Search

not physical search

but search in the head

first mentally determine a sequence of actions,

then execute those actions

if an action fails, perform a new search

State-Space ~~Problem~~ Graph

A state-space ~~problem~~ graph consists of

- a set of *states*
- a subset of states (typically just one) called the *start states* (initial state)
- a set of *actions*
- an *action function* that maps from a state and an action to a state
- a set of *goal states*, or, equivalently, a Boolean function, *goal*(s), that is true when s is a goal state
- a function that measures the quality of a solution

cookie problem

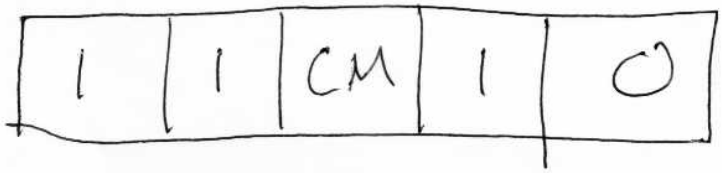
Cookie monster wants to eat all the cookies.

~~CM start in a particular~~ locations are represented as a grid.

a cookie may or may not be at a location

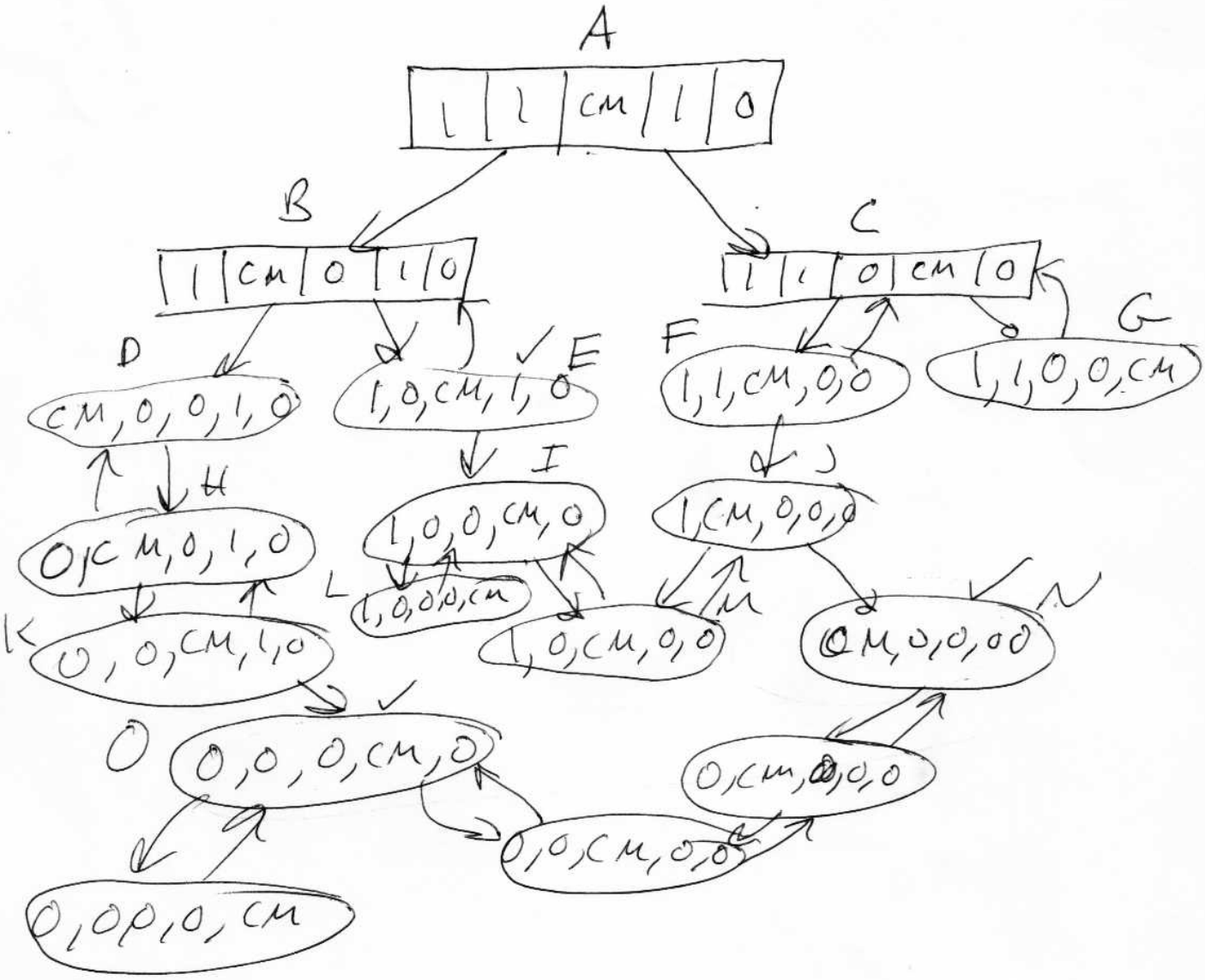
CM start at a particular location

CM knows where the cookies are.
eats cookies when it finds them.



~~R~~ R L L L

L L R R R



Frontier starts with (A)
(A) is removed
not a goal state
add ~~not~~ (A, B) and (A, C) to frontier
remove (A, C) from frontier
does not reach a goal state
add (A, C, F) and (A, C, G) to frontier

Generic Search Algorithm

Procedure *Search*(G, S, *goal*)

Inputs: ~~G: graph with nodes N and arcs A~~

S: set of start nodes (typically just one)

goal: Boolean function of nodes

Output: path from (s: initial state) to (g: goal state) s.t. $s \in S$ and *goal*(g)

or null if no solution paths are found

Frontier $\leftarrow \{(s) \mid s \in S\}$ // a set of paths

while *Frontier* is not empty

remove a path $p = (s, \ldots, t)$ from *Frontier*

if *goal*(t) then return p

for each arc from t to n (for all state we can reach from t in one action)

insert $(s, \ldots, t, n)$ into *Frontier*

return null

Generic Search Algorithm

Depth First

Procedure *Search*(G, S, *goal*)

Inputs: G: graph with nodes N and arcs A

S: set of start nodes

goal: Boolean function of nodes

Output: path from s to g s.t. $s \in S$ and *goal*(g) or null if no solution paths are found

This is a stack last in first out

Frontier $\leftarrow \{(s) \mid s \in S\}$ // a set of paths

while *Frontier* is not empty

~~remove~~ pop a path $p = (s, \ldots, t)$ from *Frontier*

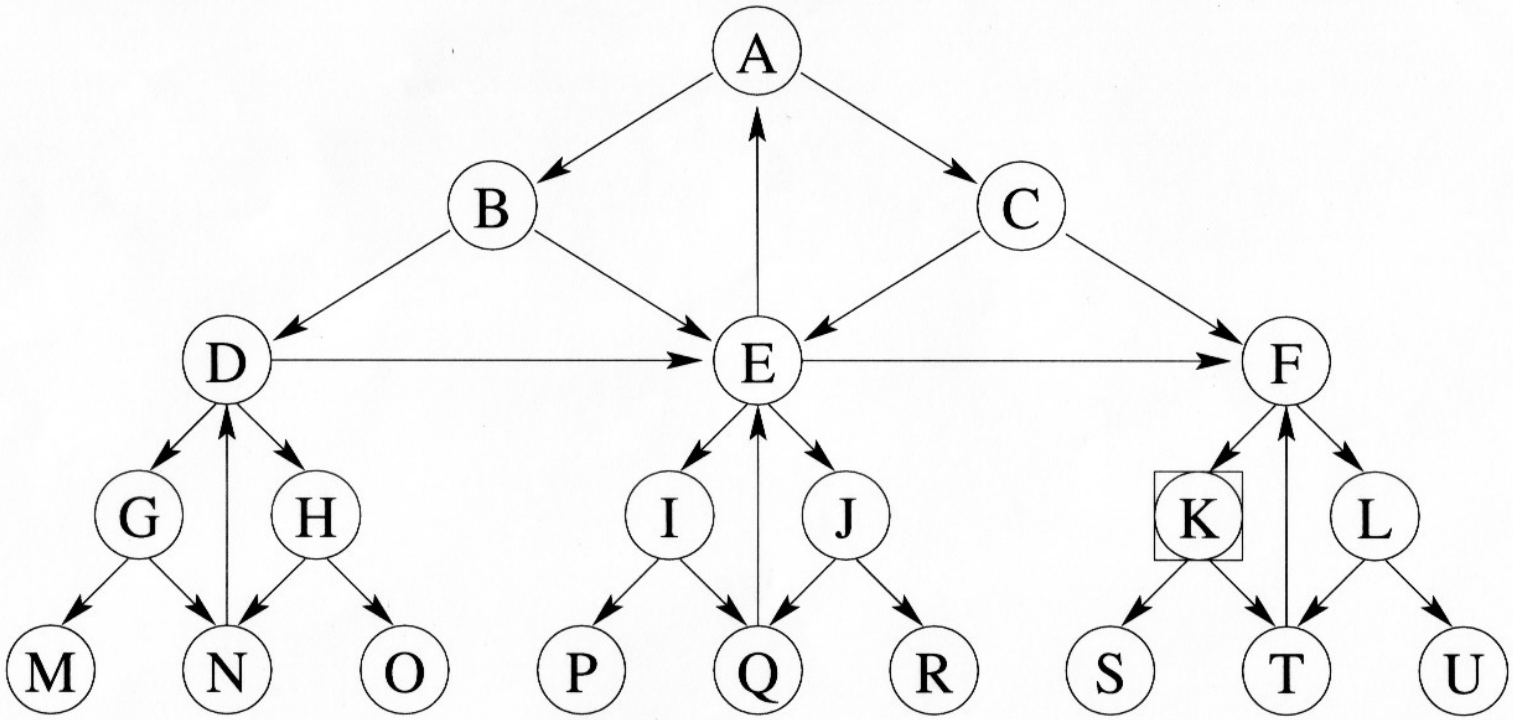
if *goal*(t) then return p

for each arc from t to n

~~insert~~ push $(s, \ldots, t, n)$ into *Frontier* if path has no cycles

return null

Uninformed Example 2



DFS with cycle-checking: A, B, D, G, M, N, H, N, O, E, I, P, Q, J, Q, R, F, K
end with (A, B, D, E, F, K) path