

# Designing heuristics; games as search

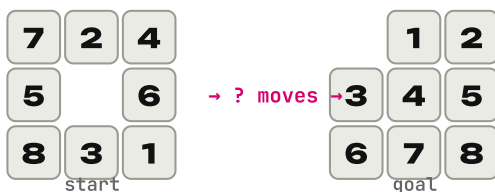
Where do good heuristics come from? Make the problem easier, solve the easy version exactly, and use its cost. Then: what changes when the world fights back.

01 COLD OPEN 0-4 min

## How far is this puzzle from solved?

The 8-puzzle from Week 2, scrambled. Before you touch it: **how many moves** will it take?

You cannot know without solving it. But you can *estimate*, and A\* only needs an estimate that never says too much.



$h_1$  = number of tiles out of place = **8**.  $h_2$  = total distance of tiles from their homes = **18**. The truth: **26** moves. Both underestimate; one is much closer. Today: why both are safe, and why the closer one saves thousands of nodes.

02 WORKED EXAMPLE 4-9 min

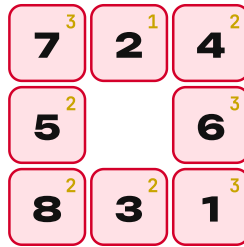
## Worked example: $h_1$ and $h_2$ by hand

GIVEN

Start: 7 2 4 / 5 \_ 6 / 8 3 1. Goal: \_ 1 2 / 3 4 5 / 6 7 8 (Russell & Norvig's example).

FIND

$h_1$  (misplaced tiles) and  $h_2$  (sum of Manhattan distances). Are they admissible? Which is better?



Method: 8-puzzle heuristics (Russell & Norvig)

## SOLUTION

1. Goal: blank top-left, then 1-8 in reading order.  $h_1$  counts tiles not on their goal square: here **all 8**.

$$h_1 = 8$$

2.  $h_2$  adds each tile's horizontal + vertical distance to its goal square:

$$h_2 = \underbrace{3}_1 + \underbrace{1}_2 + \underbrace{2}_3 + \underbrace{2}_4 + \underbrace{2}_5 + \underbrace{3}_6 + \underbrace{3}_7 + \underbrace{2}_8 = 18$$

3. The true optimal solution needs **26** moves (the lab finds it). Both are below 26: **admissible**.  $h_2$  is closer.

Each move shifts one tile one square, so it can reduce  $h_2$  by at most 1 and  $h_1$  by at most 1: neither can overestimate.

4.  $h_2(n) \geq h_1(n)$  for every state (a misplaced tile is at least 1 square away):  $h_2$  **dominates**  $h_1$ .

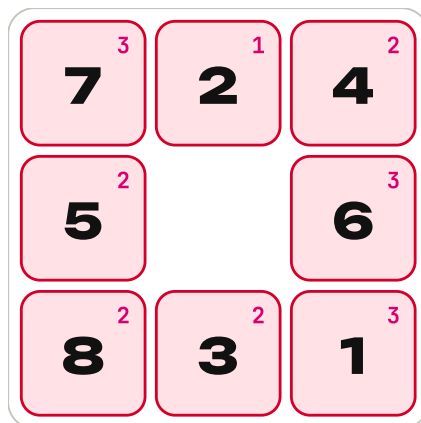
## ANSWER

$h_1 = 8$ ,  $h_2 = 18$ , true cost 26. Both admissible;  $h_2$  dominates  $h_1$ .

## Live lab: $h_0$ vs $h_1$ vs $h_2$ on the 8-puzzle

### The 8-puzzle: which heuristic knows more?

Tiles show their Manhattan distance ( $h_2$  terms); red tiles are misplaced ( $h_1$ ). Slide tiles, scramble, then let A\* solve it with each heuristic and compare the work.



$H_1$  MISPLACED

8

$H_2$  MANHATTAN

18

STATE

solvable

random moves

14

| A* with         | nodes generated | depth d | $b^*$ |
|-----------------|-----------------|---------|-------|
| $h = 0$ (UCS)   | —               | —       | —     |
| $h_1$ misplaced | —               | —       | —     |
| $h_2$ Manhattan | —               | —       | —     |

Bars are on a log scale. All three find the same optimal depth (both heuristics are admissible); the better-informed  $h_2$  generates far fewer nodes.

## Measuring a heuristic: the effective branching factor

If A\* generates  $N$  nodes and finds a solution at depth  $d$ ,  $b^*$  is the branching factor of a uniform tree of depth  $d$  with  $N + 1$  nodes:

$$N + 1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$$

| D  | IDS                        | $A^*(H_1)$          | $A^*(H_2)$         |
|----|----------------------------|---------------------|--------------------|
| 12 | $3,644,035 \cdot b^* 2.78$ | $227 \cdot 1.42$    | $73 \cdot 1.24$    |
| 24 | too many                   | $39,135 \cdot 1.48$ | $1,641 \cdot 1.26$ |

#### DOMINANCE

If  $h_2(n) \geq h_1(n)$  for every  $n$  (both admissible),  $A^*$  with  $h_2$  never expands more nodes than with  $h_1$ .  
Reason:  $A^*$  surely expands every node with  $f(n) < C^*$ , i.e.  $h(n) < C^* - g(n)$ ; a larger  $h$  makes that condition true for fewer nodes.



#### PRO TIP

A  $b^*$  close to 1 means the search walks almost straight to the goal. A well-designed heuristic keeps  $b^*$  roughly constant as problems get harder.

05 **CONCEPT** 20-25 min

## Inventing heuristics: relax the rules

The 8-puzzle rule: *a tile can move from square A to square B if A is adjacent to B **and** B is blank.*  
Drop conditions and the problem gets easier:

| RELAXED RULE                               | EXACT COST OF THE RELAXED PROBLEM |
|--------------------------------------------|-----------------------------------|
| A tile can move to B if A is adjacent to B | $h_2$ : Manhattan distance        |
| A tile can move to B if B is blank         | Gaschnig's heuristic              |
| A tile can move from A to B (teleport)     | $h_1$ : misplaced tiles           |

The optimal cost of a **relaxed** problem is an **admissible** (and consistent) heuristic for the original: the real problem has fewer moves available, so it can only cost more.

#### THREE MORE SOURCES OF H

**Max of several:**  $h = \max(h_1, h_2, \dots)$  is admissible and dominates each one.

**Pattern databases:** solve a subproblem exactly (e.g. tiles 1-4 only, others as blanks) for every configuration, store the costs, look them up during search.

**Learning:** fit  $h(n) = c_1x_1(n) + c_2x_2(n)$  to solved examples ( $x_1$  = misplaced tiles,  $x_2$  = adjacent pairs that are not adjacent in the goal). Fast, but not guaranteed admissible.

06 **YOUR TURN** 25-30 min

## Your turn • 1: heuristics for $\alpha$

# robot that moves like a king

1

EXAM CQ

A hospital delivery robot moves on a grid in **8 directions**. Straight steps cost 1. Candidate heuristics to the target at  $(\Delta x, \Delta y)$ : Manhattan  $\Delta x + \Delta y$ , Euclidean  $\sqrt{(\Delta x^2 + \Delta y^2)}$ , Chebyshev  $\max(\Delta x, \Delta y)$ .

(a) If a diagonal step costs **1**, which are admissible? (b) If it costs  $\sqrt{2}$ ? Use  $\Delta x = 3, \Delta y = 2$  as a test. (c) Why are the good ones admissible even with walls in the way?

## SOLUTION

1. **(a) Diagonal costs 1** (king moves): true cost =  $\max(\Delta x, \Delta y)$ . For  $(3, 2)$ :  $h^* = 3$ .
2. Chebyshev  $\max(\Delta x, \Delta y) = 3 = h^*$ : exact, admissible. Manhattan  $3 + 2 = 5 > 3$ : **not admissible**. Euclidean  $\sqrt{13} = 3.61 > 3$ : **not admissible** either.  
With free diagonals, even the straight line overestimates (one diagonal step covers  $\sqrt{2}$  of distance for cost 1).
3. **(b) Diagonal costs  $\sqrt{2}$** : true cost =  $\max + (\sqrt{2} - 1) \cdot \min = 3 + 0.414 \cdot 2 = 3.83$  ("octile" distance).
4. Euclidean  $3.61 \leq 3.83$  ✓, Chebyshev  $3$  ✓ (both admissible; octile itself is exact). Manhattan  $5 > 3.83$  ✗.

$$h_{\text{Cheb}} \leq h_{\text{Euc}} \leq h_{\text{Oct}} = h^* \Rightarrow \text{octile dominates}$$

5. **(c)** They are exact costs of the relaxed problem "no walls": admissible and consistent for any map of walls.

## ANSWER

**(a)** Only Chebyshev is admissible (it is exact without walls). **(b)** Chebyshev and Euclidean are admissible, Manhattan is not; octile distance is the best. **(c)** Each is the exact cost with all walls removed: a relaxed problem.

07 THINK DEEPER 30-34 min

## Critical thinking • is the

## perfect heuristic worth it?

### THINK DEEPER

$h = h^*$  (the exact remaining cost) makes A\* walk straight to the goal. Why not always use it?

Because computing the exact  $h^*$  is the search problem. A good heuristic is a compromise: **cheap to compute, close to  $h^*$** . A heuristic that saves 90 % of the nodes but takes 100× longer per node is a loss.

### A STRONG ANSWER MENTIONS

- Computing  $h^*$  exactly is as hard as solving the problem itself.
- The goal is the cheapest total: (nodes expanded) × (cost of computing  $h$  per node).
- Pattern databases and learned  $h$  move work from search time to preparation time.

### THINK DEEPER

$h_1$  is admissible and so is  $h_2$ . Is  $h_1 + h_2$  admissible? Is  $(h_1 + h_2)/2$ ?

The sum is **not** admissible (one tile one step from home:  $1 + 1 = 2 > 1$ ). The average is admissible but weaker than  $h_2$  itself. When combining admissible heuristics, take the **max**, not the sum (unless they count disjoint costs, as in disjoint pattern databases).

### A STRONG ANSWER MENTIONS

- On the textbook state  $8 + 18 = 26 = h^*$ : a coincidence. One tile one step from home gives  $h_1 + h_2 = 2 > h^* = 1$ : the sum overestimates.
- $(h_1 + h_2)/2 \leq \max(h_1, h_2) \leq h^*$ : admissible, but dominated by  $\max(h_1, h_2) = h_2$ .

08 **CONCEPT** 34–39 min

## Games: search when someone fights back

Until now the environment did what we asked. In a game, a second agent chooses half of the moves, **against** us. We study the simplest kind: two players, turn-taking, deterministic, fully observable, **zero-sum** (one's win is the other's loss).

### A GAME AS A SEARCH PROBLEM

|                          |                                        |
|--------------------------|----------------------------------------|
| $S_0$                    | initial state (empty board)            |
| TO-MOVE(s)               | whose turn: MAX or MIN                 |
| ACTIONS(s), RESULT(s, a) | legal moves and their effect           |
| IS-TERMINAL(s)           | is the game over?                      |
| UTILITY(s, p)            | final score for player p (+1 / 0 / -1) |

### WHY ORDINARY SEARCH FAILS

A\* finds a *path* to a goal. But we do not control MIN's moves, so no path is guaranteed. We need a **strategy**: a move for every reply the opponent might make.

| GAME        | B             | GAME TREE                                 |
|-------------|---------------|-------------------------------------------|
| Tic-tac-toe | $\leq 9$      | 549,946 nodes                             |
| Chess       | $\approx 35$  | $\approx 10^{154}$ ( $10^{40}$ positions) |
| Go          | $\approx 250$ | $\approx 10^{360}$                        |

09 **CONCEPT** 39–43 min

## Minimax: assume the opponent is perfect

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if } s \text{ is terminal} \\ \max_a \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if MAX to move} \\ \min_a \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if MIN to move} \end{cases}$$

- A depth-first walk to the leaves, then values are **backed up**: MAX takes the largest child, MIN the smallest.
- The minimax value is what MAX can **guarantee** against a perfect opponent.
- Complete (finite tree), optimal against an optimal opponent. Time  $O(b^m)$ , space  $O(bm)$ .



### PRO TIP

Values are always from **MAX's** point of view. A MIN node does not "want" a high number: it picks the smallest, because small is bad for MAX.



### POP CULTURE

John von Neumann proved the minimax theorem in 1928, two decades before the first computer: every finite two-player zero-sum game has a value both players can guarantee.

10 **WORKED EXAMPLE** 43–48 min

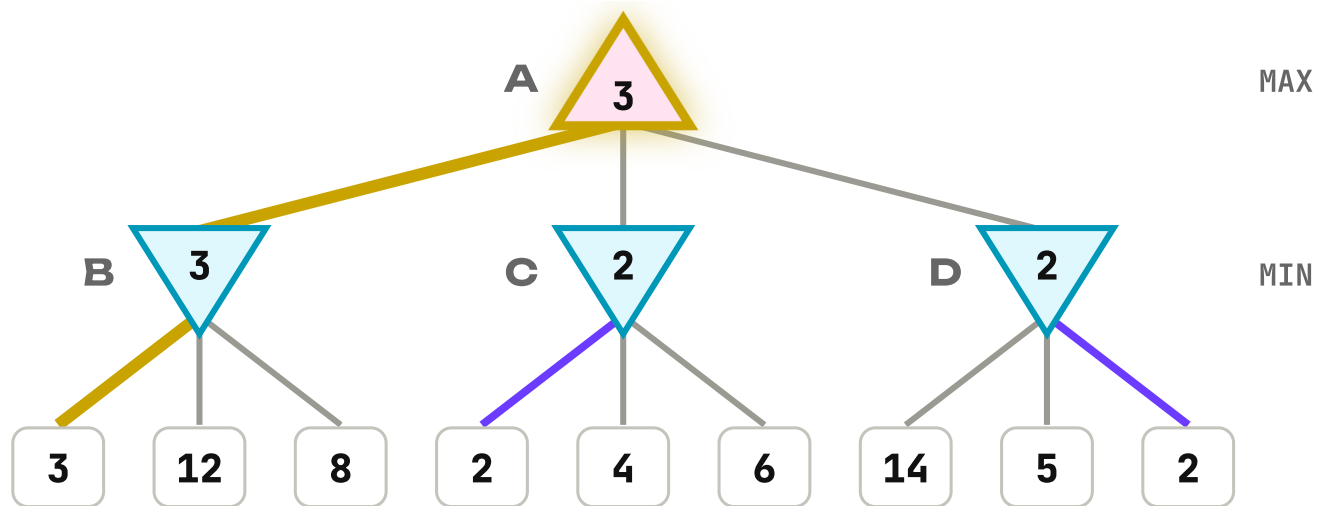
## Worked example: backing up values

### GIVEN

MAX moves, then MIN. Leaves are utilities for MAX (Russell & Norvig's example).

### FIND

The minimax value of every node and MAX's best move.



Method: MINIMAX: MIN takes the smallest, MAX the largest (Russell & Norvig)

#### SOLUTION

1. Leaves are utilities for MAX. Back up from the bottom: each MIN node takes the **smallest** child.

$$B = \min(3, 12, 8) = 3, \quad C = \min(2, 4, 6) = 2, \quad D = \min(14, 5, 2) = 2$$

2. The root is MAX: it takes the **largest** child.

$$A = \max(3, 2, 2) = 3$$

3. MAX's best move is the first one (towards B). Against a perfect MIN it guarantees 3.

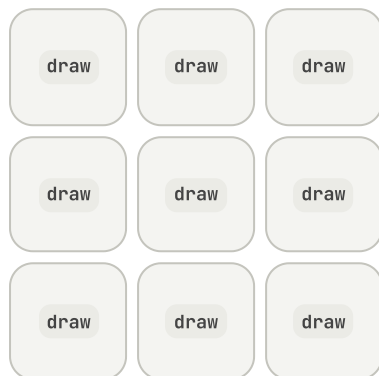
Tempting trap: the 14 under D. MIN would never allow it: from D, MIN answers with 2.

#### ANSWER

**B = 3, C = 2, D = 2, root A = 3. MAX plays the move to B.**

## Live lab: tic-tac-toe against minimax

**Tic-tac-toe against minimax** The machine searches the entire game tree before every move. Tags show the minimax outcome of each square for the player to move, assuming perfect play afterwards.



### Your move.

☒ show minimax values

NODES SEARCHED (MACHINE'S LAST MOVE)

—

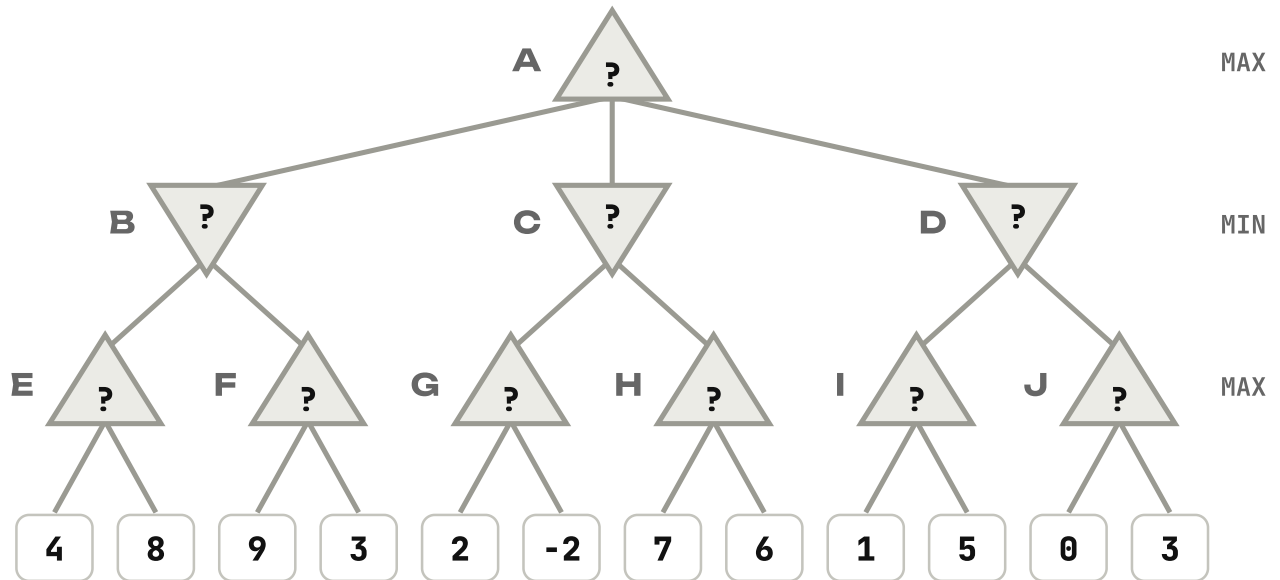
TO VALUE THIS POSITION

**549,945**

From the empty board the full game tree has 549,946 nodes (only 5,478 distinct positions: many move orders reach the same one). Tiny for a computer. Chess has about  $10^{40}$  legal positions and Go about  $10^{170}$ : full minimax is hopeless there.

## Your turn • 2: a three-ply tree

2 EXAM CQ



Find the minimax value of every internal node and MAX's best first move. Which leaf is reached if both play perfectly?

### SOLUTION

1. MAX level:  $E = \max(4, 8) = 8$ ,  $F = \max(9, 3) = 9$ ,  $G = \max(2, -2) = 2$ ,  $H = \max(7, 6) = 7$ ,  $I = \max(1, 5) = 5$ ,  $J = \max(0, 3) = 3$ .
2. MIN level:  $B = \min(8, 9) = 8$ ,  $C = \min(2, 7) = 2$ ,  $D = \min(5, 3) = 3$ .
3. Root:  $A = \max(8, 2, 3) = 8$ . Best move: towards B.

Principal variation:  $A \rightarrow B \rightarrow E \rightarrow \text{leaf } 8$ . The 9 under F is never reached: MIN at B would choose E.

### ANSWER

E 8, F 9, G 2, H 7, I 5, J 3; B 8, C 2, D 3; A = 8. MAX moves to B; expected play  $A \rightarrow B \rightarrow E \rightarrow 8$ .

## Critical thinking • what

# if MIN is not perfect?

## THINK DEEPER

Minimax assumes MIN plays perfectly. Your opponent is a beginner who often blunders. Is minimax still a good strategy? Can MAX do worse than the minimax value? Could it do better?

Minimax is a **safety** strategy: against a weaker MIN it never does worse than the minimax value, often better. But it is not the *best* response to a known weak opponent: it ignores traps the opponent would fall into. Poker programs, for example, model their opponents. The price: a modelled strategy can be exploited by a stronger player.

## A STRONG ANSWER MENTIONS

- Against any MIN, MAX's minimax strategy scores **at least** the minimax value: it is a guarantee, a lower bound.
- It can do better than the value whenever MIN blunders.
- But a strategy built for a known weak opponent (an opponent model) could score even more, at the risk of being punished by a strong one.

14 **CONCEPT** 63–67 min

## Real games are too deep: cut off and evaluate

- Stop the search at a depth limit  $d$  (IDS from Week 3: deepen until time runs out) and replace UTILITY by an **evaluation function**  $EVAL(s)$ : an estimate of the utility.
- $EVAL$  is the game version of a heuristic  $h$ . The usual form is a weighted sum of features:

$$EVAL(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

Chess material: pawn 1, knight 3, bishop 3, rook 5, queen 9;  $f_i$  = (my pieces – yours) of each kind, plus terms for king safety, mobility, pawn structure.



### THE HORIZON EFFECT

A position can look great at depth  $d$  and lose the queen at depth  $d + 1$ . Engines extend the search on "noisy" moves (captures, checks) until the position is quiet: **quiescence search**.



### PRO TIP

Same lesson as today's first half: a better evaluation (heuristic) lets a shallower search play stronger.

## Real world: from Sholo Gutti to AlphaGo



### SHOLO GUTI (শোল গুটি)

Bangladesh's village board game: 16 beads each, capture by jumping, two players, no dice, full information. Exactly the kind of game minimax with an evaluation function plays well: a good class project (b is small enough for depth 6–8).



### LUDO

Dice make it **stochastic**: minimax needs chance nodes that average over the dice (expectiminimax). Same recursion, one more node type.



### 1997 AND 2016

**Deep Blue** beat world champion Garry Kasparov in 1997 with alpha–beta minimax, custom chips and about 200 million positions per second. **AlphaGo** beat Lee Sedol 4–1 at Go in 2016: tree search guided by neural networks that had learned the evaluation function, because Go's  $b \approx 250$  defeated hand-written heuristics.

### THINK DEEPER

**Why could a hand-written EVAL work for chess but not for Go?**

Chess has simple, strong features: count the material. Go stones are all alike and their value is in large-scale patterns no one could write down. So AlphaGo **learned** its evaluation function from data.

### A STRONG ANSWER MENTIONS

- Chess features (material, mobility) correlate well with winning.
- In Go, every stone looks the same; the value depends on subtle whole-board patterns.
- AlphaGo learned EVAL (a value network) from millions of positions: a learned heuristic, as on slide 5.

## Pop quiz

1. For the 8-puzzle,  $h_2$  (Manhattan) dominates  $h_1$  (misplaced tiles). This means...

- A.  $h_2$  is always exact
- B.  $h_2(n) \geq h_1(n)$  for every  $n$ , so  $A^*$  with  $h_2$  never expands more nodes ✓
- C.  $h_2$  is not admissible
- D.  $h_2$  is faster to compute

Both are admissible; the larger one is closer to  $h^*$ . Every node  $A^*$  surely expands with  $h_2$  is also expanded with  $h_1$ .

2.  $h_1$  (misplaced tiles) is the exact cost of a relaxed 8-puzzle in which...

- A. a tile can move to any adjacent square
- B. a tile can move to any square at all ✓**
- C. two tiles can swap
- D. the blank can jump

If a tile can teleport, every misplaced tile needs exactly one move. Allow only adjacent moves (blank or not) and you get Manhattan distance,  $h_2$ .

3. A\* found a depth-5 solution after generating 52 nodes. The effective branching factor  $b^*$  is about...

- A. 1.1
- B. 1.92 ✓**
- C. 5
- D. 10.4

Solve  $53 = 1 + b + b^2 + b^3 + b^4 + b^5$ :  $b \approx 1.92$ . Good heuristics push  $b^*$  towards 1.

4. In a minimax tree, a MIN node with children 7, 3, 9 has value...

- A. 9
- B. 7
- C. 3 ✓**
- D. 19/3

MIN assumes the opponent picks what is worst for MAX: the smallest utility.

5. Minimax searches a game tree of depth  $m$  and branching  $b$  depth-first. Its time and space are...

- A.  $O(bm)$  and  $O(b^m)$
- B.  $O(b^m)$  and  $O(bm)$  ✓**
- C.  $O(b^{m/2})$  and  $O(m)$
- D.  $O(m)$  and  $O(b)$

It visits the whole tree ( $b^m$ ) but, being depth-first, keeps only one path with its siblings ( $bm$ ).  
Next class, alpha-beta cuts the time to about  $b^{m/2}$  in the best case.

17 **RECAP** 76-80 min

## Three things to remember

### 01

**Relax** the rules, solve exactly, use the cost: an admissible, consistent  $h$ . 8-puzzle:  $h_1$  misplaced,  $h_2$  Manhattan;  $h_2$  dominates.

### 02

Judge a heuristic by  $b^*$  ( $N + 1 = 1 + b^* + \dots + b^{*d}$ ) and by its cost per node. Combine admissible heuristics with **max**.

## O3

**Games:** MAX and MIN alternate; minimax backs up max/min values:  $O(b^m)$  time,  $O(bm)$  space.  
Deep games: cut off and use EVAL.

### HW

BEFORE NEXT CLASS

State **3 5 \_ / 6 1 4 / 7 2 8**, same goal as today. (a) Find  $h_1$  and  $h_2$ . (b) A\* with  $h_2$  finds the 12-move solution after generating 39 nodes; with  $h_1$ , 156 nodes. Find  $b^*$  for each. Check (a) in the lab by sliding tiles into this state.

### SOLUTION

1. (a) Misplaced: 3, 5, 6, 1, 4, 7, 2 (only 8 is home):  **$h_1 = 7$** .
2. (a) Distances: 3→1, 5→2, 6→1, 1→1, 4→1, 7→1, 2→3, 8→0:  **$h_2 = 10$** . The optimum is 12 moves: both admissible.
3. (b) A\*( $h_2$ ) generated 39 nodes for  $d = 12$ : solve  $40 = 1 + b + \dots + b^{12}$ :  **$b^* \approx 1.17$** . (A\*( $h_1$ ): 156 nodes,  $b^* \approx 1.37$ .)

### ANSWER

**$h_1 = 7$ ,  $h_2 = 10$  (true cost 12).  $b^*(h_2) \approx 1.17$ ,  $b^*(h_1) \approx 1.37$ .**

### S. M. Sakeef Sani

Adjunct Lecturer, CSE · Northern University Bangladesh

[smsakeefsani.hbitcorp.com](mailto:smsakeefsani.hbitcorp.com)