depth-first search

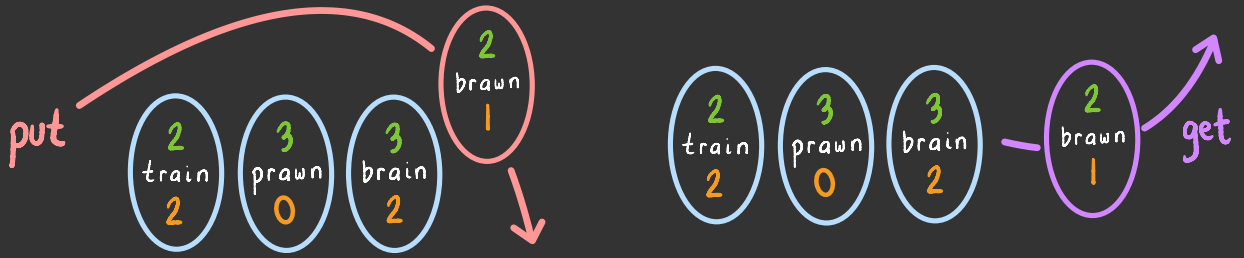
16 sept 2022

CSCI 373

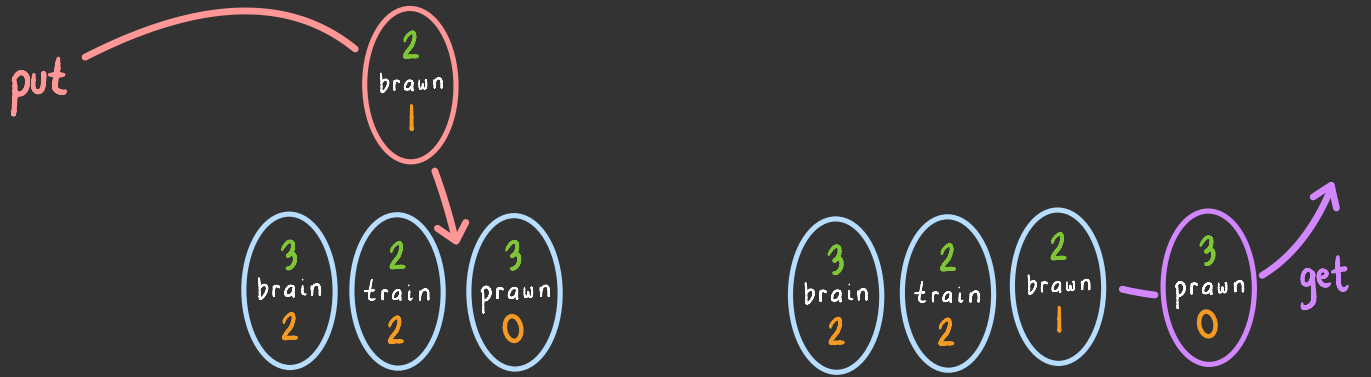
example containers

queue

stack



priority queue

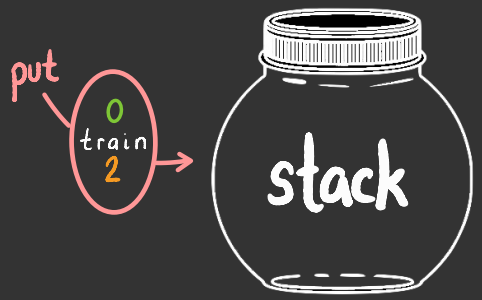



queue
put
2 brawn 1
2 train 2
3 prawn 0
3 brain 2
2 brawn 1
2 train 2
3 prawn 0
3 brain 2
get
stack
put
2 brawn 1
2 train 2
3 prawn 0
3 brain 2
2 train 2
3 prawn 0
3 brain 2
2 brawn 1
get
priority queue
put
2 brawn 1
3 brain 2
2 train 2
3 prawn 0
3 brain 2
2 train 2
2 brawn 1
3 prawn 0
get

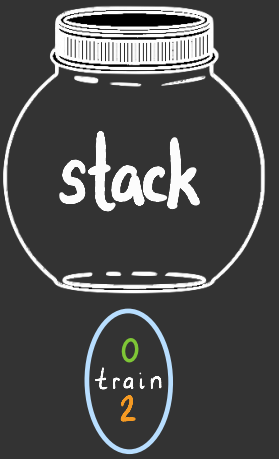
Search($M = (Q, \Sigma, \Delta, q_0, F, w)$, H):

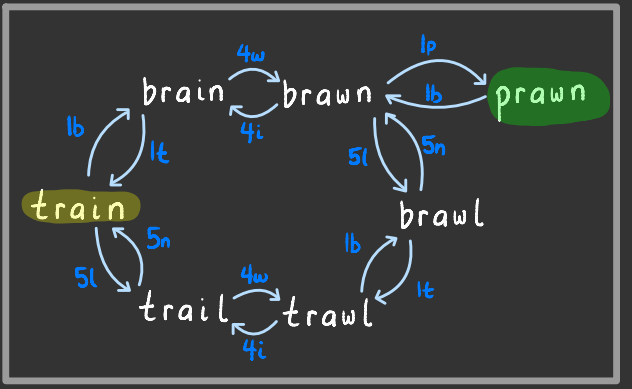
- container = new stack()
- container.put($\langle q_0, 0, H(q_0) \rangle$)
- repeat:
  - if container.empty() then return ∞
  - n = container.get()
  - if $q(n) \in F$ then return $g(n)$
  - let $\text{successors}_{M,H}(n) = \{ \langle q', g(n) + w(q(n), \sigma, q'), H(q') \rangle \mid (q(n), \sigma, q') \in \Delta \}$
  - for $n' \in \text{successors}_{M,H}(n)$:
    container.put(n')

"visit" a node

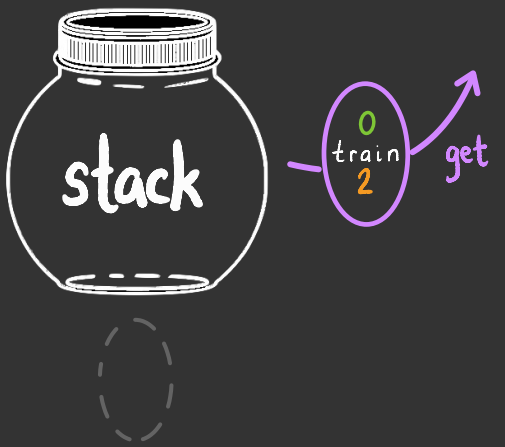


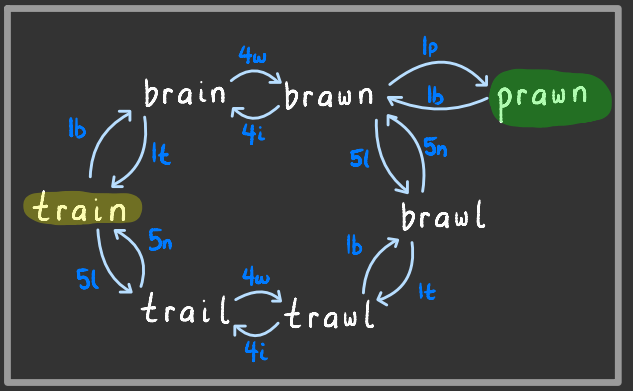

0
train
2

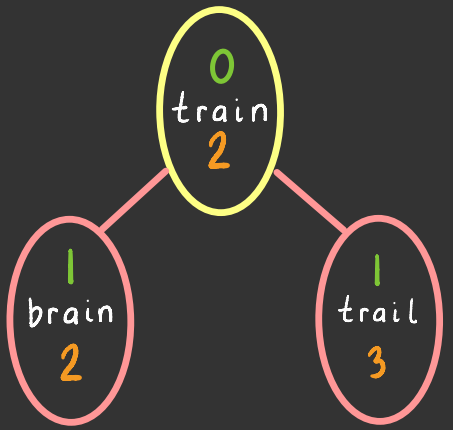

stack
0
train
2

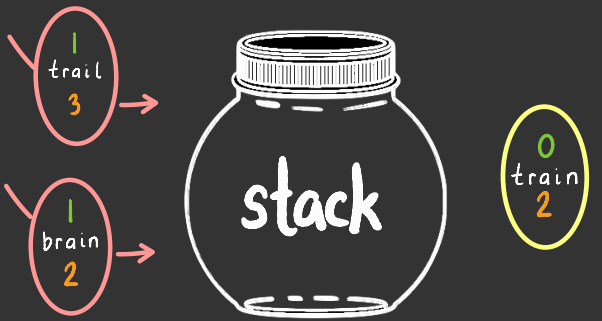

brain
brawn
prawn
train
brawl
trail
trawl
4w
4i
1p
1b
1b
1t
5l
5n
5n
5l
1b
1t
4w
4i

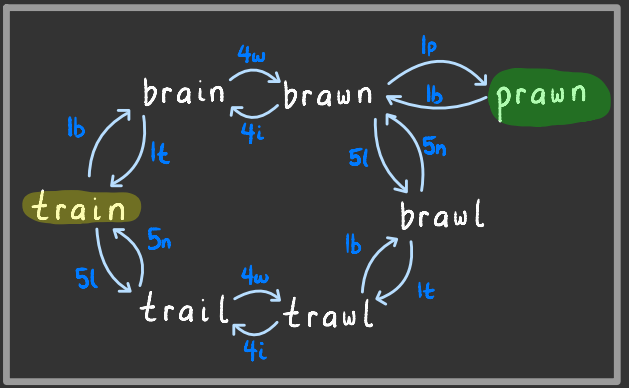

0
train
2

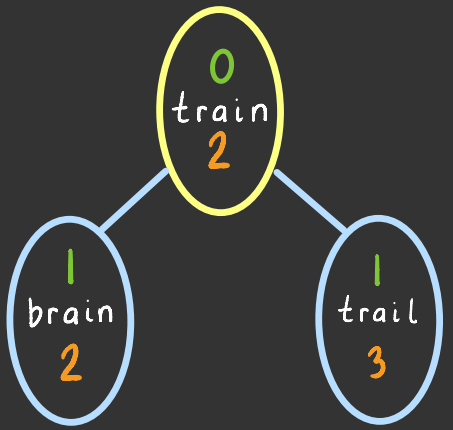

stack
0
train
2
get

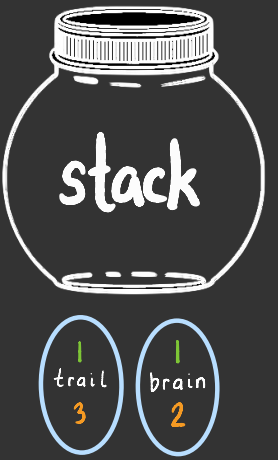

brain
brawn
prawn
train
brawl
trail
trawl
4w
4i
1p
1b
1b
1t
5l
5n
5n
5l
1b
1t
4w
4i

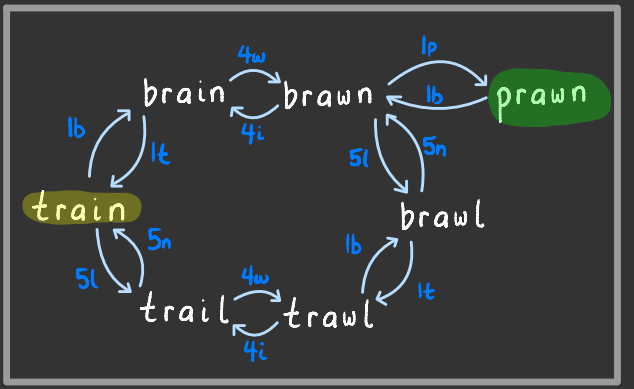

0
train
2
1
brain
2
1
trail
3

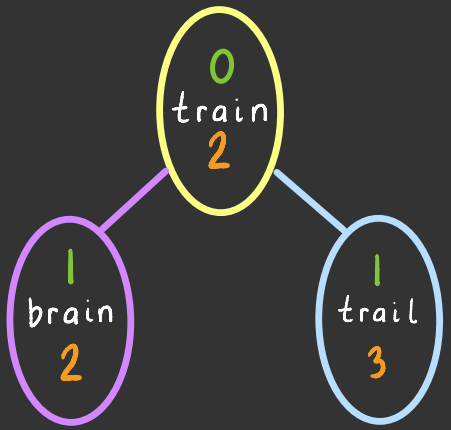

1
trail
3
1
brain
2
stack
0
train
2

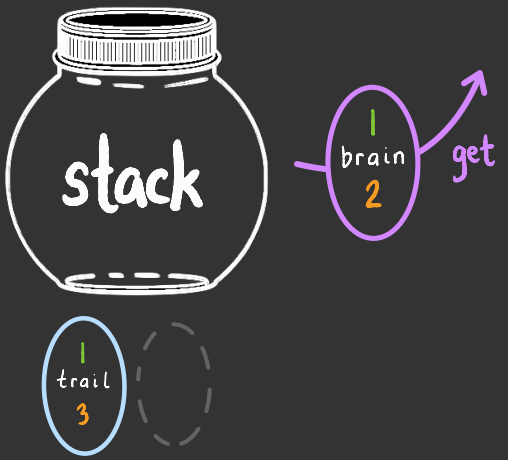

brain
brawn
prawn
train
brawl
trail
trawl
4w
4i
1p
1b
1b
1t
5l
5n
5n
5l
1b
1t
4w
4i

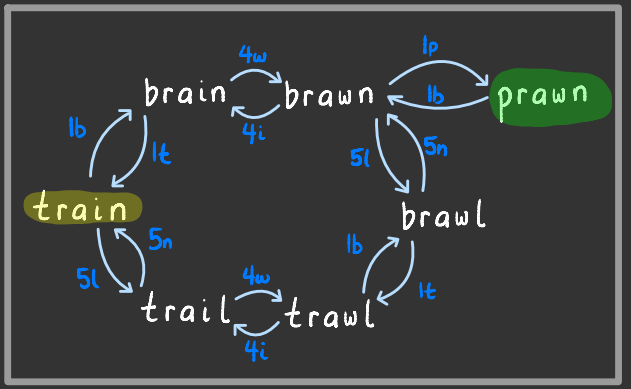

0
train
2
1
brain
2
1
trail
3

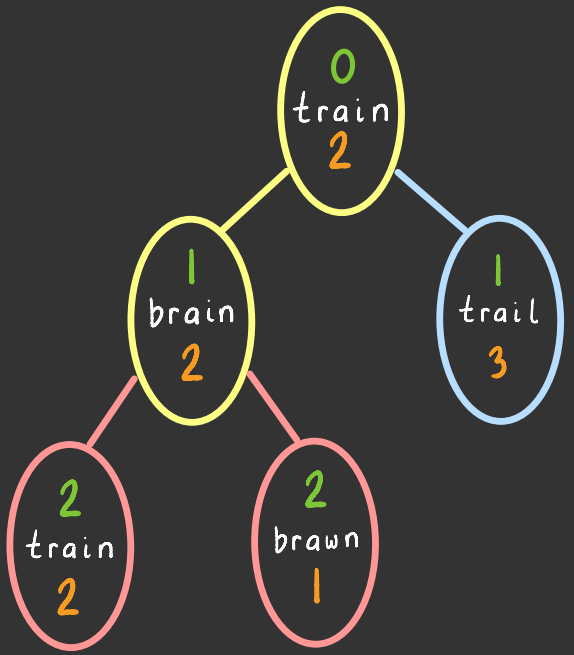

stack
1
trail
3
1
brain
2

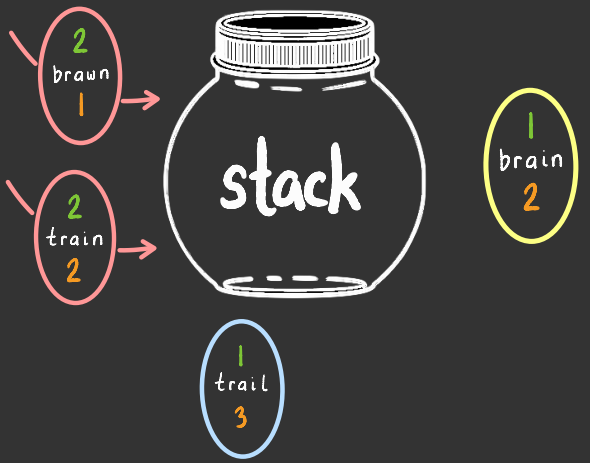

brain
brawn
prawn
train
brawl
trail
trawl
4w
4i
1p
1b
1b
1t
5l
5n
5n
5l
1b
1t
4w
4i

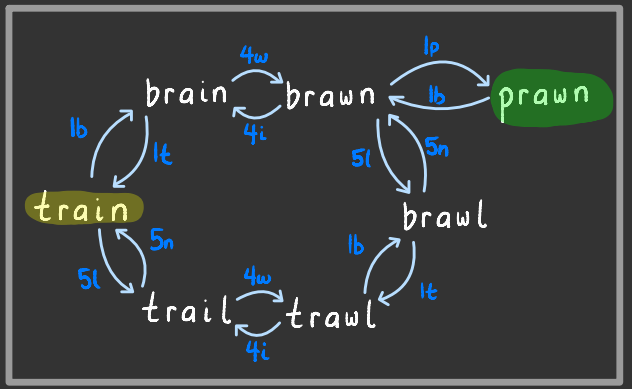

0
train
2
1
brain
2
1
trail
3

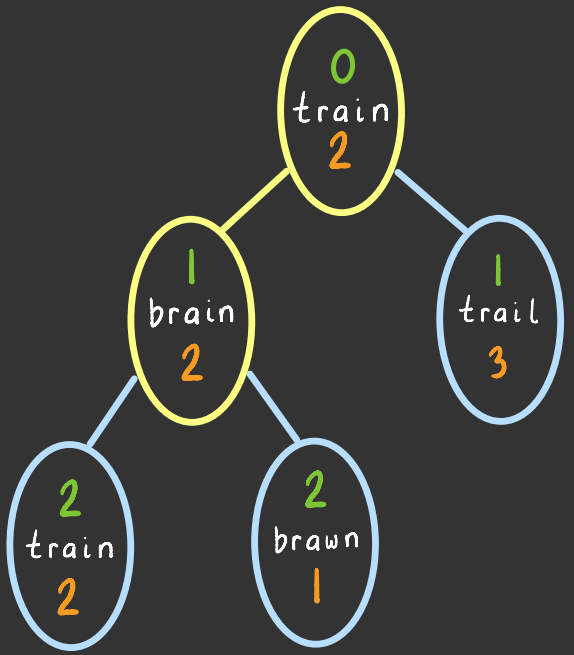

stack
1
brain
2
get
1
trail
3

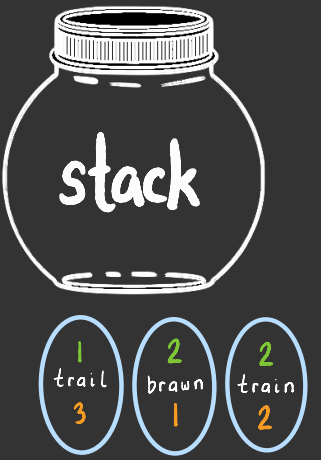

brain
brawn
prawn
train
brawl
trail
trawl
4w
4i
1p
1b
1b
1t
5l
5n
5n
5l
1b
1t
4w
4i

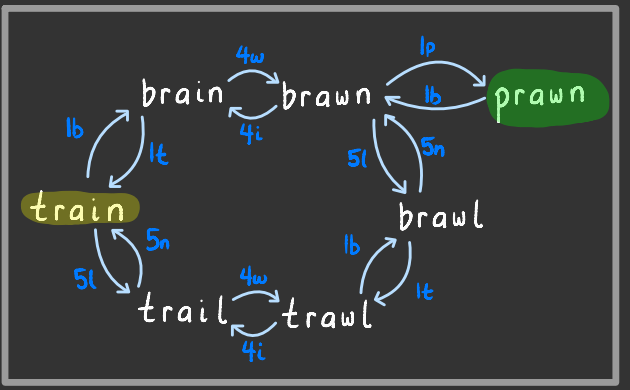

0
train
2
1
brain
2
1
trail
3
2
train
2
2
brawn
1

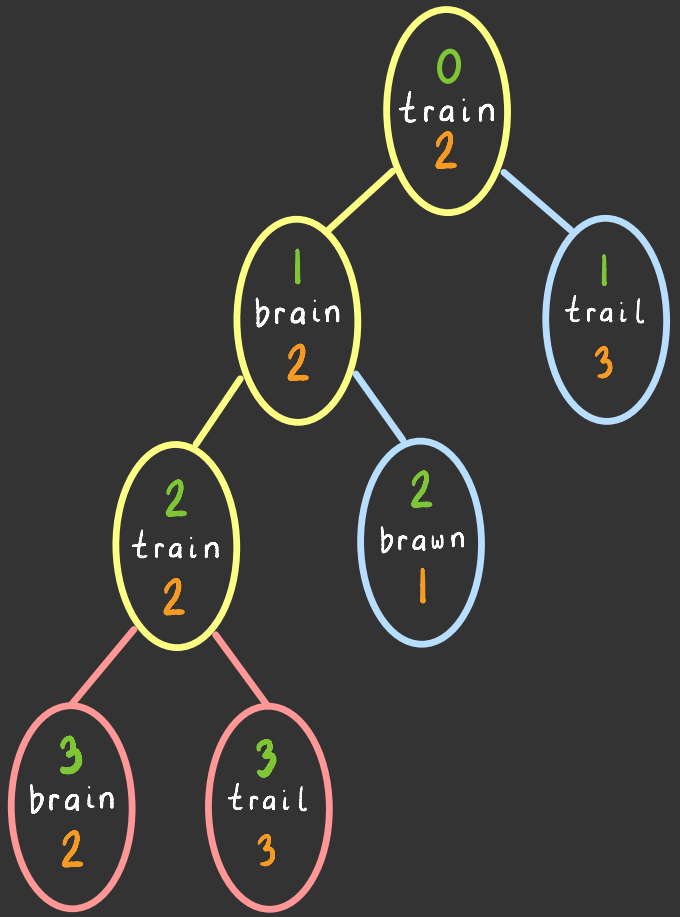

2
brawn
1
2
train
2
stack
1
brain
2
1
trail
3

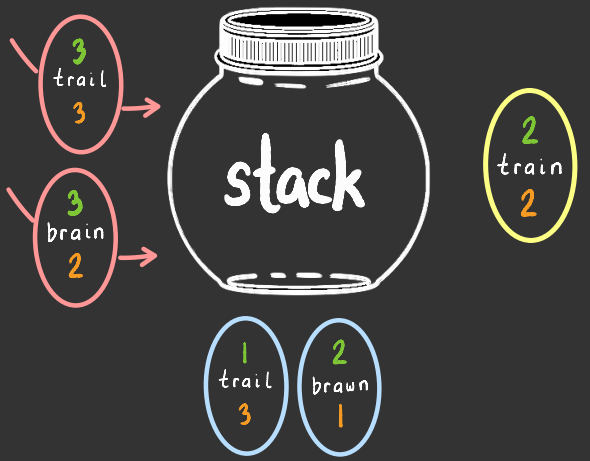

brain
brawn
prawn
train
brawl
trail
trawl
4w
4i
1p
1b
1b
1t
5l
5n
5n
5l
1b
1t
4w
4i

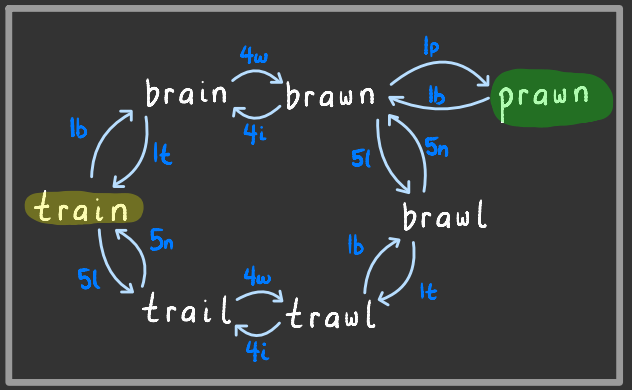

0
train
2
1
brain
2
1
trail
3
2
train
2
2
brawn
1

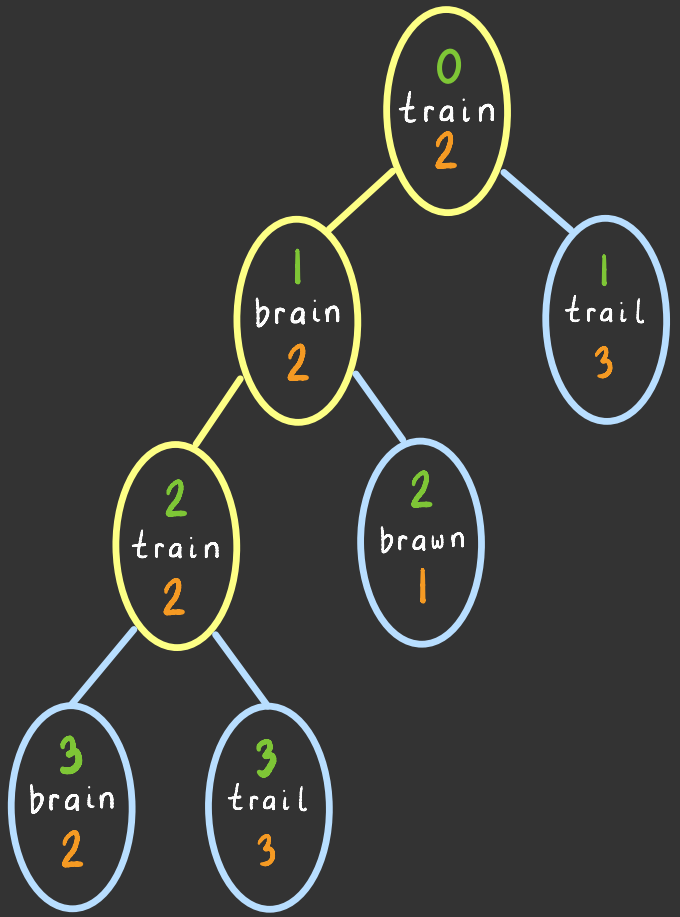

stack
1
trail
3
2
brawn
1
2
train
2

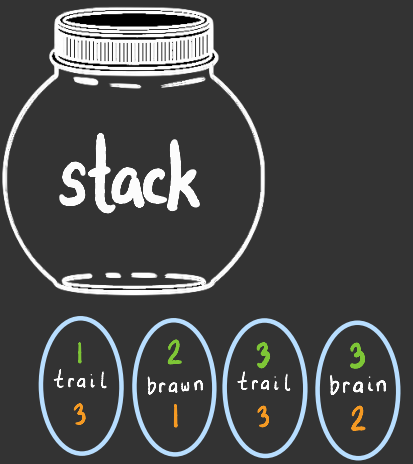

brain
brawn
prawn
train
brawl
trail
trawl
4w
4i
1p
1b
1b
1t
5t
5n
5n
5t
1b
1t
4w
4i

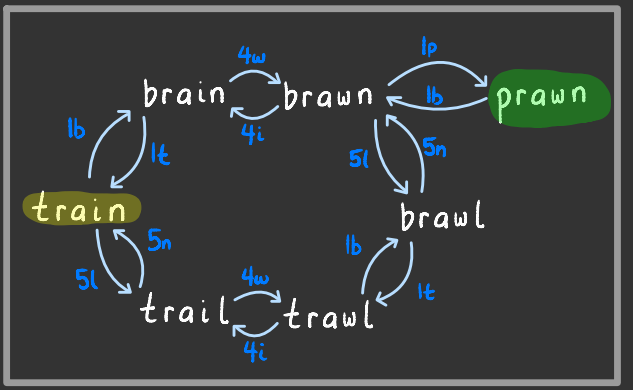

0
train
2
1
brain
2
1
trail
3
2
train
2
2
brawn
1
3
brain
2
3
trail
3

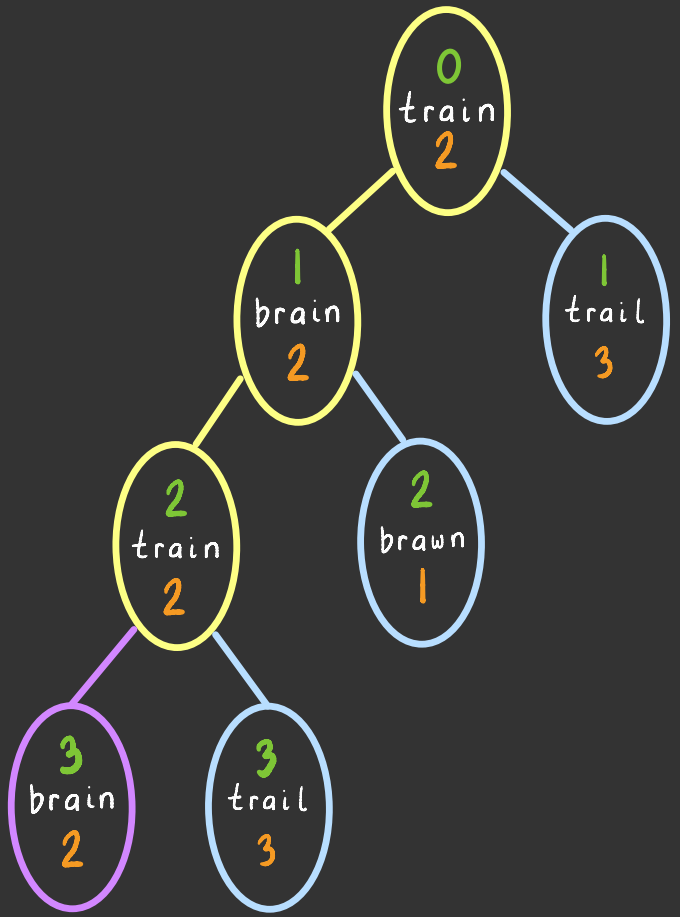

3
trail
3
3
brain
2
stack
2
train
2
1
trail
3
2
brawn
1

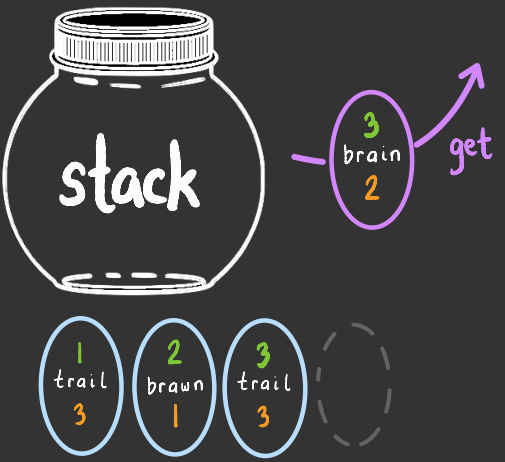

brain
brawn
prawn
train
brawl
trail
trawl
4w
4i
1p
1b
1b
1t
5l
5n
5n
5l
1b
1t
4w
4i

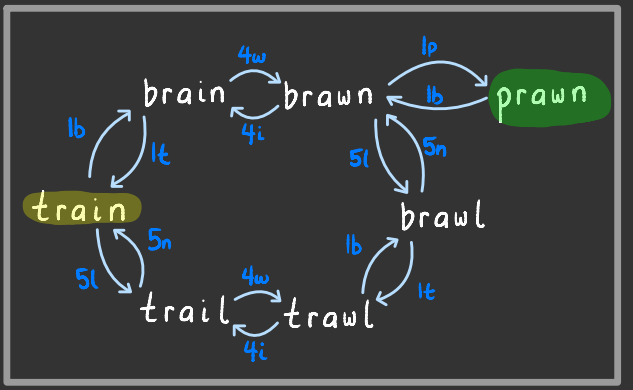

0
train
2
1
brain
2
1
trail
3
2
train
2
2
brawn
1
3
brain
2
3
trail
3

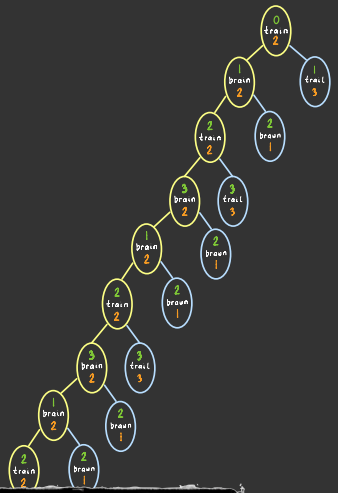

stack
1
trail
3
2
brawn
1
3
trail
3
3
brain
2


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

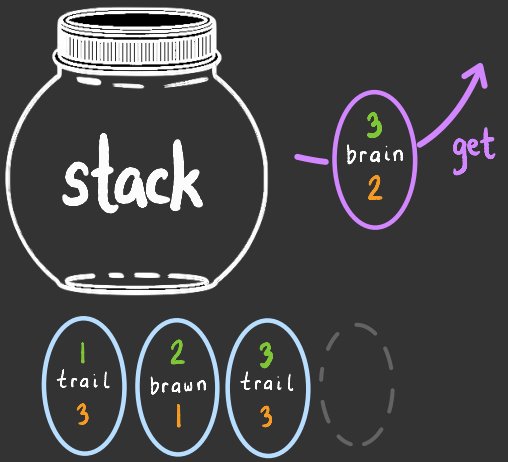

0
train
2
1
brain
2
1
trail
3
2
train
2
2
brawn
1
3
brain
2
3
trail
3

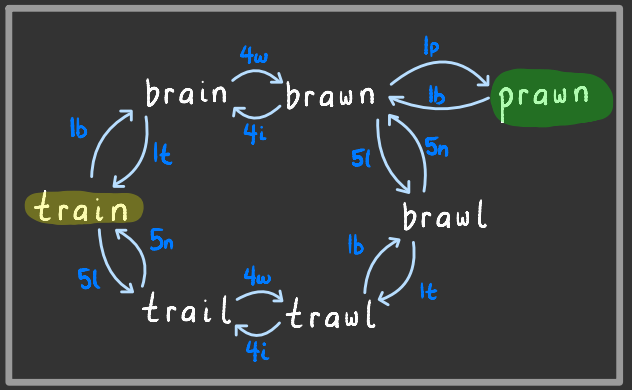

stack
3
brain
2
get
1
trail
3
2
brawn
1
3
trail
3


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


0 train 2
1 brain 2
trail 3
2 train 2
2 brawn 1
3 brain 2
3 trail 3
brain 2
2 brawn 1
2 train 2
2 brawn 1
3 brain 2
3 trail 3
1 brain 2
2 brawn 1
2 train 2
2 brawn 1


hieronymus bosch


stack
3 brain 2
get
1 trail 3
2 brawn 1
3 trail 3


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

SEARCH$(M = (Q, \Sigma, \Delta, q_0, F, w), H)$:

- container = new stack()
- container.put($\langle q_0, 0, H(q_0) \rangle$)
- repeat:
  - if container.empty() then return ∞
  - n = container.get()
  - if $q(n) \in F$ then return $g(n)$
  - let $\text{successors}_{M,H}(n) = \{ \langle q', g(n) + w(q(n), \sigma, q'), H(q') \rangle \mid (q(n), \sigma, q') \in \Delta \}$
  - for $n' \in \text{successors}_{M,H}(n)$:
    container.put(n')

"visit" a node

if we use a 

as the container, the algorithm is called depth-first search, abbreviated DFS

technique	container	optimal?	but
breadth-first search	queue	not guaranteed	guaranteed if all transitions have the same weight
depth-first search	stack	not guaranteed	
uniform cost search	priority queue (prio to lowest g)	guaranteed	

technique	container	optimal?	but
breadth-first search	queue	not guaranteed	guaranteed if all transitions have the same weight
depth-first search	stack	not guaranteed	but what?
uniform cost search	priority queue (prio to lowest g)	guaranteed	

but what if the state machine is acyclic
and all goal nodes are at the same search depth?

but what if the state machine is acyclic and all goal nodes are at the same search depth?