Sept 14: A Star Search

Consider going from Arad to Bucharest. How does (open alphabetically)

1 BFS route us?

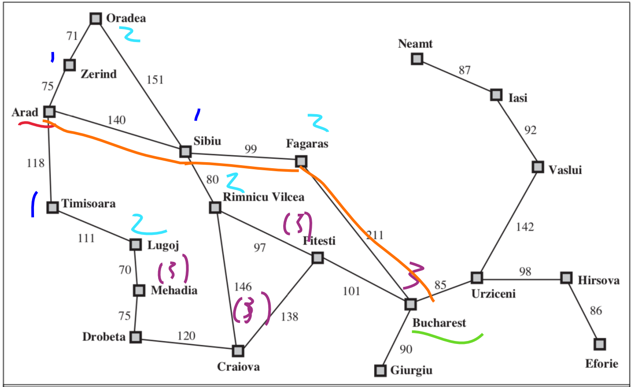


Figure 3.2 A simplified road map of part of Romania.

Sept 14: A Star Search

Consider going from Arad to Bucharest. How does

2 DFS route us?

•: alphabetic

•: reverse alphabetic

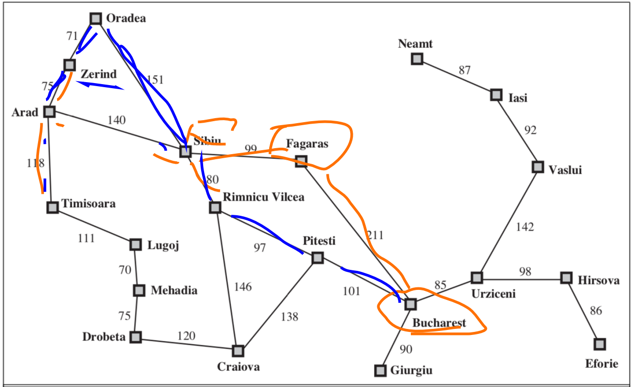


Figure 3.2 A simplified road map of part of Romania.

Sept 14: A Star Search

Consider going from Arad to Bucharest. How does

3 UCS route us?

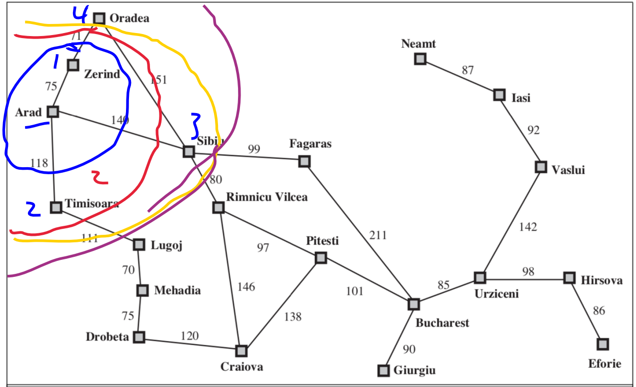


Figure 3.2 A simplified road map of part of Romania.

Zerind: 75

Timisoara: 118

Sibiu: 140

Oradea: 75+71

Announcements and To-Dos

Announcements:

1. Make sure you check Piazza (`piazza.com/colorado/fall2020/csci3202`) for common HW questions and concerns!

Last time we learned:

1. Uniform Cost searching.

To do:

1. HW02 about ready, posted tomorrow (due Sep 25)!

UCS Recap:

Last time we talked about **Uniform Cost Searching** (UCS).

1. UCS search can take into account edge weights. ☺
2. At each iterative step, expand the cheapest-to-reach frontier node first (lowest path cost).
3. In addition to the stacks/queues (DFS/BFS), UCS must track the path cost to reach each state on the frontier. This leads to a *priority* queue.
4. Goal test: when a goal state node is *selected* for expansion. This is different than BFS/DFS, which stopped when goal state reached frontier!
5. UCS can get stuck or struggle if there are sequences of no-cost actions: it will continually have to expand that branch of the tree. We think of the cost of UCS as a cost of exploring within cost *contours* of the sample space.

Upgrading UCS

The uniform cost searching algorithm was particularly appealing because it was **complete** and **optimal**: it was a more general and powerful form of the also-complete BFS.

It would be advantageous to avoid some of the inefficiencies of the all-directions contour exploration of UCS, however. One way we could do this is by tracking *another* function besides just cost.

1. Consider naming our “choice” of agent action $f(n)$. In UCS, $f(n)$ was a cost estimate: expand the lowest-evaluating state $f(state)$ first.
2. We will consider here *heuristic* functions $h(n)$ for any given state n. These may also include the *estimated* cost or distance from n to a/the goal state.

Informed Search

Zooming out for a second: all 4 of our lectures on search algorithms were base don fundamentally the same concept: wander through the state space in a structured fashion, keeping track of where we've been to avoid duplication. They differ only in their *frontier* strategies.

Uninformed

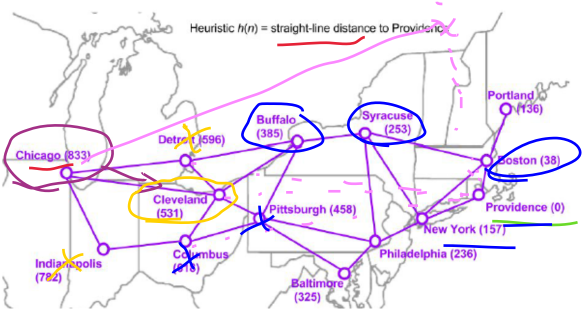
1. **Uniformed Searches** - including UCS/BFS/DFS - explore state spaces starting from the initial node with no knowledge or bias given to the goal nodes. They may be complete and optimal, but also could include our known inefficiencies.
2. **Informed Searches** include some information about where the goal or goals might be. To use these, we require a *heuristic* that estimates how close any given state is to the goal.

Heuristic Searches

One algorithm that relies on heuristics is the **greedy best-first search**. In this algorithm we will not consider a cost function, but instead our decision evaluation function $f(n)$ will just be the estimated distance-to-the goal heurstic $h(n)$. So,

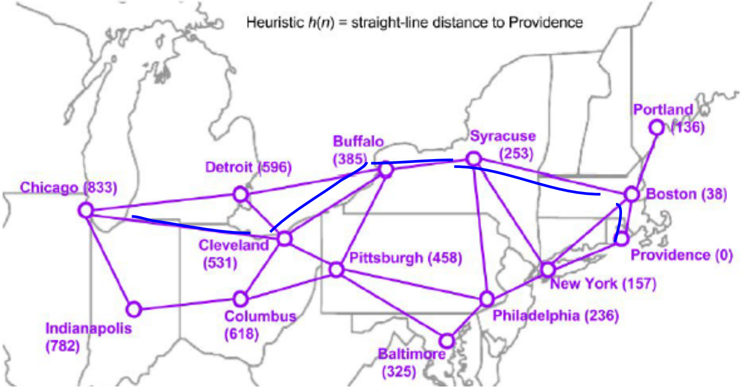
$$f(n) = h(n)$$

- So we need a heuristic $h(n)$.
- One might be "as the crow flies" straight-line distance from n to the goal.
- What would our Chicago-to-Providence route be in this case?



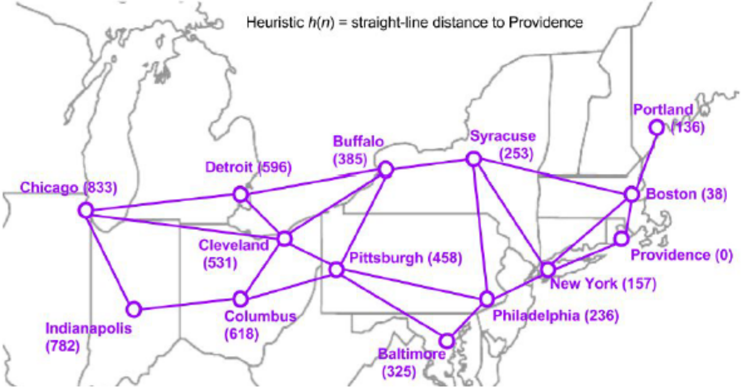
Greedy best-first on a map

What would a Chicago-to-Providence route be in this case?



Greedy best-first on a map

What would a Chicago-to-Providence route be in this case?



Solution:

1. First we choose Cleveland over Detroit and Columbus.
2. The frontier becomes only states bordering Cleveland. We choose Buffalo over Pittsburgh, Columbus, Detroit, and Chicago.
3. We continue to Syracuse, then Boston, then Providence.

Greedy best-first

Is this algorithm **complete**?

Is this algorithm **optimal**?

Why use it at all?

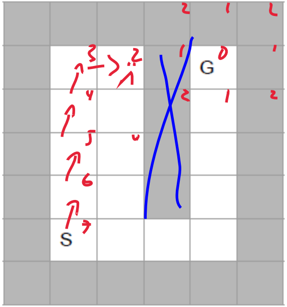
Greedy best-first

Is this algorithm **complete**? No! Consider the examples below.

Is this algorithm **optimal**? No! Consider the examples below.

Why use it at all? *Enormous* potential to save time and memory compared to the complete methods!

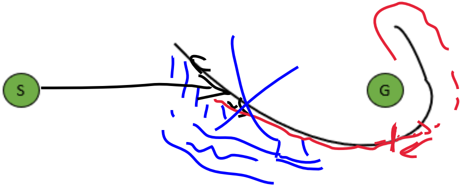
obstacles / unequal steps are issues



maze

rows different
+ # columns "

(Manhattan distance?)



Straight line

Astar Searches

The greedy algorithm was a little silly since it didn't use our edge weights. Maybe we are considering state n with cost-to-reach $g(n)$. The greedy algorithm made the evaluation

$$\underset{\text{decision "score"}}{f(n)} = \underset{\text{goal state heuristic}}{h(n)}$$

based on heuristic h.

UCS unfolded states in a closest-first order, or:

$$\overset{\text{decision}}{f(n)} = \overset{\text{"cost-to-move"}}{g(n)}.$$

An **A*** ("Astar") search uses both h and g.

- As long as the cost-units of h and g are the same (e.g. distance!), it makes a lot of sense to just add them up!
- A^* then uses:

$$f(n) = \overset{\text{distance (true) from current to n.}}{g(n)} + \overset{\text{distance from n to goal}}{\underset{\text{estimated}}{h(n)}}$$

$d(D/E/F, G) \approx 4$

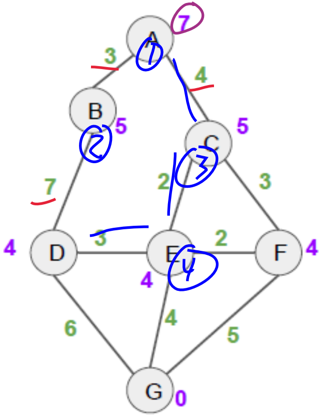
~~$(D: g=10, h=4, f=14)$~~

$\leftarrow: B$

Astar on a graph

Perform an A^* on this graph, with goal to move from A to G. $h(n)$ heuristics are in purple, step costs are in green.)

$(D: g=9, h=4, f=13)$



0 Explored: $\{\}$; Frontier:
$[(A: \underbrace{g=0}_{\text{(total) cost to get to A}}, h=7, f=7)]$.

$f + g = h$

explore

$\{A\}$

frontier

$\left\{ (B: g=3, h=5, \boxed{f=8}), (C: g=4, h=5, f=9) \right\}$

$\{A, B\}$: $\left\{ C: g=4 \quad h=5, f=9 \quad D: g=3+7, h=4, f=14 \right\}$

$\{A, B, C\}$:

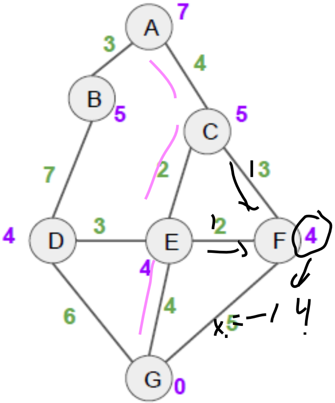
$D: g=10 \quad h=4 \quad f=14$

$E \quad g=6 \quad h=4 \quad f=10$

$F \quad g=7 \quad f=11$

Astar on a graph

Perform an A^* on this graph, with goal to move from A to G. $h(n)$ heuristics are in purple, step costs are in green.)



0 Explored: $\{\}$; Frontier: $[(A:\ g=0,\ h=7,\ f=7)]$.

1 Explored: $\{A\}$; Frontier: $[(B:\ g=3,\ h=5,\ f=8),(C:\ g=4,\ h=5,\ f=9)]$.

2 Explored: $\{A,B\}$; Frontier: $[(C:\ g=4,\ h=5,\ f=9),(D:\ g=10,\ h=4,\ f=14)]$.

3 Explored: $\{A,B,C\}$; Frontier: $[(D:\ g=9,\ h=4,\ f=13),(D:\ g=10,\ h=4,\ f=14),(E,\dots),(F,\dots)]$.

k …We will ultimately choose the same route as UCS: A-C-E-G.

Astar on the map

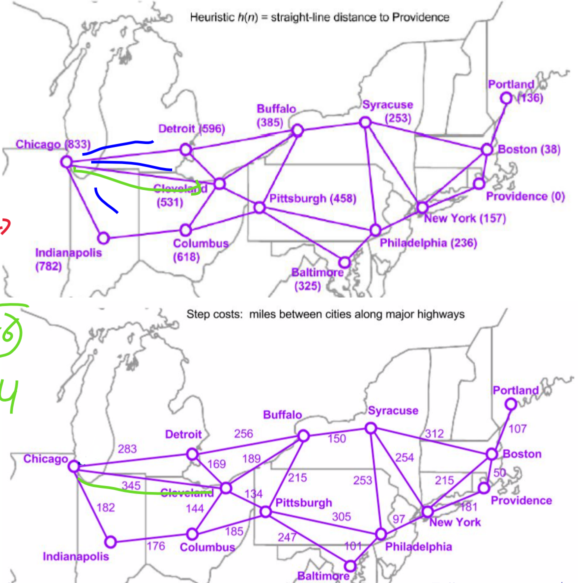
How would A^* handles the Chicago-to-Providence problem, now?

	g (rd dist from CHI)		h (dist city -> Prov) approx	
Det	283	+	596	879
Cle	345	+	531	876
Ind	182	+	782	964

explored

{CLE g=345}

Col, PIT, BUF, DET


Heuristic $h(n)$ = straight-line distance to Providence

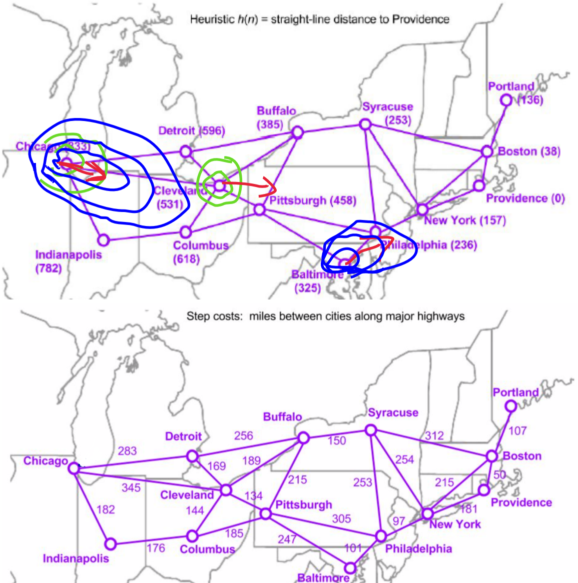
Step costs: miles between cities along major highways

Astar on the map

- UCS
- A^*

How would A^* handles the Chicago-to-Providence problem, now?

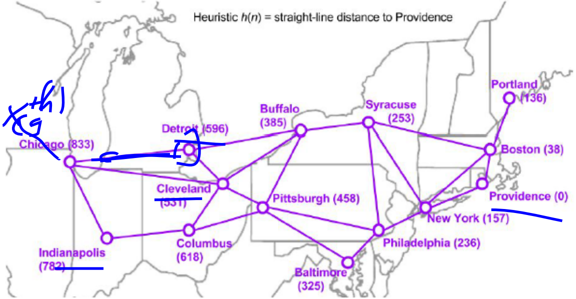
We can think of A^* as similar to contours, but instead of the circular expansion of UCS, the contours will preferentially expand towards the goal G! The better our heuristic actually is, the bigger the "pull" towards the goal state(s) will be.

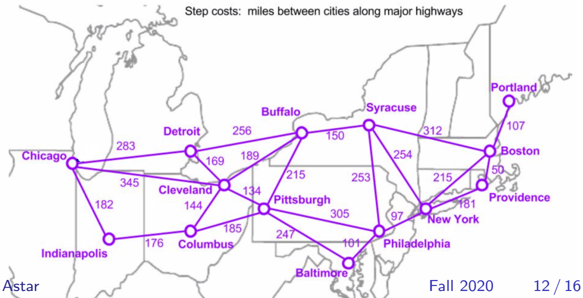


Astar ideas

A few results of A^*'s selection:

1. Contours are stretched.
2. For true optimal cost C^*, A^* will never expand a node with $f(n) > C\overset{A}{*}$, but will expand *all* nodes with $f(n) < C^*$.
3. Example: If we start in Detroit and want to move to New York, we will never expand Chicago: we find a solution before it's appealing. We call this **pruning** Chicago.


Heuristic $h(n)$ = straight-line distance to Providence


Step costs: miles between cities along major highways

Astar Theory

Ok, so A^* uses all the information. Does it do it well? Is it **optimal?** Is it **complete?**

Poll time!

Astar Theory

Ok, so A^* uses all the information. Does it do it well? Is it **optimal?** Is it **complete?**

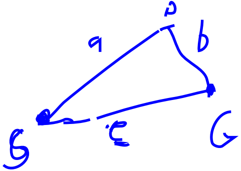
Answer: A^* is optimal on a tree if the heuristic function h is **admissible** and A^* is optimal on a graph if h is **consistent.** Make a mental note: if a search is optimal, it's definitely also complete!

Definition: a heuristic $h(n)$ is *admissible* if it never overestimates the cost to reach the goal. Because $g(n)$ is an actual cost to reach n along a current path, f will never overestimate the cost of a solution *through* n if $g(n)$ is admissible.

est. decision cost

Astar Theory

Ok, so A^* uses all the information. Does it do it well? Is it **optimal?** Is it **complete?**



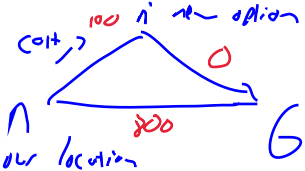
$|a| + |b| \geq |c|$

sum of smaller side length

hypotenuse

Definition: a heuristic $h(n)$ is *consistent* if it satisfies the triangle inequality. In a sentence: the cost to go from n to the goal must be less than or equal to the cost to move from n to n' plus the *estimated* cost $h(n')$,

$$\underline{h(n) \leq c(n, a, n') + h(n')}.$$



Astar Theory

Ok, so A^* uses all the information. Does it do it well? Is it **optimal?** Is it **complete?**

Definition: a heuristic $h(n)$ is *admissible* if it never overestimates the cost to reach the goal. Because $g(n)$ is an actual cost to reach n along a current path, f will never overestimate the cost of a solution *through* n if $g(n)$ is admissible.

Definition: a heuristic $h(n)$ is *consistent* if it satisfies the triangle inequality. In a sentence: the cost to go from n to the goal must be less than or equal to the cost to move from n to n' plus the *estimated* cost $h(n')$,

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

Consistent and Admissible

So we have two criteria:

1. **Admissible**: $0 \leq h(n) \leq h^*(n)$
2. **Consistent**: $h(n) \leq c(n, a, n') + h(n'))$

Theorem: Any consistent heuristic is also admissible.
Proof: is by induction.

Consistent and Admissible

So we have two criteria:

1. **Admissible**: $0 \leq h(n) \leq h^*(n)$
2. **Consistent**: $h(n) \leq c(n, a, n') + h(n'))$

Theorem: Any consistent heuristic is also admissible.
Proof: is by induction.

Base Case: Suppose a heuristic h is consistent for $k = 1$ nodes along the shortest path from n to the goal node (so we start one step away from the finish)

Let n' be the goal, 1 step away. Then we have $h(n) \leq c(n, a, n') + h(n')$. Since n' is the goal, $h(n') = 0$ and $c(n, a, n') = h^*(n)$

Then:

$$\implies h(n) \leq c(n, a, n') + h(n') = h^*(n)$$

which means that h does not overestimate the actual cost to get to the goal from n, so it is admissible.

Consistent and Admissible

So we have two criteria:

1. **Admissible**: $0 \leq h(n) \leq h^*(n)$
2. **Consistent**: $h(n) \leq c(n, a, n') + h(n'))$

Theorem: Any consistent heuristic is also admissible.
Proof: is by induction.

Inductive Step: Suppose *if* a heuristic h is consistent for k nodes along the shortest path from n to the goal node, then it is admissible.

Let n' be another node on the shortest path from n to the goal, k steps away. Then we have $h(n) \leq c(n, a, n') + h(n')$.

Since h is admissible along that path, $h(n') \leq h^*(n')$. Then:

$$\implies h(n) \leq c(n, a, n') + h(n') \leq c(n, a, n') + h^*(n') = h^*(n)$$

which means that h does not overestimate the actual cost to get to the goal from n, so it is admissible.

Search Underview

In general our search algorithms only work when:

1. domain is fully observable
2. domain must be known
3. domain must be deterministic
4. domain must be static

implementation: use a node (dictionary)

1. state - indicates state at end of path
2. action - action taken to get here
3. cost - total cost
4. parent - pointer to another node

Moving Forward

- This week:
  1. HW2 released this week.
  2. Additional Office Hours starting this week!
- Next time: more on heuristics!