

Search

Agenda

- problem formulation
- search algorithms
- Informed search

Problem Formulation

- **states**: description of “world of interest”
- **initial state**
- **successor function**: generates set of legal next states from available actions
- **goal test**: how do we know we’re done?
- **path cost**: way of choosing between multiple solutions (e.g., shortest route)

Vacuum World Problem

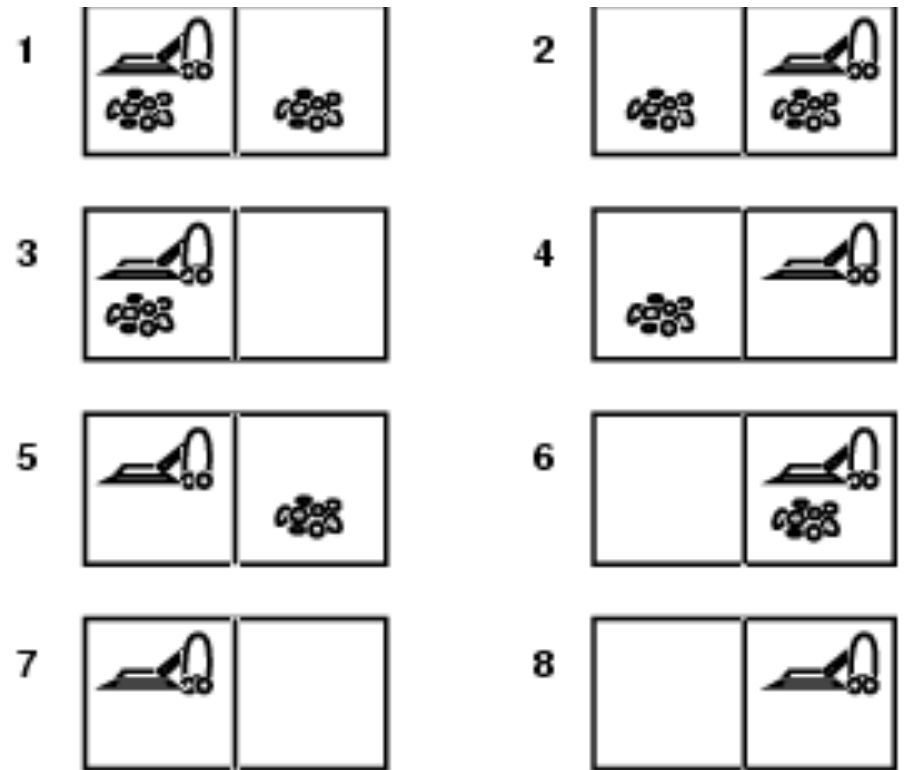
- **successor function:**
 - move left (L), move right (R), suck (S)
- **goal test:**
 - no dirt left in any square
- **path cost:**
 - each step costs 1



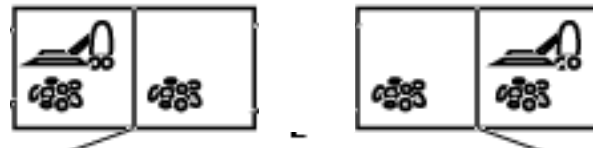
Exercise

Starting from states 1 and 2, generate the state tree, showing all possible actions and the associated successor states

(don't redraw existing states; use loops to indicate "no change")



Solution



Missionaries & Cannibals

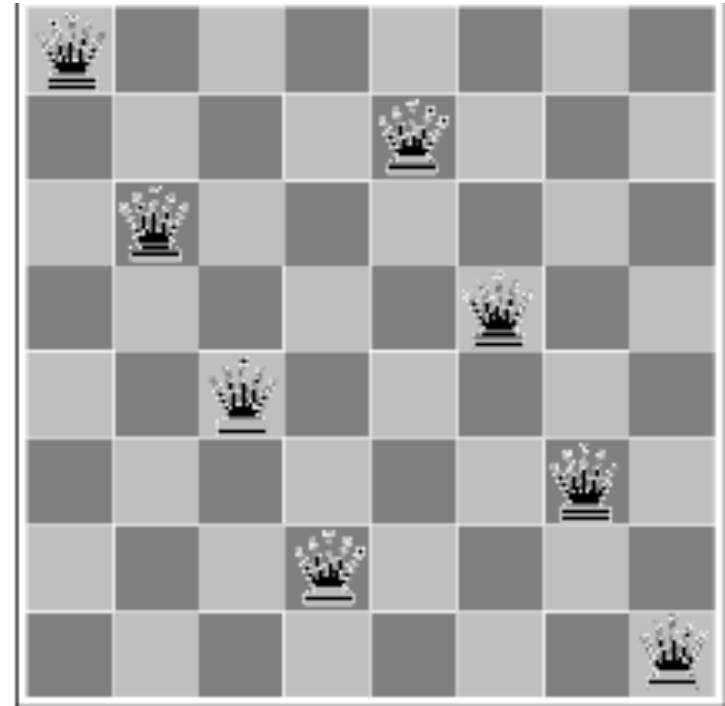
- 3 missionaries and 3 cannibals need to cross crocodile-infested river
- boat can hold 1 or 2 people
- can't leave any missionaries outnumbered by cannibals

Missionaries & Cannibals: Formulation

- **states:** (# missionaries, # cannibals, # of boats) on left bank of river
- **initial state:** (3,3,1)
- **successor function:** move (# missionaries, # cannibals) from one bank to other
- **goal test:** (0,0,0)
- **path cost:** # of river crossings

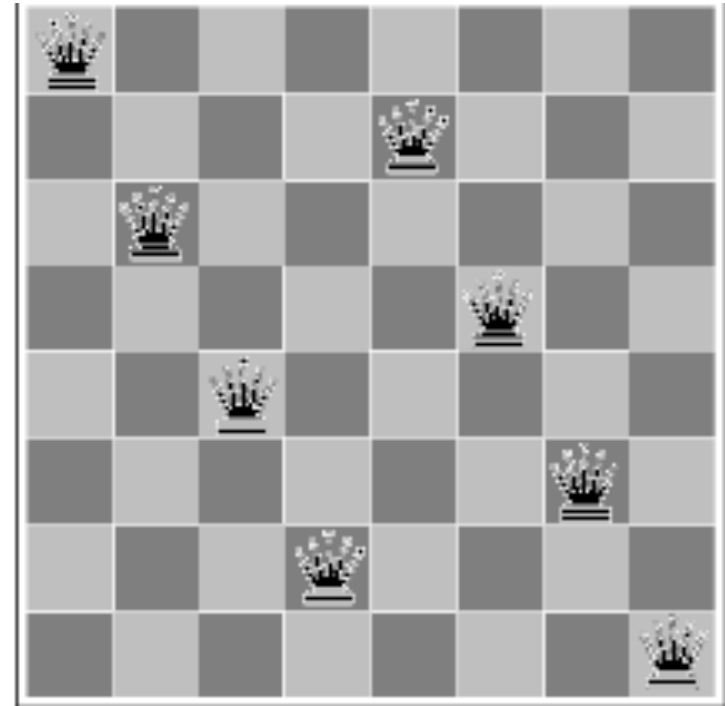
8-Queens problem

- arrange 8 queens on a chessboard so that no two queens are on the same row, column or diagonal (i.e., attack each other)
- applications to parallel memory storage, VLSI testing, traffic control, and deadlock prevention



Naïve approach

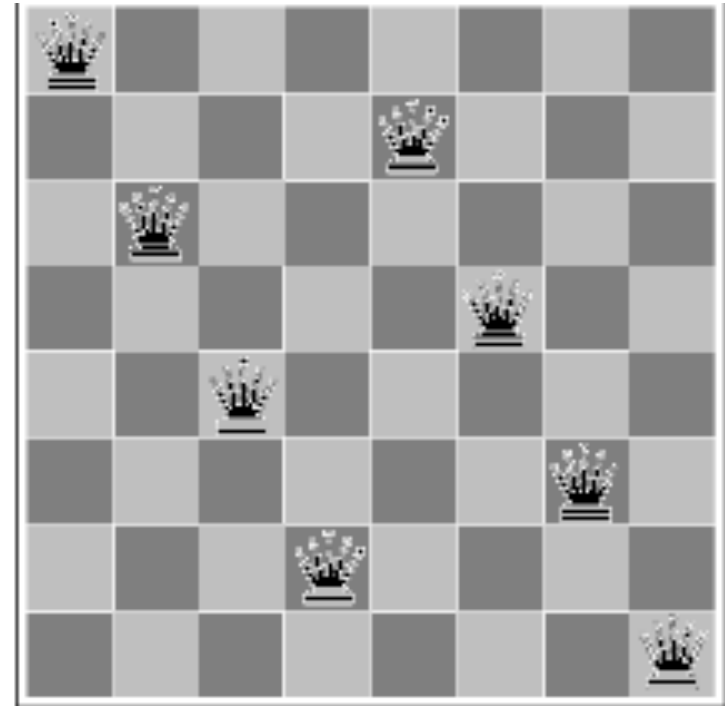
- **state**: any arrangement of [0,8] queens on the board
- **successor function**: add a queen to any empty square



State space: 3×10^{14}

Better approach

- **state**: any arrangement of $n=[0,8]$ queens, one per column in the leftmost n columns, with no queen attacking another
- **successor function**: add a queen to any square in the leftmost empty column such that it is not attacked by any other queen



State space: 2057

8-squares problem

5	4	
6	1	8
7	3	2

initial state

1	2	3
8		4
7	6	5

goal state

Search Methods

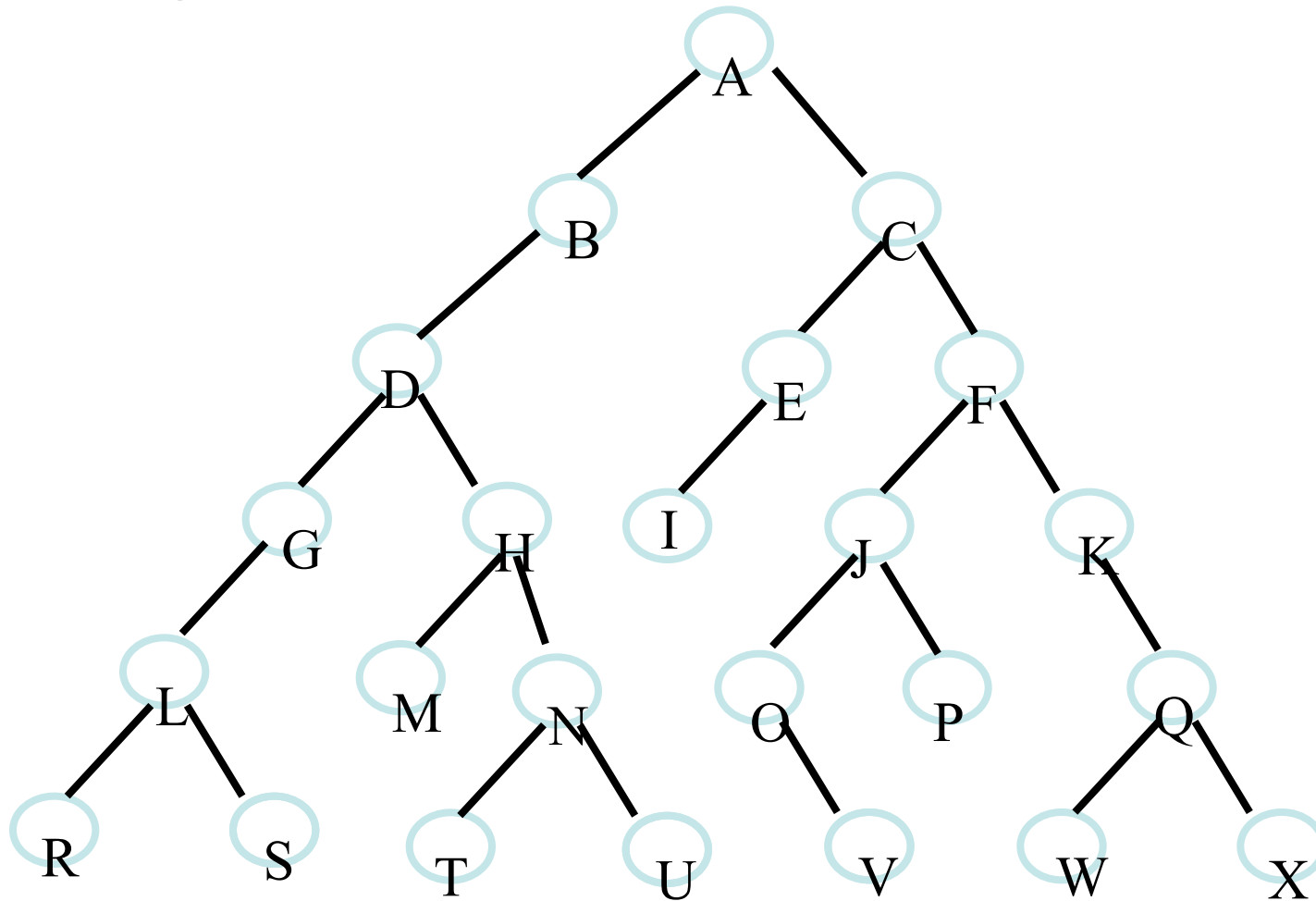
- use to explore state space for solution to a problem
- can be uninformed (blind) or use some reasonable knowledge (heuristics) to guide search

Uninformed Search

- **breadth-first**
 - expand shallowest nodes first (FIFO)
- **depth-first**
 - expand deepest nodes first (LIFO)
- **depth-limited search**
 - depth-first with cutoff
- **iterative-deepening**
 - combines benefits of BFS and DFS
- **bidirectional**
 - applicable when operators are reversible

Exercise:

Compare performance of search algorithms in locating 'G', 'K' and 'U'



Correct Answer

- **BF:** abcdefghijklmnopqrstuvwxyz
- **DF:** abdglrshmntuceifjovpkqwx
- **ID:** a, abc, abdcef, abdghceifjk,
abdglhmnceifjopkq,
abdglrshmntuceifjovpkqwx

Informed Search

- greedy search
 - minimize estimated cost to goal, $h(n)$
 - start by expanding minimal cost node
- A^* search
 - minimize estimated cost to goal: $f(n) = g(n) + h(n)$
 - $g(n)$ = cost of solution from start to n and $h(n)$ is estimated cost of cheapest solution from n to goal
 - A^* is optimal if $h(n)$ never over-estimates true cost to reach goal

Homework Exercise - Part 1

5	4	
6	1	8
7	3	2

initial state

1	2	3
8		4
7	6	5

goal state

- provide a compact state representation and successor function for the 8-squares problem
 - State Representation = {contents of cells in matrix or vector form}, e.g., [5,4,-,6,1,8,7,3,2]
 - Naive successor function: move any tile <N|S|E|W>
 - Better successor function: move the “empty square” <N|S|E|W>

Homework Exercise - Part 2

- for this problem, are:
 - $h(n)$ = number of displaced tiles or
 - $h(n)$ = sum of Manhattan distances of the displaced tilesvalid (**admissible**) heuristics for A* search?
- which heuristic is better?
- is A* search **optimal** in these cases (with respect to other algorithms using the same heuristic)?

Exercise - Part 2

- for this problem, are:
 - $h(n)$ = number of displaced tiles or
 - $h(n)$ = sum of Manhattan distances of the displaced tilesvalid (**admissible**) heuristics for A* search?
 - $h(n)$ valid if it never overestimates actual cost to reach goal
- which heuristic is better?
 - Second one: it provides a higher lower bound on the estimate
- is A* search **optimal** in this case (with respect to other algorithms using the same heuristic)?
 - if $h(n)$ is valid, then A* is optimal because it is optimistic: the true cost of a path through node n to the goal will be at least as great as $g(n) + h(n)$
 - A* will never overlook the possibility of a lower-cost path