Artificial Intelligence

Search - more
(Chapter 3)

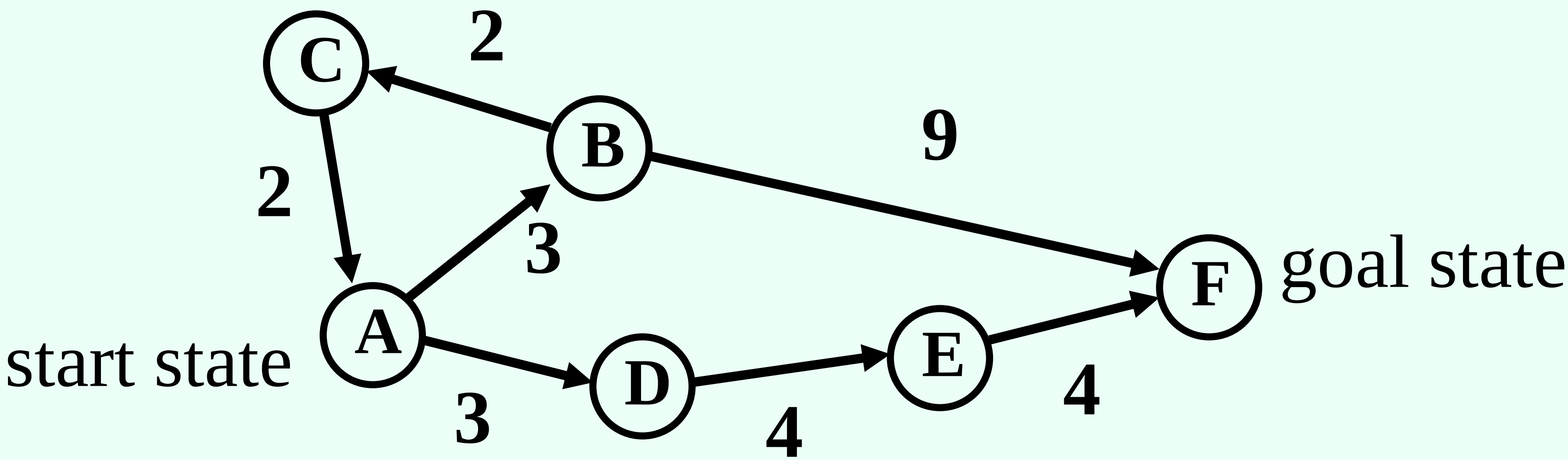
Instructor: Qiang Yu

Informed search

- So far, have assumed that no nongoal state looks better than another
- Unrealistic
  - Even without knowing the road structure, some locations seem closer to the goal than others
  - Some states of the 8s puzzle seem closer to the goal than others
- Makes sense to expand closer-seeming nodes first

Heuristics

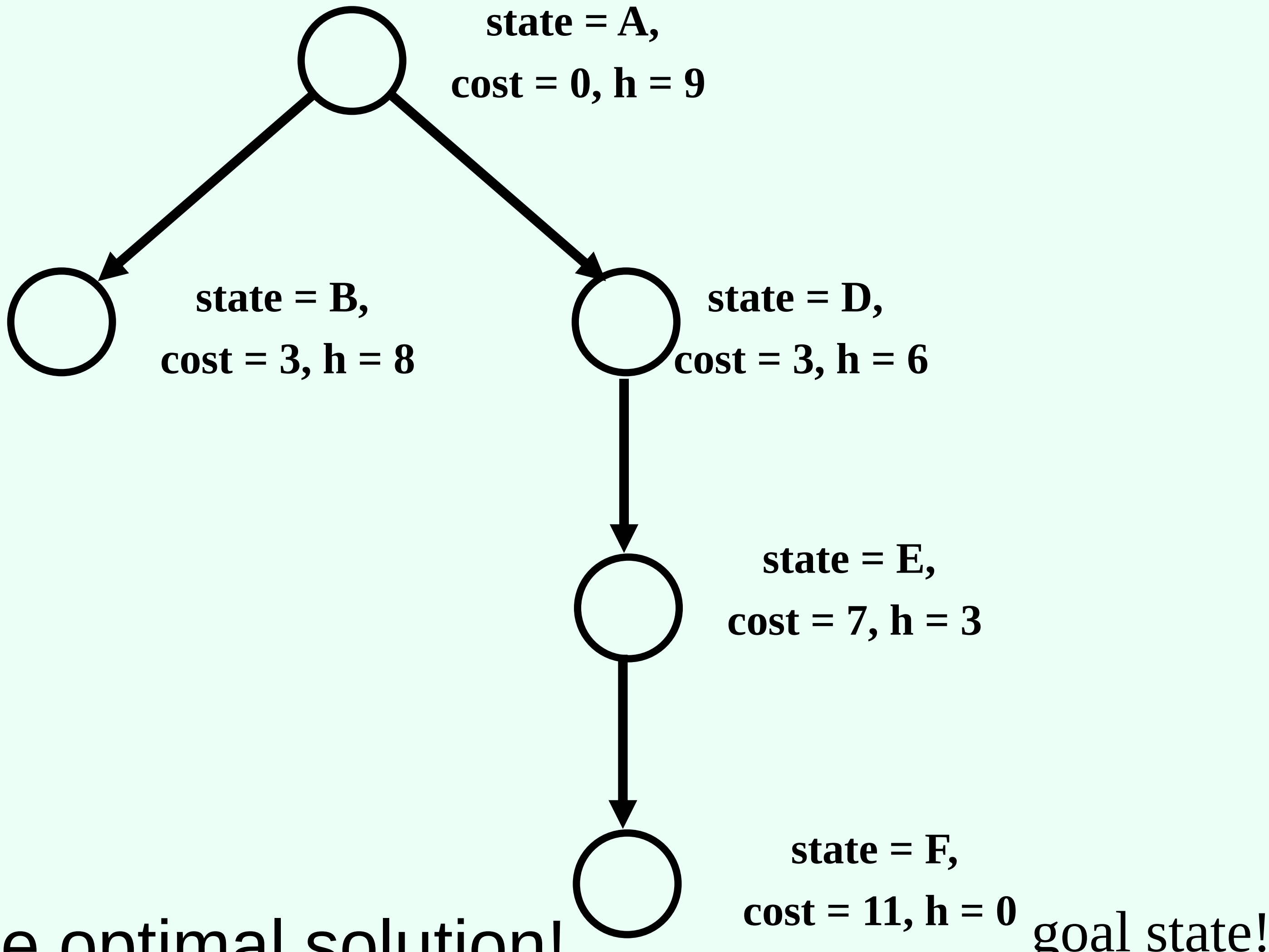
- Key notion: heuristic function h(n) gives an estimate of the distance from n to the goal
  - h(n)=0 for goal nodes
- E.g. straight-line distance for traveling problem



- Say: h(A) = 9, h(B) = 8, h(C) = 9, h(D) = 6, h(E) = 3, h(F) = 0
- We're adding something new to the problem!
- Can use heuristic to decide which nodes to expand first

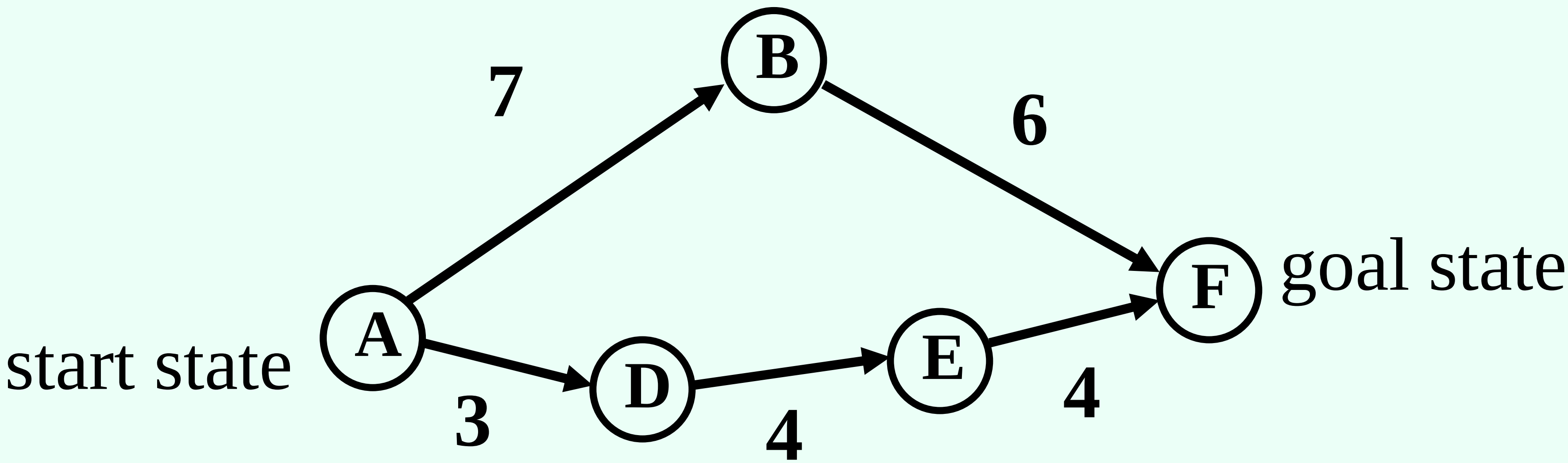
Greedy best-first search

- Greedy best-first search: expand nodes with lowest h values first



- Rapidly finds the optimal solution!
- Does it always?

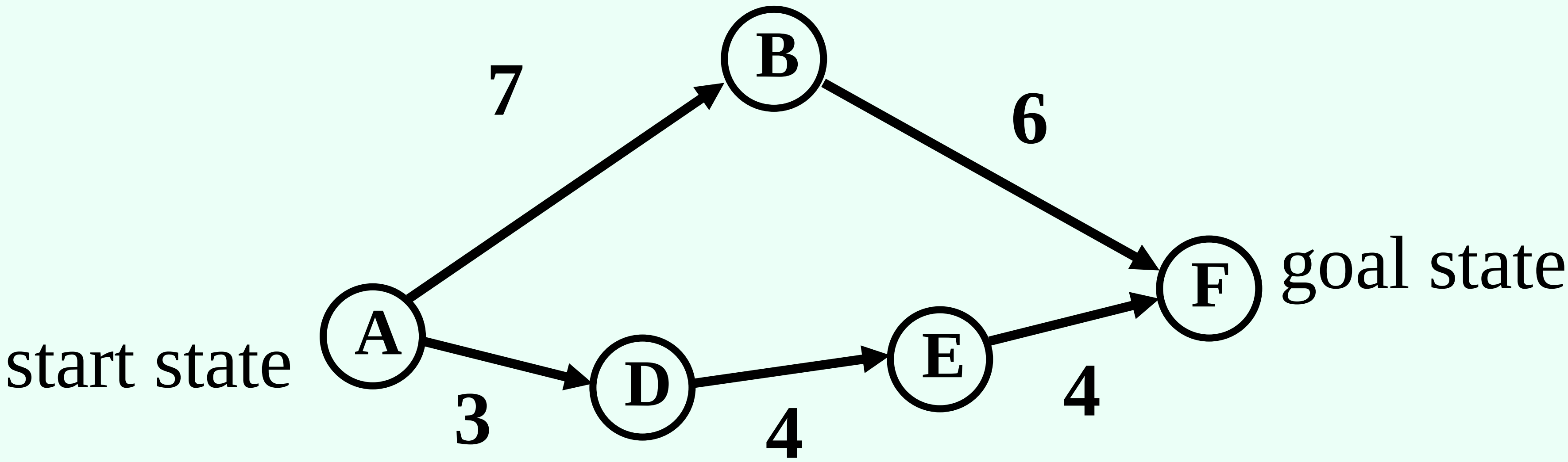
A bad example for greedy



- Say: h(A) = 9, h(B) = 5, h(D) = 6, h(E) = 3, h(F) = 0
- Problem: greedy evaluates the promise of a node only by how far is left to go, does not take cost occurred already into account

A*

- Let g(n) be cost incurred already on path to n
- Expand nodes with lowest g(n) + h(n) first



- Say: h(A) = 9, h(B) = 5, h(D) = 6, h(E) = 3, h(F) = 0

- Note: if h=0 everywhere, then just uniform cost search

Admissibility

- A heuristic is admissible if it never overestimates the distance to the goal
  - If n is the optimal solution reachable from n’, then g(n) ≥ g(n’) + h(n’)
- Straight-line distance is admissible: can’t hope for anything better than a straight road to the goal
- Admissible heuristic means that A* is always optimistic

Consistency

- A heuristic is consistent if the following holds: if one step takes us from n' to n, then $h(n') \leq h(n)$ + cost of step from n' to n
  - Similar to triangle inequality
  - Equivalently, $g(n')+h(n') \leq g(n)+h(n)$
- Implies admissibility

Iterative Deepening A*

- One big drawback of A* is the space requirement: similar problems as uniform cost search, BFS
- Limited-cost depth-first A*: some cost cutoff c, any node with $g(n)+h(n) > c$ is not expanded, otherwise DFS

More search:
When the path to the solution doesn't matter

(Chapter 4，6)

Search where the path doesn't matter

- So far, looked at problems where the path was the solution
  - Traveling on a graph
  - Eights puzzle
- However, in many problems, we just want to find a goal state
  - Doesn't matter how we get there

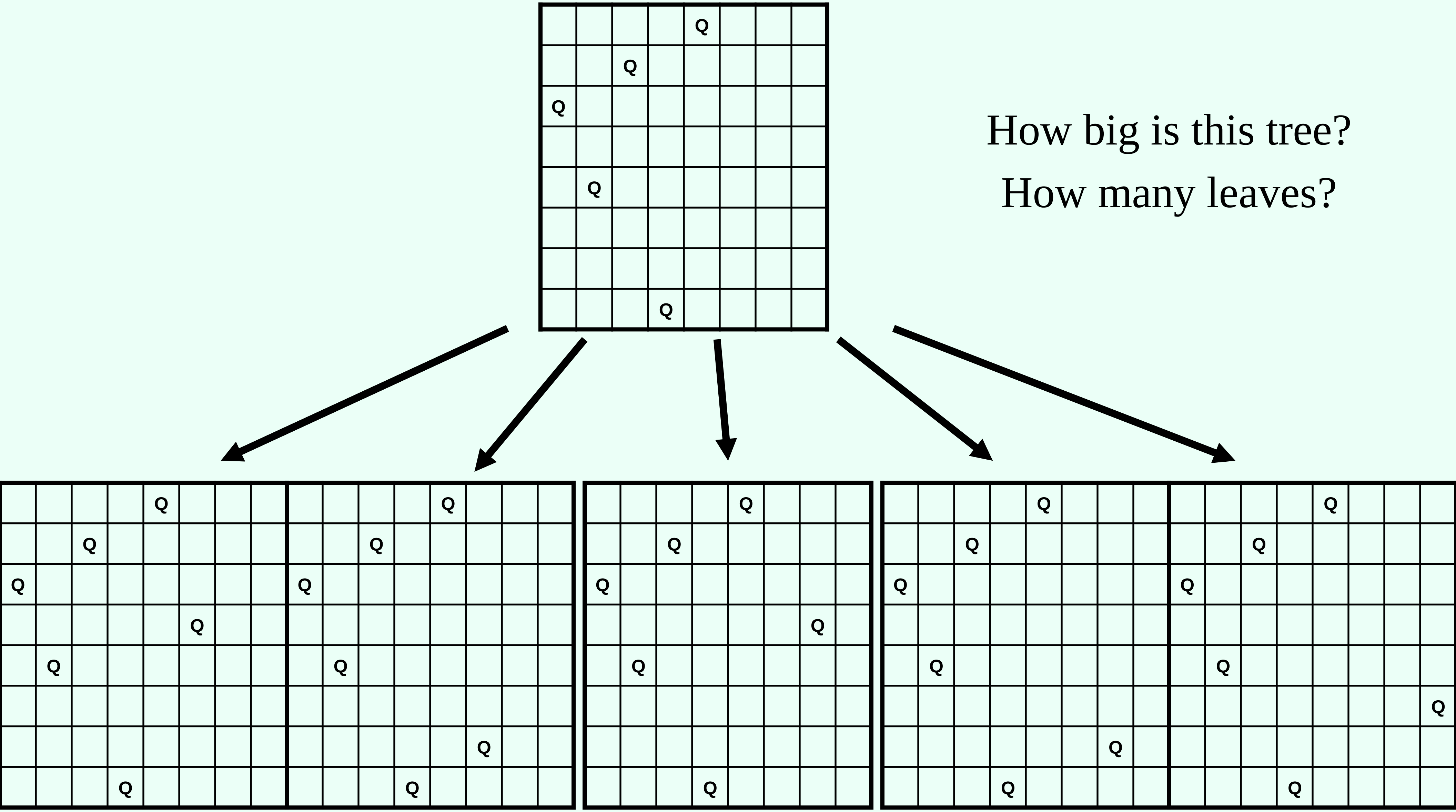
Queens puzzle

- Place eight queens on a chessboard so that no two attack each other

				Q			
		Q					
Q							
						Q	
	Q						
							Q
					Q		
			Q				

Search formulation of the queens puzzle

- **Successors**: all valid ways of placing additional queen on the board; **goal**: eight queens placed

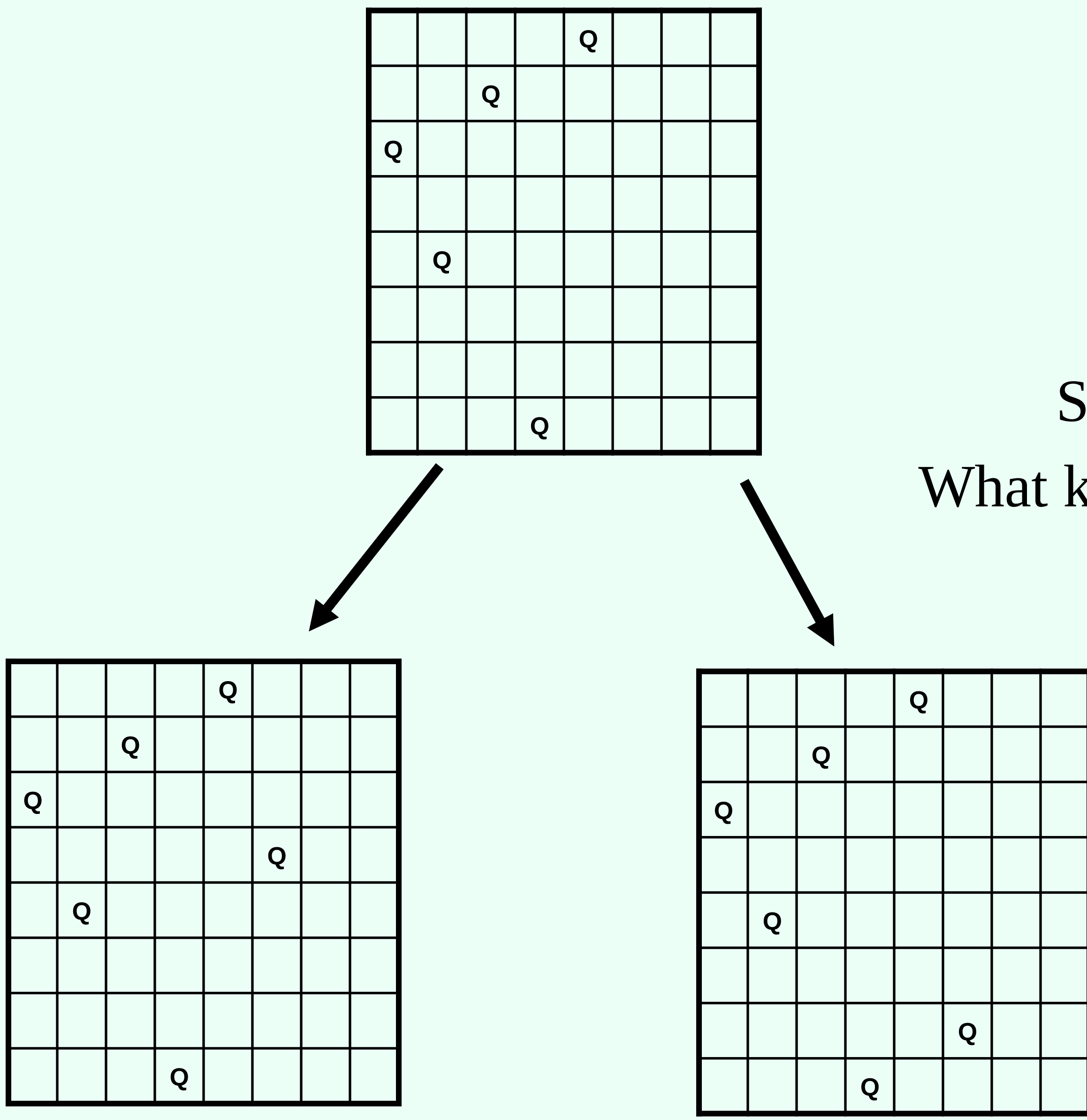


How big is this tree?

How many leaves?

Search formulation of the queens puzzle

- **Successors**: all valid ways of placing a queen in the next column; **goal**: eight queens placed



Search tree size?

What kind of search is best?

Constraint satisfaction problems (CSPs)

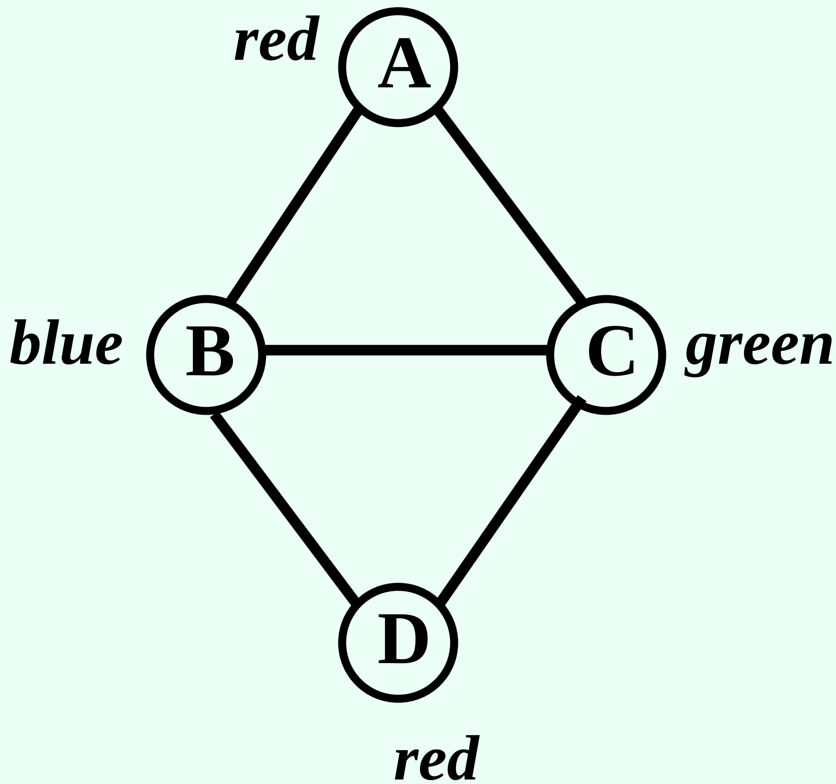
- Defined by:
  - A set of variables $x_1, x_2, \ldots, x_n$
  - A domain D_i for each variable x_i
  - Constraints $c_1, c_2, \ldots, c_m$
- A constraint is specified by
  - A subset (often, two) of the variables
  - All the allowable joint assignments to those variables
- Goal: find a complete, consistent assignment
- Queens problem: (other examples in next slides)
  - x_i in {1, …, 8} indicates in which row in the ith column to place a queen
  - For example, constraint on x_1 and x_2: {(1,3), (1,4), (1,5), (1,6), (1,7), (1,8), (2,4), (2,5), …, (3,1), (3,5), … …}

Meeting scheduling

- Meetings A, B, C, … need to be scheduled on M, Tu, W, Th, F
- A and B cannot be scheduled on the same day
- B needs to be scheduled at least two days before C
- C cannot be scheduled on Th or F
- Etc.
- How do we model this as a CSP?

Graph coloring

- Fixed number of colors; no two adjacent nodes can share a color



Recap

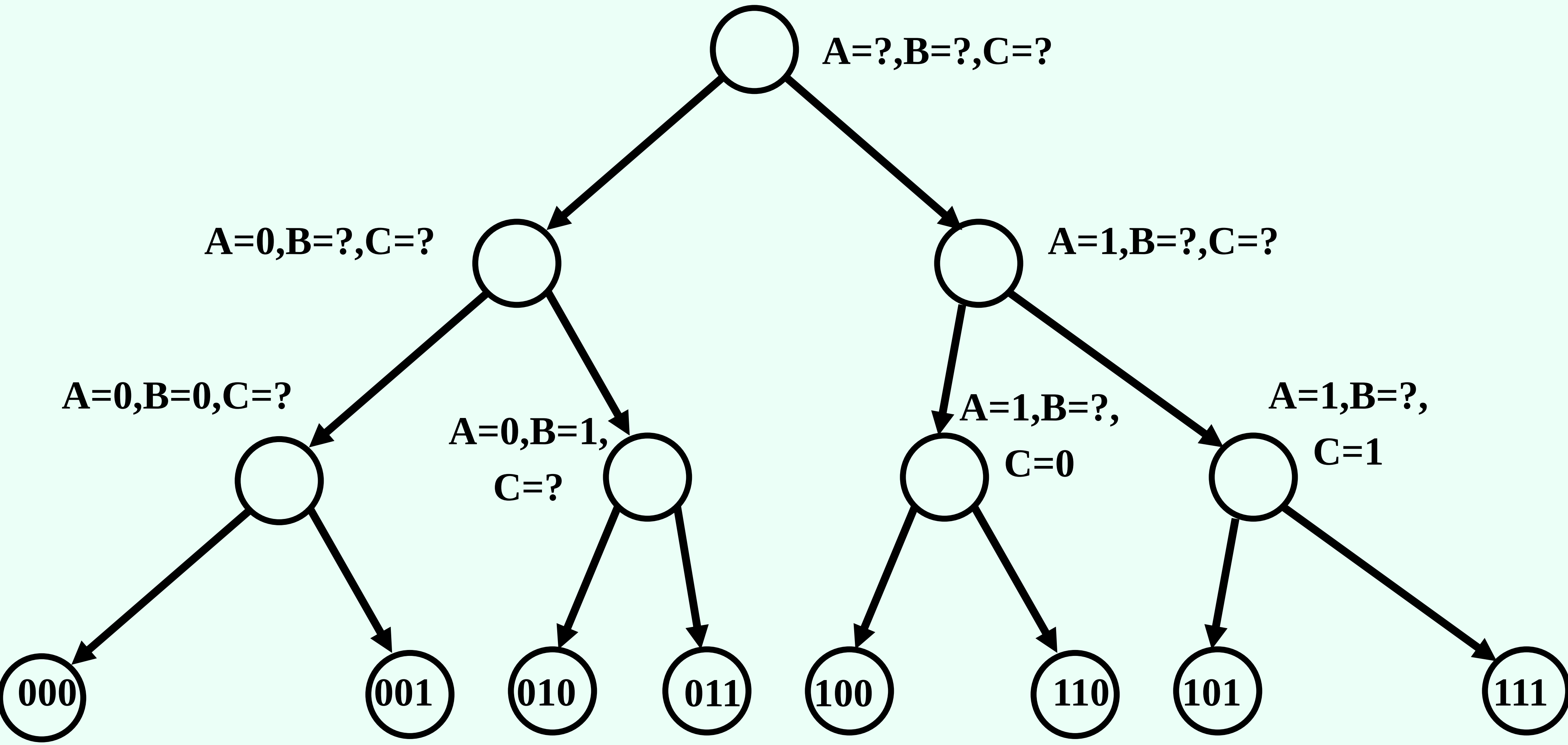
- Greedy best-first search: expand nodes with lowest h values first
- A* search: f=g+h
- Admissibility: A heuristic is admissible if it never overestimates the distance to the goal.
- Consistency: if one step takes us from n’ to n, then h(n’) ≤ h(n) + cost of step from n’ to n
- Constraint satisfaction problems (CSPs)

Generic approaches to solving CSPs

- State: some variables assigned, others not assigned
- Naïve successors definition: any way of assigning a value to an unassigned variable results in a successor
  - How many leaves do we get in the worst case?
- CSPs satisfy commutativity: order in which actions applied does not matter
- Better idea: only consider assignments for a single variable at a time
  - How many leaves?

Choice of variable to branch on is still flexible!

- Each of variables A, B, C takes values in {0,1}



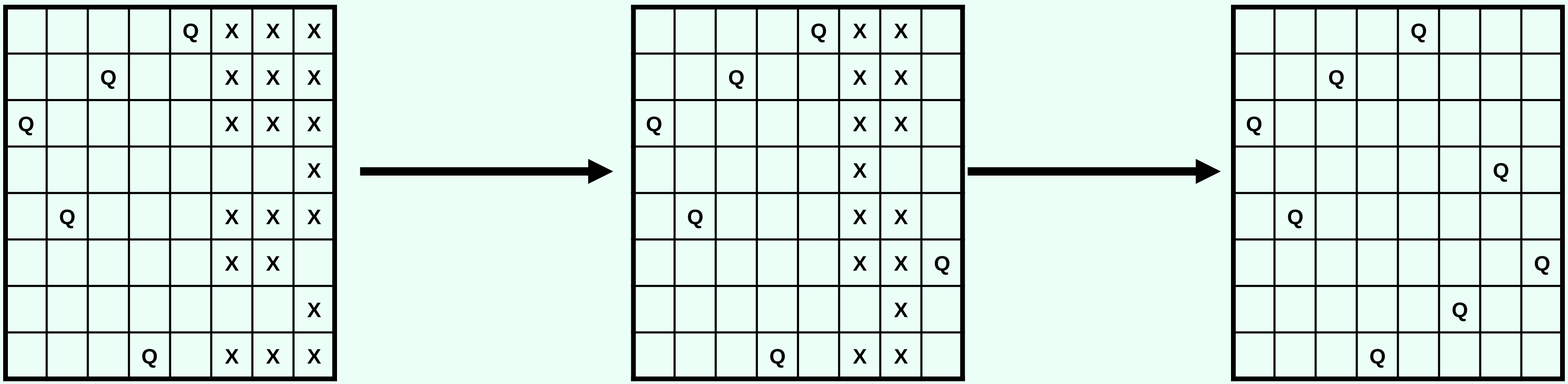
A generic recursive search algorithm

(*assignment* is a partial assignment)

- **Search(*assignment*, *constraints*)**
- If *assignment* is complete, return it
- Choose an unassigned variable *x*
- For every value *v* in *x*'s domain, if setting *x* to *v* in *assignment* does not violate *constraints:*
  - Set *x* to *v* in *assignment*
  - *result* := Search(*assignment*, *constraints*)
  - If *result* != *failure* return *result*
  - Unassign *x* in *assignment*
- Return *failure*

Keeping track of remaining possible values

- For every variable, keep track of which values are still possible



only one possibility for last column; might as well fill in

now only one left for other two columns

done!
(no real branching needed!)

- General heuristic: branch on variable with fewest values remaining

Arc consistency

- Take two variables connected by a constraint
- Is it true that for **every** remaining value *d* of the first variable, there exists **some** value *d'* of the other variable so that the constraint is satisfied?
  - If so, we say the arc from the first to the second variable is consistent
  - If not, can remove the value *d*
- General concept: constraint propagation

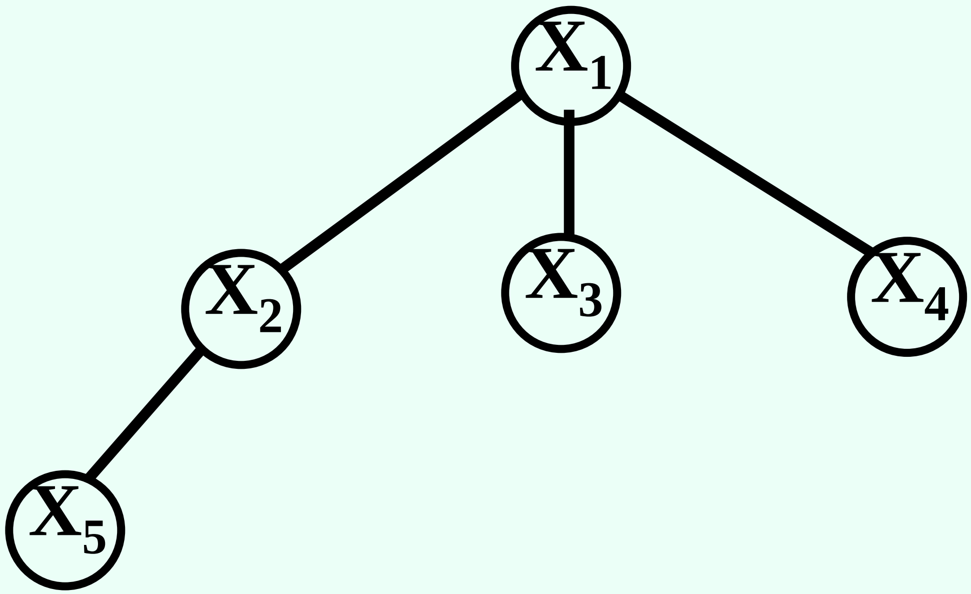
Q				x			x
				x			
		Q		x			x
				x			
				x	Q		x
				x			

Is the arc from the fifth to the eighth column consistent?

What about the arc from the eighth to the fifth?

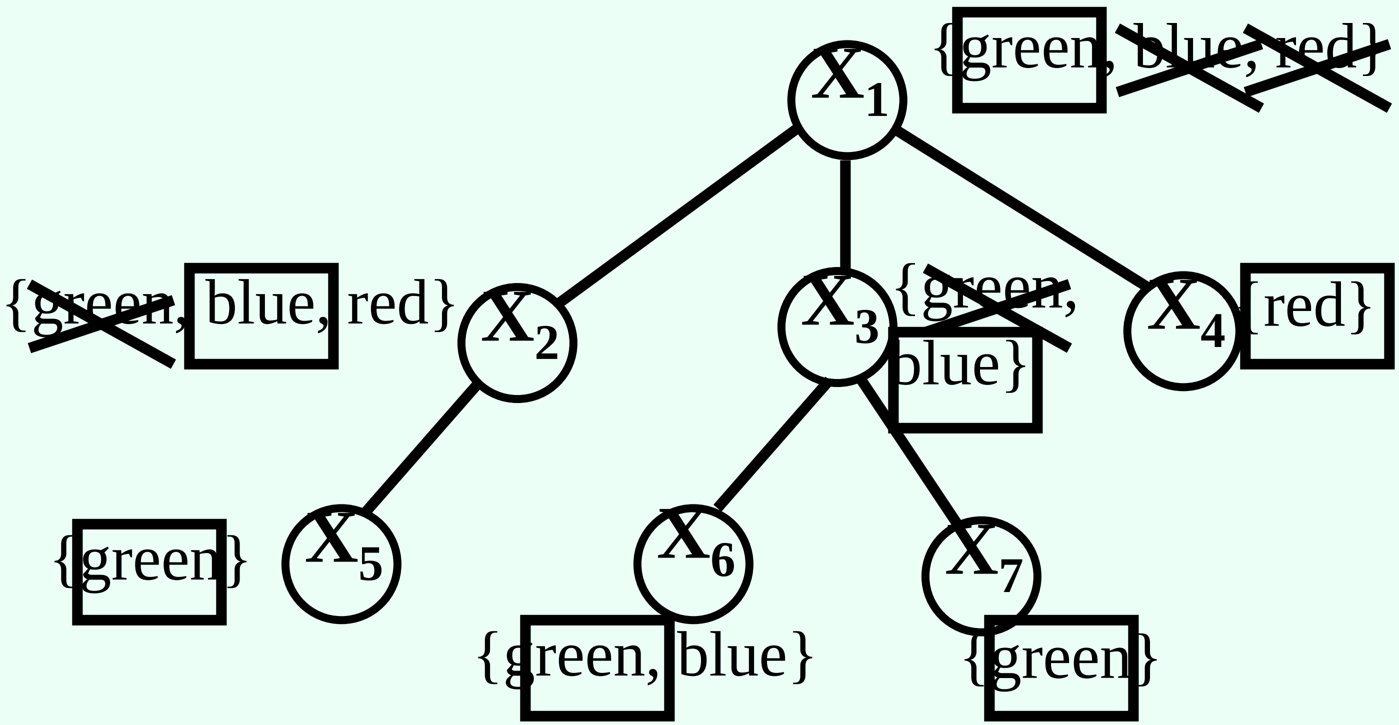
Tree-structured constraint graphs

- Suppose we only have pairwise constraints and the graph is a tree (or forest = multiple disjoint trees)



- Dynamic program for solving this (linear in #variables):
  - Starting from the leaves and going up, for each node *x*, compute all the values for *x* such that the subtree rooted at *x* can be solved
    - Equivalently: apply arc consistency from each parent to its children, starting from the bottom
  - If no domain becomes empty, once we reach the top, easy to fill in solution

Example: graph coloring with limited set of colors per node



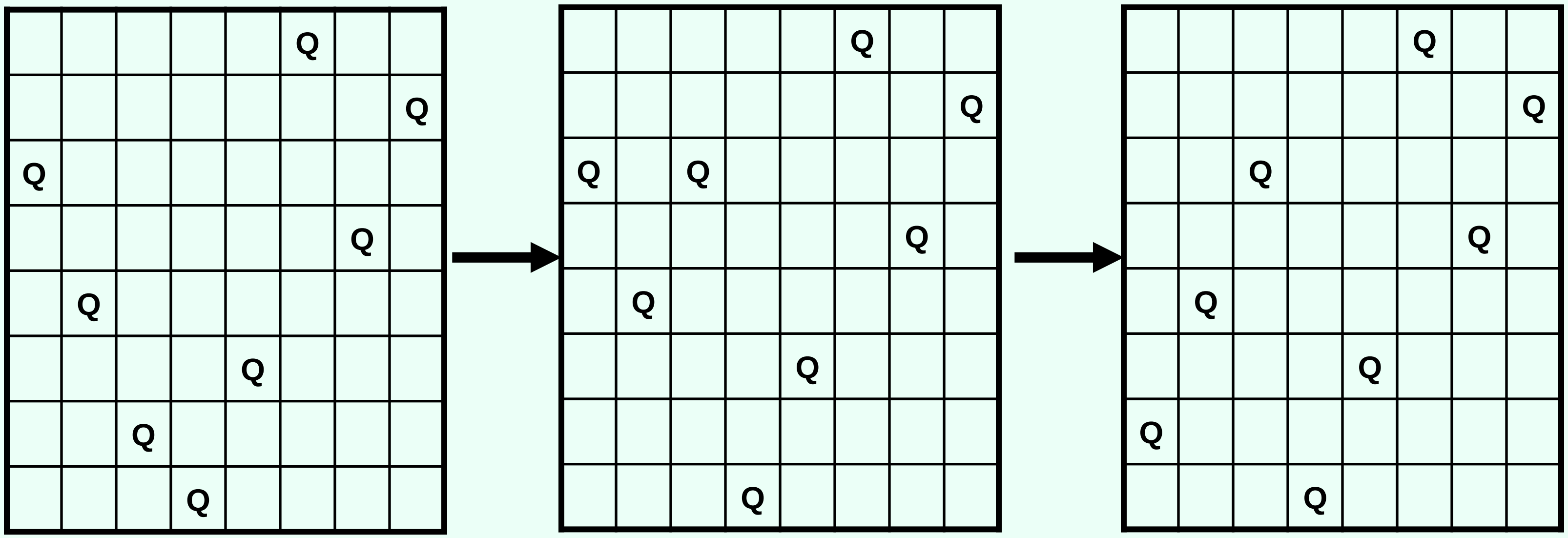
- Stage 1: moving upward, cross out the values that cannot work with the subtree below that node
- Stage 2: if a value remains at the root, there is a solution: go downward to pick a solution

A different approach: optimization

- Let’s say every way of placing 8 queens on a board, one per column, is feasible
- Now we introduce an objective: minimize the number of pairs of queens that attack each other
  - More generally, minimize the number of violated constraints
- Pure optimization

Local search: hill climbing

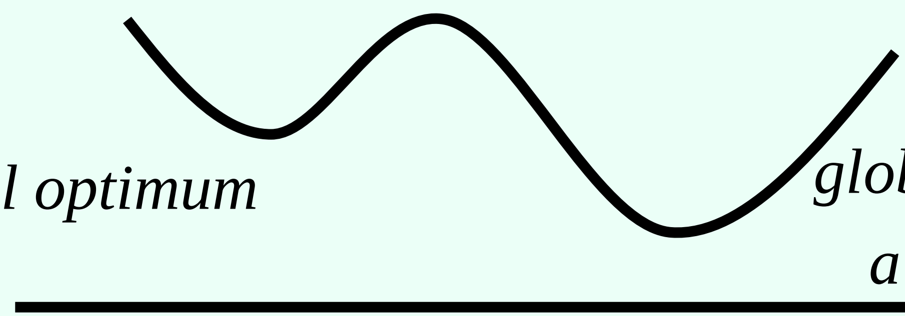
- Start with a complete state
- Move to successor with best (or at least better) objective value
  - Successor: move one queen within its column



no more improvements

4 attacking pairs *3 attacking pairs* *2 attacking pairs*

- Local search can get stuck in a local optimum



local optimum

global optimum (also a local optimum)

Avoiding getting stuck with local search

- Random restarts: if your hill-climbing search fails (or returns a result that may not be optimal), restart at a random point in the search space
  - Not always easy to generate a random state
  - Will **eventually** succeed (why?)
- Simulated annealing:
  - Generate a random successor (possibly worse than current state)
  - Move to that successor with some probability that is sharply decreasing in the badness of the state
  - Also, over time, as the “temperature decreases,” probability of bad moves goes down