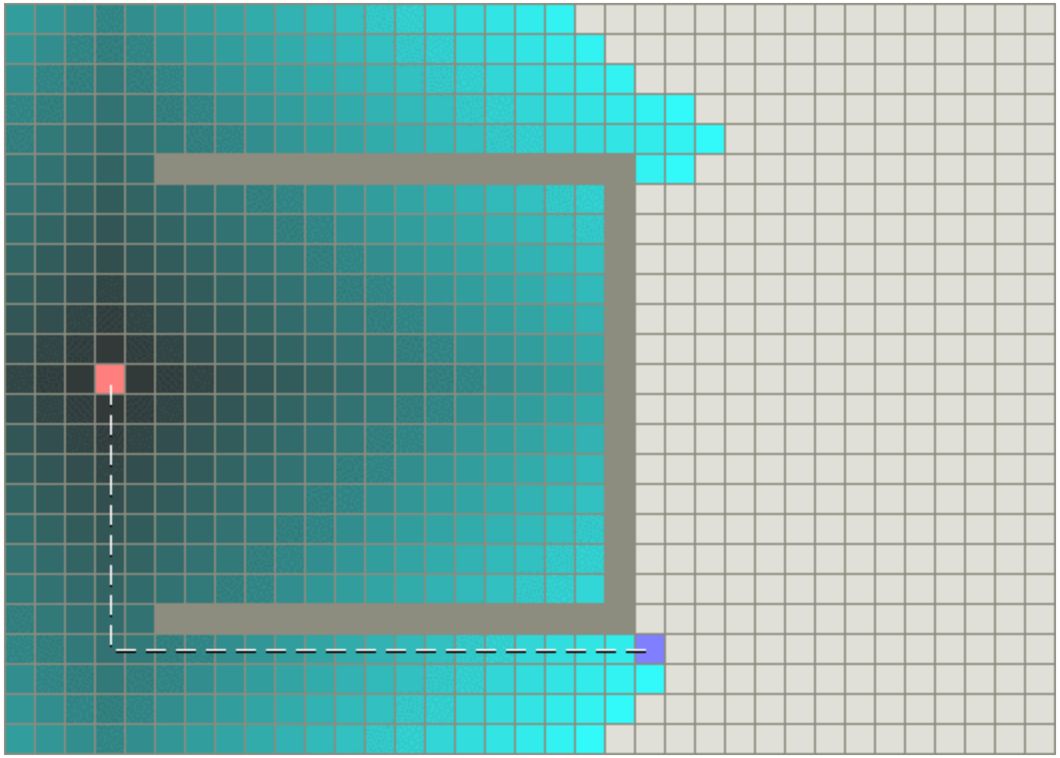
vs. an optimal *informed search* algorithm (A*)



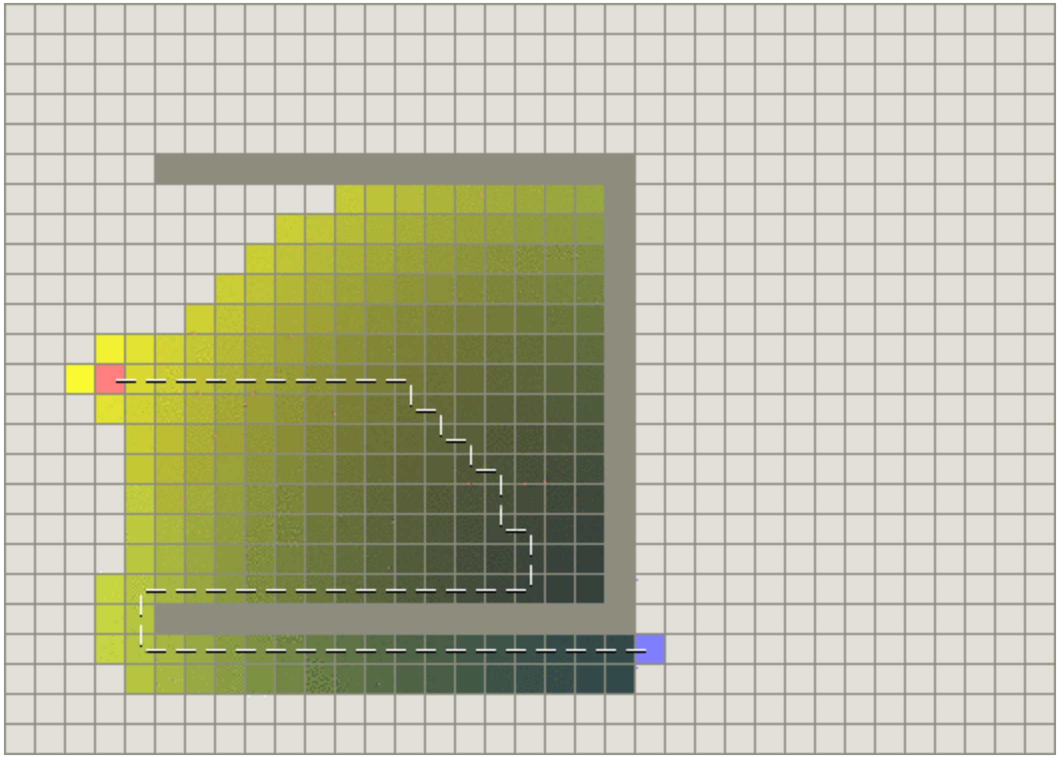
- We add a *heuristic estimate* of distance to the goal

- Yellow: examined nodes with *high* *estimated* distance
- Blue: examined nodes with *low* *estimated* distance

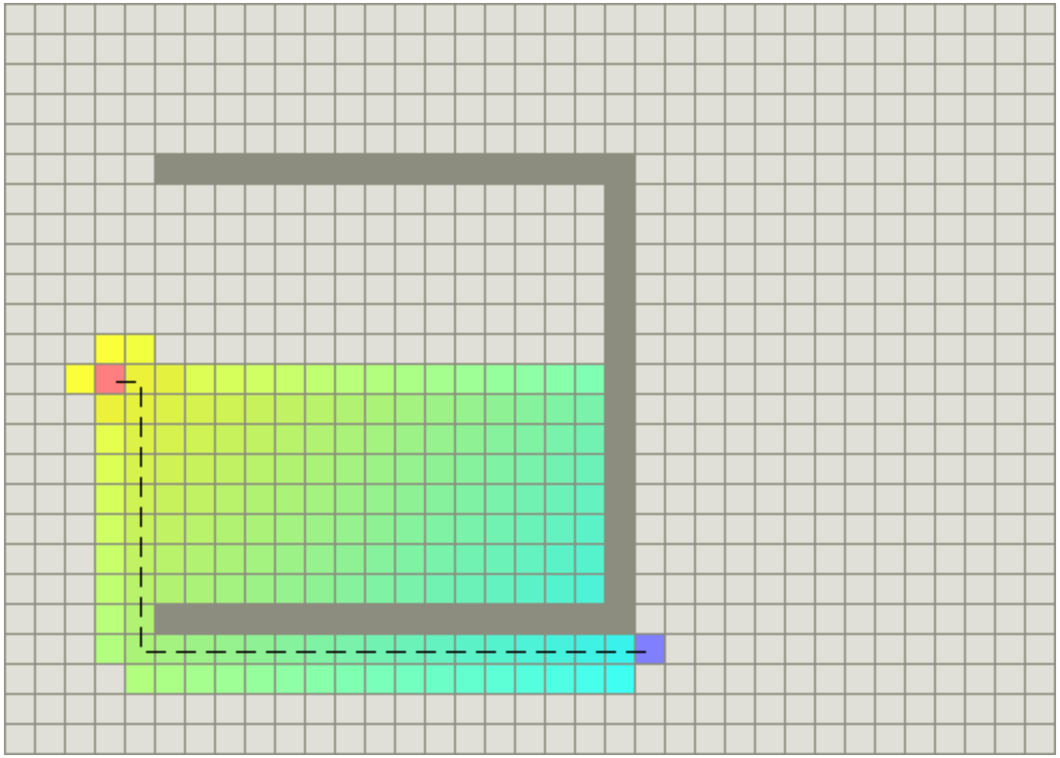
Breadth first in a world with obstacles



Greedy best-first search in a world with obstacles



Informed search (A*) in a world with obstacles



Outline for today's lecture

Uninformed Search

- **Briefly: Bidirectional Search**
- **"Uniform Cost" Search (UCS)**

Informed Search

- **Introduction to Informed search**
  - Heuristics
- ***1st attempt: Greedy Best-first search (AIMA 3.5.1)***

Review: Best-first search

Basic idea:

- ***select node for expansion*** **with minimal *evaluation function $f(n)$***
  - where $f(n)$ is some function that includes ***estimate heuristic*** $h(n)$ of the remaining distance to goal

- **Implement using priority queue**
- **Exactly UCS with $f(n)$ replacing $g(n)$**

Greedy best-first search: $f(n) = h(n)$

- **Expands the node that *is estimated* to be closest to goal**
- **Completely ignores *g(n):* the cost to get to *n***
- **Here, $h(n) = h_{SLD}(n)$ = straight-line distance from ` to Bucharest**

Greedy best-first search example

Frontier
queue:

Arad 366

- **Initial State = Arad**
- **Goal State = Bucharest**

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	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 example

Frontier queue:

Sibiu 253

Timisoara 329

Zerind 374

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	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 example

Frontier queue:

Fagaras 176

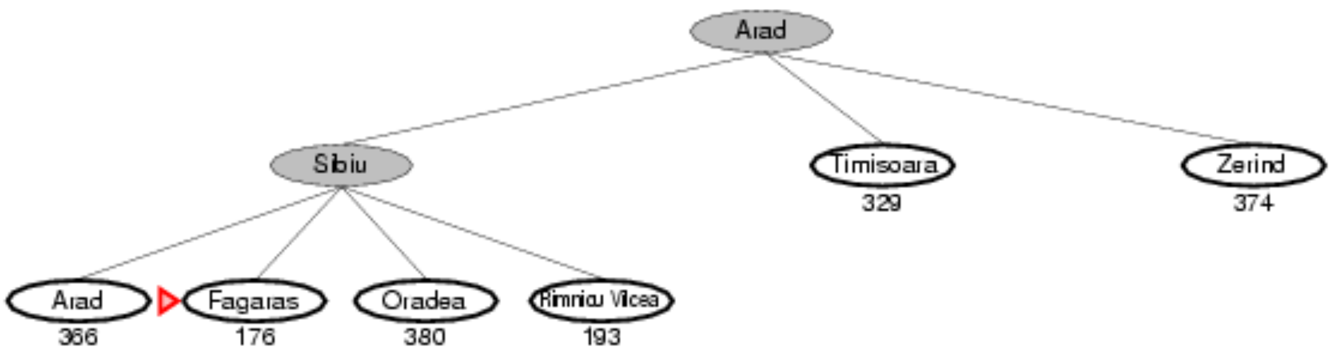
Rimnicu Vilcea 193

Timisoara 329

Arad 366

Zerind 374

Oradea 380



Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	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 example

Frontier queue:

Bucharest 0

Rimnicu Vilcea 193

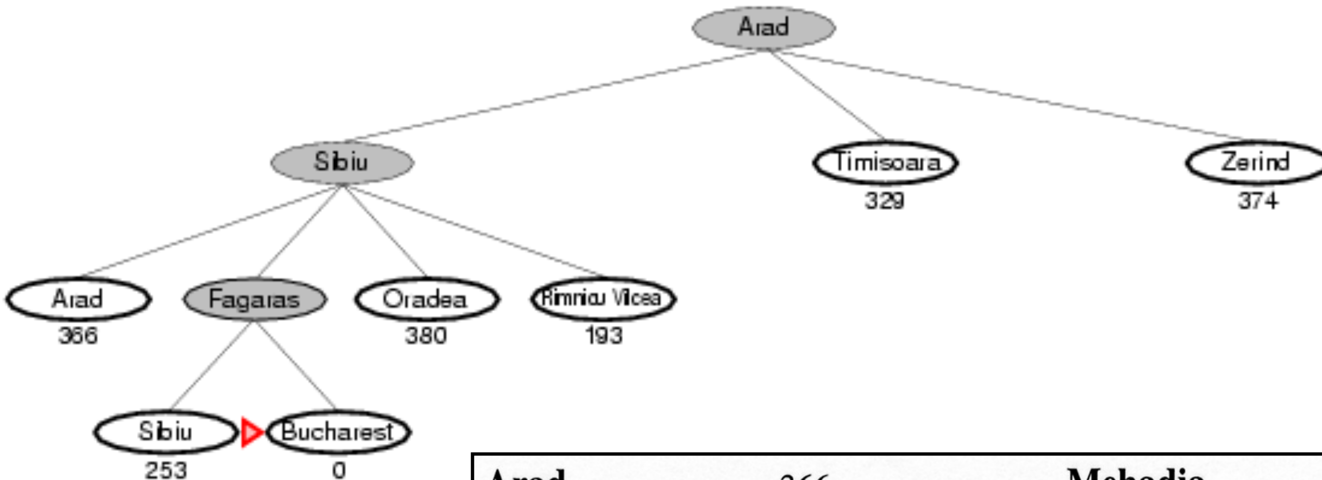
Sibiu 253

Timisoara 329

Arad 366

Zerind 374

Oradea 380

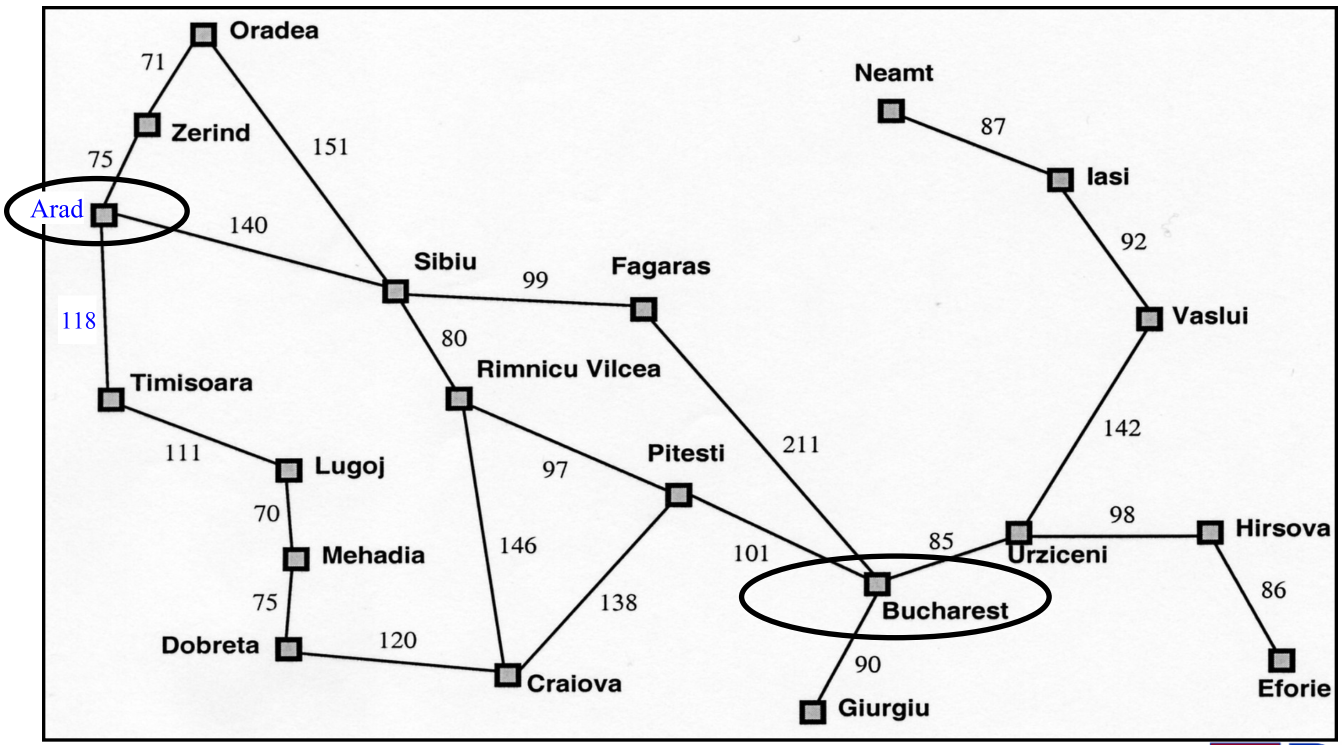


Goal reached !!

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	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

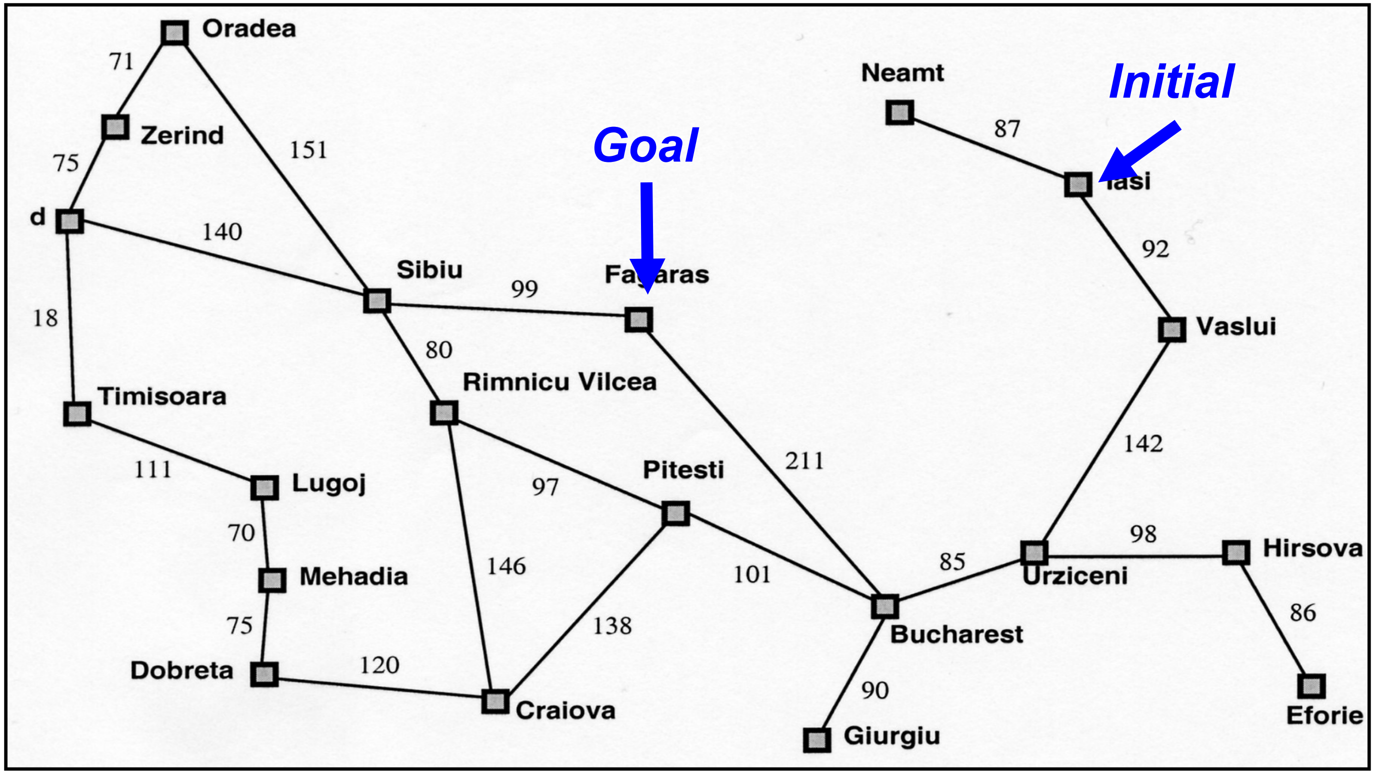
Properties of greedy best-first search

- ***Optimal?***
  - No!
    - —Found: *Arad → Sibiu → Fagaras → Bucharest (450km)*
    - —Shorter: *Arad → Sibiu → Rimnicu Vilcea → Pitesti → Bucharest (418km)*



Properties of greedy best-first search

- ***Complete?***
  - No – can get stuck in loops,
  - e.g., Iasi → Neamt → Iasi → Neamt →…



Properties of greedy best-first search

- ***Complete?*** **No – can get stuck in loops,**
  - e.g., Iasi → Neamt → Iasi → Neamt → …
- ***Time?*** ***$O(b^m)$* – *worst case* (like Depth First Search)**
  - But a good heuristic can give dramatic improvement of *average cost*
- ***Space?*** ***$O(b^m)$* – priority queue, so worst case: keeps all (unexpanded) nodes in memory**
- ***Optimal?*** **No**

IF TIME

- *Optimal informed search: A* (AIMA 3.5.2)*

A* search

- **Best-known form of best-first search.**
- **Key Idea: avoid expanding paths that are already expensive, but expand most promising first.**
- ***Simple idea*: $f(n)=g(n) + h(n)$**
  - $g(n)$ the actual cost (so far) to *reach* the node
  - $h(n)$ estimated cost to *get from the node to the goal*
  - $f(n)$ estimated *total cost* of path through *n* to goal
- **Implementation: Frontier queue as priority queue by increasing $f(n)$ *(as expected…)***

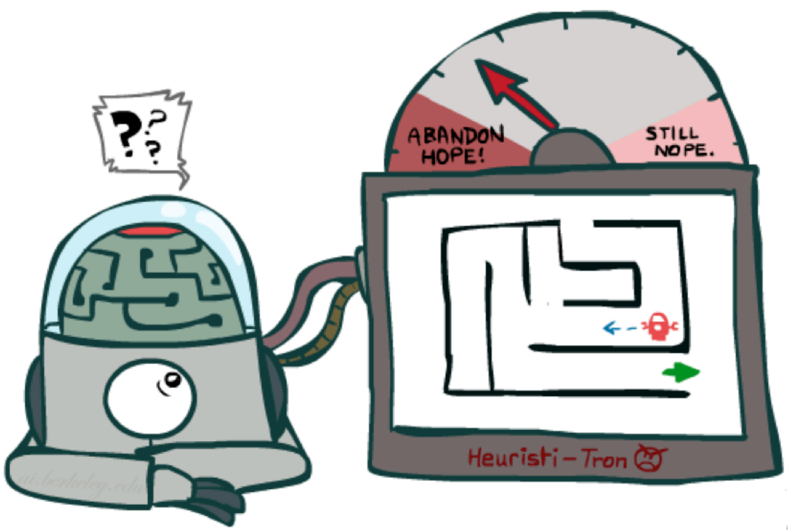
Key concept: Admissible heuristics

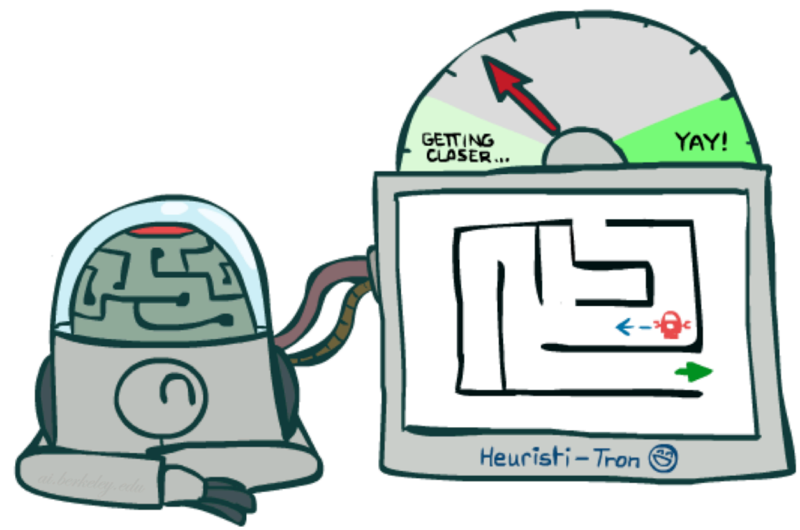
- **A heuristic $h(n)$ is *admissible* if it *never overestimates* the cost to reach the goal; i.e. it is *optimistic***
  - Formally: $\forall n$, n a node:
    1. $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from n
    2. $h(n) \geq 0$ so $h(G)=0$ for any goal G.

- ***Example:* $h_{SLD}(n)$ never overestimates the actual road distance**

Theorem: If $h(n)$ is *admissible*, A* using Tree Search is *optimal*

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

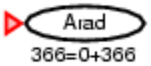
Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Penn
University of Pennsylvania

A* search example

Frontier queue:

Arad 366



A* search example

Frontier queue:

Sibiu 393

Timisoara 447

Zerind 449

We add the three nodes we found to the Frontier queue.

We sort them according to the g()+h() calculation.

A* search example

Frontier queue:

Rimricu Vicea 413

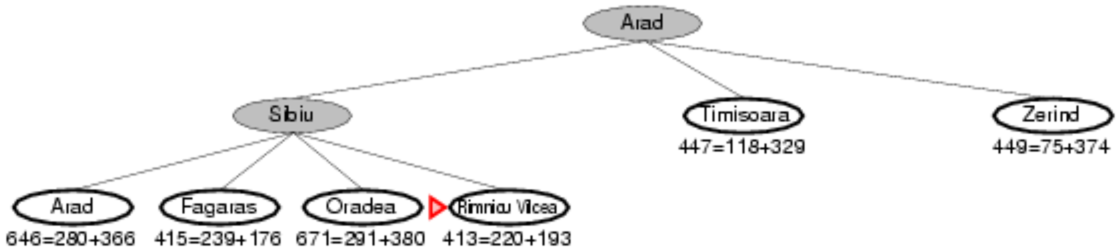
Fagaras 415

Timisoara 447

Zerind 449

Arad 646

Oradea 671



When we expand Sibiu, we run into Arad again. Note that we've already expanded this node once; but we still add it to the Frontier queue again.

A* search example

Frontier queue:

Fagaras 415

Pitesti 417

Timisoara 447

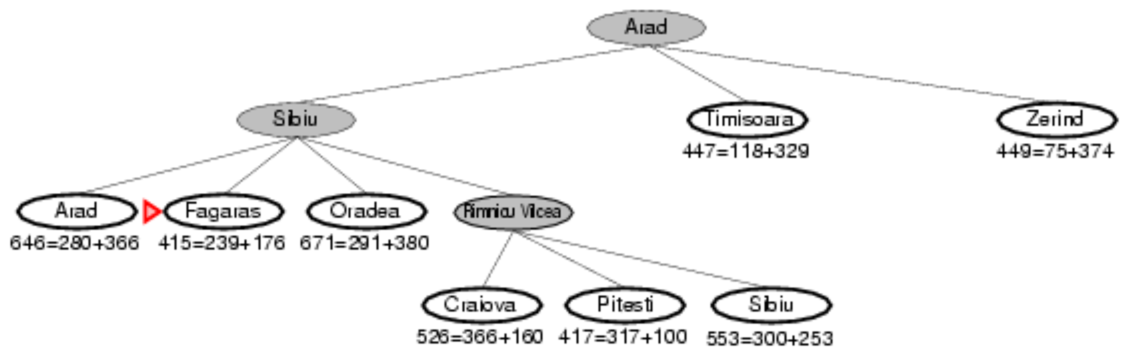
Zerind 449

Craiova 526

Sibiu 553

Arad 646

Oradea 671



We expand Rimricu Vicea.

A* search example

Frontier queue:

Pitesti 417

Timisoara 447

Zerind 449

Bucharest 450

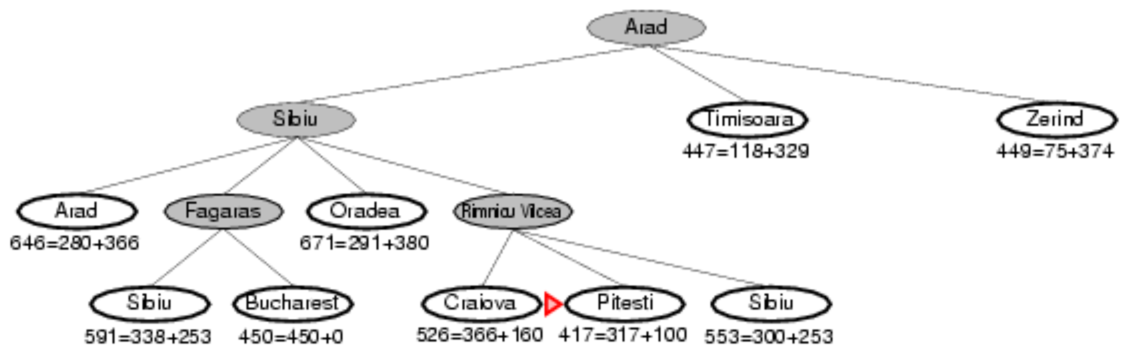
Craiova 526

Sibiu 553

Sibiu 591

Arad 646

Oradea 671



When we expand Fagaras, we find Bucharest, but we're not done. The algorithm doesn't end until we "expand" the goal node – it has to be at the top of the Frontier queue.

A* search example

Frontier queue:

Bucharest 418

Timisoara 447

Zerind 449

Bucharest 450

Craiova 526

Sibiu 553

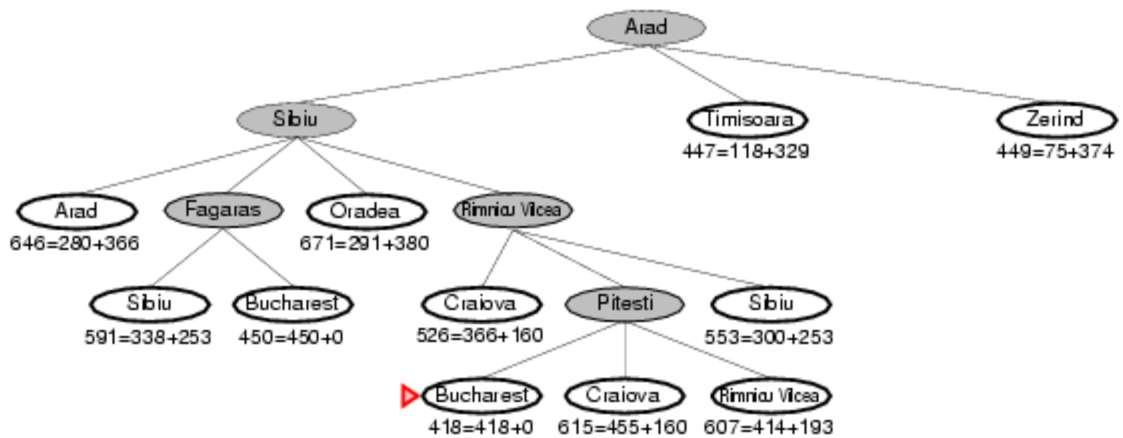
Sibiu 591

Rimricu Vicea 607

Craiova 615

Arad 646

Oradea 671



Note that we just found a better value for Bucharest!

Now we expand this better value for Bucharest since it's at the top of the queue.

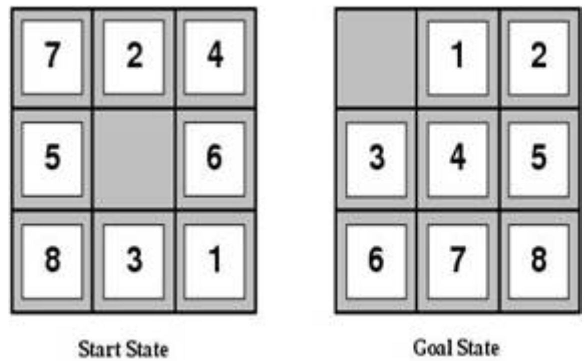
We're done and we know the value found is optimal!

Outline for today's lecture

Informed Search

- **Optimal informed search: A***
- ***Creating good heuristic functions (AIMA 3.6)***
- **Hill Climbing**

Heuristic functions



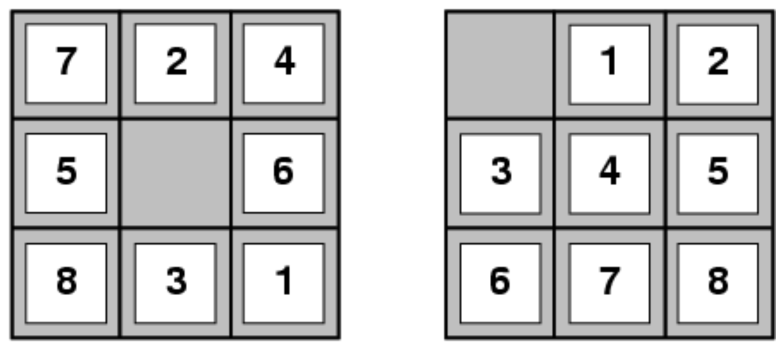
- For the 8-puzzle
  - Avg. solution cost is about 22 steps
    - —(branching factor ≤ 3)
  - Exhaustive search to depth 22: ***3.1×10^{10} states***
  - A good heuristic function can reduce the search process

Example Admissible heuristics

For the 8-puzzle:

- ***$h_{oop}(n)$*** **= number of out of place tiles**

- ***$h_{md}(n)$*** **= total Manhattan distance (i.e., # of moves from desired location of each tile)**



Start State Goal State

- **$h_{oop}(S) = 8$**
- **$h_{md}(S) = 3+1+2+2+2+3+3+2 = 18$**

Relaxed problems

- A problem with fewer restrictions on the actions than the original is called a *relaxed problem*
- ***The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem***
- If the rules of the 8-puzzle are relaxed so that a tile can move *anywhere*, then $h_{oop}(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to *any adjacent square,* then $h_{md}(n)$ gives the shortest solution

Defining Heuristics: *h(n)*

- **Cost of an exact solution to a *relaxed* problem (fewer restrictions on operator)**

- **Constraints on *Full* Problem:**

  A tile can move from square A to square B *if* A is adjacent to B *and* B is blank.

  - Constraints on *relaxed* problems:
    - —A tile can move from square A to square B *if* A is adjacent to B. (h_{md})
    - —A tile can move from square A to square B *if* B is blank.
    - —A tile can move from square A to square B. (h_{oop})

Dominance: A metric on *better* heuristics

- **If $h_2(n) \geq h_1(n)$ for all *n* (both admissible)**
  - then h_2 *dominates* h_1
- **So h_2 is optimistic, but more accurate than h_1**
  - h_2 is therefore better for search
  - Notice: h_{md} dominates h_{oop}

- **Typical search costs (average number of nodes expanded):**
  - *d=12* Iterative Deepening Search = 3,644,035 nodes
    $A^*(h_{oop})$ = 227 nodes
    $A^*(h_{md})$ = 73 nodes
  - *d=24* IDS = too many nodes
    $A^*(h_{oop})$ = 39,135 nodes
    $A^*(h_{md})$ = 1,641 nodes

The best and worst admissible heuristics

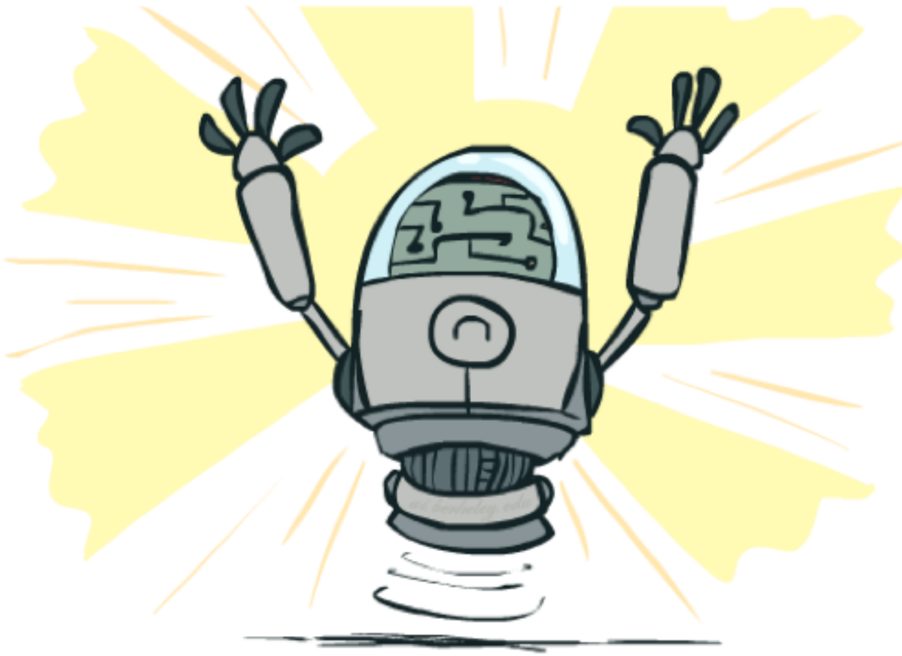
$h^*(n)$ - the (unachievable) Oracle heuristic

- $h^*(n)$ = the true distance from the root to n

$h_{\text{we're here already}}(n) = h_{\text{teleportation}}(n) = 0$

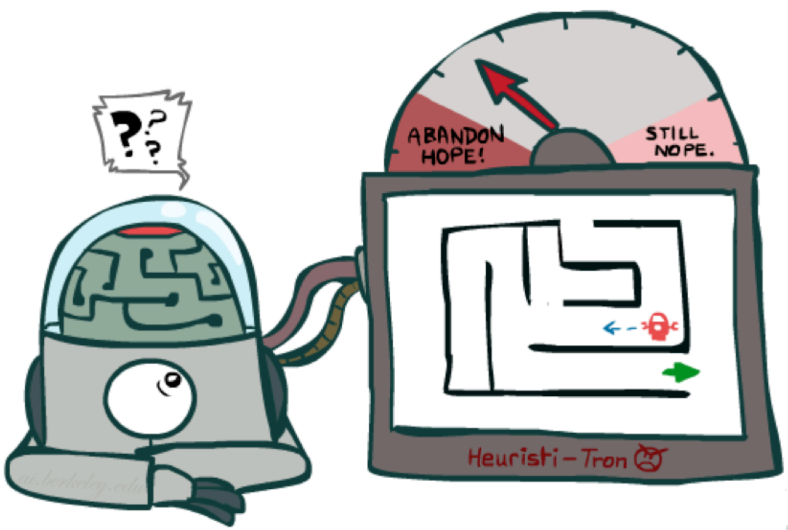
- **Admissible: both yes!!!**
- **$h^*(n)$ *dominates all other heuristics***
- **$h_{\text{teleportation}}(n)$ *is dominated by all heuristics***

Optimality of A* Tree Search

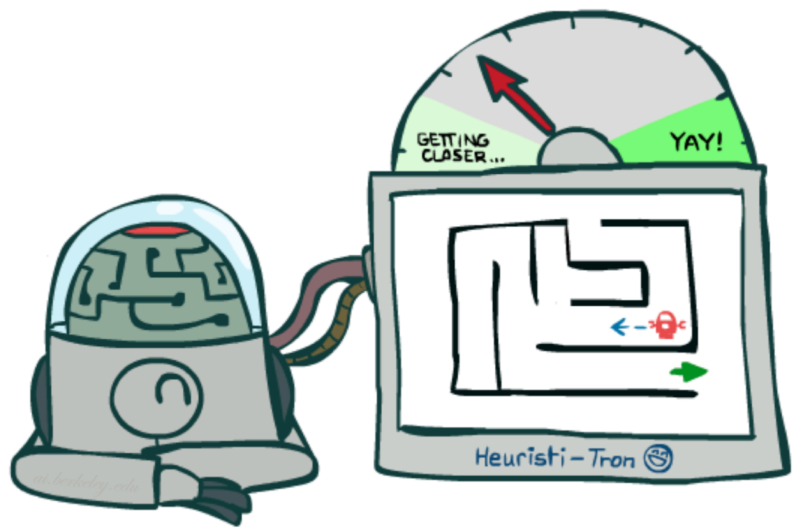


Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

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

Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

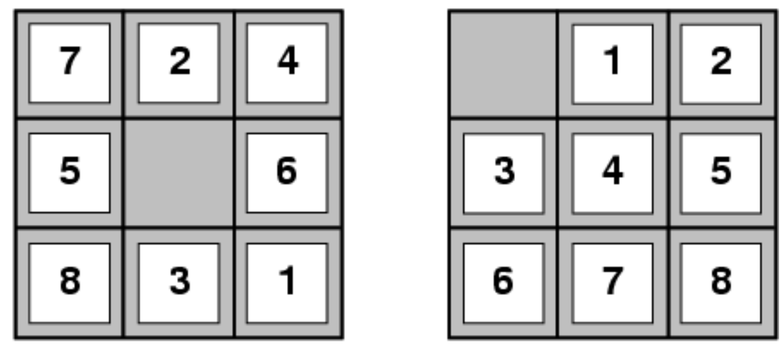
Admissible Heuristics

- 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

- **Is Manhattan Distance admissible?**



Start State Goal State

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

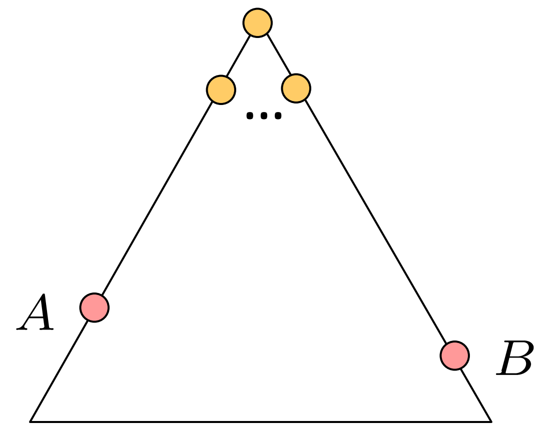
Optimality of A* Tree Search

Assume:

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

Claim:

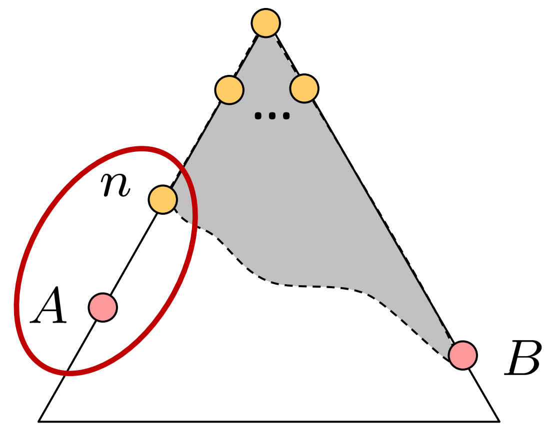
- **A will exit the fringe before B**



Optimality of A* Tree Search: Blocking

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)



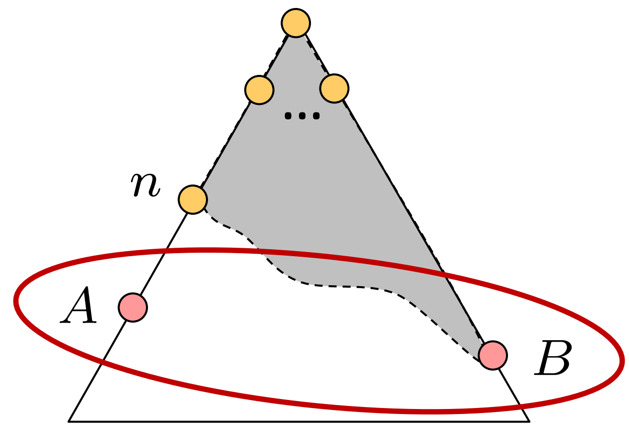
$f(n) = g(n) + h(n)$	Definition of f-cost
$f(n) \leq g(A)$	Admissibility of h
$g(A) = f(A)$	h = 0 at a goal

Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Optimality of A* Tree Search: Blocking

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

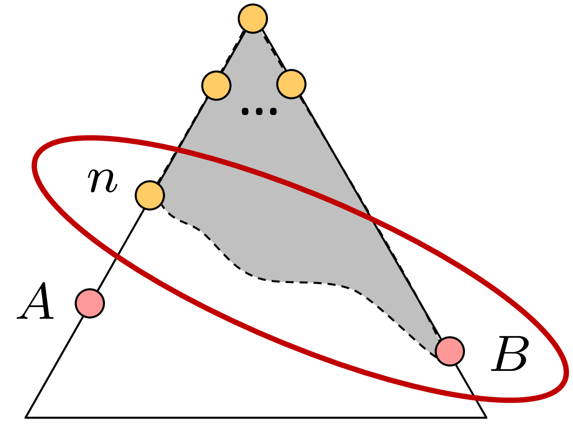
$f(A) < f(B)$ h = 0 at a goal

Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Optimality of A* Tree Search: Blocking

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**



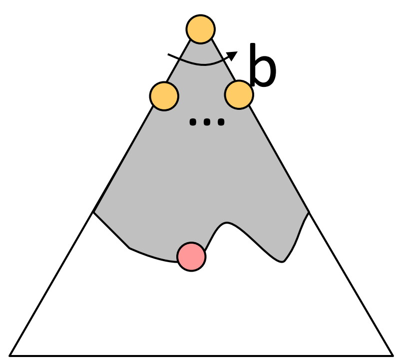
$$f(n) \leq f(A) < f(B)$$

Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

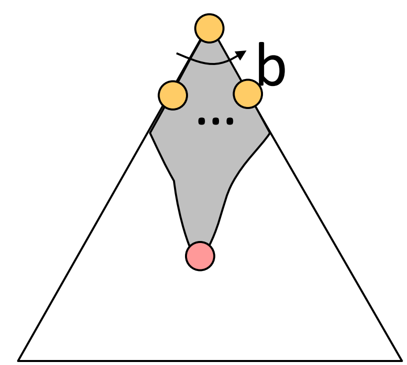
Properties of A*

Properties of A*

Uniform-Cost



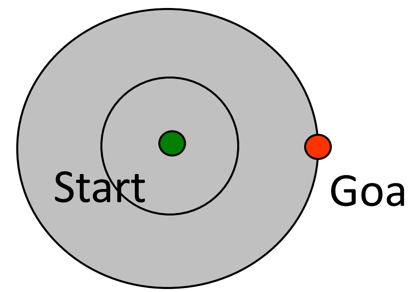
A*



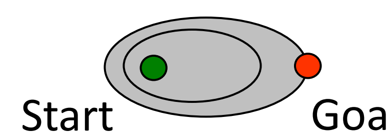
Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

UCS vs A* Contours

- **Uniform-cost expands equally in all "directions"**



- **A* expands mainly toward the goal, but does hedge its bets to ensure optimality**



Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

A* Applications

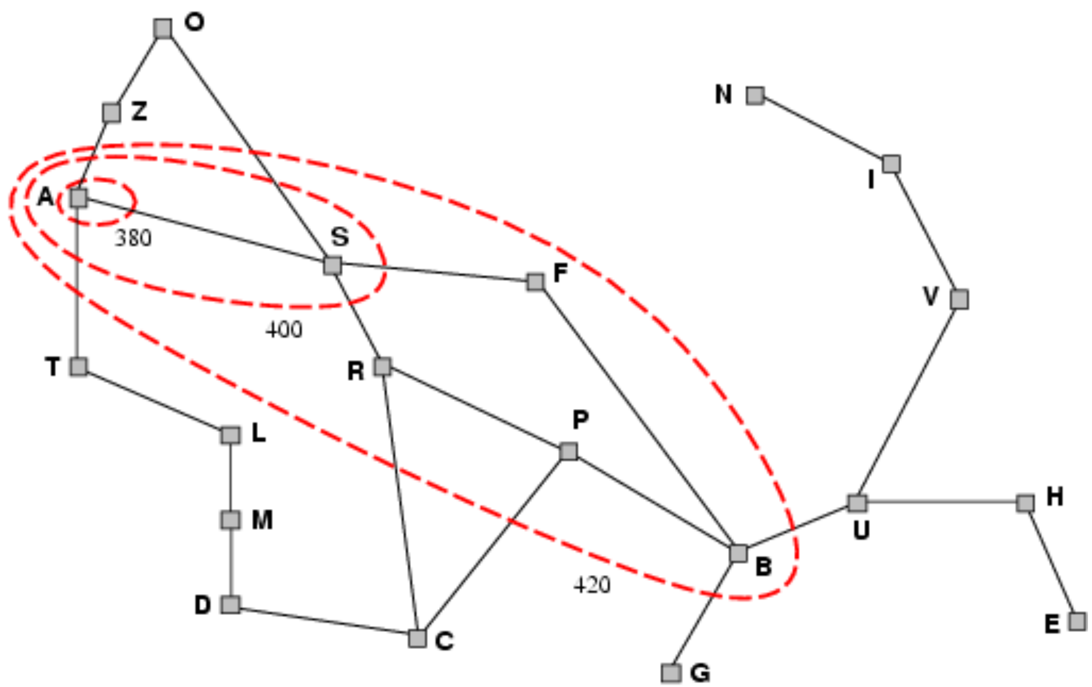
- Video games
- Pathing / routing problems (A* is in your GPS!)
- Resource planning problems
- Robot motion planning
- …

Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Optimality of A* (intuitive)

- Lemma: A* expands nodes on frontier in order of increasing f value

- Gradually adds "f-contours" of nodes
- Contour i has all nodes with $f=f_i$, where $f_i < f_{i+1}$
- (After all, A* is just a variant of uniform-cost search….)



Optimality of A^* using Tree-Search (proof idea)

- **Lemma: A^* expands nodes on frontier in order of increasing *f* value**
- **Suppose some suboptimal goal G_2 *(i.e a goal on a suboptimal path)* has been generated and is in the frontier along with an optimal goal G.**

Must prove: $f(G_2) > f(G)$

(Why? Because if $f(G_2) > f(n)$, then G_2 will never get to the front of the priority queue.)

Proof:

1. $g(G_2) > g(G)$ — since G_2 is suboptimal
2. $f(G_2) = g(G_2)$ — since $f(G_2)=g(G_2)+h(G_2)$ & $h(G_2) = 0$, since G_2 is a goal
3. $f(G) = g(G)$ — similarly
4. $f(G_2) > f(G)$ — from 1,2,3

Also must show that G is added to the frontier before G_2 is expanded – see AIMA for argument in the case of Graph Search

A* search, evaluation

- **Completeness: YES**
  - Since bands of increasing *f* are added
  - As long as *b* is finite
    - (guaranteeing that there aren't infinitely many nodes n with *f (n) < f(G))*
- **Time complexity: Same as UCS worst case**
  - Number of nodes expanded is still exponential in the length of the solution.
- **Space complexity: Same as UCS worst case**
  - It keeps all generated nodes in memory so exponential
  - Hence space is the major problem not time
- **Optimality: YES**
  - Cannot expand f_{i+1} until f_i is finished.
  - A* expands all nodes with $f(n) < f(G)$
  - A* expands one node with $f(n)=f(G)$
  - A* expands no nodes with $f(n)>f(G)$

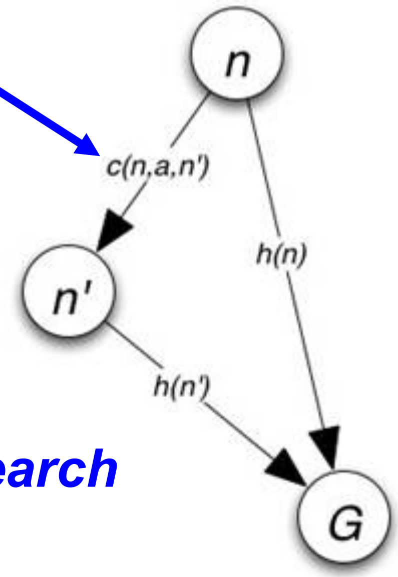
Consistency

- A heuristic is ***consistent*** if

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

- Consistency enforces that h(n) is optimistic

Cost of getting from n to n' by any action a



Theorem: if h(n) is consistent, *A* using Graph-Search is optimal*

See book for details