

# Online Methods

# Last Time

- Value Iteration
- Policy Iteration

# Guiding Questions

- What are the differences between *online* and *offline* solutions?
- Are there solution techniques that require computation time *independent* of the state space size?

# Why Do We Need Something Else?

- Problems Policy and Value Iteration may struggle with?
  - Path planning across the country, or interplanetary
  - More realistic car dynamics (continuous states)
- Why are these problems hard?
  - State Space is massive (or infinite)

# Curse of Dimensionality

1 dimension, 5 segments

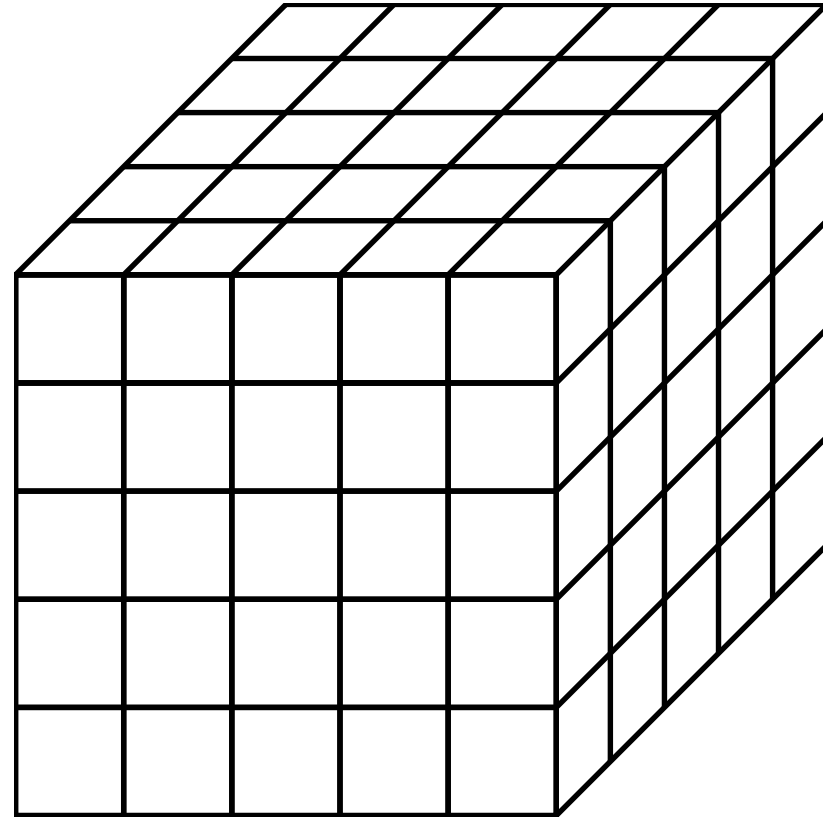
$$|\mathcal{S}| = 5$$

2 dimensions, 5 segments

$$|\mathcal{S}| = 25$$

3 dimensions, 5 segments

$$|\mathcal{S}| = 125$$

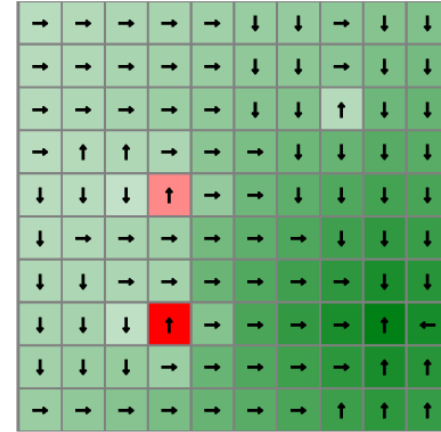


$$n \text{ dimensions, } k \text{ segments} \rightarrow |\mathcal{S}| = k^n$$

# Offline vs Online Solutions

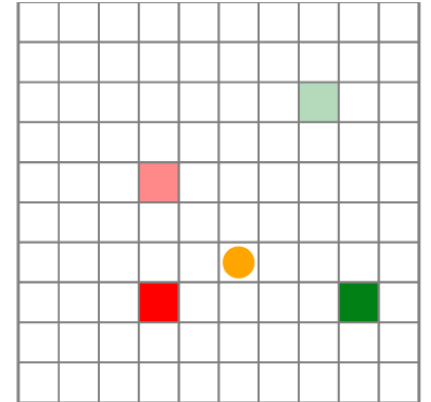
## Offline

- Before Execution: find  $V^*/Q^*$
- During Execution:  $\pi^*(s) = \operatorname{argmax} Q^*(s, a)$



## Online

- Before Execution: <nothing>
- During Execution: Consider actions and their consequences (everything)



- Why?
- Online methods are insensitive to the size of  $S$  !

# One Step Lookahead

```
randstep( $\mathcal{P}::\text{MDP}$ , s, a) =  $\mathcal{P}.\text{TR}(s, a)$ 
```

```
function rollout( $\mathcal{P}$ , s,  $\pi$ , d)
    ret = 0.0
    for t in 1:d
        a =  $\pi(s)$ 
        s, r = randstep( $\mathcal{P}$ , s, a)
        ret +=  $\mathcal{P}.\gamma^{(t-1)} * r$ 
    end
    return ret
end
```

```
function ( $\pi::\text{RolloutLookahead}$ )(s)
    U(s) = rollout( $\pi.\mathcal{P}$ , s,  $\pi.\pi$ ,  $\pi.d$ )
    return greedy( $\pi.\mathcal{P}$ , U, s).a
end
```

```
function greedy( $\mathcal{P}::\text{MDP}$ , U, s)
    u, a = findmax(a → lookahead( $\mathcal{P}$ , U, s, a),  $\mathcal{P}.\mathcal{A}$ )
    return (a=a, u=u)
end
```

```
function lookahead( $\mathcal{P}::\text{MDP}$ , U, s, a)
    S, T, R,  $\gamma$  =  $\mathcal{P}.S$ ,  $\mathcal{P}.T$ ,  $\mathcal{P}.R$ ,  $\mathcal{P}.\gamma$ 
    return  $R(s, a) + \gamma * \text{sum}(T(s, a, s') * U(s') \text{ for } s' \text{ in } S)$ 
end
```

# Forward Search

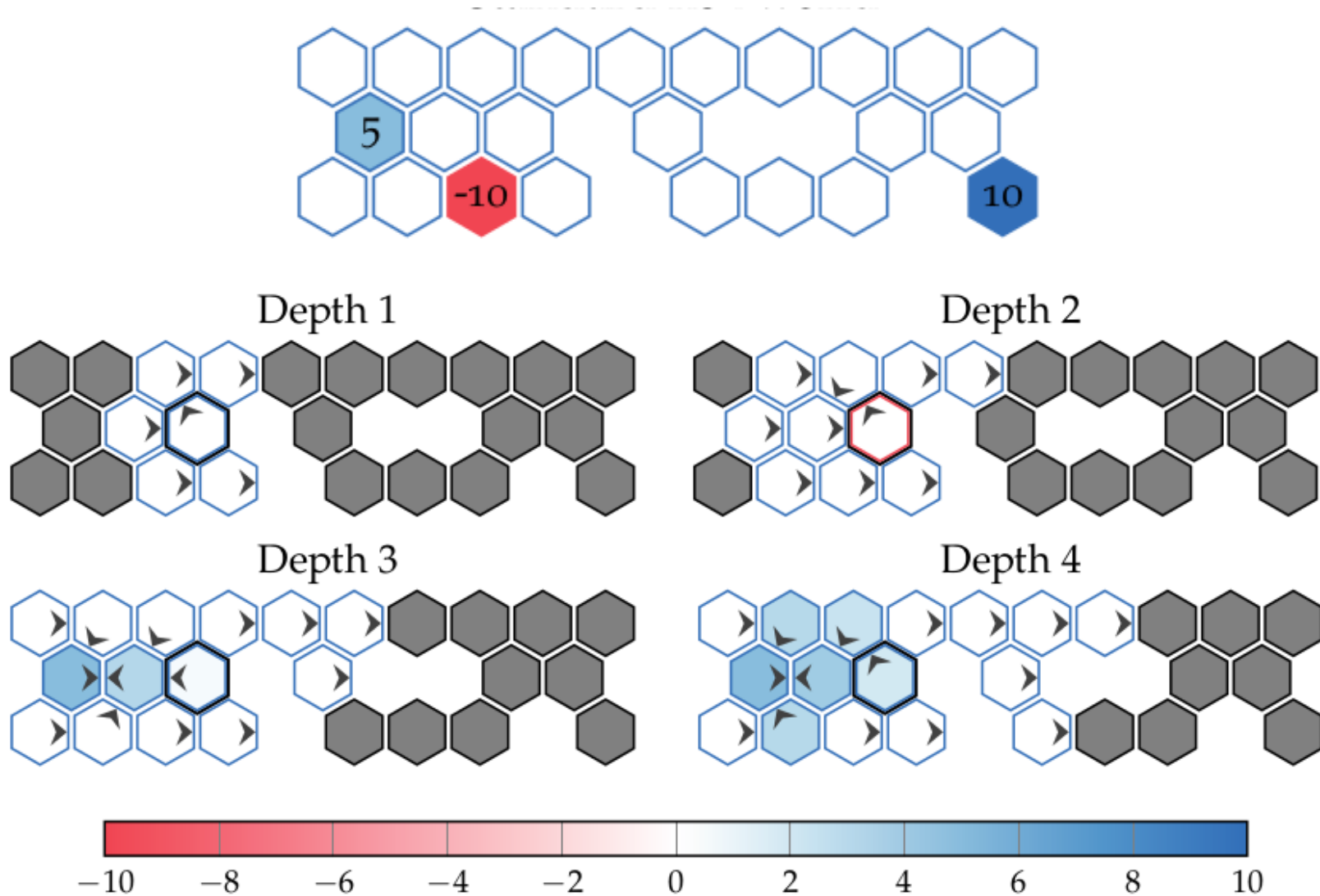
```
function forward_search( $\mathcal{P}$ , s, d, U)
    if d ≤ 0
        return (a=nothing, u=U(s))
    end
    best = (a=nothing, u=-Inf)
    U'(s) = forward_search( $\mathcal{P}$ , s, d-1, U).u
    for a in  $\mathcal{P}.A$ 
        u = lookahead( $\mathcal{P}$ , U', s, a)
        if u > best.u
            best = (a=a, u=u)
        end
    end
    return best
end

function lookahead( $\mathcal{P}::\text{MDP}$ , U, s, a)
    S, T, R,  $\gamma$  =  $\mathcal{P}.S$ ,  $\mathcal{P}.T$ ,  $\mathcal{P}.R$ ,  $\mathcal{P}.\gamma$ 
    return R(s,a) +  $\gamma*\text{sum}(T(s,a,s')*U(s'))$  for s' in S
end
```

$$O(|S| \times |A|^d)$$



# Forward Search depth



# Sparse Sampling

```
function sparse_sampling( $\mathcal{P}$ , s, d, m, U)
    if d ≤ 0
        return (a=nothing, u=U(s))
    end
    best = (a=nothing, u=-Inf)
    for a in  $\mathcal{P}.A$ 
        u = 0.0
        for i in 1:m
            s', r = randstep( $\mathcal{P}$ , s, a)
            a', u' = sparse_sampling( $\mathcal{P}$ , s', d-1, m, U)
            u += (r +  $\mathcal{P}.\gamma * u'$ ) / m
        end
        if u > best.u
            best = (a=a, u=u)
        end
    end
    return best
end
```

$O((m|A|)^d)$        $|V^{\text{SS}}(s) - V^*(s)| \leq \epsilon$        $m, \epsilon$ , and  $d$  related, but independent of  $|S|$

# Break

Draw the trees produced by the following algorithms for a problem with 2 actions and 3 states:

1. One-step lookahead with rollout
2. Forward search ( $d=2$ )
3. Sparse sampling ( $d=2, m=2$ )

# Forward Search Sparse Sampling

(FSSS)

Paper: <https://cdn.aaai.org/ojs/7689/7689-13-11219-1-2-20201228.pdf>

- Sparse Sampling, but only look at potentially valuable states

Things it keeps track of:

$Q(s, a)$ : Estimate of the value for the  
state action pair

$U(s)$ : Upper bound for value of state  $s$

$L(s)$ : Lower bound for value of state  $s$

$U(s, a)$ : Upper bound for value of state-  
action

$L(s, a)$ : Lower bound for value of state-  
action

# Forward Search Sparse Sampling

---

## Algorithm 3 FSSS( $s, d$ )

---

**if**  $d = 1$  (leaf) **then**

$$L^d(s, a) = U^d(s, a) = R(s, a), \forall a$$

$$L^d(s) = U^d(s) = \max_a R(s, a)$$

**else if**  $n_{sd} = 0$  **then**

**for each**  $a \in A$  **do**

$$L^d(s, a) = V_{\min}$$

$$U^d(s, a) = V_{\max}$$

**for**  $C$  times **do**

$$s' \sim T(s, a, \cdot)$$

$$L^{d-1}(s') = V_{\min}$$

$$U^{d-1}(s') = V_{\max}$$

$$K^d(s, a) = K^d(s, a) \cup \{s'\}$$

$$a^* = \operatorname{argmax}_a U^d(s, a)$$

$$s^* = \max_{s' \in K^d(s, a^*)} (U^{d-1}(s') - L^{d-1}(s'))$$

$$\text{FSSS}(s^*, d-1)$$

$$n_{sd} = n_{sd} + 1$$

$$L^d(s, a^*) = R(s, a^*) + \gamma \sum_{s' \in K^d(s, a^*)} L^{d-1}(s') / C$$

$$U^d(s, a^*) = R(s, a^*) + \gamma \sum_{s' \in K^d(s, a^*)} U^{d-1}(s') / C$$

$$L^d(s) = \max_a L^d(s, a)$$

$$U^d(s) = \max_a U^d(s, a)$$


---

If  $L(s, a^*) \geq \max_{a \neq a^*} U(s, a)$  for best action ( $a^* = \arg \max_a U(s, a)$ ):  
then, the node is closed because the best action is found.

# Monte Carlo Tree Search (MCTS/UCT)

- FSSS, but with less to keep track of

Keep track of:

$Q(s, a)$ : Value estimate of that  
state and action combo

$N(s, a)$ : Number of times we  
visit a state and action combo

$$Q(s, a) + c \sqrt{\frac{\log N(s)}{N(s, a)}} \quad Q(s, a) + c \frac{N(s)^\beta}{\sqrt{N(s, a)}}$$

low  $N(s, a)/N(s)$  = high bonus  
start with  $c = 2(\bar{V} - \underline{V})$ ,  $\beta = 1/4$

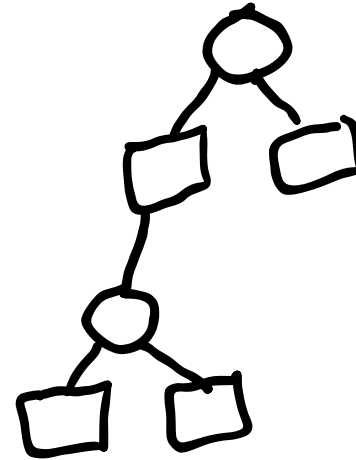
Full story can be found in  
<https://arxiv.org/pdf/1902.05213.pdf>

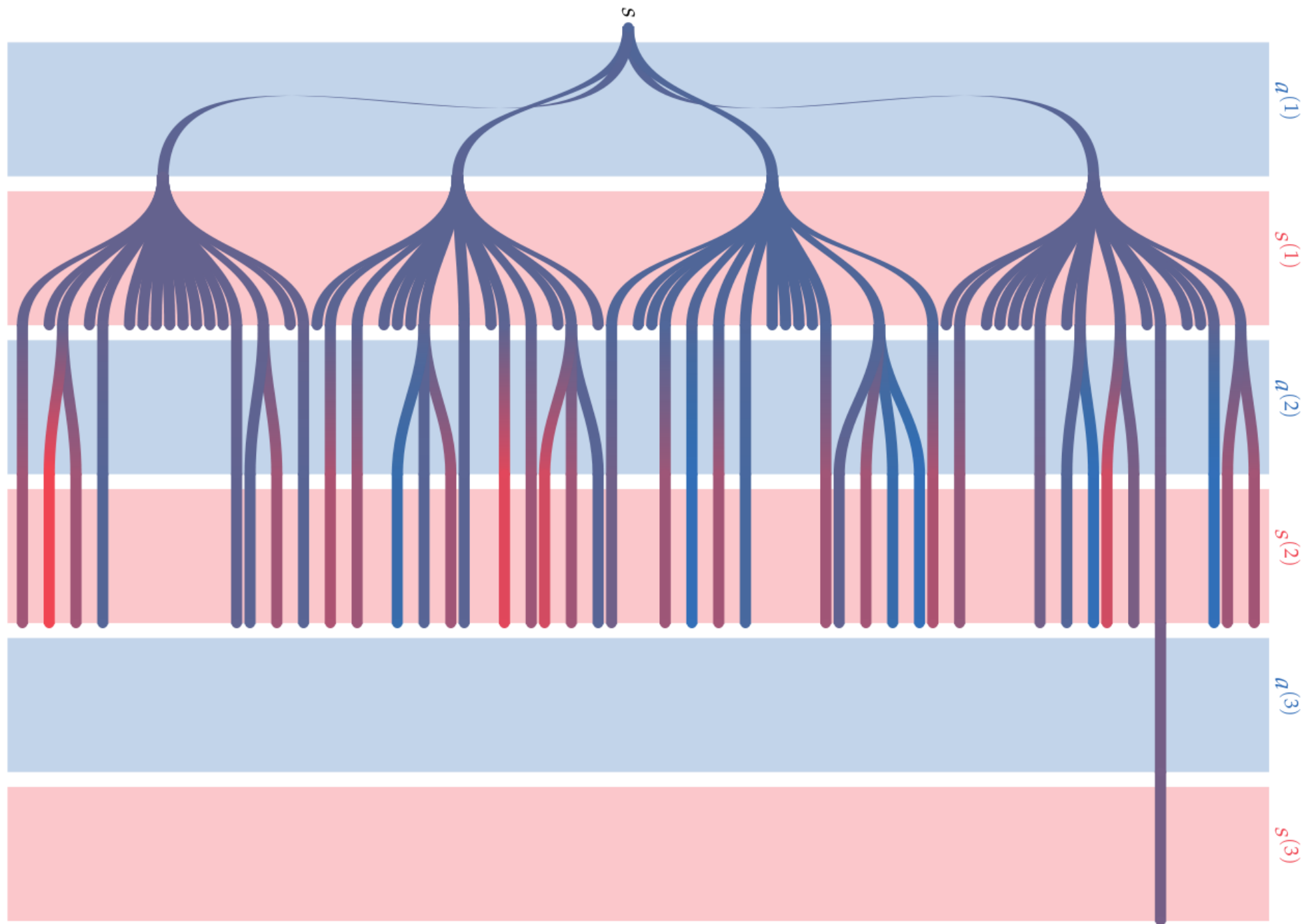
# Monte Carlo Tree Search (MCTS/UCT)

- FSSS, but with less to keep track of

```
function ( $\pi$ ::MonteCarloTreeSearch)(s)
  for k in 1: $\pi$ .m
    simulate!( $\pi$ , s)
  end
  return argmax( $a \rightarrow \pi.Q[(s,a)]$ ,  $\pi.P.A$ )
end
```

```
function simulate!( $\pi$ ::MonteCarloTreeSearch, s, d= $\pi.d$ )
  if d  $\leq$  0
    return  $\pi.U(s)$ 
  end
   $P, N, Q, c = \pi.P, \pi.N, \pi.Q, \pi.c$ 
   $A, TR, \gamma = P.A, P.TR, P.\gamma$ 
  if !haskey(N, (s, first(A)))
    for a in A
       $N[(s,a)] = 0$ 
       $Q[(s,a)] = 0.0$ 
    end
    return  $\pi.U(s)$ 
  end
  a = explore( $\pi$ , s)
   $s', r = TR(s,a)$ 
  q = r +  $\gamma * \text{simulate}!(\pi, s', d-1)$ 
   $N[(s,a)] += 1$ 
   $Q[(s,a)] += (q - Q[(s,a)]) / N[(s,a)]$ 
  return q
end
```







# Guiding Questions

- What are the differences between online and offline solutions?
- Are there solution techniques that are *independent* of the state space size?