search spaces

12 sept 2022

CSCI 373

train

rules

- change one letter at a time to get from train to prawn
- each intermediate word must be valid english

prawn

train

rules

- change one letter at a time to get from train to prawn
- each intermediate word must be valid english

prawn

train
trail
trawl
brawl
brawn
prawn

train

your answer here

prawn

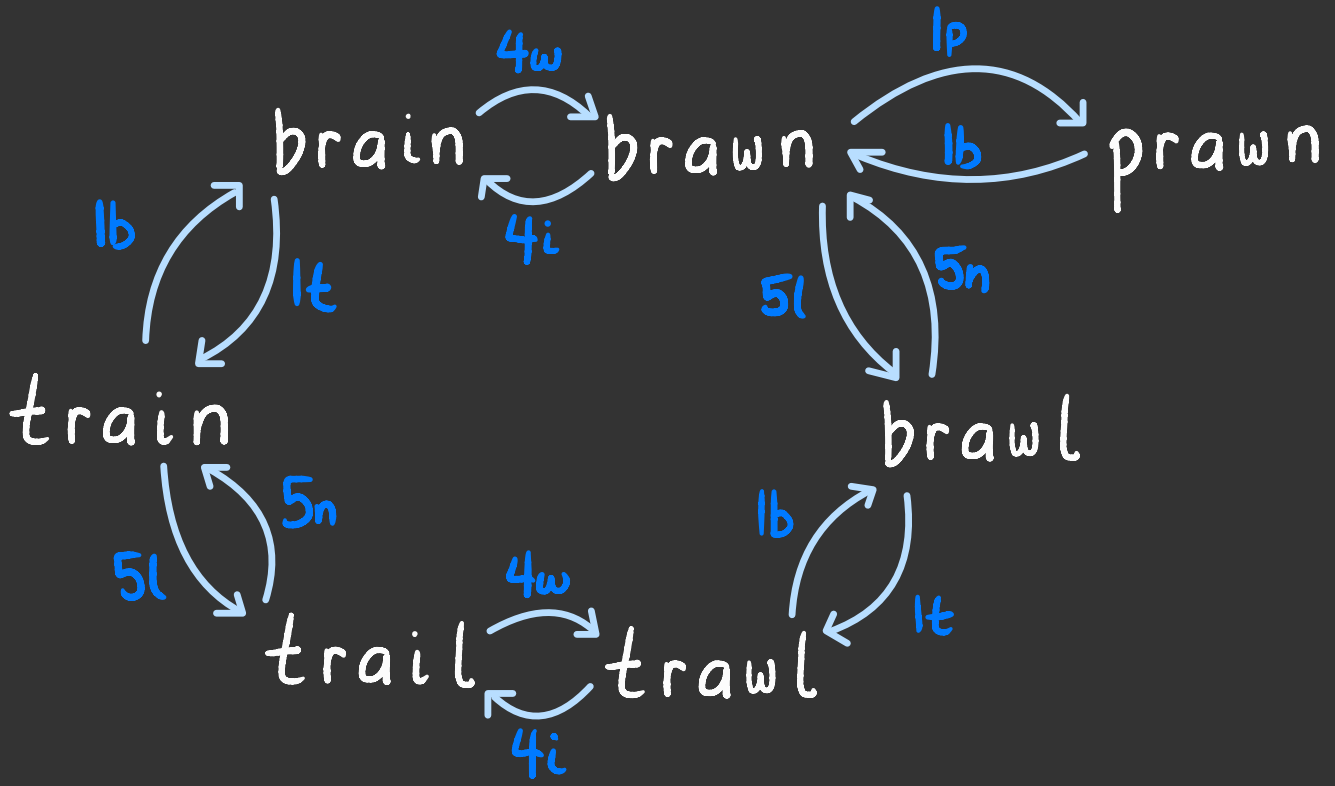
train

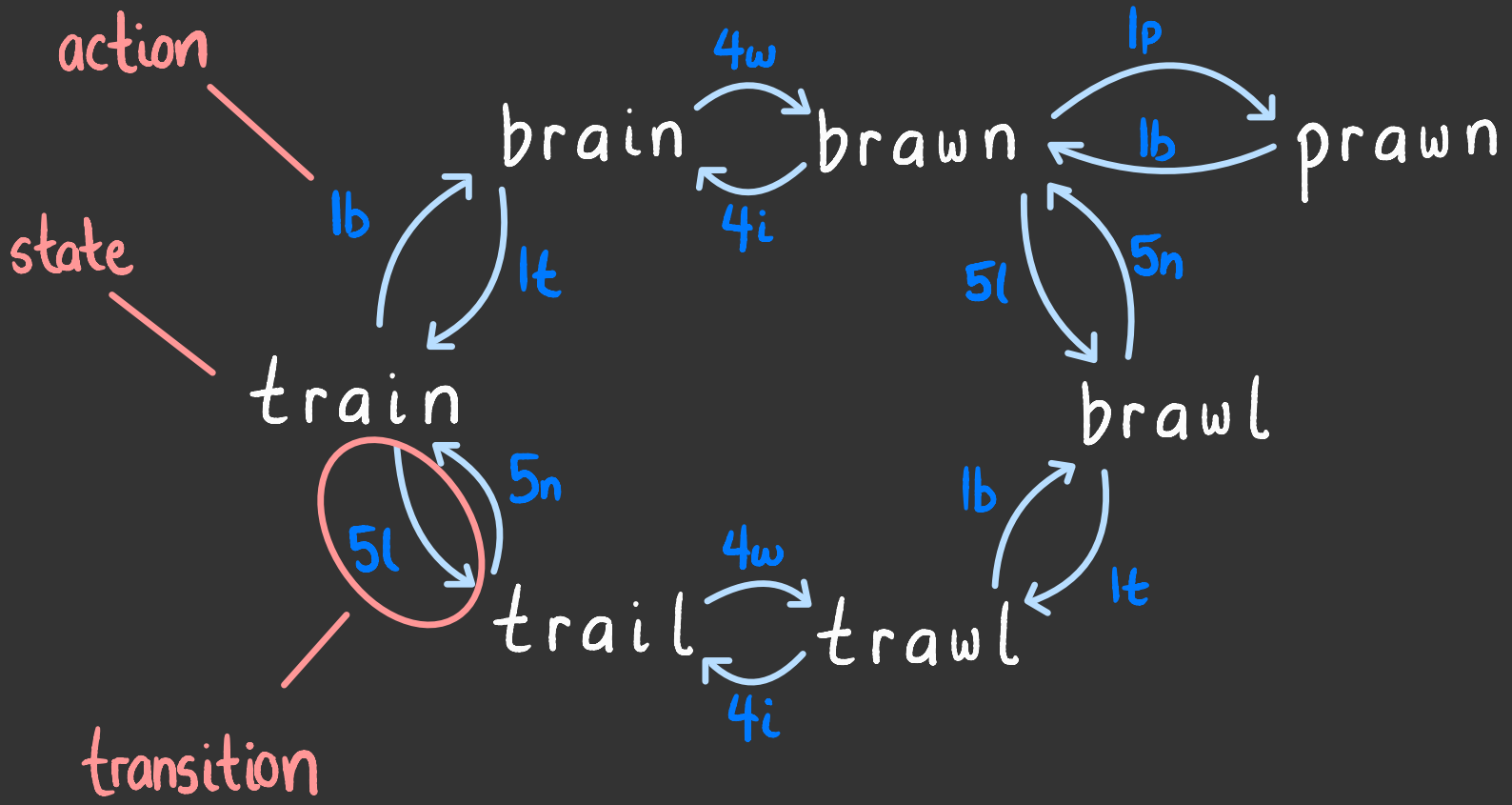
brain

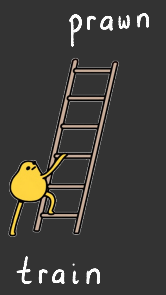
brawn

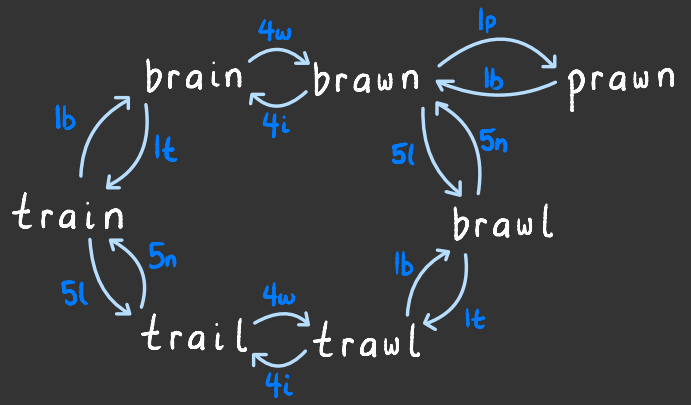
prawn

how do we describe word ladder?


brain
brawn
prawn
train
brawl
trail
trawl
4w
4i
1p
1b
1b
1t
5l
5n
5n
5l
1b
1t
4w
4i

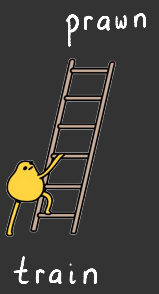

action
state
transition
brain
brawn
prawn
train
brawl
trail
trawl
4w
4i
1p
1b
1b
1t
5l
5n
5n
5l
1b
1t
4w
4i

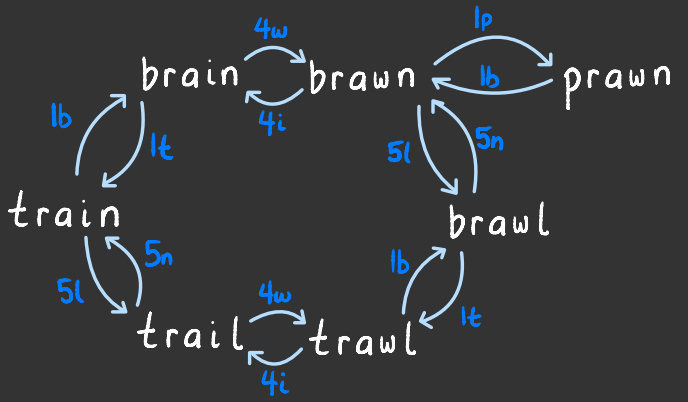




define a state machine as a tuple $(Q, \Sigma, \Delta, q_0, F)$ where:

- Q is a set of states
- Σ is a set of actions
- $\Delta \subseteq Q \times \Sigma \times Q$ is a set of permitted transitions
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is a set of final states





define a state machine as a tuple $(Q, \Sigma, \Delta, q_0, F)$ where:

- Q is a set of states
- Σ is a set of actions
- $\Delta \subseteq Q \times \Sigma \times Q$ is a set of permitted transitions
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is a set of final states

A search **problem** can be defined formally as follows:

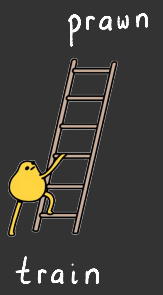
- A set of possible **states** that the environment can be in. We call this the **state space**.
- The **initial state** that the agent starts in. For example: *Arad*.
- A set of one or more **goal states**. Sometimes there is one goal state (e.g., *Bucharest*), sometimes there is a small set of alternative goal states, and sometimes the goal is defined by a property that applies to many states (potentially an infinite number). For example, in a vacuum-cleaner world, the goal might be to have no dirt in any location, regardless of any other facts about the state. We can account for all three of these possibilities by specifying an IS-GOAL method for a problem. In this chapter we will sometimes say "the goal" for simplicity, but what we say also applies to "any one of the possible goal states."
- The **actions** available to the agent. Given a state s, ACTIONS(s) returns a finite[2] set of actions that can be executed in s. We say that each of these actions is **applicable** in s. An example:

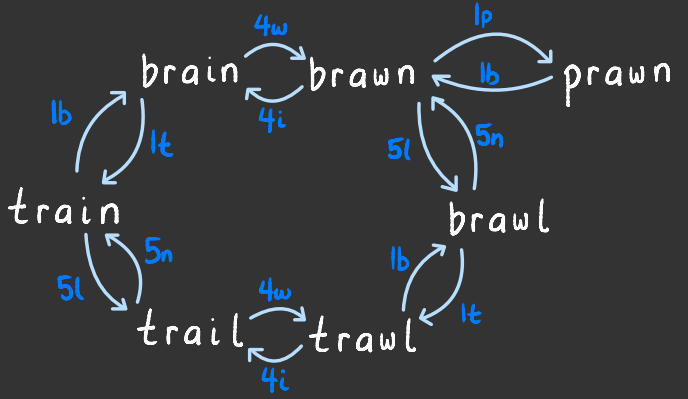
  $$\text{ACTIONS}(Arad) = \{ToSibiu, ToTimisoara, ToZerind\}.$$

- A **transition model**, which describes what each action does. RESULT(s, a) returns the state that results from doing action a in state s. For example,

  $$\text{RESULT}(Arad, ToZerind) = Zerind.$$

- An **action cost function**, denoted by ACTION-COST(s, a, s') when we are programming or $c(s, a, s')$ when we are doing math, that gives the numeric cost of applying action a in state s to reach state s'. A problem-solving agent should use a cost function that reflects its own performance measure; for example, for route-finding agents, the cost of an action might be the length in miles (as seen in Figure 3.1), or it might be the time it takes to complete the action.





define a state machine as a tuple $(Q, \Sigma, \Delta, q_0, F)$ where:

- Q is a set of states
- Σ is a set of actions
- $\Delta \subseteq Q \times \Sigma \times Q$ is a set of permitted transitions
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is a set of final states

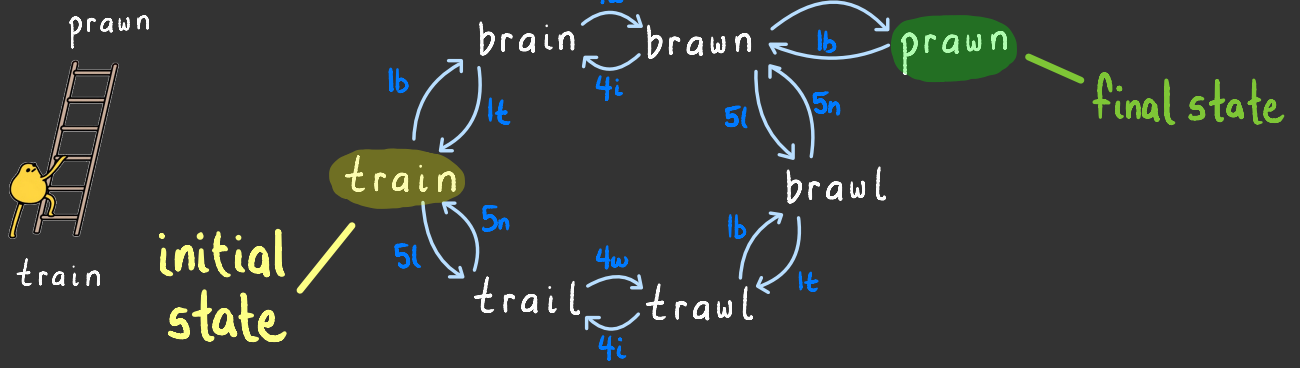
$Q = \{\text{train}, \text{brain}, \text{trail}, \ldots\}$

$\Sigma = \{1a, \ldots, 5z\}$

$\Delta = \{(\text{train}, 1b, \text{brain}), \ldots\}$

$q_0 = \text{train}$

$F = \{\text{prawn}\}$



define a state machine as a tuple $(Q, \Sigma, \Delta, q_0, F)$ where:

- Q is a set of states
- Σ is a set of actions
- $\Delta \subseteq Q \times \Sigma \times Q$ is a set of permitted transitions
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is a set of final states

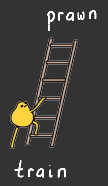
$Q = \{\text{train}, \text{brain}, \text{trail}, \ldots\}$

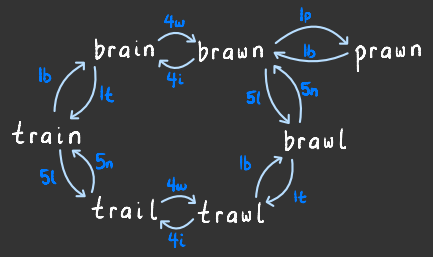
$\Sigma = \{\text{la}, \ldots, \text{5z}\}$

$\Delta = \{(\text{train}, \text{lb}, \text{brain}), \ldots\}$

$q_0 = \text{train}$

$F = \{\text{prawn}\}$





define a state machine as a tuple $(Q, \Sigma, \Delta, q_0, F)$ where:

- Q is a set of states
- Σ is a set of actions
- $\Delta \subseteq Q \times \Sigma \times Q$ is a set of permitted transitions
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is a set of final states

$Q = \{\text{train}, \text{brain}, \text{trail}, \ldots\}$

$\Sigma = \{1a, \ldots, 5z\}$

$\Delta = \{(\text{train}, 1b, \text{brain}), \ldots\}$

$q_0 = \text{train}$

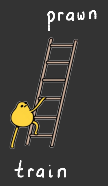
$F = \{\text{prawn}\}$

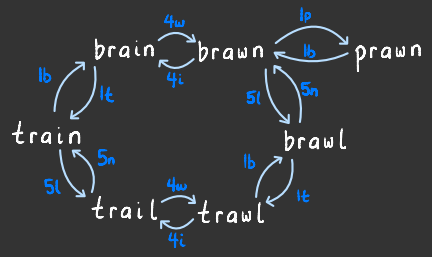
define a weighted state machine as a tuple $(Q, \Sigma, \Delta, q_0, F, w)$ where:

- $(Q, \Sigma, \Delta, q_0, F)$ is a state machine
- $w : \Delta \to \mathbb{R}$ assigns a real-valued weight to each transition

what is the weight function w for word ladder?

your answer here





define a state machine as a tuple $(Q, \Sigma, \Delta, q_0, F)$ where:

- Q is a set of states
- Σ is a set of actions
- $\Delta \subseteq Q \times \Sigma \times Q$ is a set of permitted transitions
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is a set of final states

$Q = \{\text{train}, \text{brain}, \text{trail}, \dots\}$

$\Sigma = \{\text{la}, \dots, \text{5z}\}$

$\Delta = \{(\text{train}, \text{lb}, \text{brain}), \dots\}$

$q_0 = \text{train}$

$F = \{\text{prawn}\}$

define a weighted state machine as a tuple $(Q, \Sigma, \Delta, q_0, F, w)$ where:

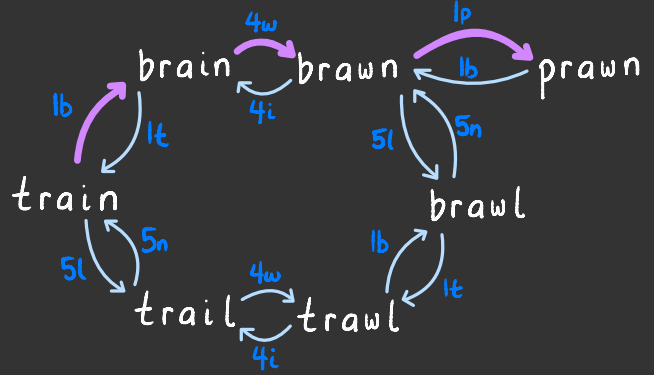
- $(Q, \Sigma, \Delta, q_0, F)$ is a state machine
- $w : \Delta \to \mathbb{R}$ assigns a real-valued weight to each transition

what is the weight function w for word ladder?

$$w = \{\delta \mapsto 1 \mid \delta \in \Delta\}$$

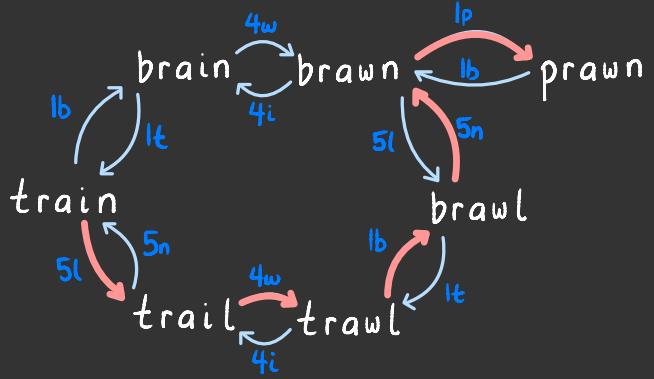
search paths

train
brain
brawn
prawn



formally: your answer here

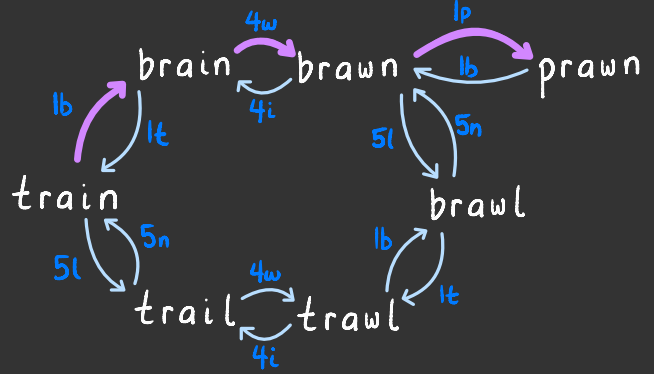
train
trail
trawl
brawl
brawn
prawn



formally: your answer here

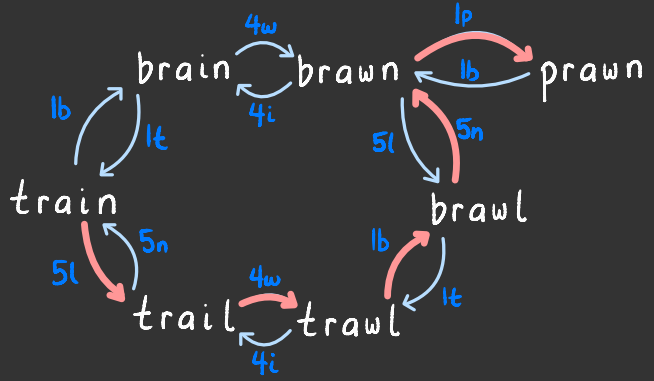
search paths

train
brain
brawn
prawn



formally: ⟨(train, lb, brain), (brain, 4w, brawn), (brawn, lp, prawn)⟩

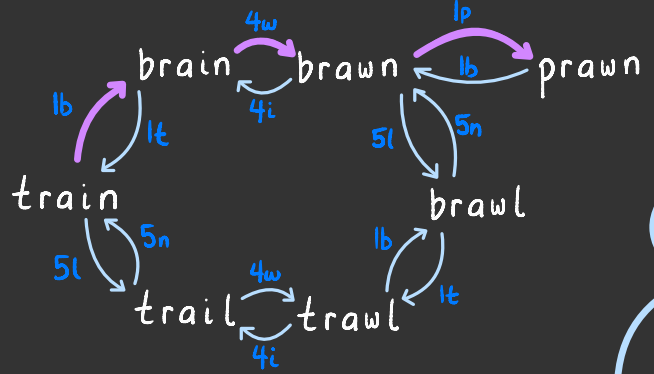
train
trail
trawl
brawl
brawn
prawn



formally: ⟨(train, 5l, trail), (trail, 4w, trawl), (trawl, lb, brawl), (brawl, 5n, brawn), (brawn, lp, prawn)⟩

search paths

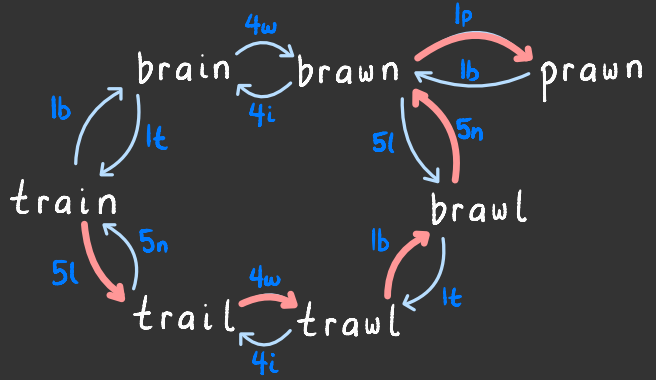
train
brain
brawn
prawn



why do we include the **action**?

formally: ⟨(train, lb, brain), (brain, 4w, brawn), (brawn, lp, prawn)⟩

train
trail
trawl
brawl
brawn
prawn

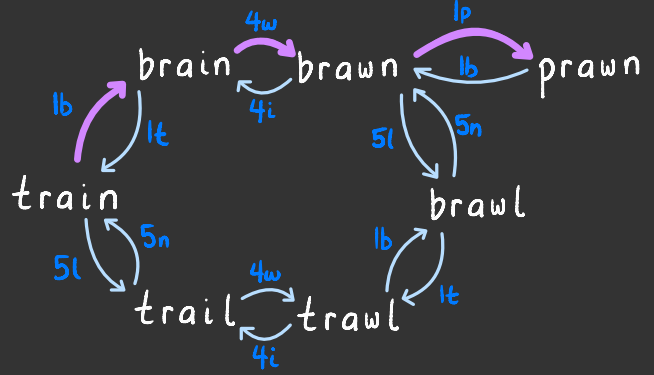


why do we include the next **state**?

formally: ⟨(train, 5l, trail), (trail, 4w, trawl), (trawl, lb, brawl), (brawl, 5n, brawn), (brawn, lp, prawn)⟩

search paths

train
brain
brawn
prawn



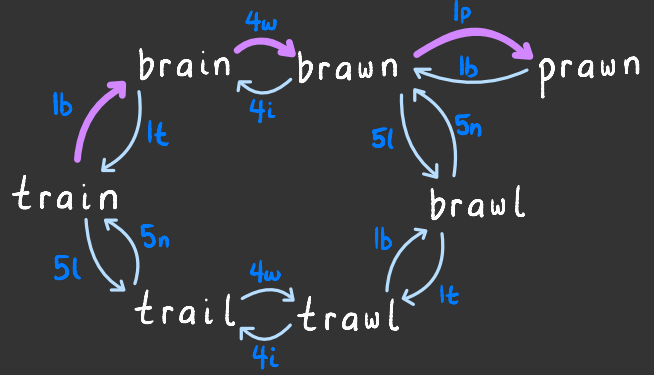
formally: $\langle (\text{train}, \text{lb}, \text{brain}), (\text{brain}, \text{4w}, \text{brawn}), (\text{brawn}, \text{lp}, \text{prawn}) \rangle$

states	$Q = \{\text{train}, \text{brain}, \text{trail}, \dots\}$
actions	$\Sigma = \{\text{la}, \dots, \text{5z}\}$
transitions	$\Delta = \{(\text{train}, \text{lb}, \text{brain}), \dots\}$
initial state	$q_0 = \text{train}$
final states	$F = \{\text{prawn}\}$
weight function	$w = \{\delta \mapsto 1 \mid \delta \in \Delta\}$

a **search path** is a sequence $\langle \delta_0, \dots, \delta_k \rangle$ of **transitions** from Δ such that there exist **states** $q_1, \dots, q_{k+1} \in Q$ and **actions** $\sigma_0, \dots, \sigma_k \in \Sigma$ such that $\delta_i = (q_i, \sigma_i, q_{i+1}) \quad \forall i \in \{0, \dots, k\}$

search paths

train
brain
brawn
prawn



formally: $\langle (\text{train}, \text{lb}, \text{brain}), (\text{brain}, \text{4w}, \text{brawn}), (\text{brawn}, \text{lp}, \text{prawn}) \rangle$

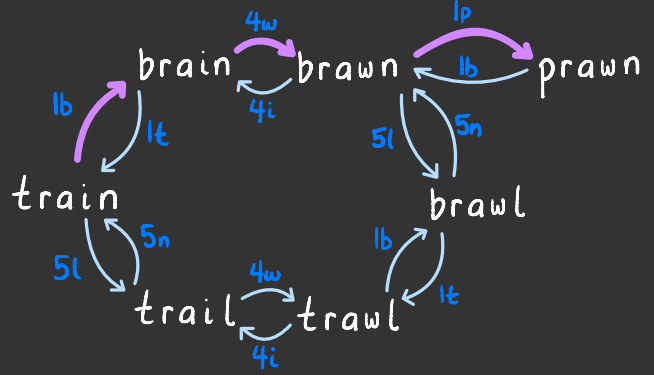
states	$Q = \{\text{train}, \text{brain}, \text{trail}, \dots\}$
actions	$\Sigma = \{\text{la}, \dots, \text{5z}\}$
transitions	$\Delta = \{(\text{train}, \text{lb}, \text{brain}), \dots\}$
initial state	$q_0 = \text{train}$
final states	$F = \{\text{prawn}\}$
weight function	$w = \{\delta \mapsto 1 \mid \delta \in \Delta\}$

a search path is a sequence $\langle \delta_0, \dots, \delta_k \rangle$ of transitions from Δ such that there exist states $q_1, \dots, q_{k+1} \in Q$ and actions $\sigma_0, \dots, \sigma_k \in \Sigma$ such that $\delta_i = (q_i, \sigma_i, q_{i+1}) \quad \forall i \in \{0, \dots, k\}$

how does this differ from an arbitrary sequence of transitions?

search paths

train
brain
brawn
prawn



formally: $\langle (\text{train}, \text{lb}, \text{brain}), (\text{brain}, \text{4w}, \text{brawn}), (\text{brawn}, \text{lp}, \text{prawn}) \rangle$

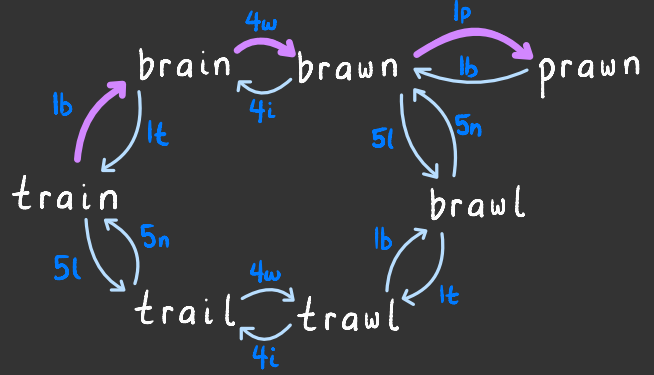
states	$Q = \{\text{train}, \text{brain}, \text{trail}, \dots\}$
actions	$\Sigma = \{\text{la}, \dots, \text{5z}\}$
transitions	$\Delta = \{(\text{train}, \text{lb}, \text{brain}), \dots\}$
initial state	$q_0 = \text{train}$
final states	$F = \{\text{prawn}\}$
weight function	$w = \{\delta \mapsto 1 \mid \delta \in \Delta\}$

a **search path** is a sequence $\langle \delta_0, \dots, \delta_k \rangle$ of transitions from Δ such that there exist states $q_1, \dots, q_{k+1} \in Q$ and actions $\sigma_0, \dots, \sigma_k \in \Sigma$ such that $\delta_i = (q_i, \sigma_i, q_{i+1}) \quad \forall i \in \{0, \dots, k\}$

why do these start at different indices?

search paths

train
brain
brawn
prawn



formally: ⟨(train, lb, brain), (brain, 4w, brawn), (brawn, lp, prawn)⟩

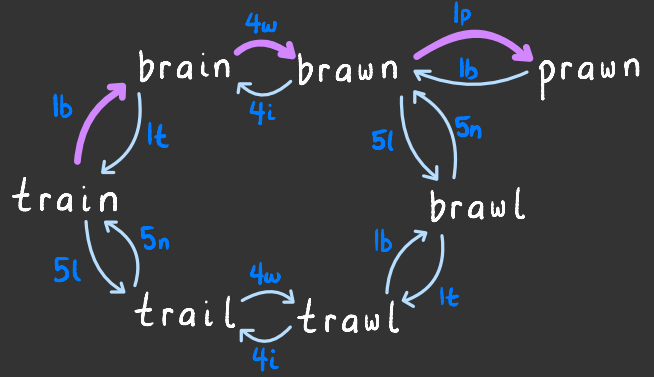
this must be the initial state

these must match

these must match

in a search path

train
brain
brawn
prawn



a search path is a sequence $\langle \delta_0, \dots, \delta_k \rangle$ of transitions from Δ such that there exist states $q_1, \dots, q_{k+1} \in Q$ and actions $\sigma_0, \dots, \sigma_k \in \Sigma$ such that $\delta_i = (q_i, \sigma_i, q_{i+1}) \quad \forall i \in \{0, \dots, k\}$

formally: $\langle (\text{train}, \text{lb}, \text{brain}), (\text{brain}, \text{4w}, \text{brawn}), (\text{brawn}, \text{lp}, \text{prawn}) \rangle$

states $Q = \{\text{train}, \text{brain}, \text{trail}, \dots\}$

actions $\Sigma = \{\text{la}, \dots, \text{5z}\}$

transitions $\Delta = \{(\text{train}, \text{lb}, \text{brain}), \dots\}$

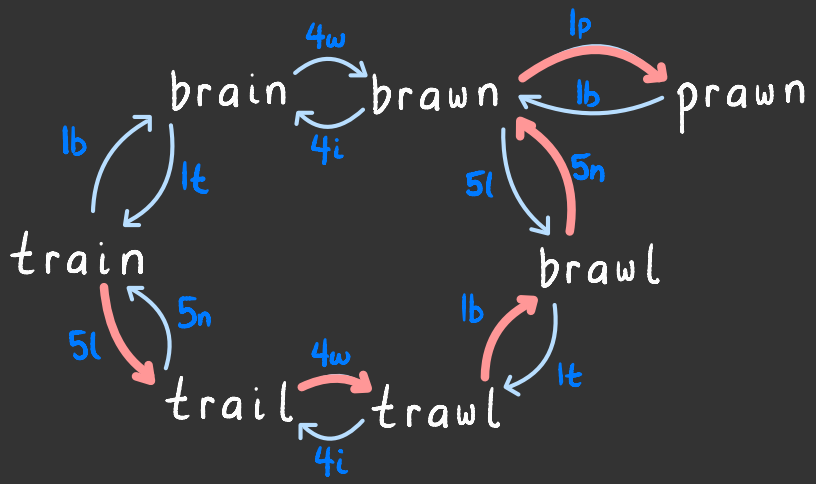
initial state $q_0 = \text{train}$

final states $F = \{\text{prawn}\}$

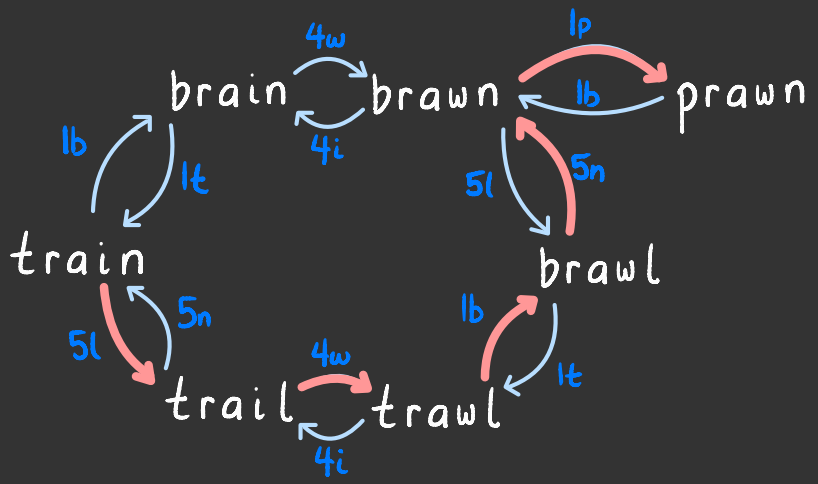
weight function $w = \{\delta \mapsto 1 \mid \delta \in \Delta\}$

the cost of a search path is the sum of the weights of the transitions

what is the cost of this search path?

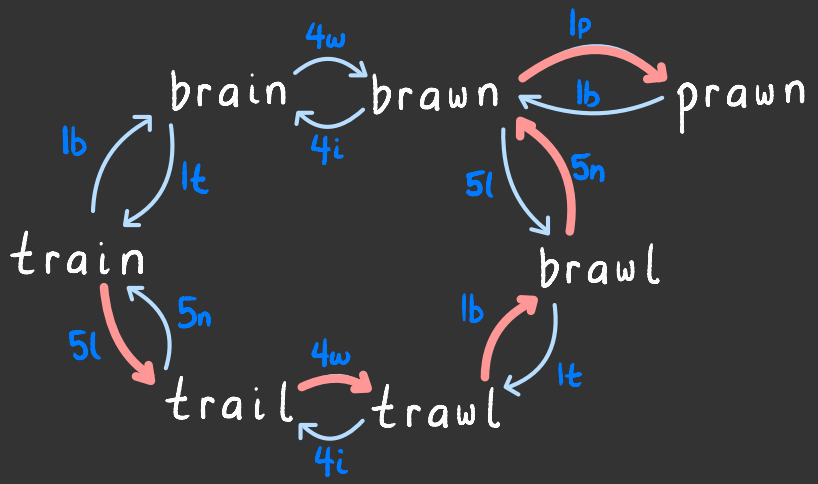


what is the cost of this search path?



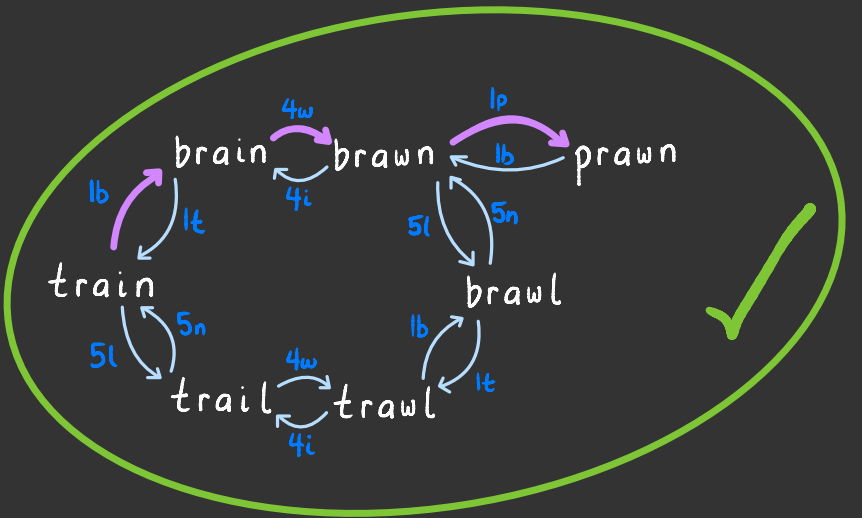
$$w((\text{train}, 5\text{l}, \text{trail}))$$
$$+ w((\text{trail}, 4\text{w}, \text{trawl}))$$
$$+ w((\text{trawl}, 1\text{b}, \text{brawl}))$$
$$+ w((\text{brawl}, 5\text{n}, \text{brawn}))$$
$$+ w((\text{brawn}, 1\text{p}, \text{prawn}))$$

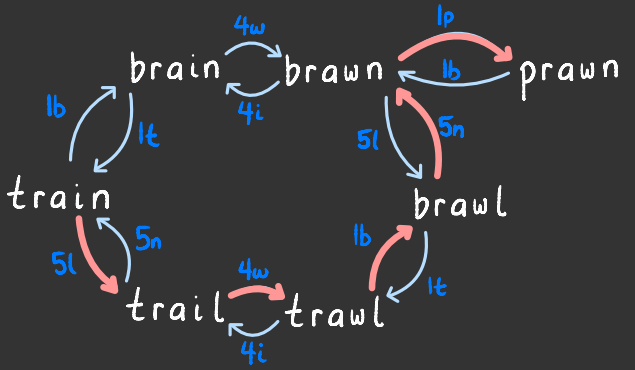
what is the cost of this search path?



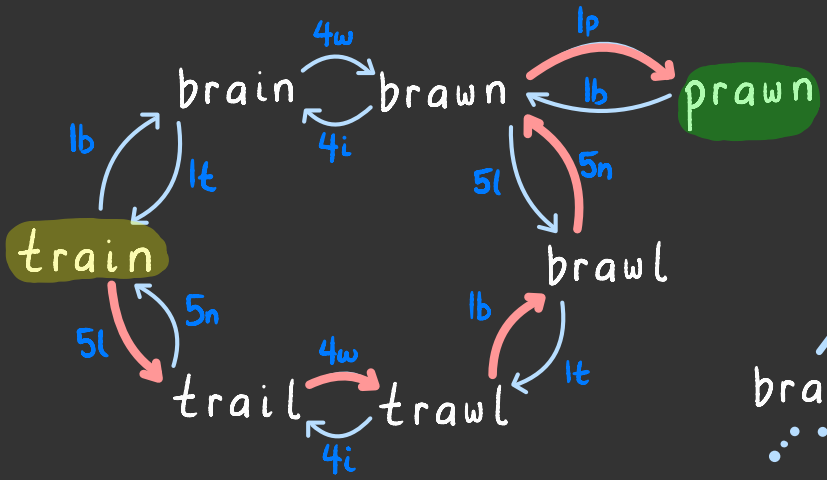
$$
\begin{aligned}
& w((\text{train}, 5\text{l}, \text{trail})) && 1 \\
+\ & w((\text{trail}, 4\text{w}, \text{trawl})) && +\ 1 \\
+\ & w((\text{trawl}, 1\text{b}, \text{brawl})) && +\ 1 \\
+\ & w((\text{brawl}, 5\text{n}, \text{brawn})) && +\ 1 \\
+\ & w((\text{brawn}, 1\text{p}, \text{prawn})) && +\ 1 \\
\hline
& && 5
\end{aligned}
$$

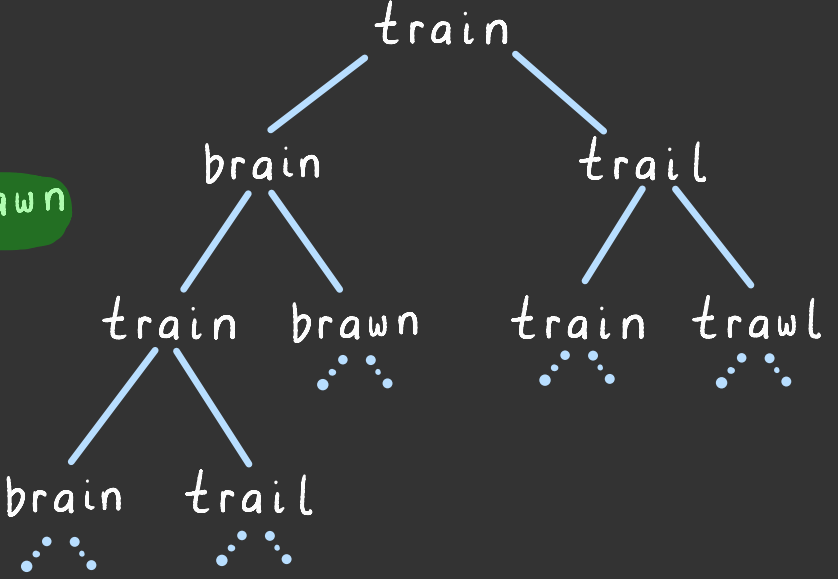
the goal of search:
to find a search path
of minimal cost whose
last transition leads to
a final state





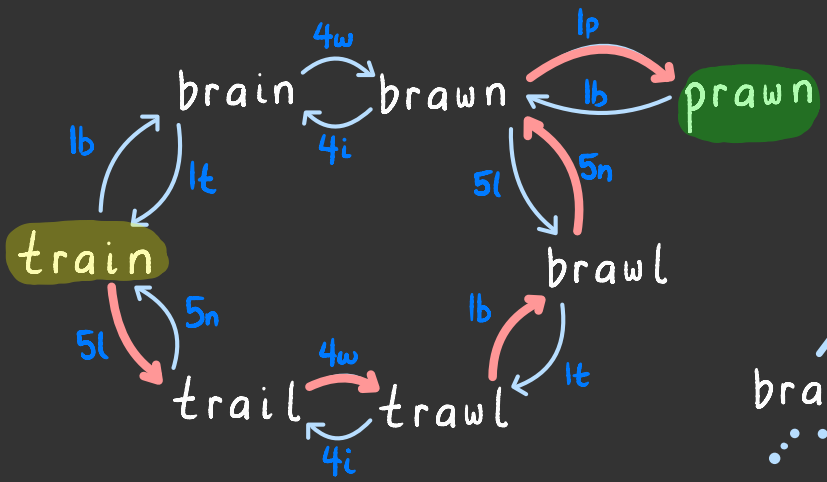
a state machine can be unraveled into a search tree

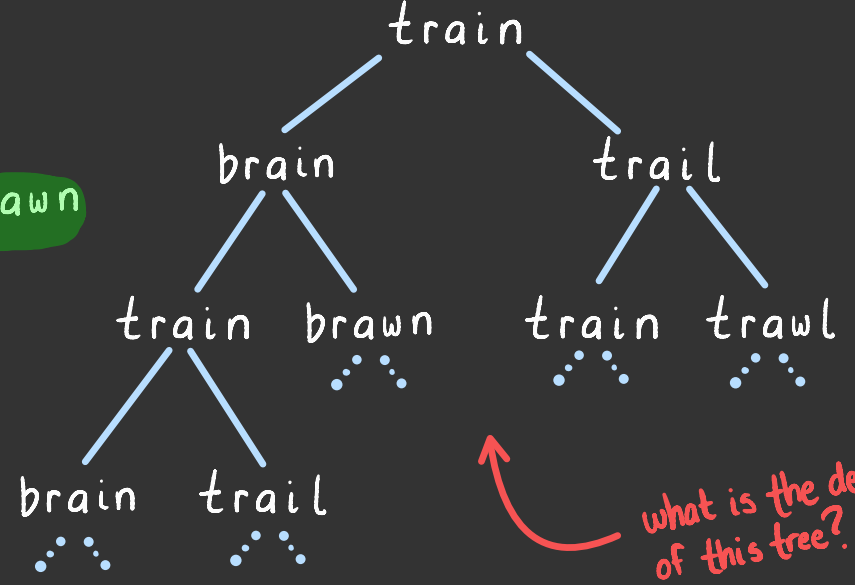




the search tree is a compact representation of all the search paths

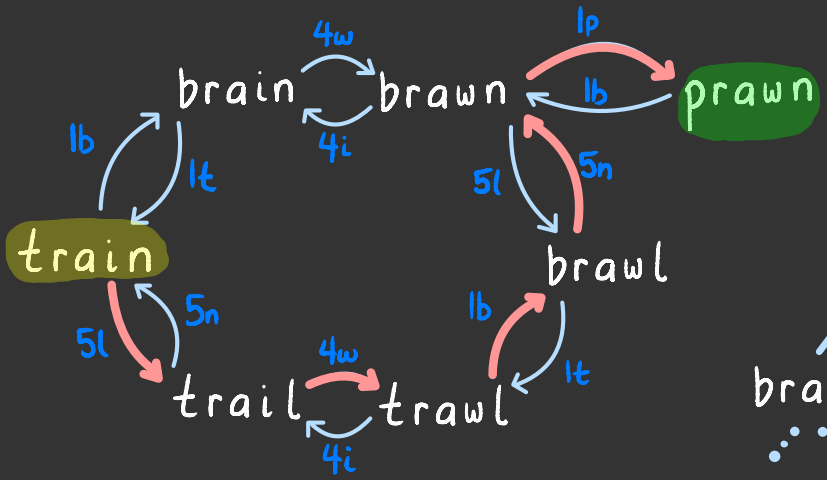
a state machine can be unraveled into a search tree

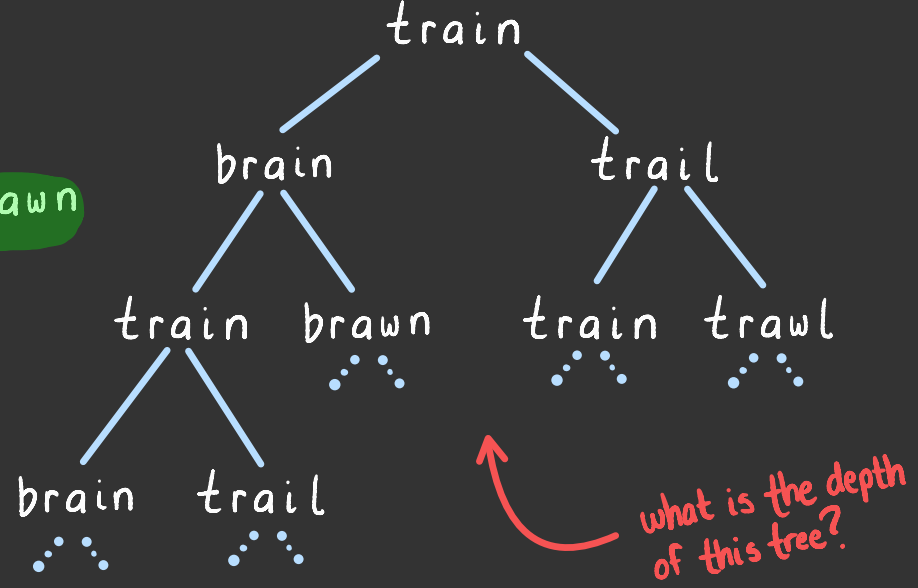




your **answer** here

a state machine can be unraveled into a search tree





infinite. in general, if a state machine has reachable cycles, then its search tree has infinite depth