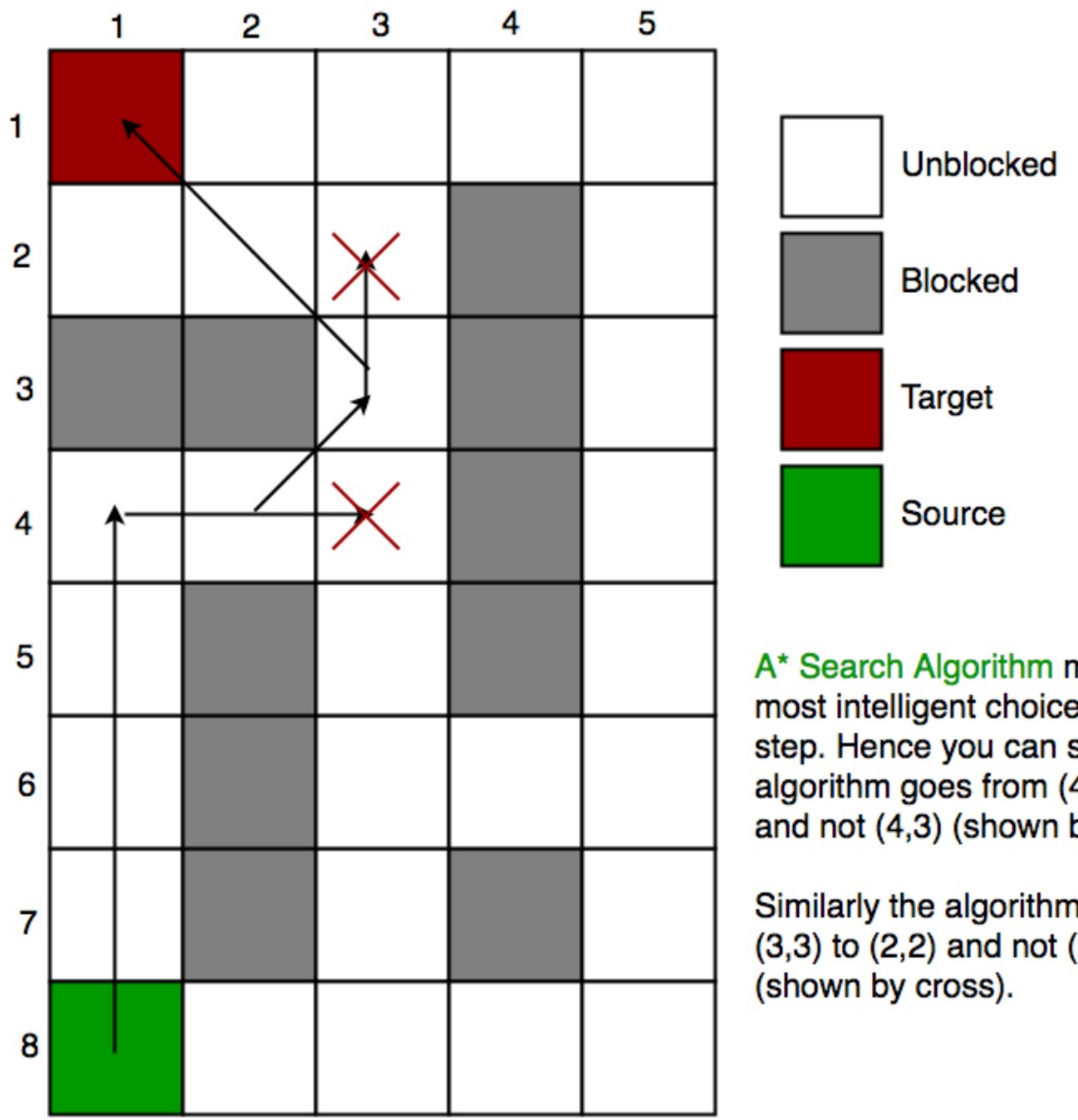
CSCI 3202: Intro to Artificial Intelligence
Lecture 8: A* Search and Heuristics

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).

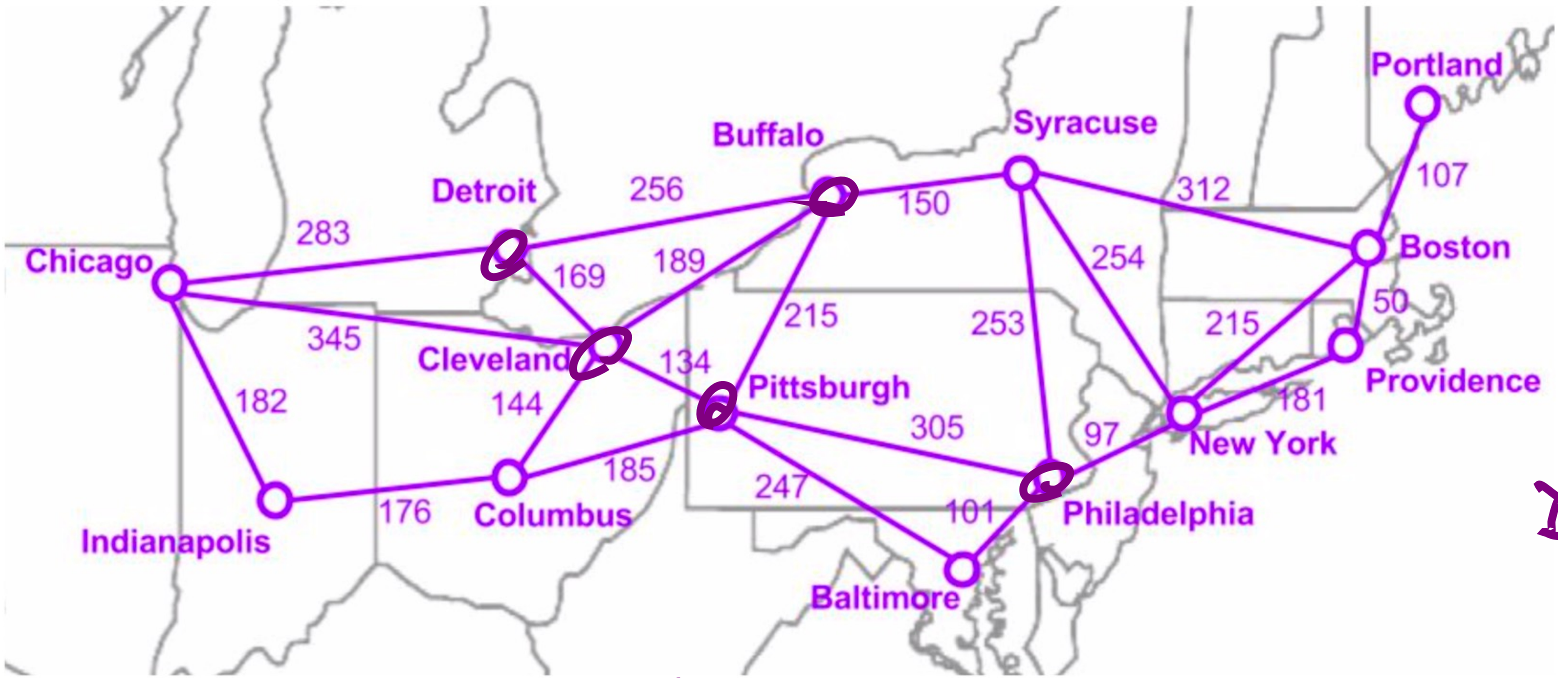
Similarly the algorithm goes from (3,3) to (2,2) and not (2,3) (shown by cross).

Source

Review: 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

Uniform–cost Search (UCS)



Exp	Frontier
D	Chi = 283, Buf = 256, Cle = 169
D, Cle	Chi = 283, Buf = 256, Col = 313, Pit = 303
D, Cle, Buf	Chi = 283, Col = 313, Pit = 303, Syr = 406
D, Cle, Buf, Chi	Pit = 303, Col = 313, Syr = 406, Ind = 465
D, Cle, Buf, Chi, Pit	Col = 313, Syr = 406, Ind = 465

When Phi added to explored, Solution found.

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

Exp	Frontier
D, Cle, Buf, Chi, Pit, Col, Syr	Ind = 465, Phi = 608, Bal = 530, NY = 660, Bost = 718.

D, Cle, But, Chi, Pit, Col, Syr, Ind, Bal, Phi

Phi = 608
Bal = 550

Uniform–cost Search (UCS)

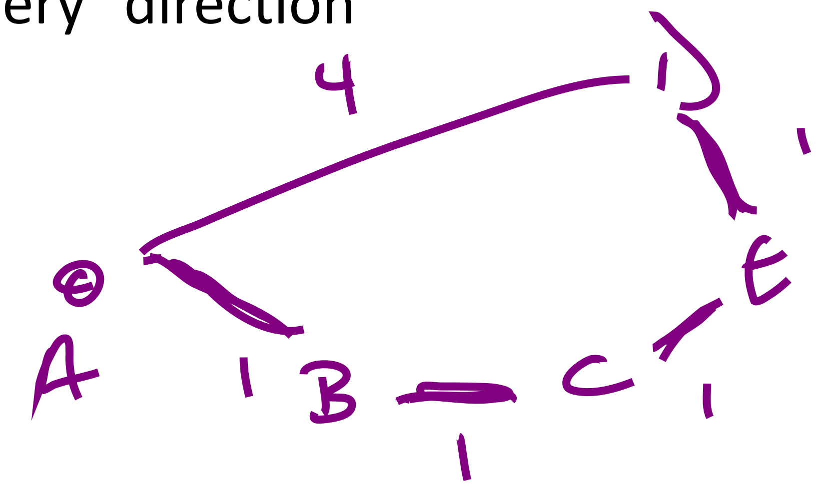
- Can get stuck if there are sequences of no-cost actions. Optimality requires positive edge weights

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

b = branching factor

- Worst-case in time and space complexity:
  - C* is cost of optimal solution
  - ϵ is minimal action cost

- Potential inefficiency: Explores in every “direction”

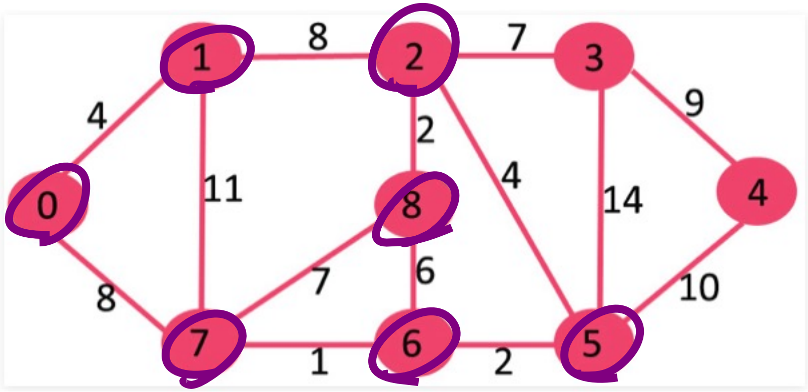


Dijkstra's Shortest Path Algorithm

❖ Uniform Cost Search is a variant of Dijkstra's shortest path algorithm.

When a node pops from queue, we have shortest path to that node.

Example: Use Dijkstra's algorithm to find the shortest path from 0 **to all other nodes** (Shortest Path Tree)



Path 0-4

Order that nodes added to explored

0, 1, 7, 6, 5, 2, 8,
3, 4

Explored	Frontier
0	0-1=4, 0-7=8
0, 1	0-7=8, 0-1-2=12
0, 1, 7	0-1-2=12, 0-7-8=15 0-7-6=9
0, 1, 7, 6	0-1-2=12, 0-7-8=15 0-7-6-5=11
0, 1, 7, 6, 5	0-7-6-5-4=21 0-7-6-5-3=25 0-1-2=12 0-7-8=15

Search Algorithms

0, 1, 7, 6, 5, 2

-s ‘ =21

0, 1, 7, 6, 5, 2, 8, 3, 4

❖ Search algorithms we’ve seen are fundamentally the same except for their frontier strategies.

Uninformed Search: e.g. Uniform Cost Search

- the good: UCS is complete and optimal → if a solution exists, it will find it with the least cost path

- the bad: explores in every direction

Informed Search: include information about where the goal is

- what do we need to have? A heuristic.

heuristic: A function that estimates how close a state is to a goal.

Informed guess based on domain knowlege.

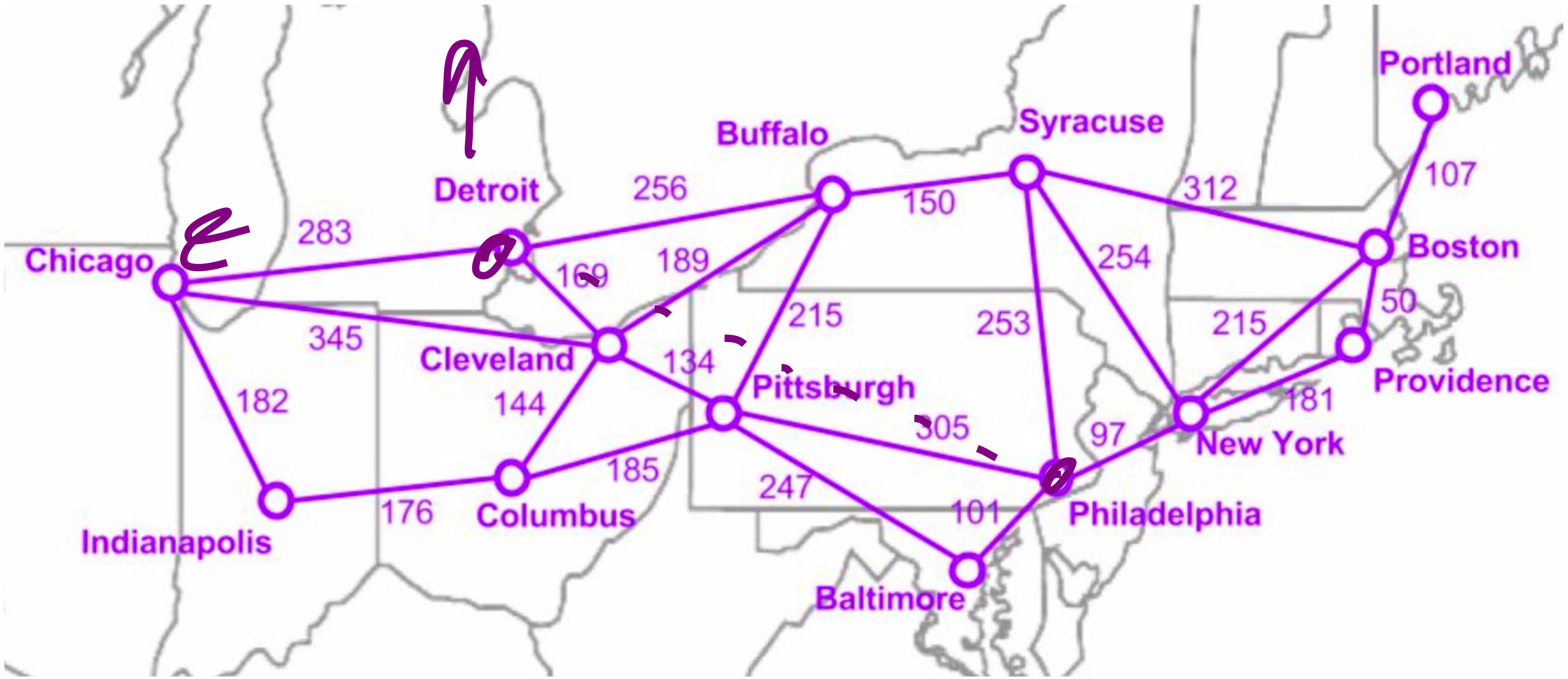
Greedy best–first search

❖ 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.

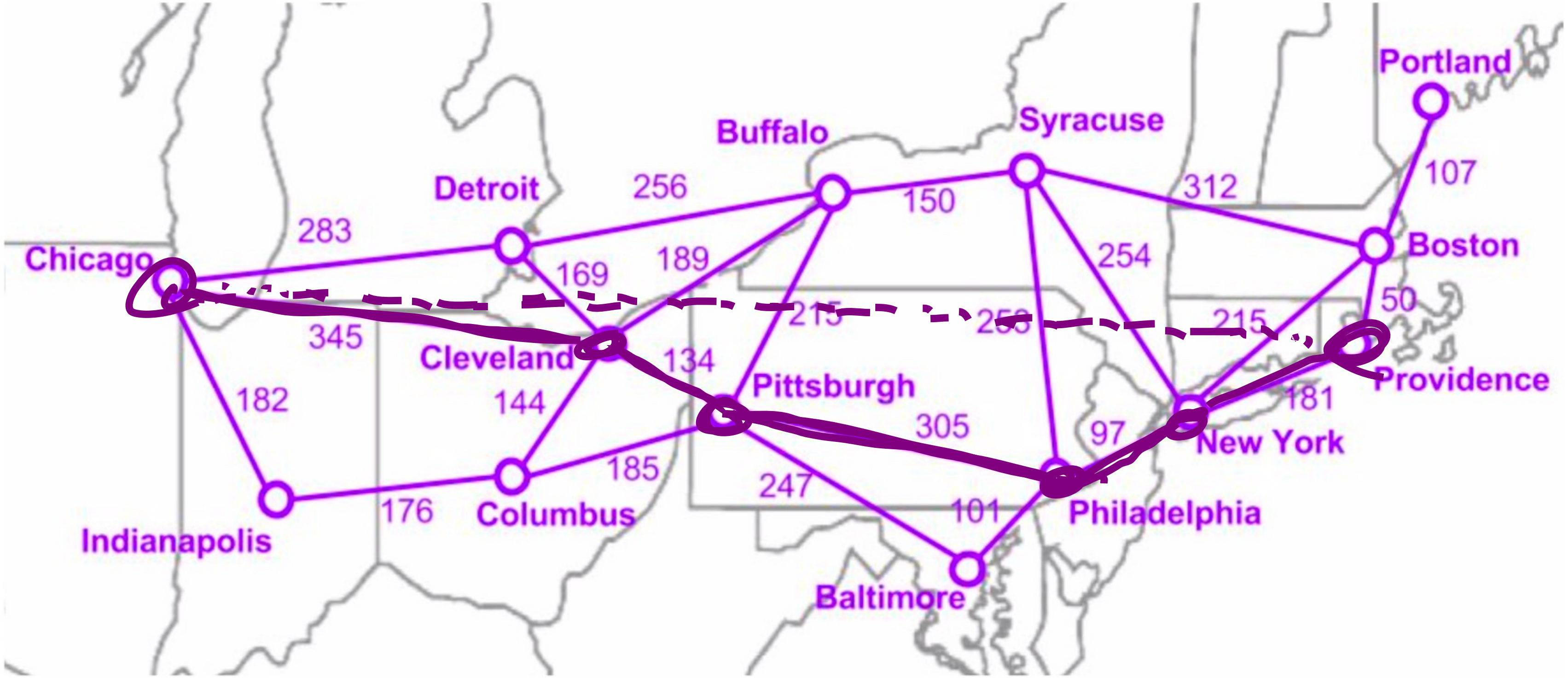
Step costs: miles between cities along major highways



Greedy best–first search

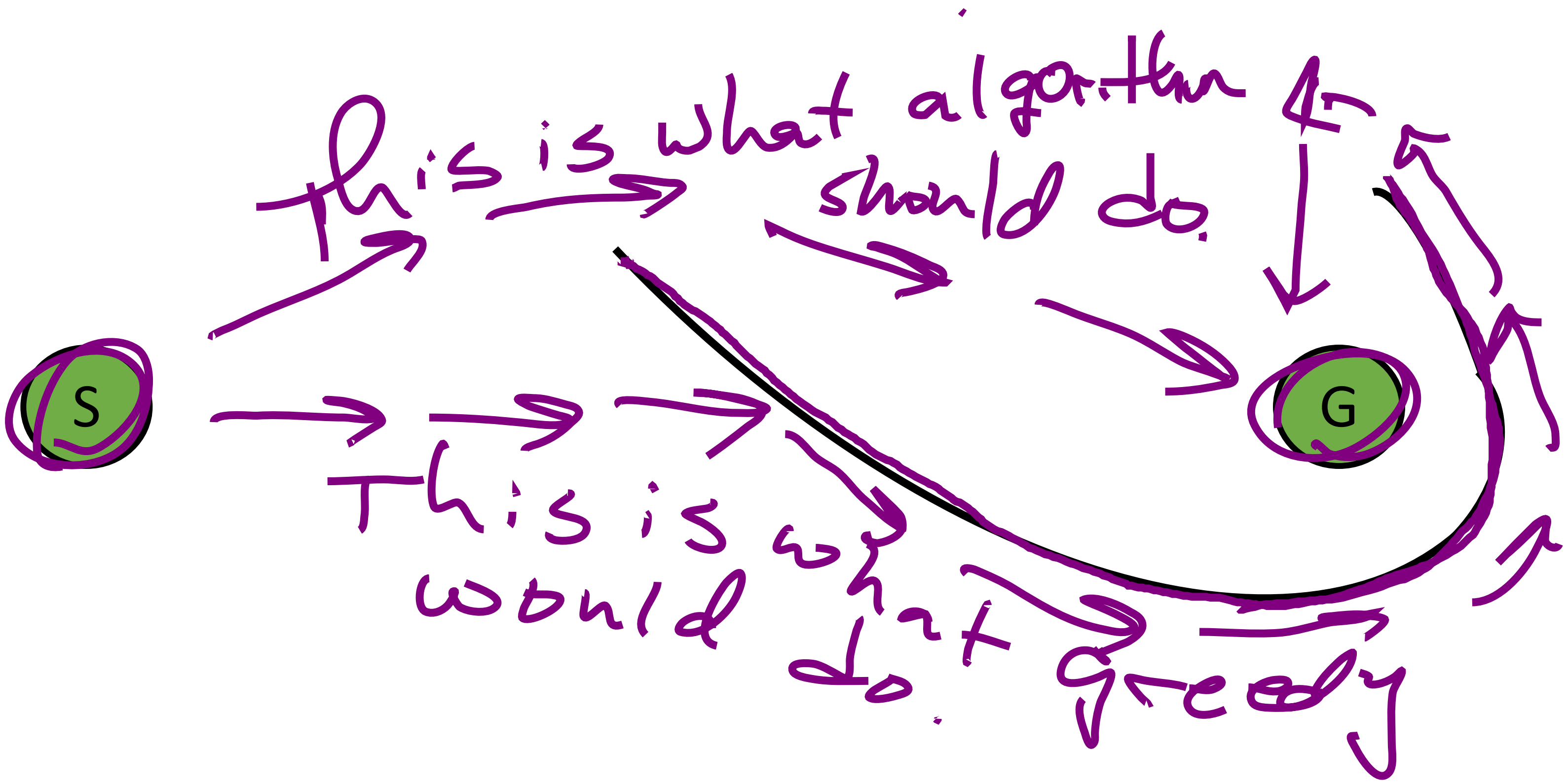
Example: Use the greedy best-first search to find a route from Chicago to Providence.

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



Greedy best_first search

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



A* Search

$$A \xrightarrow{g(n)} n \xrightarrow{h(n)} G$$

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)

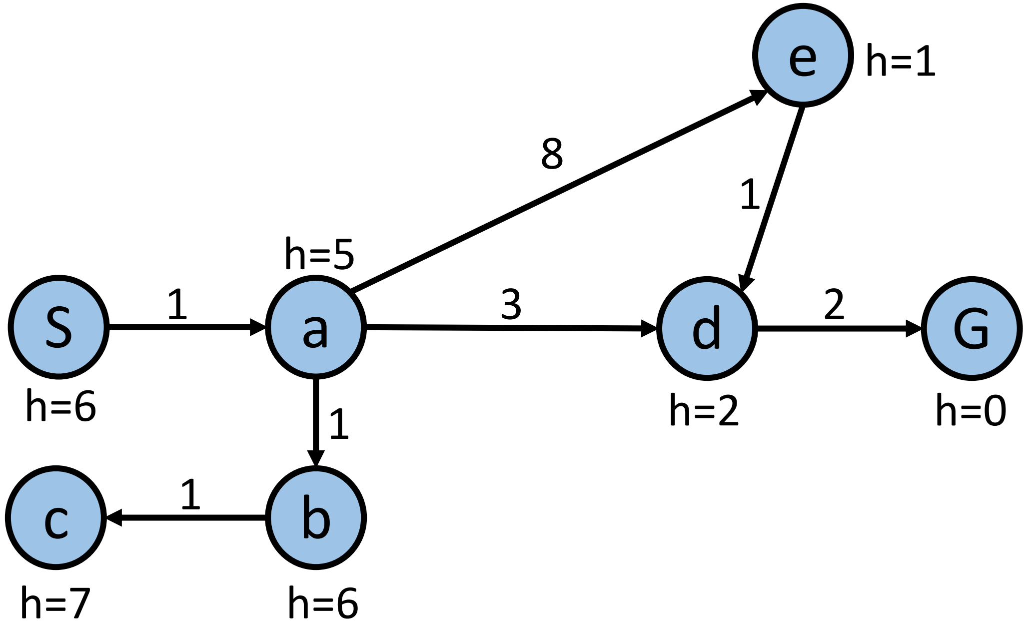
A*:

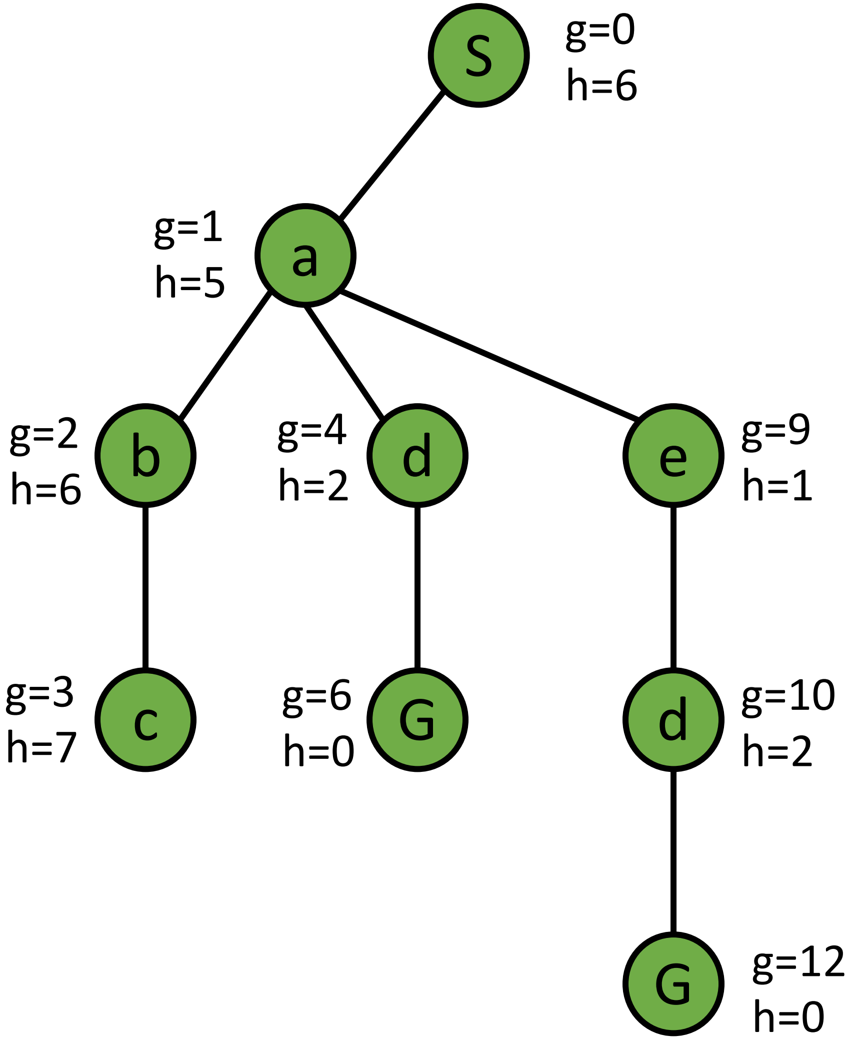
$f(n) = g(n) + h(n)$ (estimated total cost of cheapest solution through *n*)

A* Search

Will be covered on Monday 1/31

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

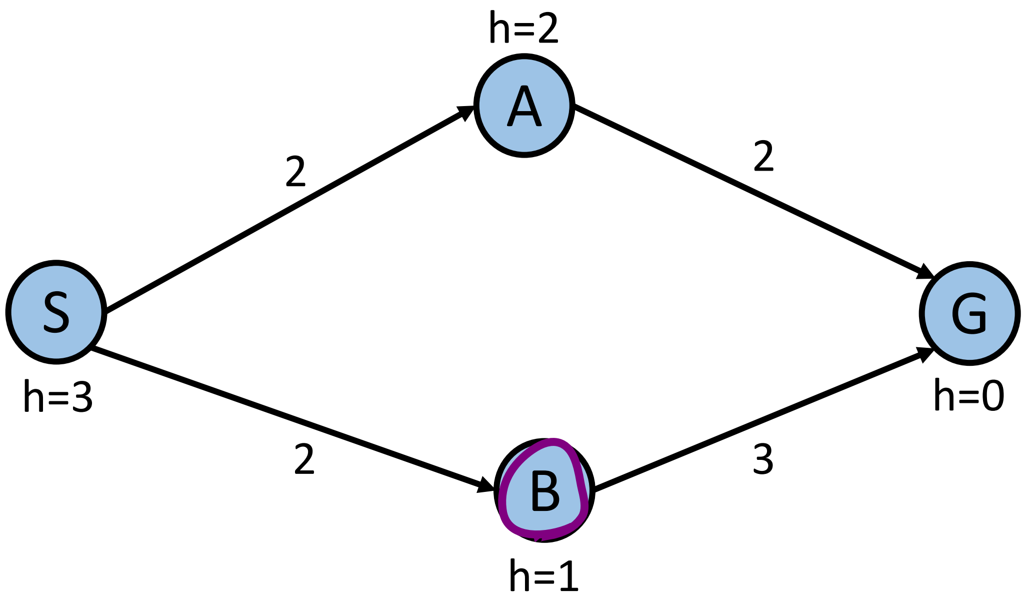




A* Search

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

Example: When should A* search terminate?



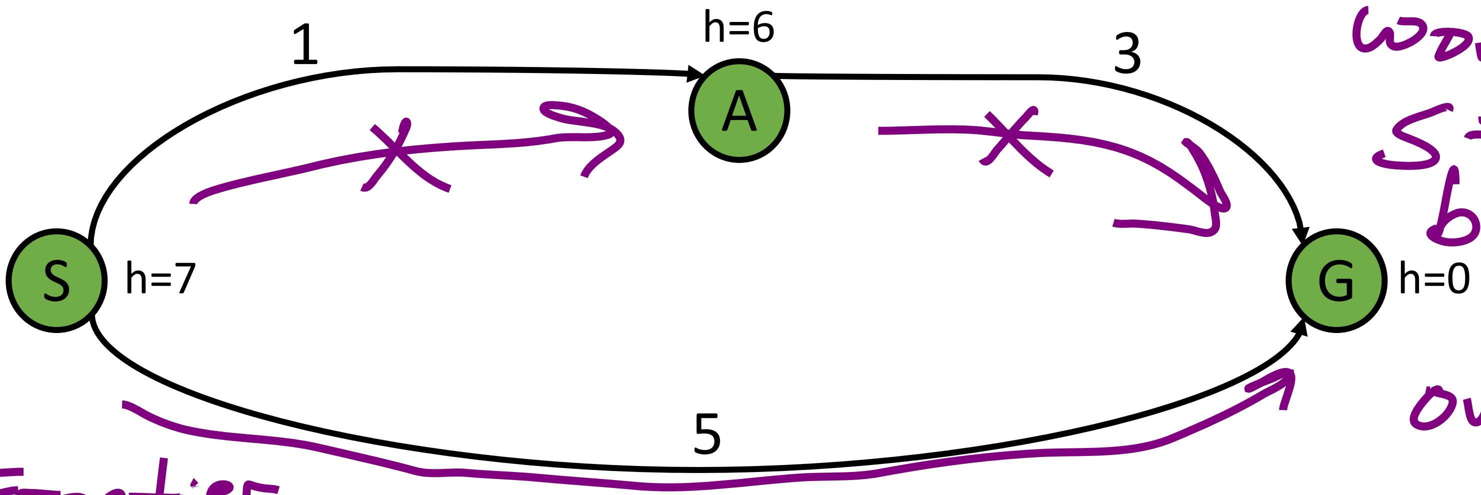
Explored	Frontier
S	(A,4), (B,3)
S, B	(SA,4), (SBG,5)
S, B, A	(SBG,5), (SAG,4)

Once G is in explored set, we have solution for G.

$S \to A \to G = 4$

A* Search

Is A* optimal?



Algorithm won't find S→A→G path because h(n) overestimates cost

Exp	Frontier
S	S→G = 5
	S→A = 1+6=7
S, G	

G will pop from Frontier before paths through A explored.

A* Search

Start here on Monday 1/31

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.

A* Search

Search only works when:

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

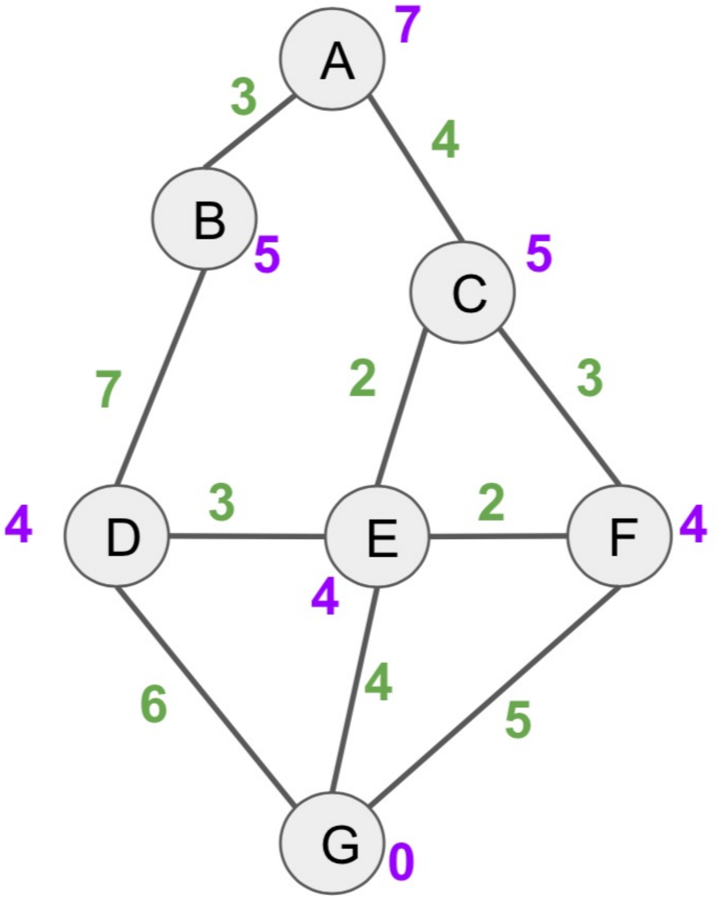
implementation: use a **node**

- state - indicates state at end of path
- action - action taken to get here
- cost - total cost
- parent - pointer to another node

A* Search

A* Search:

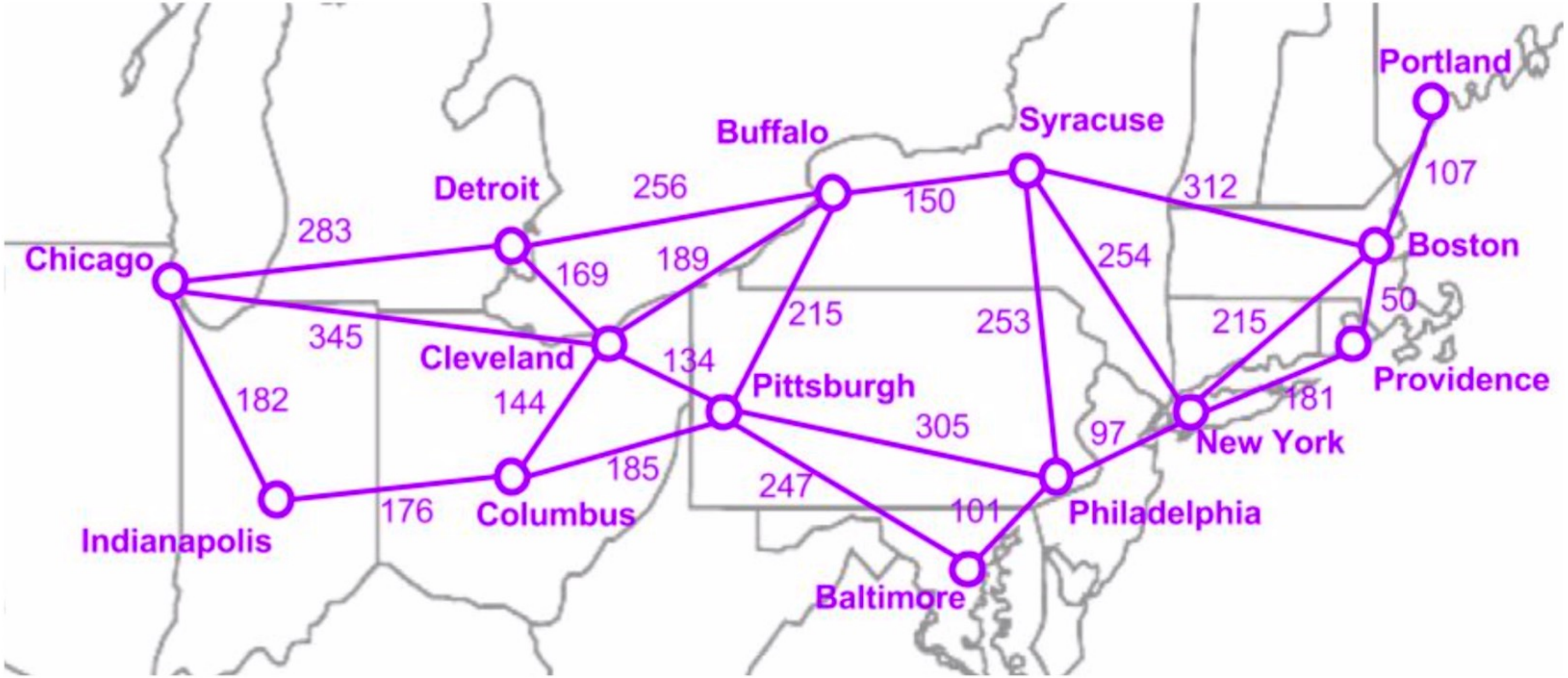
- Find the cheapest path from A to G
- $h(n)$ values are given in **purple**
- Step costs are given in **green**



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



A* Search

Any consistent heuristic is also admissible (but not the other way around).

<u>Example</u>: Prove the above statement by induction.

Next Time

Optimality and Variants of A*