Search

These slides are primarily based on the slides created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley.

The artwork is by Ketrina Yim.

Today

A*

- Creating Heuristics

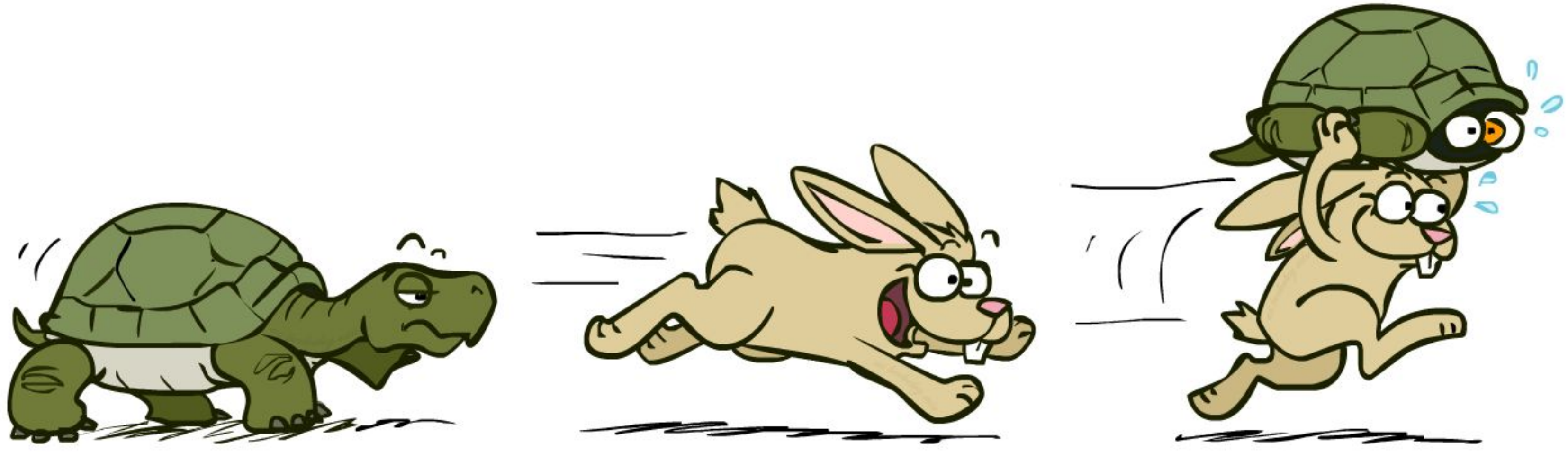
Local Search

- Hill Climbing

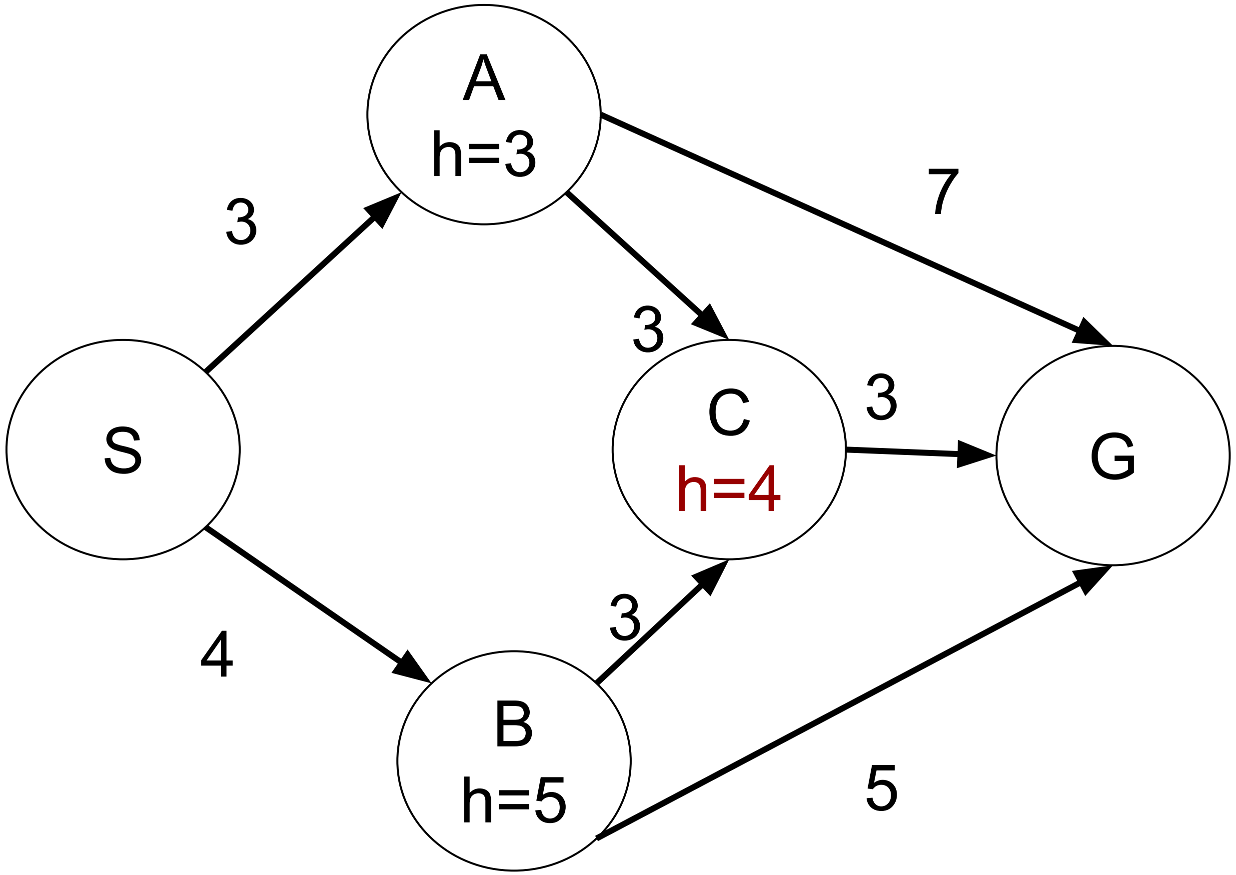
A* Recap

- ✓ A* uses both backward costs and (estimates of) forward costs
- ✓ A* is optimal with admissible / consistent heuristics

➢ Heuristic design is key

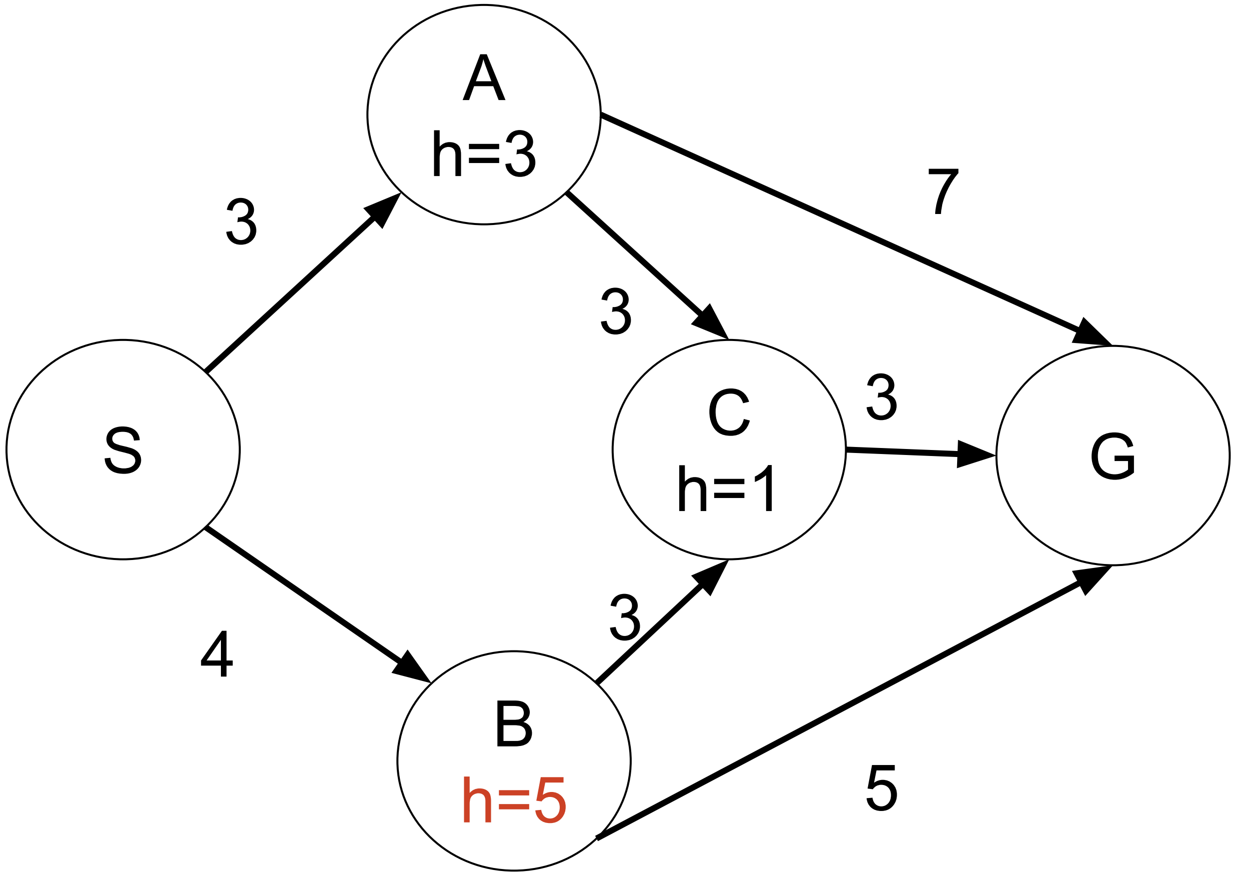


Admissible Heuristic?



A. Yes
B. No

Consistent Heuristic?

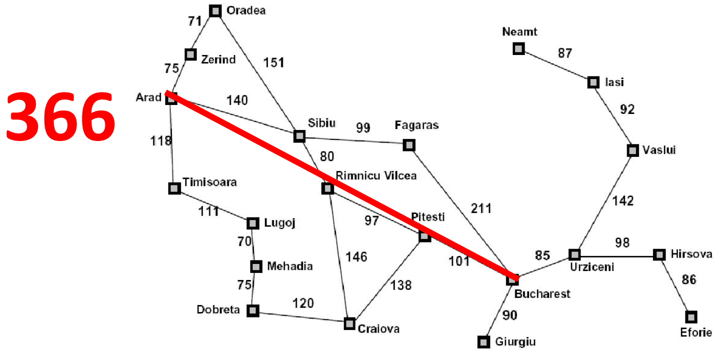


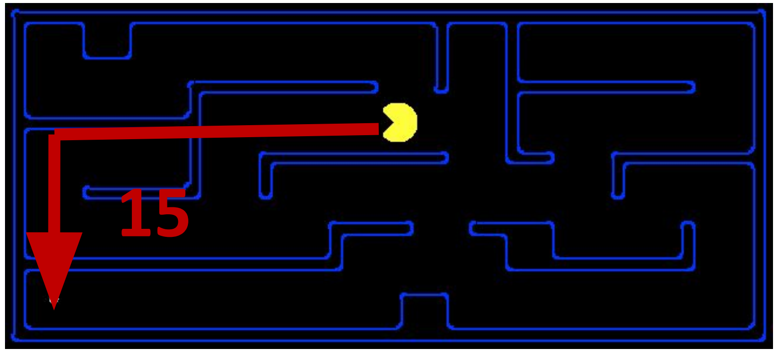
A. Yes
B. No

Creating Admissible Heuristics

- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics

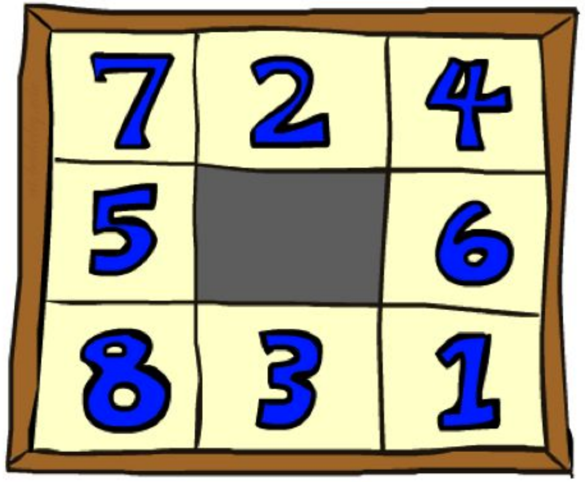
- Often, admissible heuristics are solutions to *relaxed problems,* where new, 'easier' actions are available



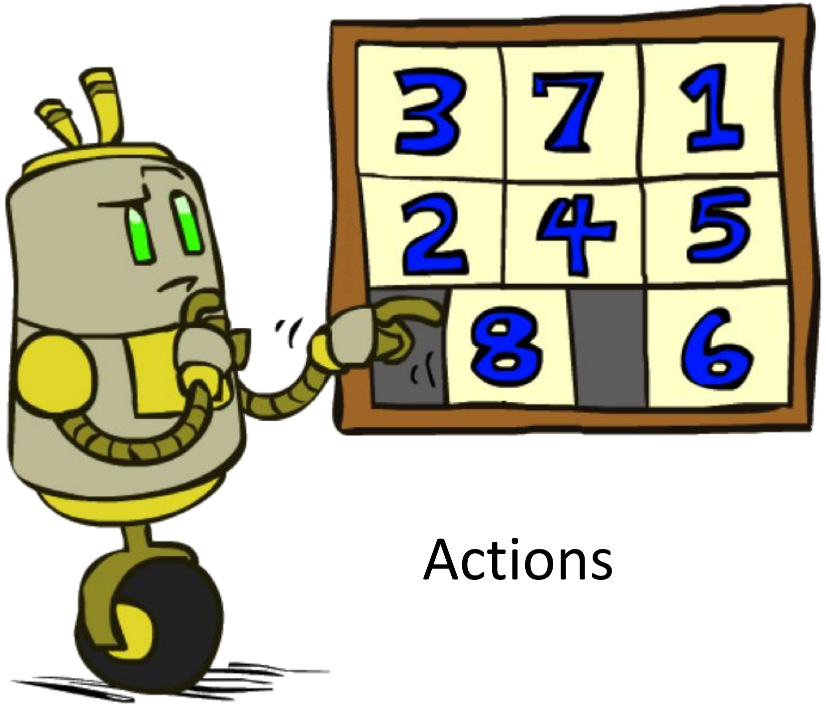


- Inadmissible heuristics are often useful too - suboptimal is sometimes ok

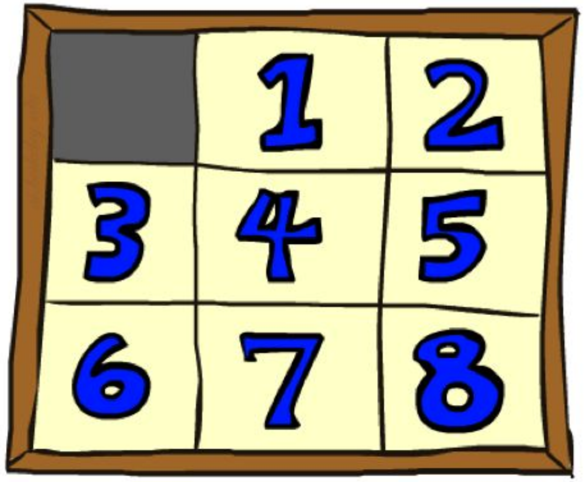
Example: 8 Puzzle



Start State



Actions



Goal State

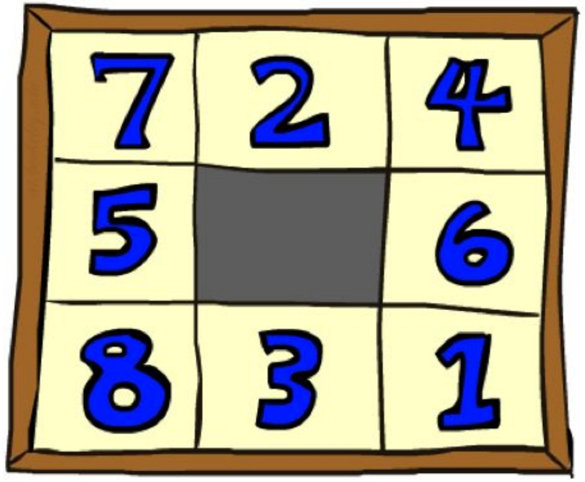
What are the states?

- permutations of 9 squares

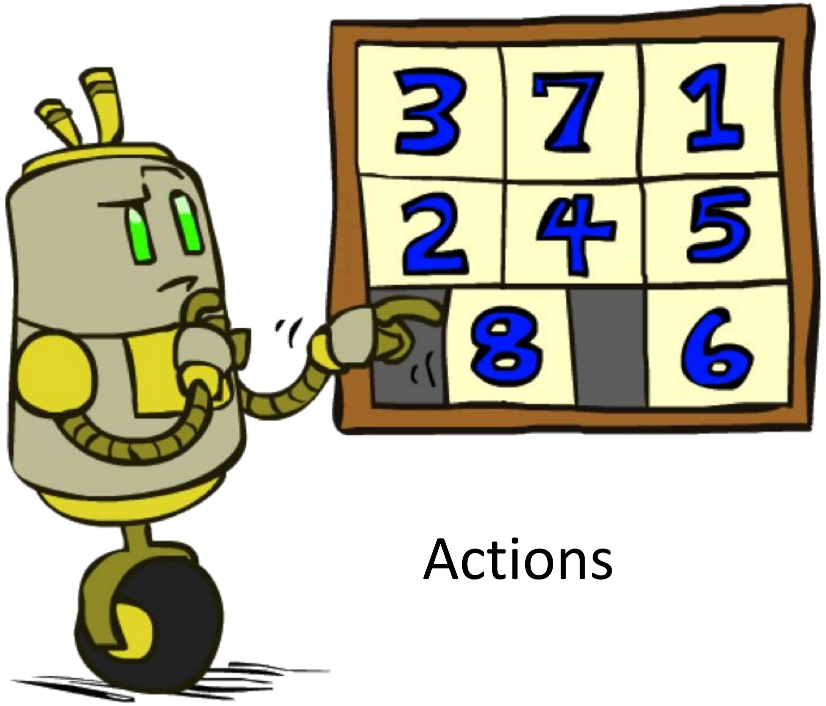
How many states?

- 9!

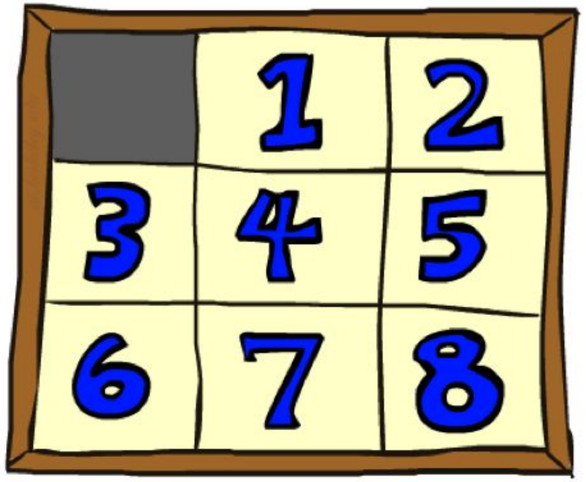
Example: 8 Puzzle



Start State



Actions



Goal State

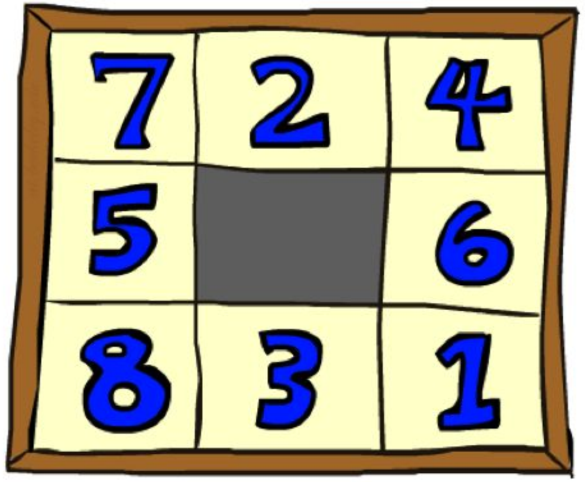
What are the actions?

- move a number square into the empty space

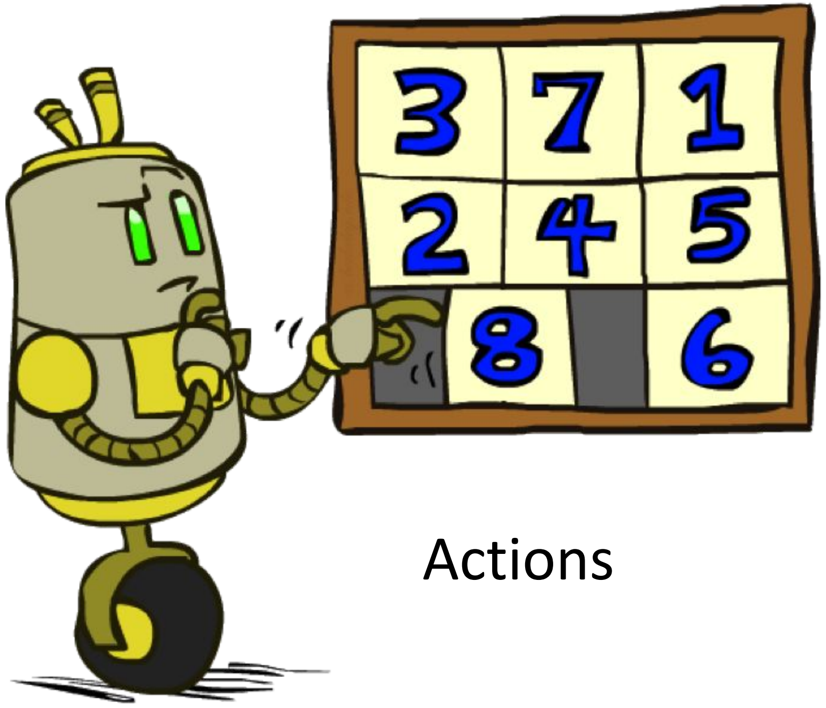
How many successors from the start state?

- 4

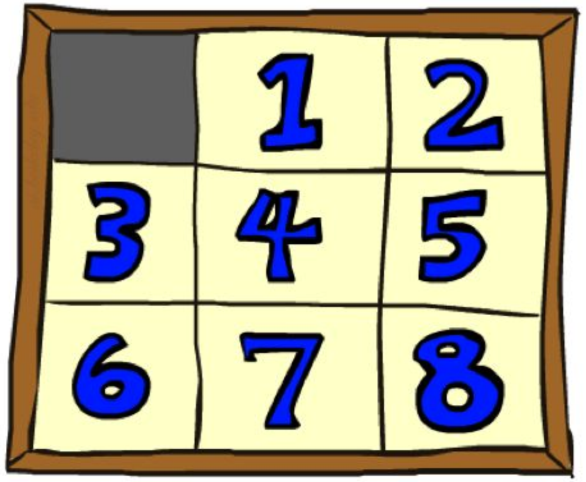
Example: 8 Puzzle



Start State



Actions



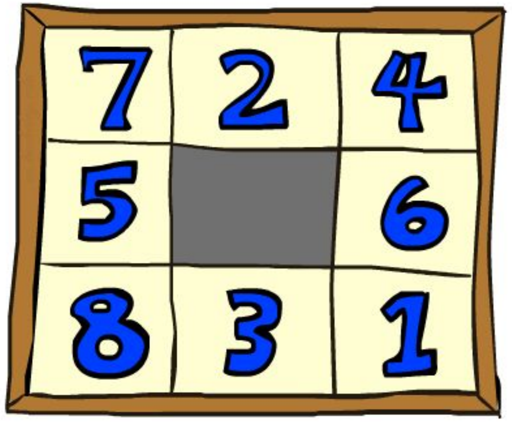
Goal State

Costs?

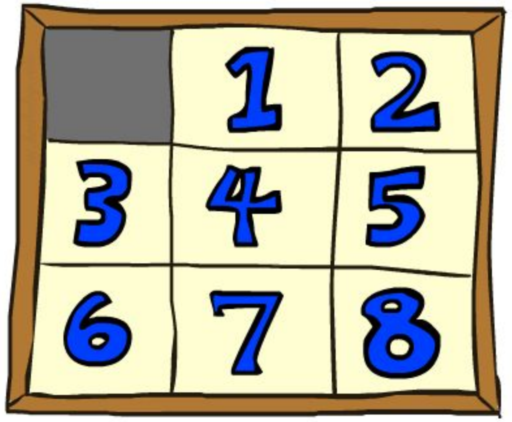
- 1

8 Puzzle I

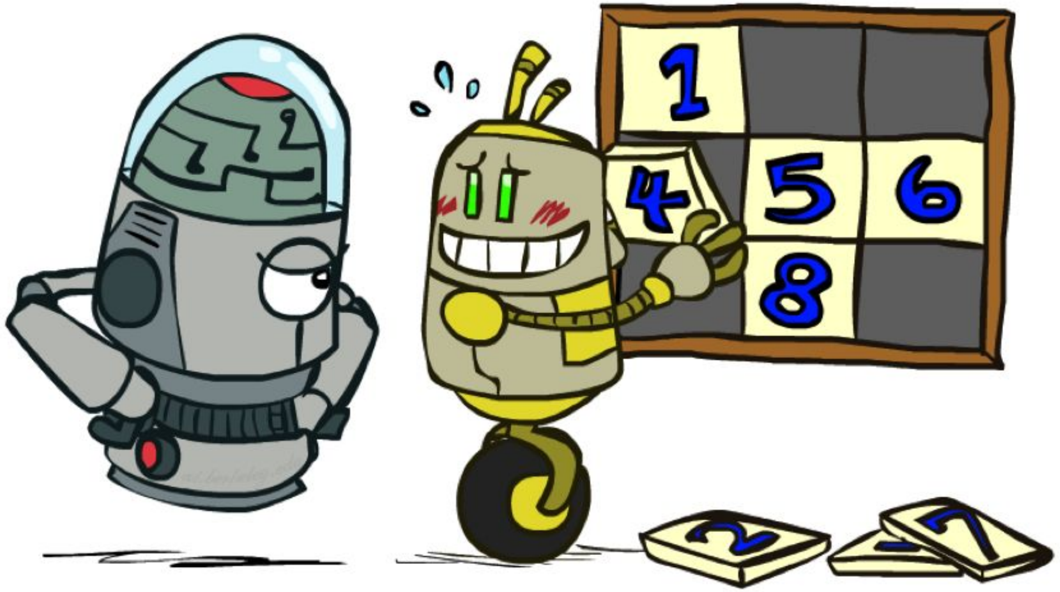
- Heuristic: Number of tiles misplaced
- h(start) = 8
- Why is it admissible?
- This is a *relaxed-problem* heuristic



Start State



Goal State



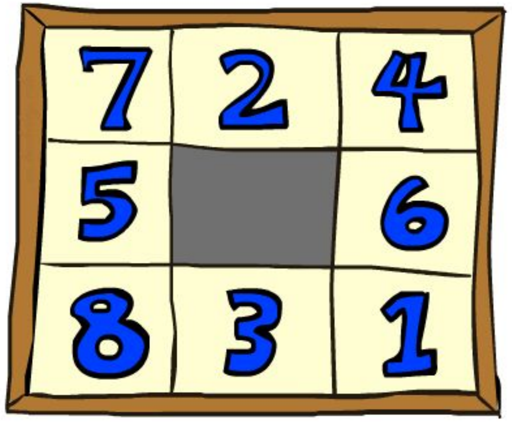
	Average nodes expanded when the optimal path has…		
	…4 steps	…8 steps	…12 steps
UCS	112	6,300	3.6×10^6
TILES	13	39	227

Statistics from Andrew Moore

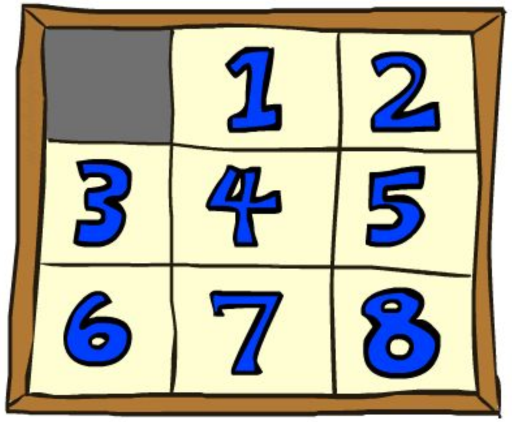
8 Puzzle II

- What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?
- Total *Manhattan* distance
- Why is it admissible?
- h(start) =

  $3 + 1 + 2 + 2 + 3 + 2 + 2 + 3 = 18$



Start State

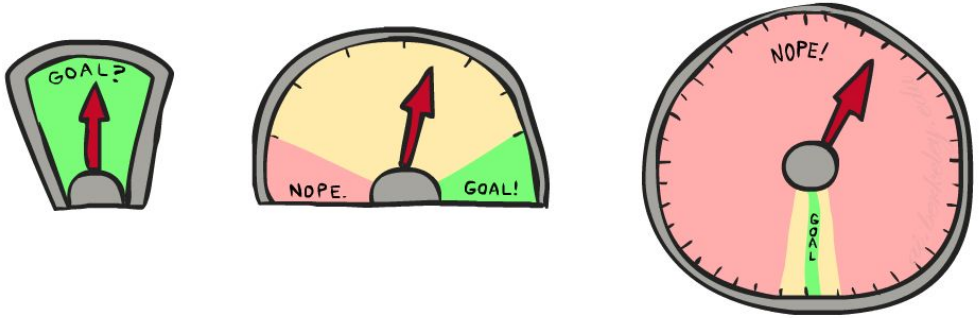


Goal State

	Average nodes expanded when the optimal path has…		
	…4 steps	…8 steps	…12 steps
TILES	13	39	227
MANHATTAN	12	25	73

8 Puzzle III

- How about using the *actual cost* as a heuristic?
  - Would it be admissible?
  - Would we save on nodes expanded?
  - 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, we'll expand fewer nodes but usually do more work per node to compute the heuristic itself

Trivial Heuristics, Dominance

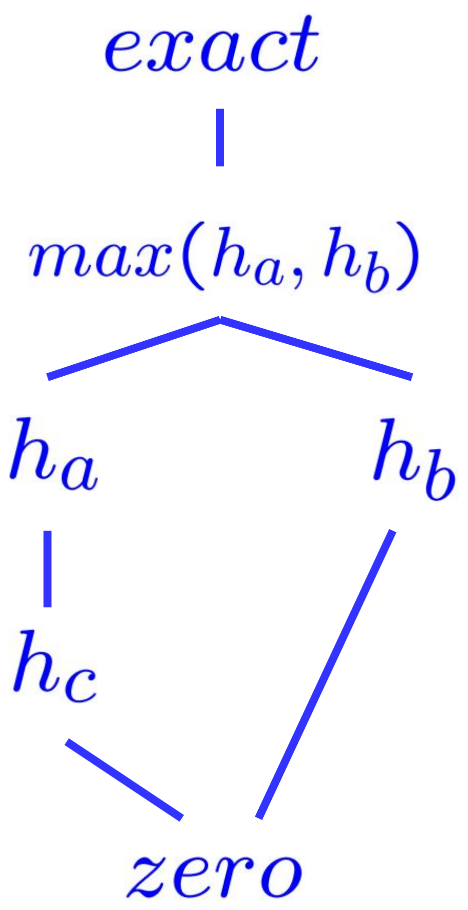
- Dominance: $h_a \geq h_c$ if

$$\forall n : h_a(n) \geq h_c(n)$$

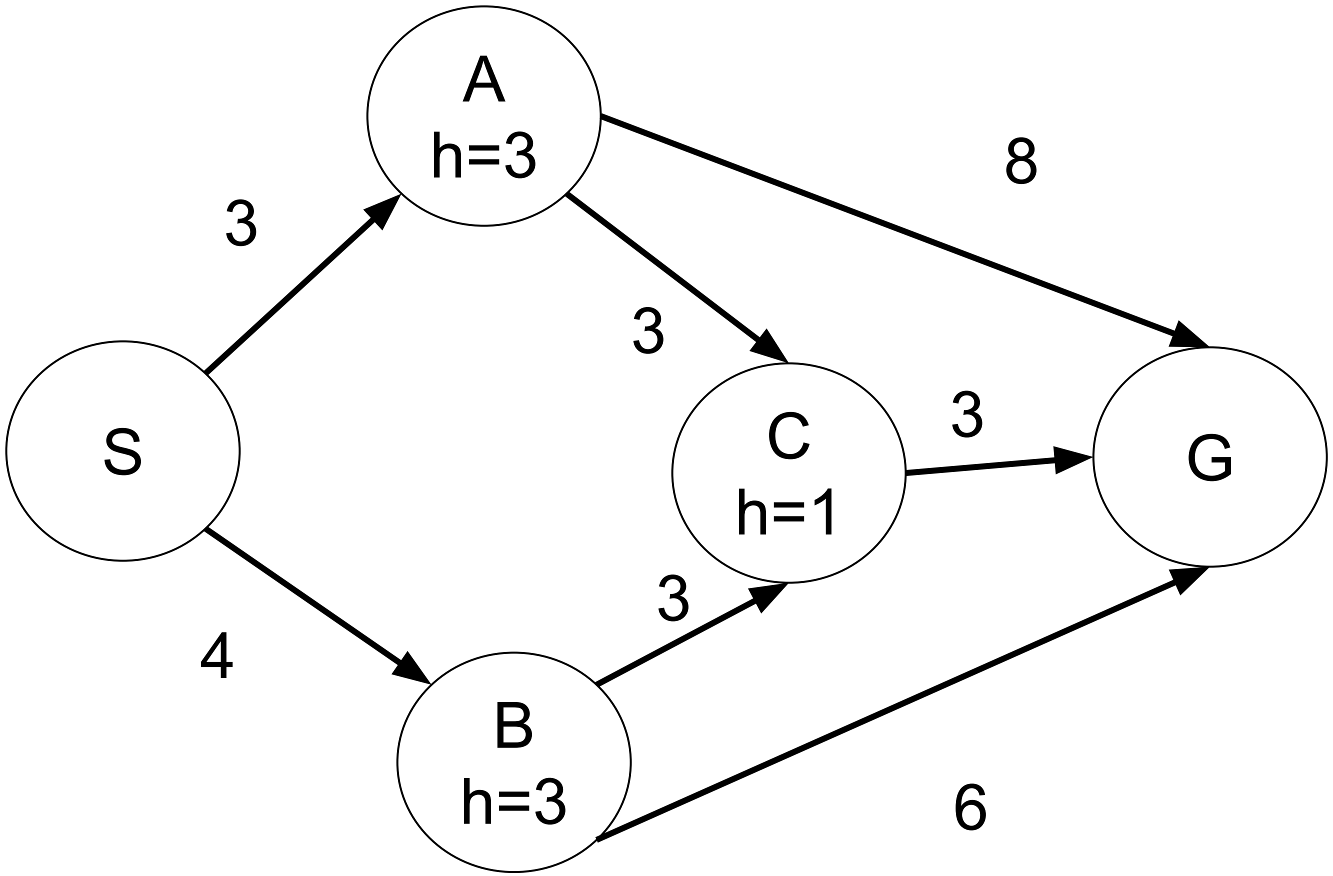
- Heuristics form a semi-lattice:
  - Max of admissible heuristics is admissible

$$h(n) = max(h_a(n), h_b(n))$$

- Trivial heuristics
  - Bottom of lattice is the zero heuristic (what does this give us?)
  - Top of lattice is the exact heuristic



A * Step by Step Expansion

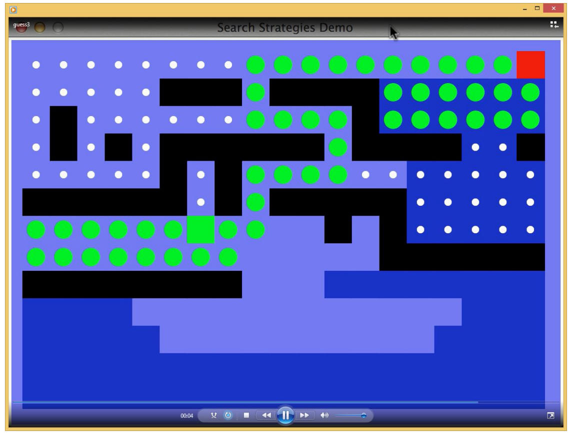


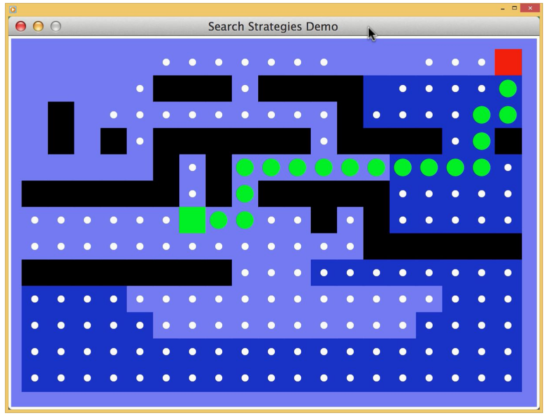
Fringe path	g	h	f
~~S~~	0		
~~S-A~~	3	3	6
~~S-B~~	4	3	7
~~S-A-C~~	6	1	7
S-A-G	11	0	11
S-B-C	7	1	8
S-B-G	10	0	10
S-A-C-G	9	0	9

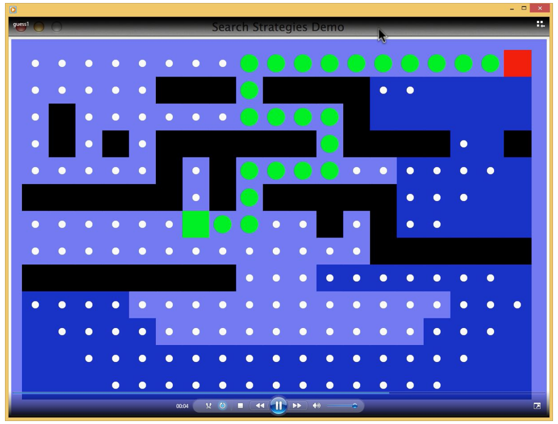
Nodes expanded: SABCG

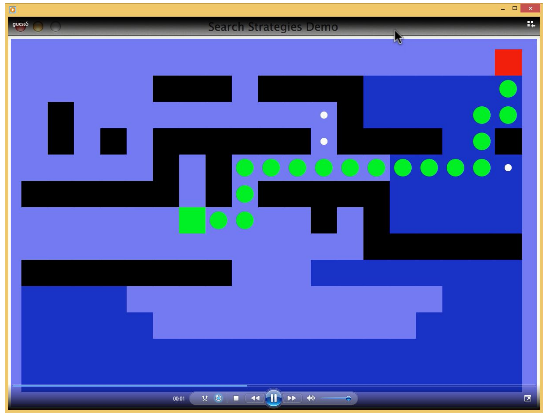
Search Recap

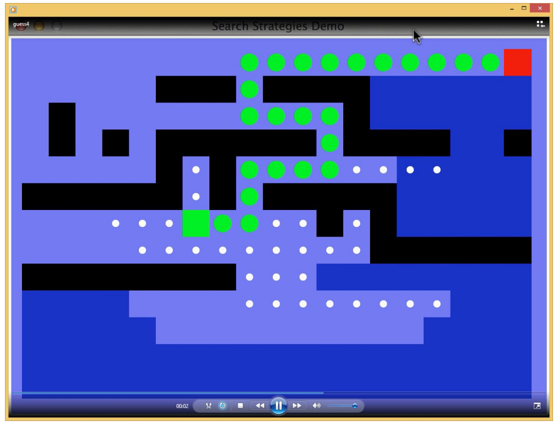
A. DFS
B. BFS
C. UCS
D. GREEDY
E. ASTAR











Different Problems, Different Search

- In some problems, the goal state itself is important, not the path
- Optimization problem: find the best state given an objective function
- Identification problems, scheduling problems, etc...
- We can use iterative improvement algorithms (local search) such as hill climbing

Local Search

Tree/Graph search keeps unexplored alternatives on the fringe to ensure completeness

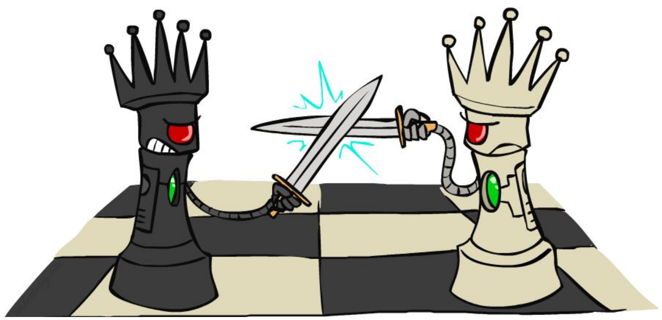
Local search:

- keep a single 'current' node (no fringe)
- try to improve it until we can’t make it better

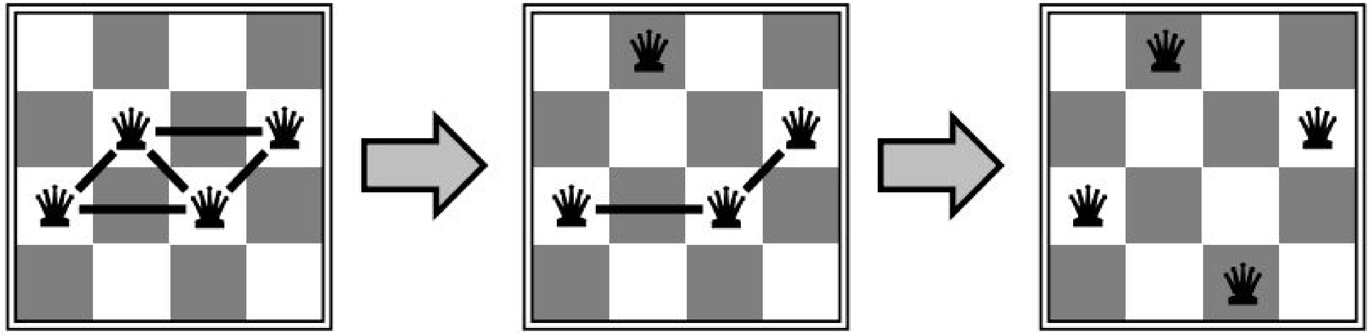
Generally much faster and more memory efficient (but incomplete and suboptimal)

Example: n-Queens

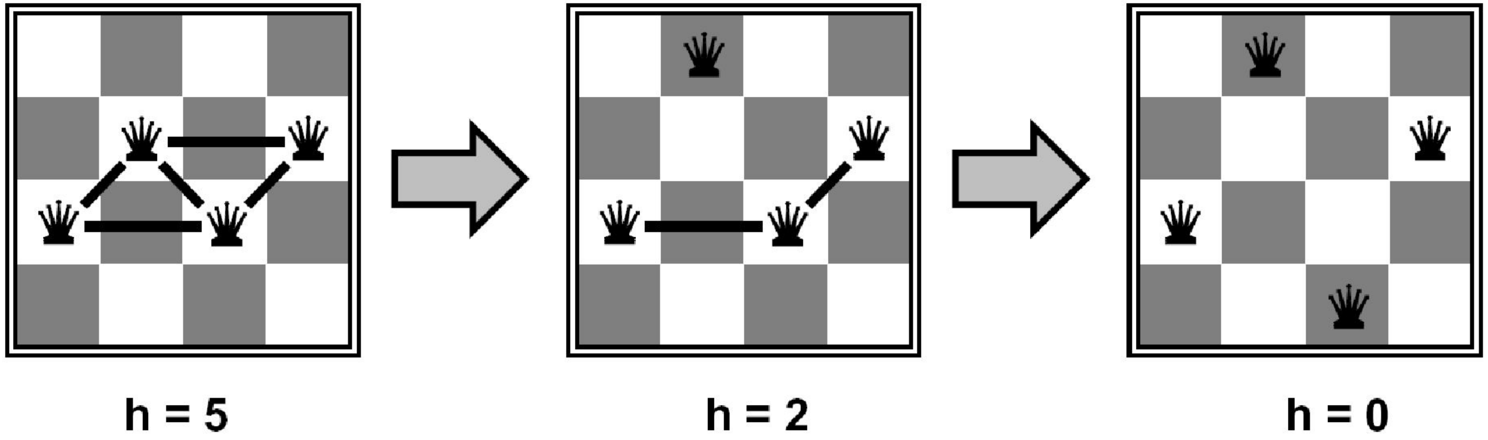
Place n queens on an n × n board with no two queens on the same row, column, or diagonal.



Start with any configuration.   Move a queen to reduce number of conflicts.



Example: 4-Queens



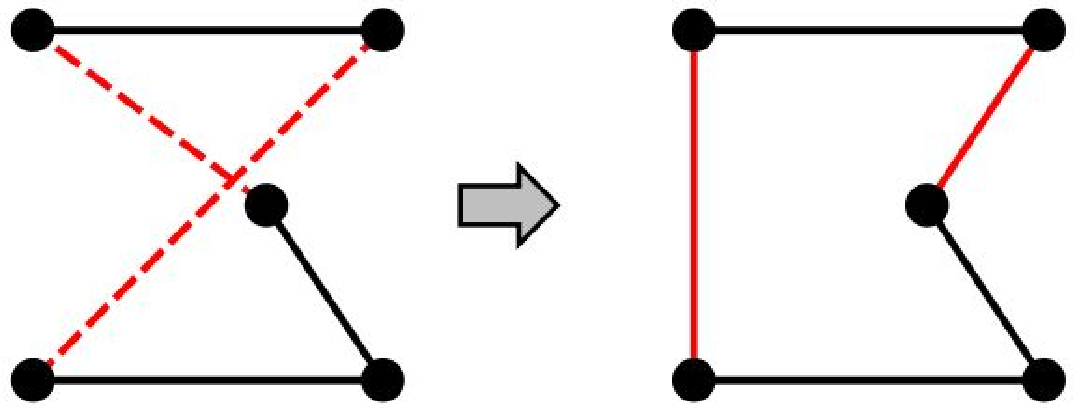
- States: 4 queens in 4 columns ($4^4 = 256$ states)
- Actions: move queen in column
- Goal test: no conflict
- Evaluation: c(n) = number of conflicts

Example: Travelling SalesPerson Problem

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? It is an NP-hard problem.

Start with any complete tour, perform pairwise exchanges.

Variants of this approach get within 1% of optimal very quickly with thousands of cities

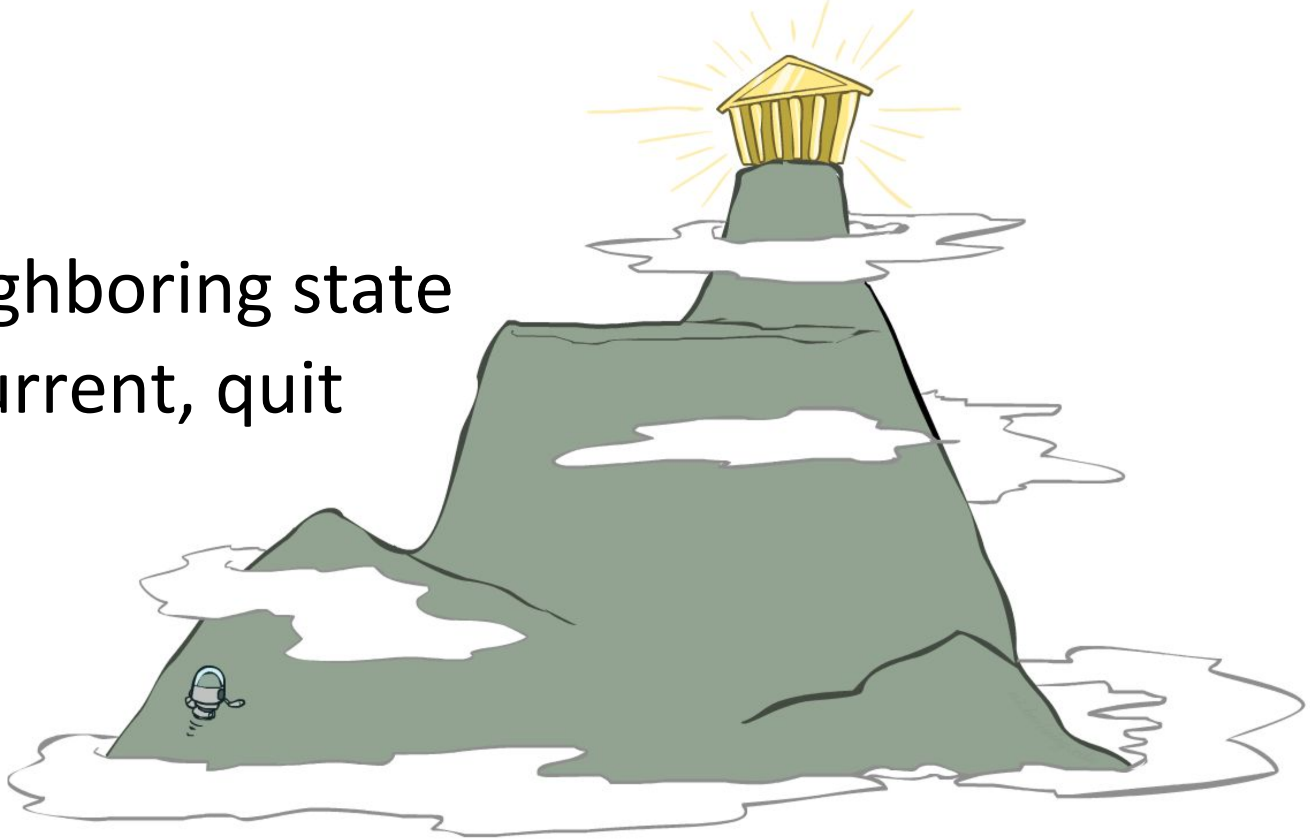


Hill Climbing (gradient ascent/descent)

'Like climbing Everest in thick fog with amnesia'

Simple, general idea:

- Start wherever
- Repeat: move to the best neighboring state
- If no neighbors better than current, quit



Hill Climbing (gradient ascent/descent)

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
    end
```

Hill Climbing

Starting from X, where do we end up ?
B
Starting from Y, where do we end up ?
D
Starting from Z, where do we end up ?
E