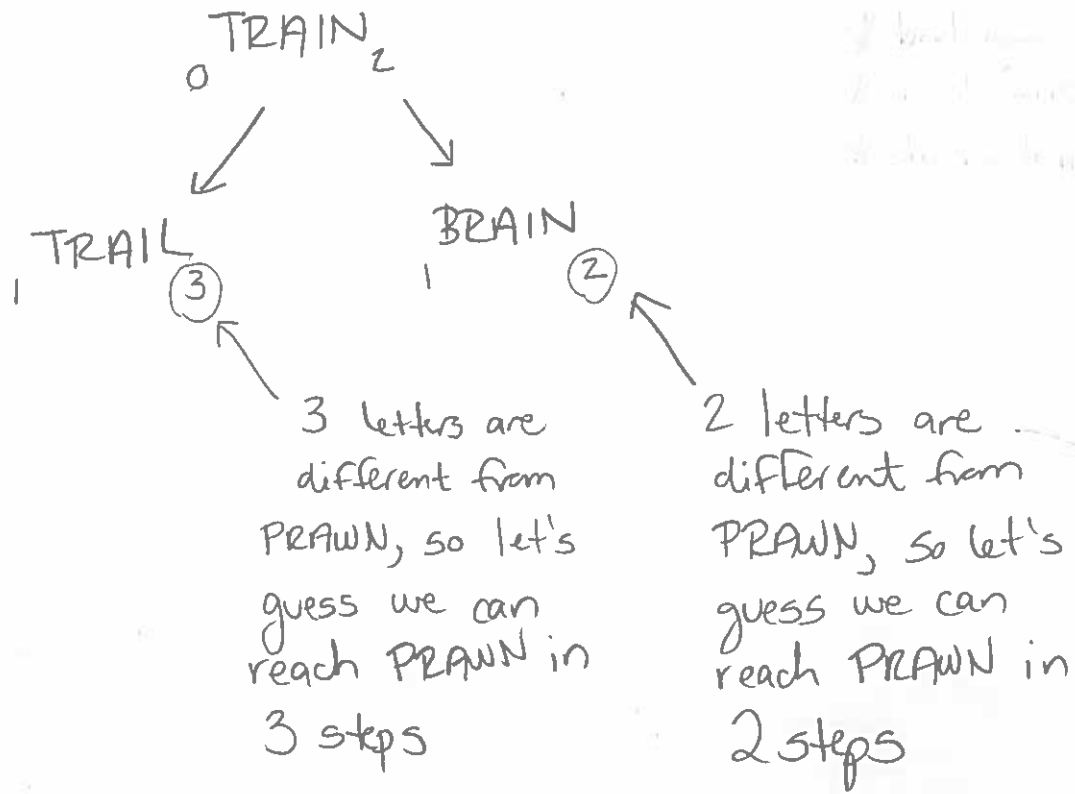


# HEURISTIC SEARCH

① So why were we making these guesses anyway?



② So far, the most "intelligent" container we had was a priority queue that prioritized nodes by g-value:



Why wouldn't we want to factor in our guess about how much it should cost to complete our solution?

# HEURISTIC SEARCH

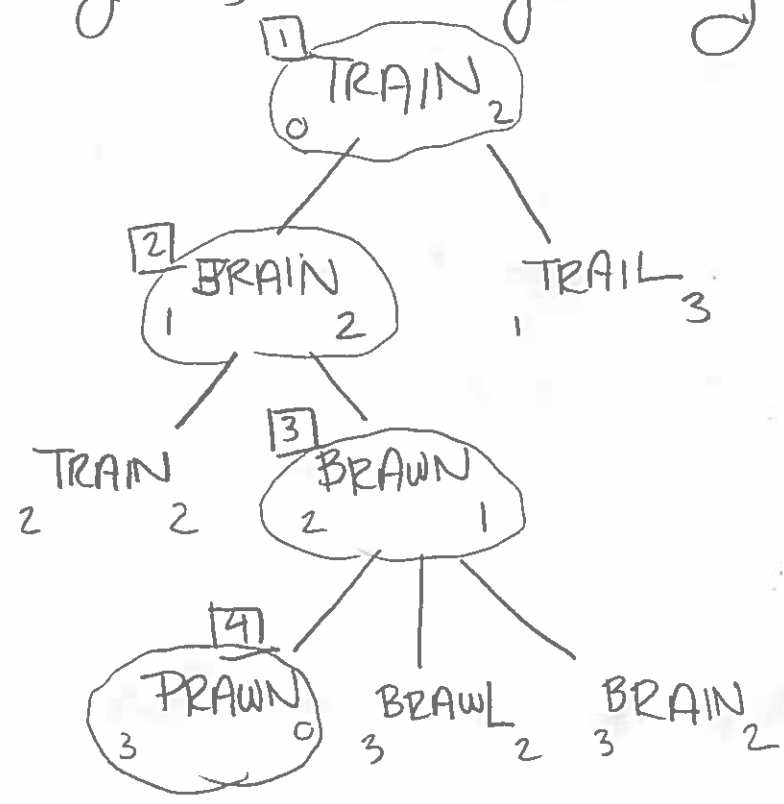
3) It would seem to make sense to have a priority queue that prioritizes nodes by:

the cost so far + the cost to reach the goal from this point on (guessed)

↑  
g-value

↑  
h-value

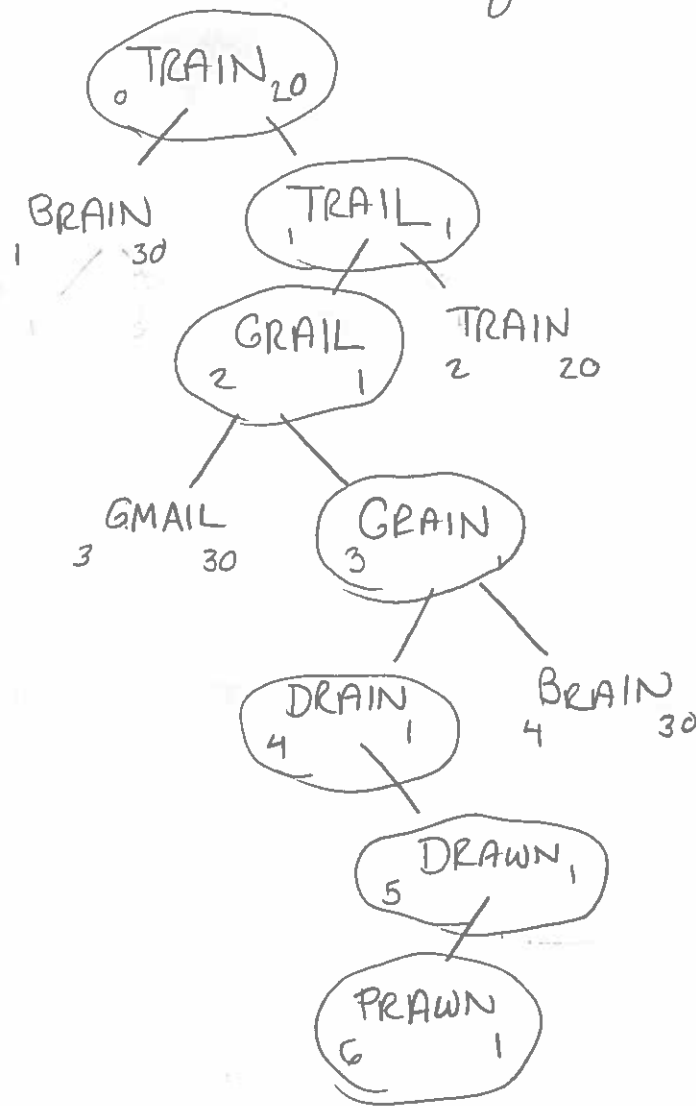
4) If we use a priority queue that prioritizes search nodes based on  $g + h$ , something amazing happens



We go straight to the optimal solution!

## HEURISTIC SEARCH

5) If our guesses are terrible, bad things can happen:



6) How do we choose a heuristic function such that amazing things, as opposed to bad things, happen?

# HEURISTIC SEARCH

⑦ To study this question, it will help to have some extra terminology.

Suppose we have a <sup>weighted</sup> state machine  $M = (Q, \Sigma, \Delta, q_0, F, w)$ .

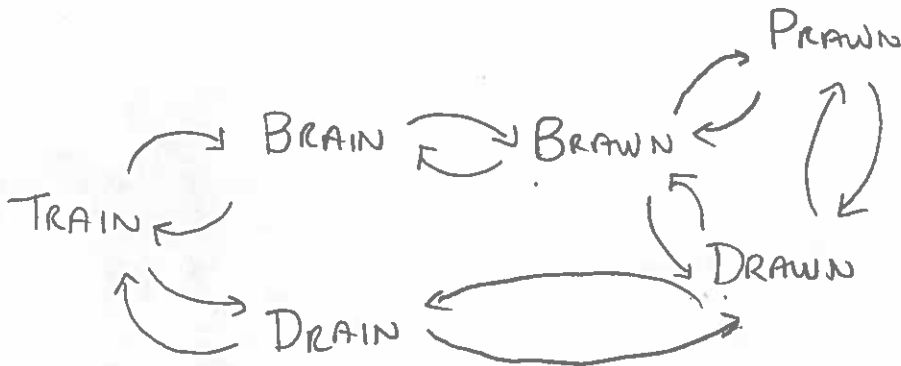
Define a completion path from state  $q \in Q$  as a sequence  $\langle \sigma_1, \dots, \sigma_k \rangle$  of transitions where:

-  $\forall 1 \leq i \leq k, \sigma_i \in \Delta$  and  $\sigma_i = (q_i, \sigma_i, q_{i+1})$

-  $q_1 = q$

-  $q_{k+1} \in F$

e.g.



If  $q_0 = \text{TRAIN}$  and  $F = \{\text{PRAWN}\}$ , then the following are examples of completion paths from DRAIN:

$\langle (\text{DRAIN}, 4W, \text{DRAWN}), (\text{DRAWN}, 1P, \text{PRAWN}) \rangle$

$\langle (\text{DRAIN}, 4W, \text{DRAWN}), (\text{DRAWN}, 1B, \text{BRAWN}), (\text{BRAWN}, 1P, \text{PRAWN}) \rangle$

# HEURISTIC SEARCH

⑧ The cost of a completion path is the sum of the weights of its transitions.

The optimal completion path <sup>(from state  $q$ )</sup> is the completion path of minimal cost.

Define  $H^*(q)$  as the cost of the optimal completion path.

e.g.  $H^*(\text{DRAIN}) = 2$  (i.e. the cost of completion path  $\langle \text{DRAIN}, 4W, \text{DRAWN} \rangle$  <  $\langle \text{DRAWN}, 1P, \text{PRAWN} \rangle$ )

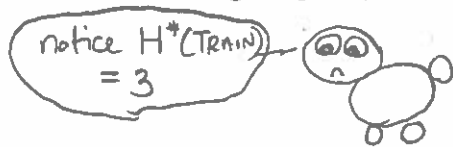
⑨ Let's consider what happens if we use an "optimistic" heuristic function  $H$ , in other words, one that never overestimates the cost of the optimal completion cost.

In other, other words:

$$H(q) \leq H^*(q) \quad \text{for all } q \in Q$$

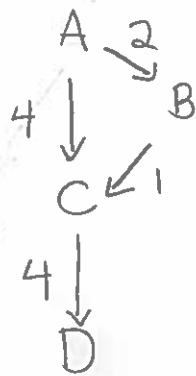
An example (for word ladder) would estimate the cost of completion to be the number of letters that are different from the solution word!

TRAIN  
different different, so  $H(\text{TRAIN}) = 2$



# HEURISTIC SEARCH

⑩ Such a heuristic is called admissible. Let's show a simple example. Consider the state machine:



$$Q = \{A, B, C, D\}$$

$$q_0 = A$$

$$F = \{D\}$$

Suppose our heuristic function is defined:

$$H(A) = 7$$

$$H(B) = 5$$

$$H(C) = 1$$

$$H(D) = 0$$

We can check easily that it is admissible:

$$H^*(A) = 8 \geq 7 = H(A)$$

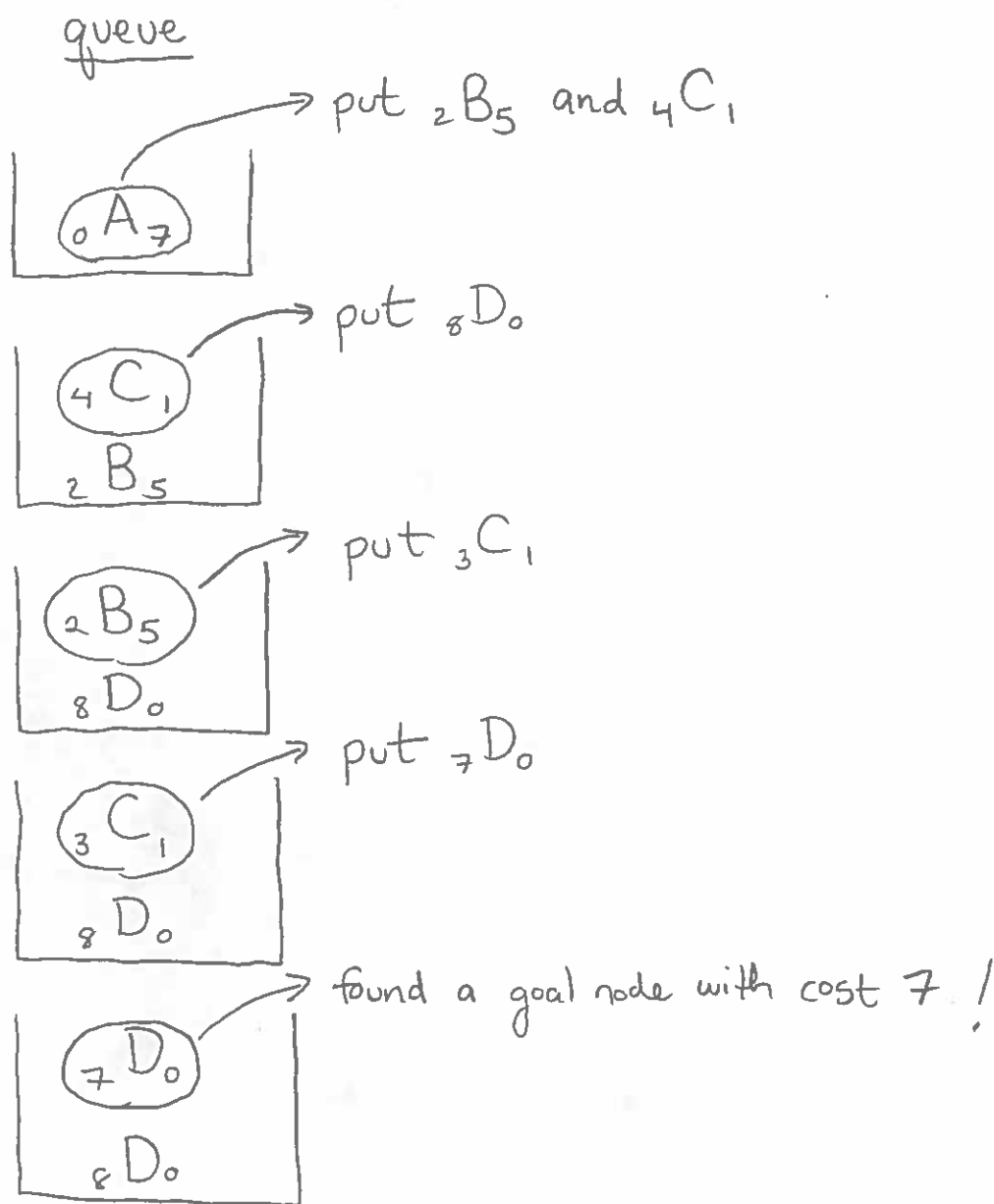
$$H^*(B) = 5 \geq 5 = H(B)$$

$$H^*(C) = 4 \geq 1 = H(C)$$

$$H^*(D) = 0 \geq 0 = H(D)$$

# HEURISTIC SEARCH

11) Let's see how the basic SEARCH algorithm performs if we use this heuristic, and a priority queue prioritized by  $g+h$ :

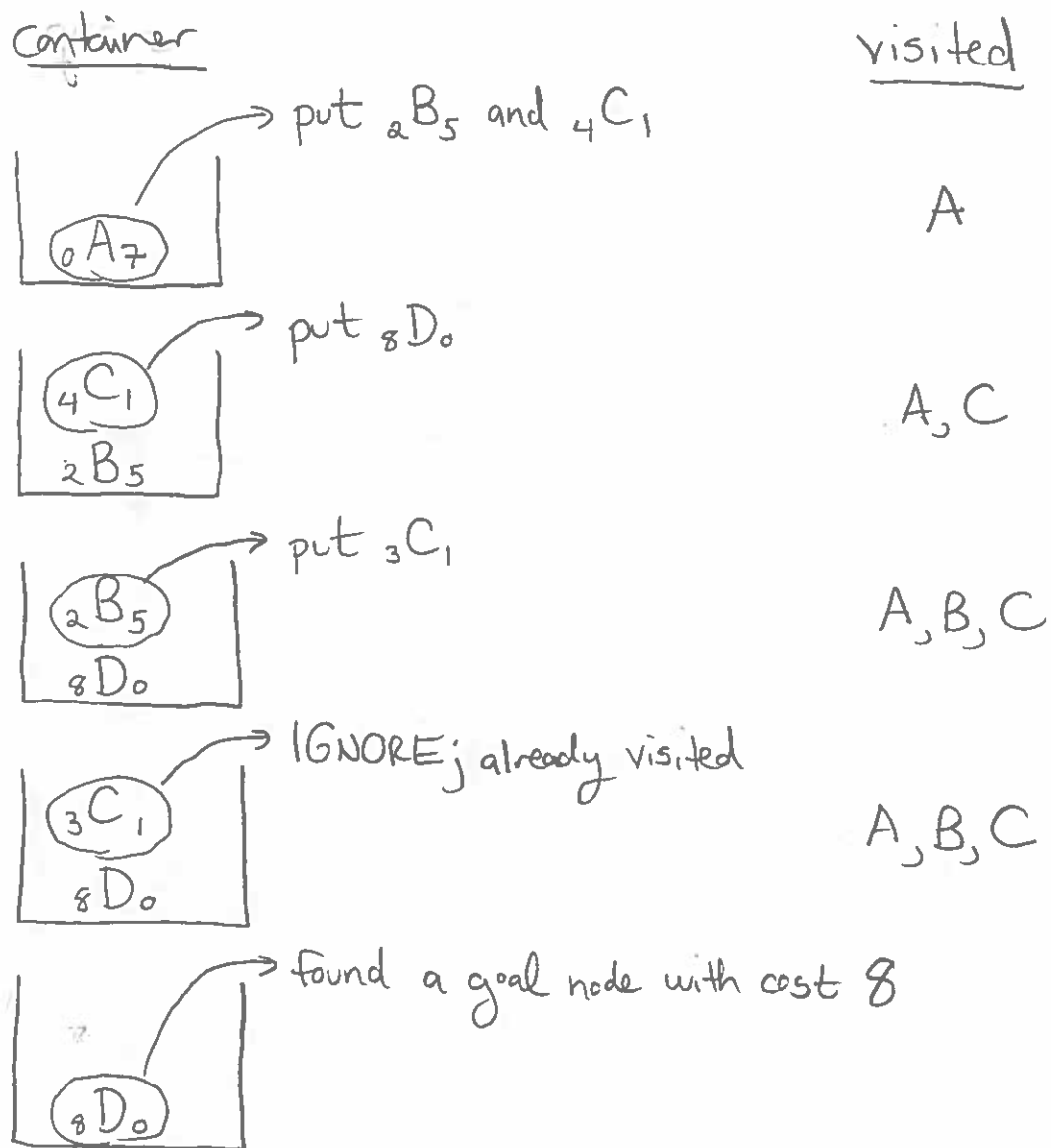


It returns the correct (optimal) solution! In fact, this is true in general. If  $H$  is admissible, and we run SEARCH using a priority queue prioritized by  $g+h$ , then we are guaranteed the optimal solution.

# HEURISTIC SEARCH

⑫ Guaranteed, that is, as long as we don't memoize.

Suppose we run MEMOIZED SEARCH with the same container and heuristic.



Clearly admissibility is not enough to ensure optimality for MEMOIZED SEARCH.



## HEURISTIC SEARCH

- ⑬ Turns out we need a slightly stronger condition for MEMORIZED SEARCH.

A heuristic function  $H$  is consistent if  $H(q) = 0 \forall q \in F$  and

$$H(q) \leq w(q, \sigma, q') + H(q')$$

for all  $q, q' \in Q, \langle q, \sigma, q' \rangle \in \Delta$ .

In our simple example,  $H$  is not consistent.

$$H(A) = 7 > 5 = w(A, \cdot, C) + H(C)$$

$\begin{array}{ccc} & \uparrow & \uparrow \\ & 4 & 1 \end{array}$

- ⑭ We can prove that if  $H$  is consistent, then both SEARCH and MEMORIZED SEARCH, using a priority queue prioritized by  $g+h$ , will be optimal.

First, observe that if node  $n' \in \text{Successors}_{M, H}(n)$ , then:

$$\begin{aligned} g(n') + h(n') &= [g(n) + w(q(n), \sigma, q(n'))] + h(n') \\ &= g(n) + [w(q(n), \sigma, q(n')) + h(n')] \\ &\geq g(n) + [w(q(n), \sigma, q(n')) + H(q(n'))] \\ &\geq g(n) + H(q(n)) \quad [\text{b/c } H \text{ is consistent}] \\ &= g(n) + h(n) \end{aligned}$$

## HEURISTIC SEARCH

---

(15) Next, let's show that if  $n_1, n_2, \dots, n_k$  are the search nodes visited by SEARCH or MEMORIZED SEARCH (in order), then  $j > i \Rightarrow g(n_j) + h(n_j) \geq g(n_i) + h(n_i)$ .

Assume (for contradiction) that there exists  $j > i$  for which  $g(n_j) + h(n_j) < g(n_i) + h(n_i)$ .

When we visit node  $n_i$ , some ancestor of node  $n_j$  must still be in the container (or else  $n_j$  will never be visited). Call this ancestor  $n$ .

From (14), we know  $g(n) + h(n) \leq g(n_j) + h(n_j)$   
 $< g(n_i) + h(n_i)$

So the priority queue should have visited  $n$ , not  $n_i$ .  
Contradiction!

---

(16) Finally, consider the goal nodes. Suppose goal node  $n_i$  is visited before goal node  $n_j$ . From (15):

$$\begin{aligned} i < j &\Rightarrow g(n_i) + h(n_i) \leq g(n_j) + h(n_j) \\ &\Rightarrow g(n_i) \leq g(n_j) \end{aligned}$$

So we visit goal nodes in order of their cost.  
Thus the algorithm is optimal!

## HEURISTIC SEARCH

---

⑪ The basic SEARCH algorithm whose container is a priority queue prioritized by  $g+h$  is called  $A^*$ .

What we've asserted so far:

- if heuristic  $H$  is admissible, then the nonmemoized  $A^*$  returns the optimal solution
- if heuristic  $H$  is consistent, then  $A^*$  (memoized or nonmemoized) returns the optimal solution