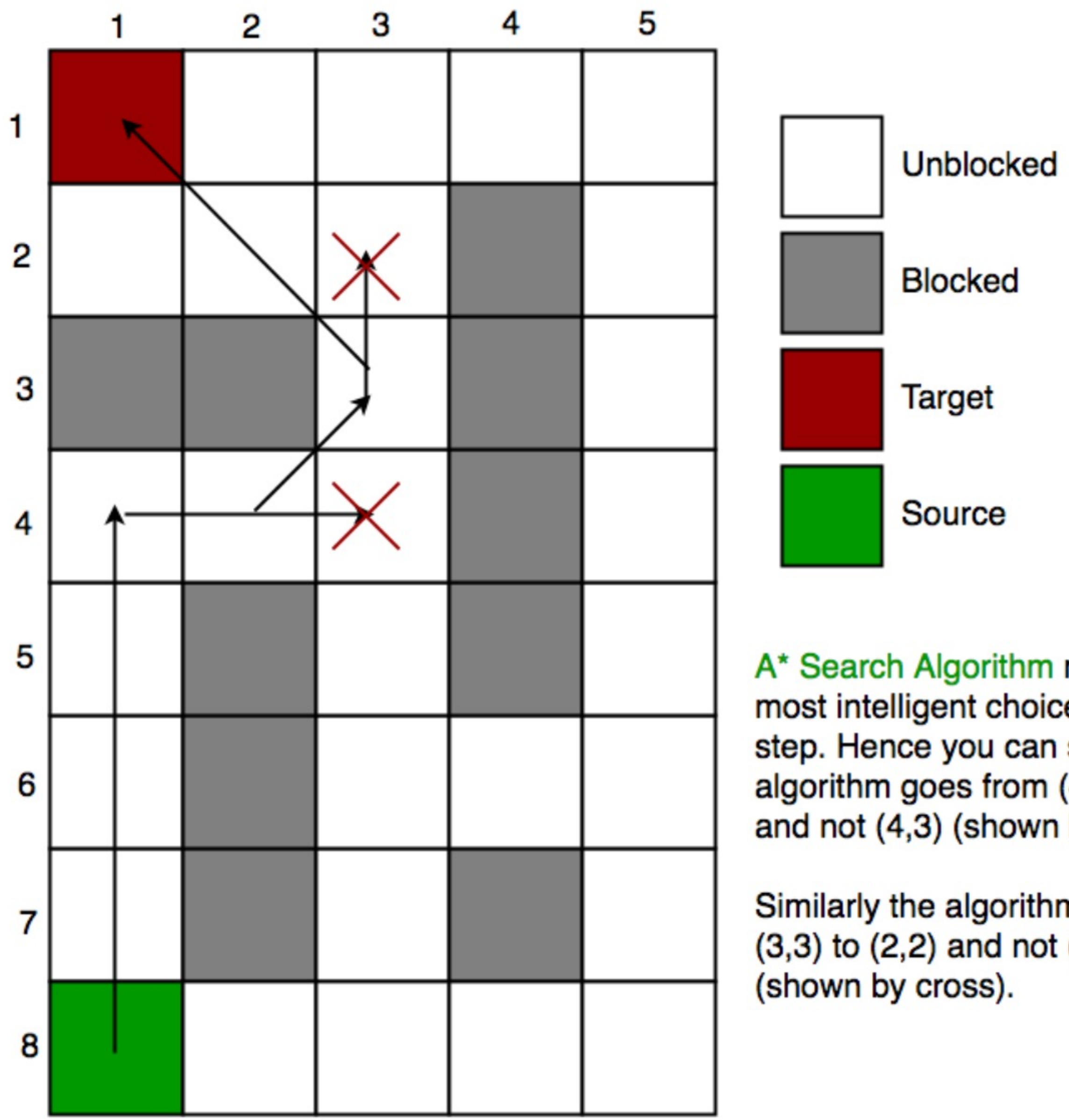
CSCI 3202: Intro to Artificial Intelligence
UCS, A* Search and Heuristics

Rhonda Hoenigman

Department of Computer Science



A* Search Algorithm makes the most intelligent choice at each step. Hence you can see that algorithm goes from (4,2) to (3,3) and not (4,3) (shown by cross).

Similarly the algorithm goes from (3,3) to (2,2) and not (2,3) (shown by cross).

Source

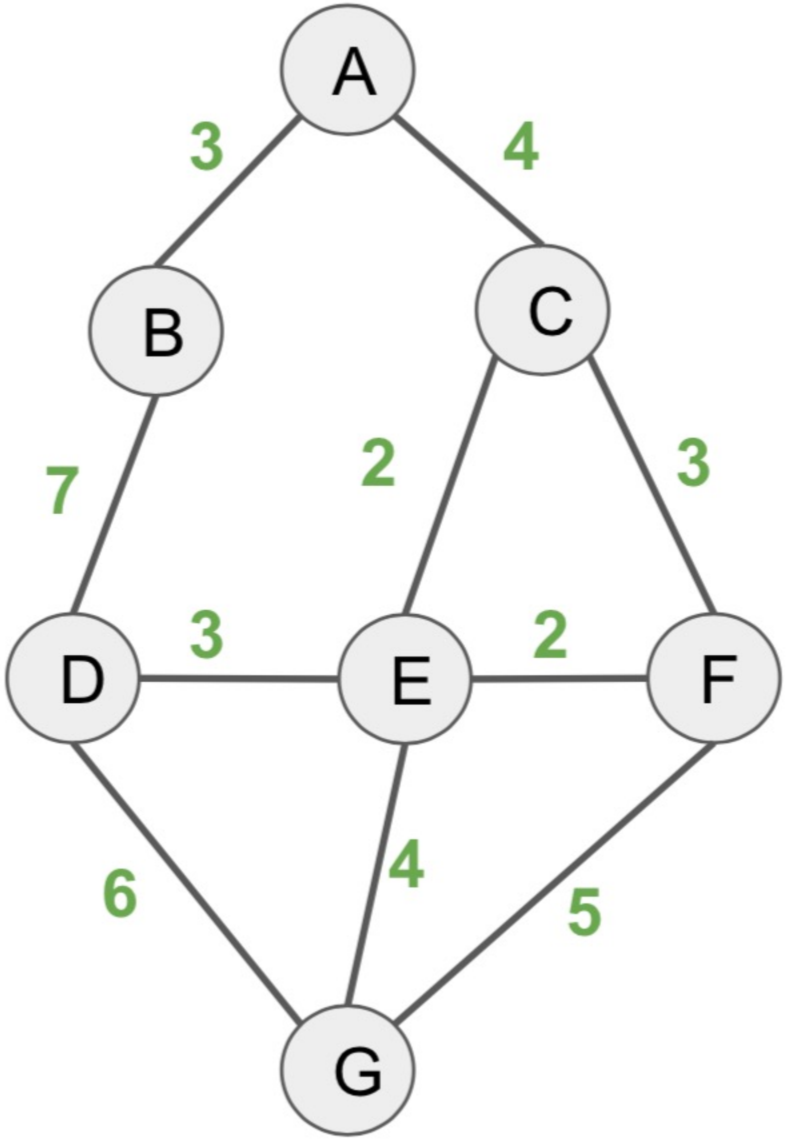
Uniform–cost Search (UCS)

- Expand out in contours, where least cost dictates which nodes we explore.
- Eventually, we will find a path to the goal - but the search is not directed
- BFS strategy
- Expand cheapest node first (lowest path cost)
- Frontier is a priority queue
- Cost function sets priority

Uniform–cost Search (UCS) – pseudocode

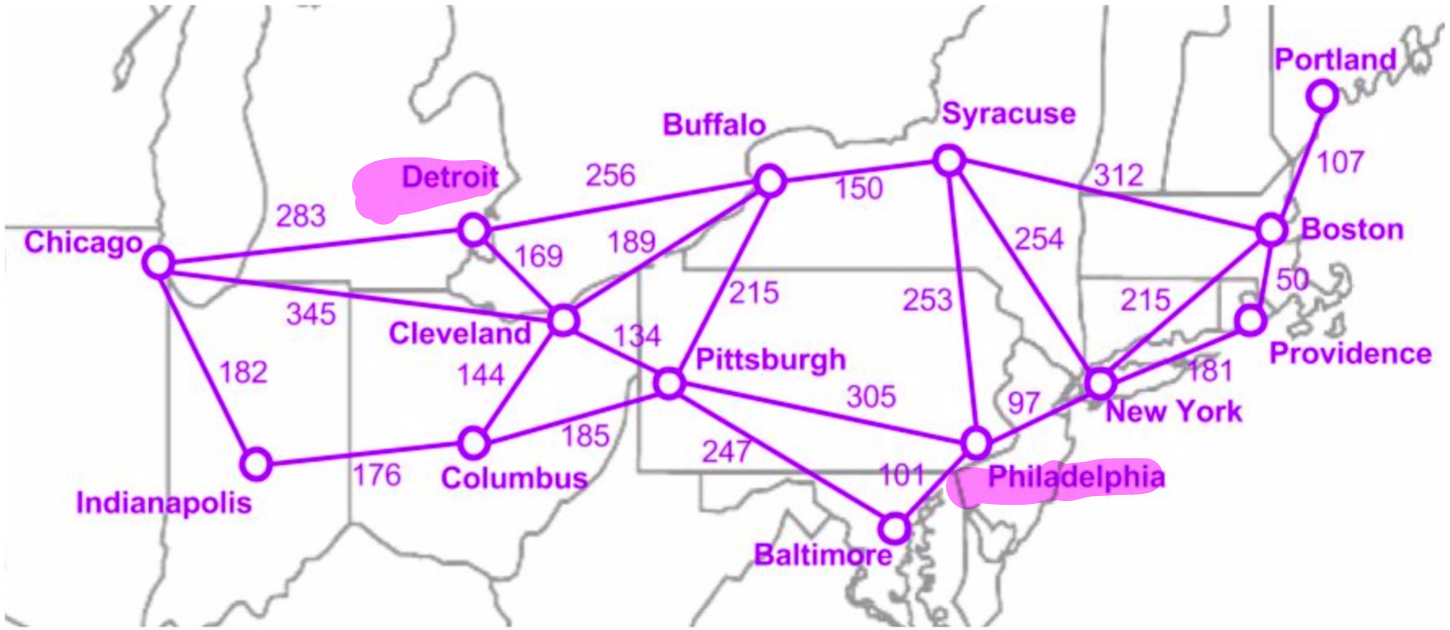
```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
     if EMPTY?(frontier) then return failure
     node ← POP(frontier)   /* chooses the lowest-cost node in frontier */
     if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
     add node.STATE to explored
     for each action in problem.ACTIONS(node.STATE) do
        child ← CHILD-NODE(problem, node, action)
        if child.STATE is not in explored or frontier then
           frontier ← INSERT(child, frontier)
        else if child.STATE is in frontier with higher PATH-COST then
           replace that frontier node with child
```

Uniform–cost Search (UCS)



Example: Perform a UCS on the graph below. A is the starting point; G is the goal.

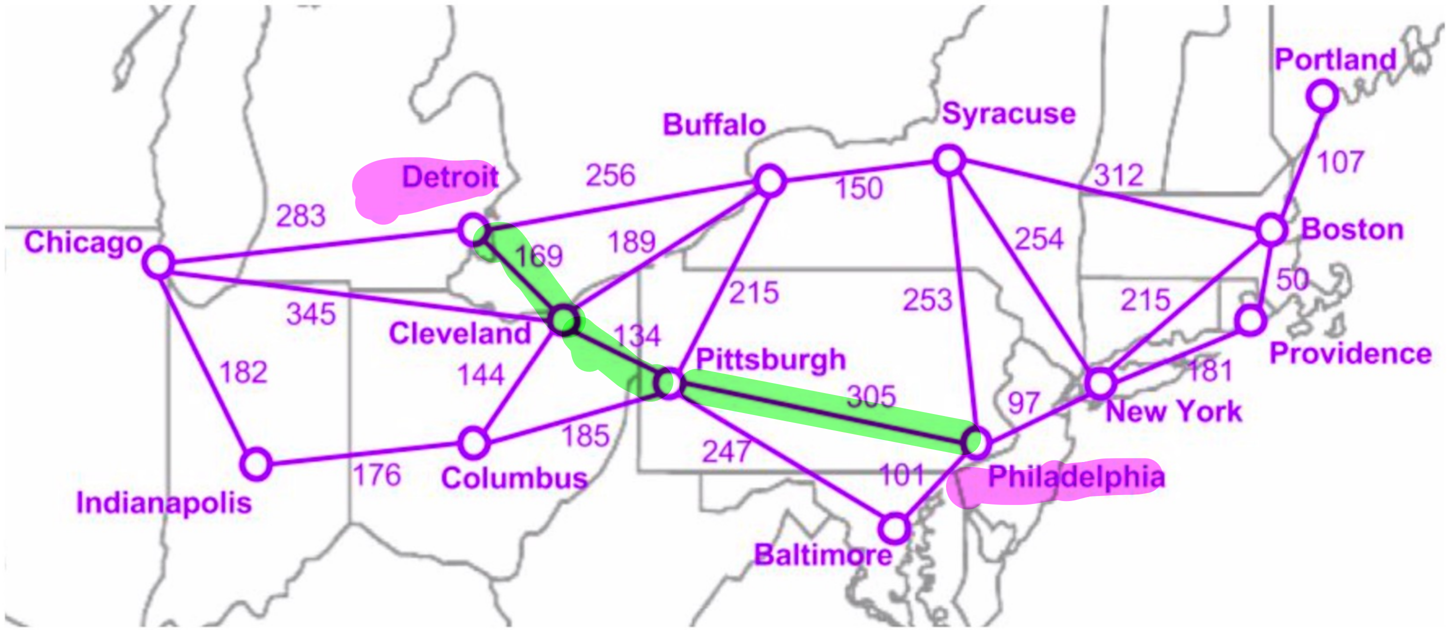
Uniform-cost Search (UCS)



Example: Use UCS to find a route from Detroit to Philadelphia.

	F	E	ACT
1.	De(ϕ)		
2.	Bu(256), Cl(169), Ch(283)	De	De → Bu, Cl, Ch
3.	Bu(256), Pi(303), Ch(514)	De, Cl	Cl → Bu, Pi, Ch; Bu = 169+189 ✗; Pi = 169+134; Ch = 169+345
4.	Sy(406), Pi(303), Ch(514)	De, Cl, Bu	Bu → Sy, Pi, ~~Cl~~; Sy = 256+150; Pi = 256+215 ~~471~~
5.	Sy(406), Ch(514), Ph(608), Ba(550)	De, Cl, Bu, Pi	Pi → Ph, Ba, ~~Bu~~; Ph = 303+305; Ba = 303+247
6.	Ch(514), Ph(608), Ba(550), Bo(718), Ny(660)	De, Cl, Bu, Pi, Sy	Sy → Ny, Bo, Ph; Ny = 406+254; Bo = 406+312; Ph = 406+256 = 659 ✗

Uniform–cost Search (UCS)



Example: Use UCS to find a route from Detroit to Philadelphia.

6. Ch(514), Ph(608), Ba(550) Bo(718) Ny(660)   De, Cl, Bu, Pi, Sy

Sy → Ny, Bo, Ph
Ny = 406 + 254
Bo = 406 + 312
Ph = 406 + 256 = 659 X

Det → Clv, Pit, Ph = 169 + 134 + 305 = 608

F	E	ACT
7. In(696) Ph(608) Ba(550) Bo(718) Ny(660)	De, Cl, Bu Pi Sy Ch	Ch → ~~De~~, ~~Cl~~, In In → 514 + 182
8. In(696) Ph(608) Bo(718) Ny(660)	De, Cl, Bu, Pi, Sy Ch, Ba	Ba → Ph, ~~Pi~~, Ph = 550 + 101 651 X

9. Ph → 608

Uniform–cost Search (UCS)

- ***Goal test occurs when node is selected for expansion***
- Because we know we've taken the cheapest path to get there, UCS is **optimal if all edge weights > 0**
- It is also **complete** because it's a more general form of BFS (which is complete)

Uniform–cost Search (UCS)

- Can get stuck if there are sequences of no-cost actions. Optimality requires positive edge weights

$$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$$

- Worst-case in time and space complexity:

  - C* is cost of optimal solution
  - ϵ is minimal action cost

- Potential inefficiency: Explores in every “direction”

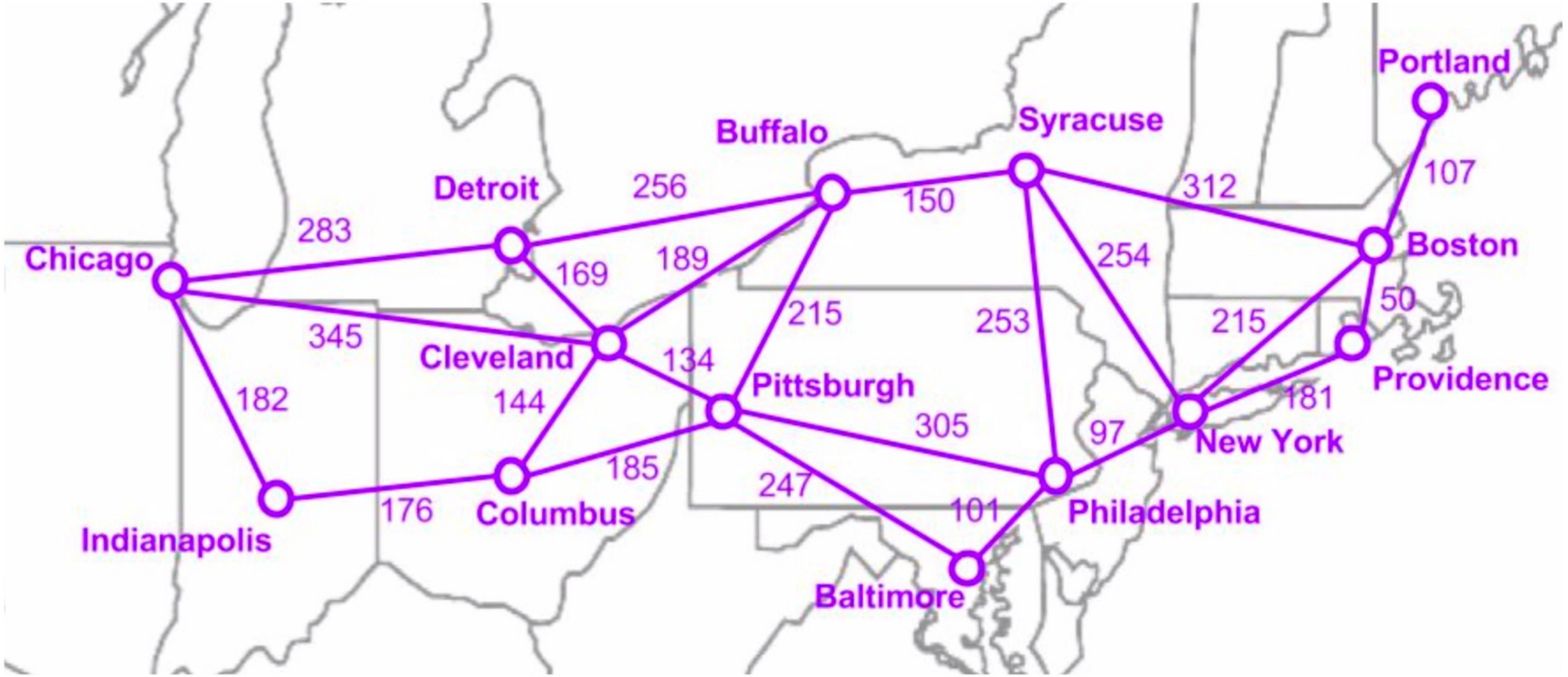
Slightly more informed search – Greedy best–first search

❖ First expand the path that's closest to the goal.

To determine what's closest to the goal, we need to define a heuristic function.

Example: For the traveling in the northeast problem, let's estimate the distance to the goal as the straight-line distance between city and the goal city.

Step costs: miles between cities along major highways



FOR YOUR NEXT STEP, PICK THE NODE THAT IS CLOSEST TO YOUR GOAL

Greedy best–first search

Example: Use the greedy best-first search to find a route from Chicago to Providence.

Distances:
Chicago->833
Detroit->597
Indianapolis->783
Columbus->618
Cleveland->530
Pittsburgh->456
Buffalo->388
Syracuse->256
Baltimore->324
Philadelphia->235
New York->155
Boston->41
Portland->140

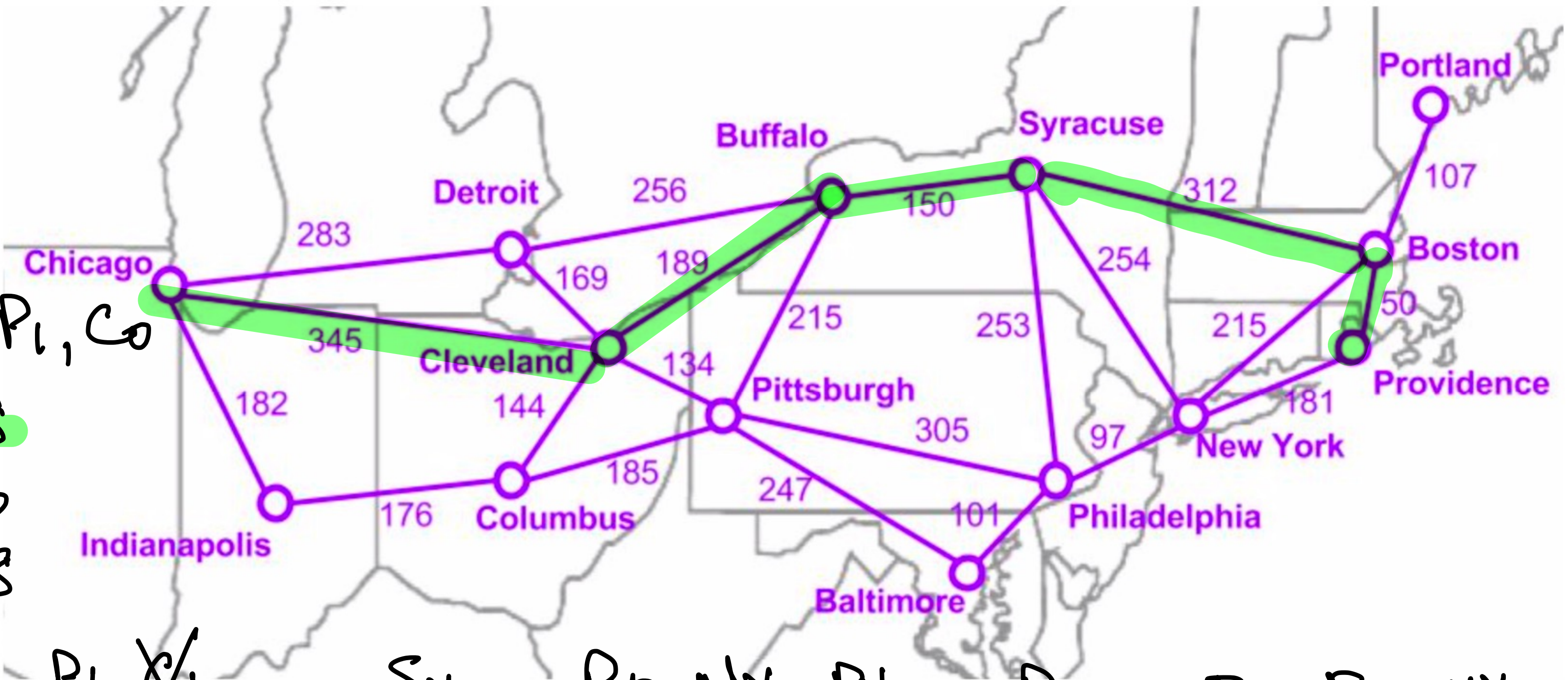
Heuristic: $h(n)$ = straight-line distance to Providence

CH → De, CL, IN

De = 597
CL = 530
IN = 783

CL → Bu, Pi, Co

Bu = 388
Pi = 456
CO = 618



Bu → Sy, Pi, ~~CL~~

Sy = 256
Pi = 456

Sy → Bo, NY, PH

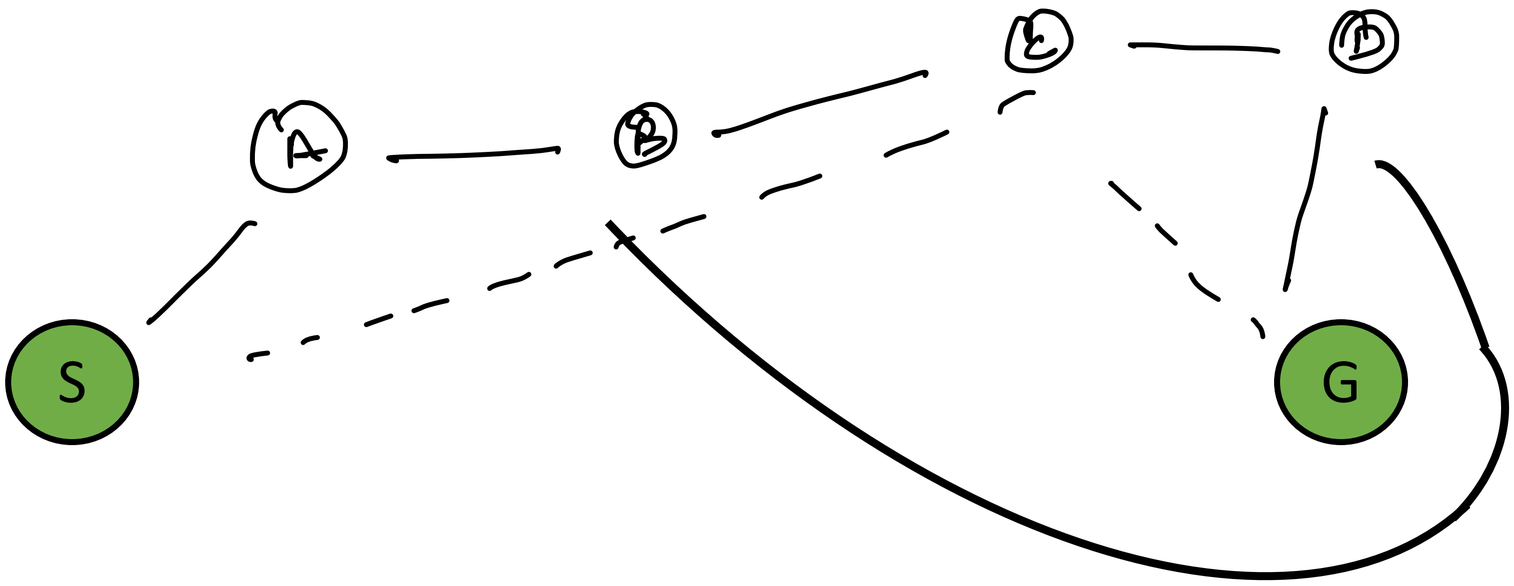
Bo = 41
NY = 155
PH = 235

Bo → PR, PO, NY

PR = 0 ✓
Po = 140
NY = 155

Greedy best_first search

Possible Issue: Won't necessarily find the optimal path. Can get stuck in local optimum.



A* Search

Uniform-cost search:

$f(n) = g(n)$ (cost to get to *n*)

Greedy:

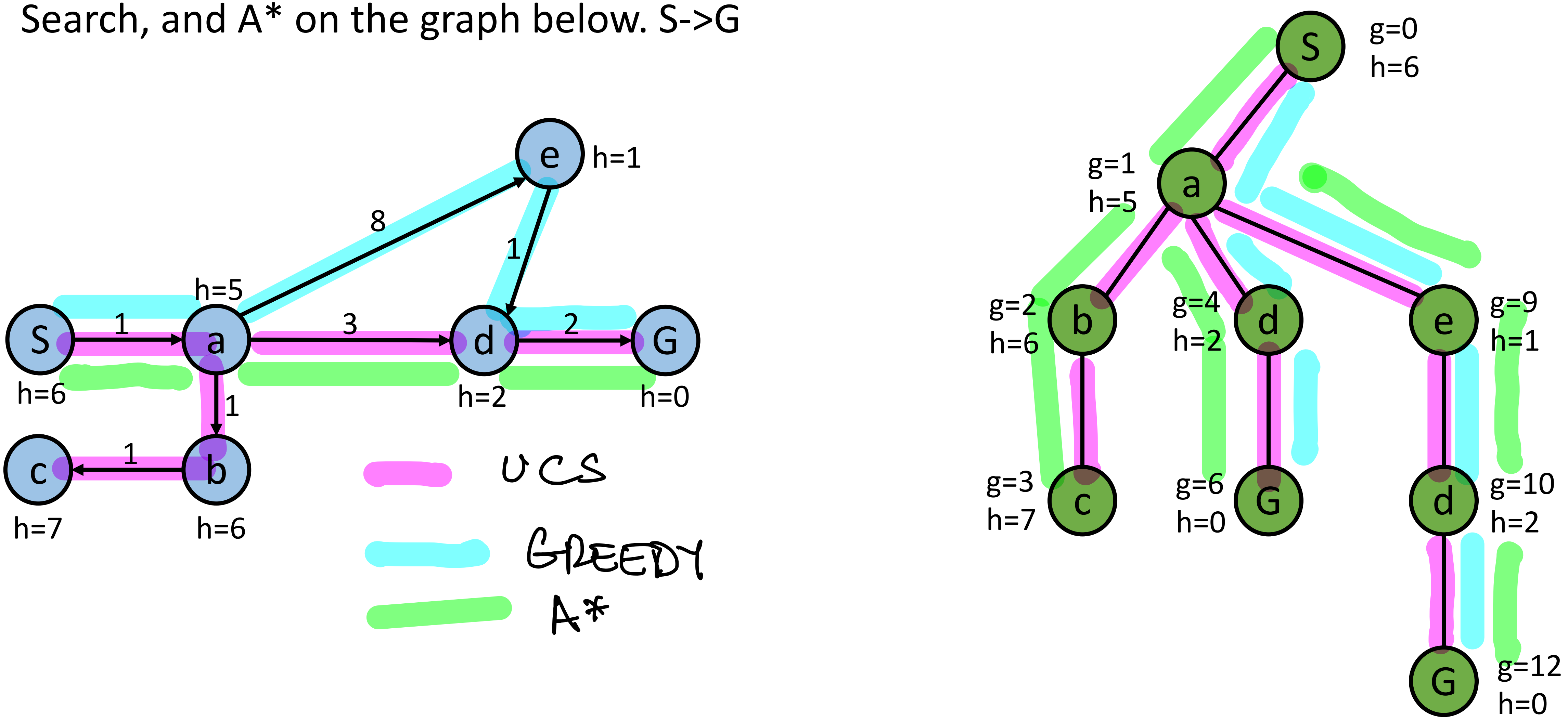
$f(n) = h(n)$ (estimated cost to get from *n* to goal)

A*:

$f(n) = g(n) + h(n)$ (estimated total cost of cheapest solution through *n*)

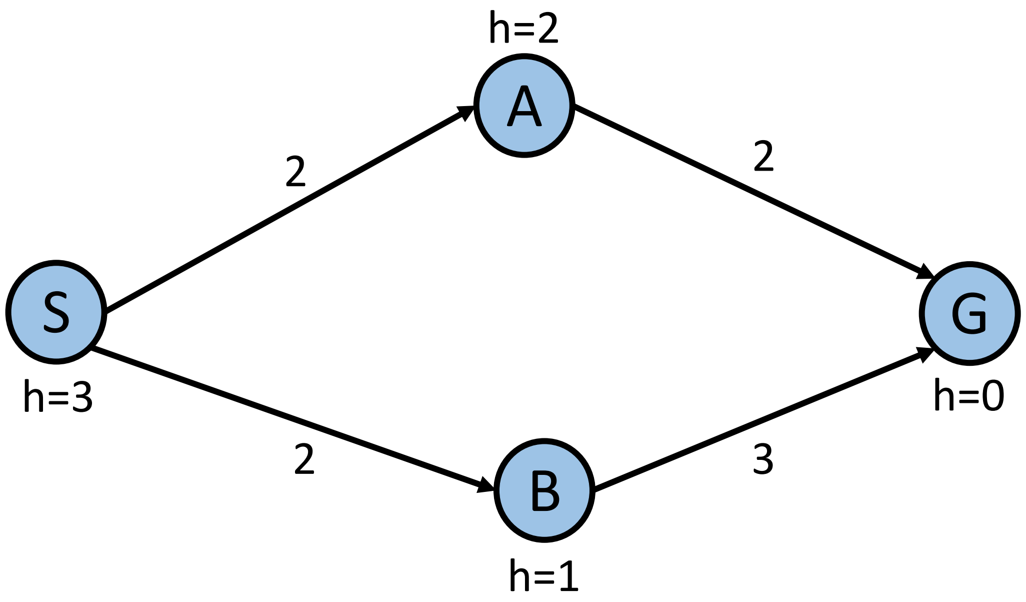
A* Search

Example: Compare Uniform Cost, Greedy Search, and A* on the graph below. S->G



A* Search

Example: When should A* search terminate?



S → G

1. S(ϕ)

2. A(4)
   B(3)

3. A(4)
   G(5)

4. G(4)

5. G(4)

S → A, B

A = (2+2) = 4

B = (2+1) = 3

B → G

G = (2+3+0) = 5

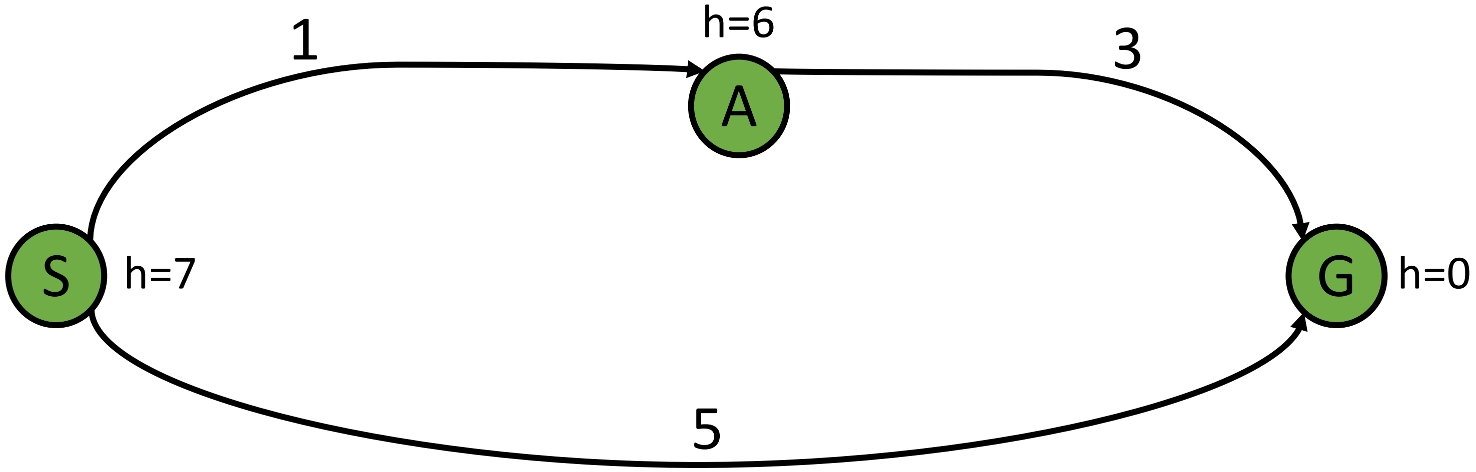
A → G

G = (2+2+0) = 4

Done

A* Search

Is A* optimal?



A* Search

Consistent: for every node n and successor n' of n, generated by some action a, the estimated cost of reaching the goal from n is no greater than the step cost from n to n', plus the estimated cost of reaching the goal from n'

- That is: $h(n) \leq c(n, a, n') + h(n')$
- General **triangle inequality** between n, n', and the goal

A heuristic h is **admissible** (optimistic) if $0 \leq h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to the nearest goal.

Optimality

Conditions for Optimality: Admissibility & Consistency

- $h(n)$ must be **admissible** - an admissible heuristic is one that never overestimates the cost to reach the goal.

- $h(n)$ is **consistent** if, for every node n and every successor n' of n generated by any 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':

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

Optimality of A* Search

A* is **optimally efficient** for any given heuristic: No other optimal algorithm is guaranteed to expand fewer nodes than A*

- Recall: A* expands all nodes with $f(n) <$ C*, where C* is the cost of the optimal solution path.

- Any algorithm that does not expand all nodes with $f(n) <$ C* risks missing a better solution path.

Optimality of A* Search

A* (graph) is optimal if the heuristic $h(n)$ is consistent.

Based on two key facts:

1. If $h(n)$ is consistent, then the values of $f(n)$ along any path are nondecreasing.

2. Whenever A* selects a node *n* for expansion, the optimal path to that node has been found.

Optimality of A* Search

A* (graph) is optimal if the heuristic $h(n)$ is consistent.

Based on two key facts:

1. If $h(n)$ is consistent, then the values of $f(n)$ along any path are nondecreasing.

2. Whenever A* selects a node *n* for expansion, the optimal path to that node has been found.

- **So the first goal node to be expanded took the lowest-cost path, and all later goal node expansions are at least as expensive.**

Optimality of A* Search

So A* is **optimal**, **complete**, and **optimally efficient**.

Why do we even care about other search algorithms?

- **Number of nodes** to expand along the goal contour is still **exponential** in depth of solution/length of solution path.

- Absolute error: $\Delta \coloneqq h^* - h$
  - o h* = actual cost from root to goal
  - o h = heuristic you used

- Relative error: $\epsilon \coloneqq (h^* - h)/h^*$

A* Search

Search only works when:

- domain is fully observable
- domain must be known
- domain must be deterministic
- domain must be static

A* Search

Complexity depends strongly on state space characterization

- Single goal, tree, reversible actions $\rightarrow O\left(b^{\Delta}\right)$, or $O\left(b^{\epsilon d}\right)$ with constant step costs (*d* is solution depth)

  Δ typically is proportional to the path cost $h*$, so ϵ is pretty much constant (or growing with d), and we can rewrite: $O\left((b^{\epsilon})^{d}\right)$

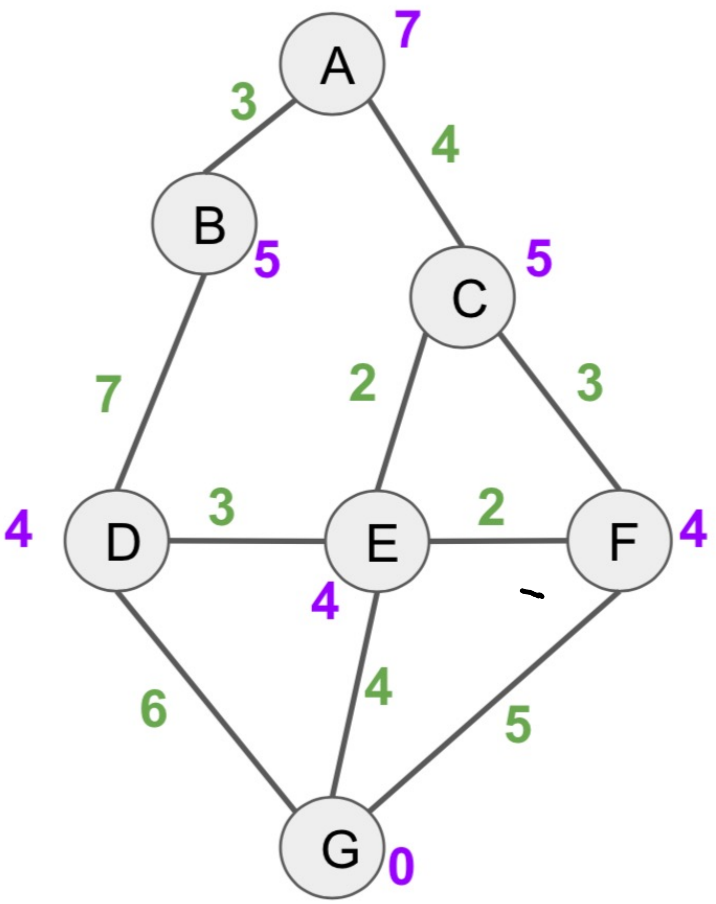
  $\rightarrow$ The effective branching factor is really b^{ϵ}.
  $\rightarrow$ Important to choose as good of a heuristic as we can.

- Many goal states/near-goal states can be a problem -- need to expand a ***lot*** of branches.

A* Search

A* Search:

- Find the minimum cost path from A to G
- $h(n)$ values are given in **purple**
- Step costs are given in **green**



A → G

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

path cost + heuristic

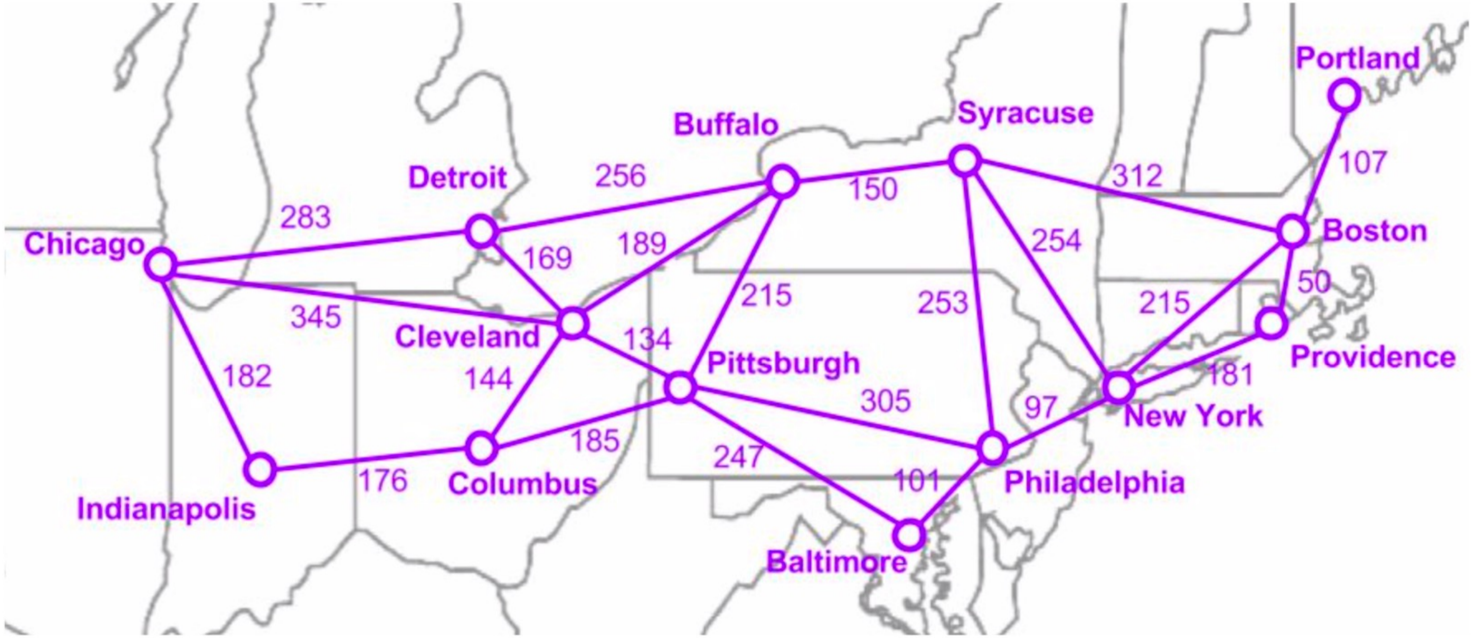
	F	E	ACTION
1	A(7)		
2.	B(8), C(9)	A	A → B(3+5), C(4+5)
3.	C(9) D(14)	A, B	B → D(3+7+4)
4.	D(14), E(10), F(11)	A, B, C	C → E(4+2+4), F(4+3+4) 12 X
5.	D(14), F(11), G(10)	A, B, C, E	E → F(4+2+2+4), G(4+2+4+0)
6.	D(14), F(11)	A, B, C, E, G	G → F(4+2+4+5+4) = 19 X DONE!

A* Search

Example: Use A* search to find a route from Chicago to Providence.

$h(n)$ = straight-line distance to Providence
$g(n)$ = Path cost so far

Distances:
Chicago->833
Detroit->597
Indianapolis->783
Columbus->618
Cleveland->530
Pittsburgh->456
Buffalo->388
Syracuse->256
Baltimore->324
Philadelphia->235
New York->155
Boston->41
Portland->140



h(Chicago) = 833

	F	E
1	Chi (833)	0
2.	Det (880), Cle (875), Ind (965)	Chi
3.	Buf (922), Pit (935), Col (1107), Det (880), Ind (965)	Chi, Cle
4.	Buf (922), Pit (935), Col (1107), Ind (965)	Chi, Cle, Det

ACTION

Chi → Det, Cle, Ind

Det = 283 + 597 = 880
Cle = 345 + 530 = 875
Ind = 182 + 783 = 965

Cle → Buf, Pit, Col

Buf = 345 + 189 + 388 = 922
Pit = 345 + 134 + 456 = 935
Col = 345 + 144 + 618 = 1107

Det → ~~Chi~~, ~~Cle~~, Buf

Buf = 345 + 169 + 256 + 388 = 1158 X

5. Pit (935) Chi
Ind (965) Cle
Col (1107) Det
Syr (940) Buf

Buf → ~~Det~~, Pit, Syr.

Pit = 345 + 189 + 215 + 456 = 1205 X
Syr = 345 + 189 + 150 + 256 = 940

6. Syr (940) Chi
Col (1107) Cle
Ind (965) Det
Phi (1109) Buf
Bal (1050) Pit

Pit → Phi, Bal, ~~Buf~~

Phi = 345 + 134 + 395 + 235 = 1109
Bal = 345 + 134 + 247 + 324 = 1050

7. Ind (965) Chi
Col (1107) Cle
Phi (1109) Det
Bal (1050) Buf
Bos (1042) Pit
New (1098) Syr
Phi (1177)

Syr → Bos, New, Phi

Bos = 283 + 256 + 150 + 312 + 41 = 1042
New = 283 + 256 + 150 + 254 + 155 = 1098
Phi = 283 + 256 + 150 + 253 + 235 = 1177 X

8. Col(976) Chi Ind → Col, ~~Chi~~
Phi (1109) Cle
Bal (1050) Det
Bos (1042) Buf
New (1098) Pit
Syr
Ind

$Col = 182 + 176 + 618 = 976$

9. Phi (1109) Chi, Cle, Col → ~~Cle~~, ~~Pit~~
Bal (1050) Pit, Det,
Bos (1042) Buf, Syr,
New (1098) Ind, Col

10. Phi (1109) Bos → Pro, Por, ~~Syr~~
Bal (1050)
New (1098)
Pro (1051)
Port (1248)

$Pro = 283 + 256 + 150 + 312 + 50 = 1051$

$Port = 283 + 256 + 150 + 312 + 107 + 140 = 1248$

11.

1062
Phi (1109)
New (1098)
Pro (1051)
Port (1248)

Bal → Phi, ~~Det~~

Phi = 345 + 134 + 247 + 101 + 235 = 1062

Pro → Done

12.

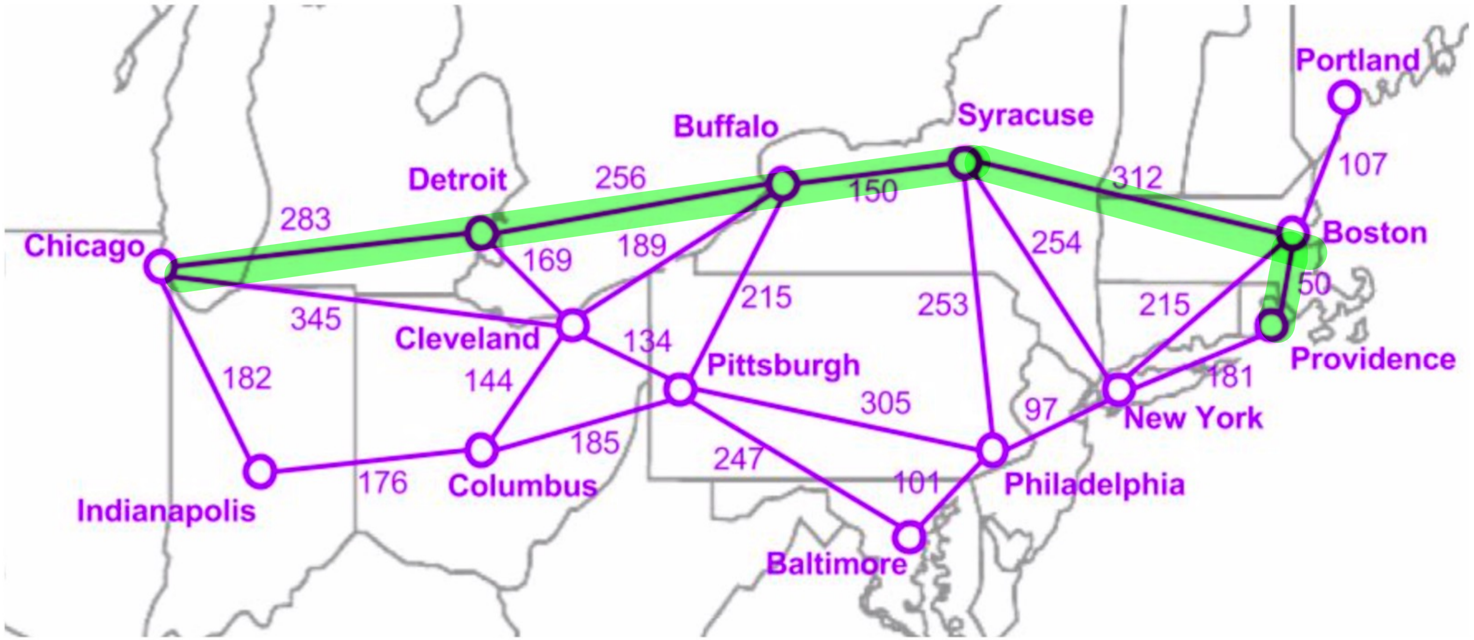
Path : Chi → Det → Buf → Syr → Bos → Providence

A* Search

Example: Use A* search to find a route from Chicago to Providence.

Distances:
Chicago->833
Detroit->597
Indianapolis->783
Columbus->618
Cleveland->530
Pittsburgh->456
Buffalo->388
Syracuse->256
Baltimore->324
Philadelphia->235
New York->155
Boston->41
Portland->140

$h(n)$ = straight-line distance to Providence
$g(n)$ = Path cost so far



PATH DISTANCE = 1051 miles