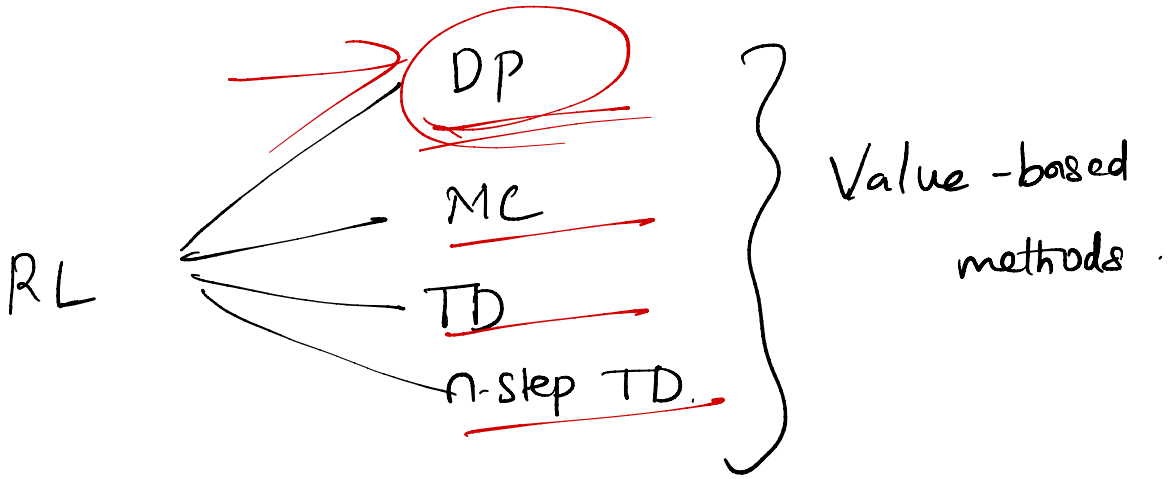
Lecture - 09



Model-free algorithms.

- REINFORCE
- " with baseline
- Actor Critic

Policy gradient methods.

↑ Policy + value.

Model-based RL

Model of the world → dynamics fn.

given $S_t, A_t \rightarrow S_{t+1}, R_{t+1}$

→ $P(S_{t+1}, R_{t+1} \mid S_t, A_t)$

RL
- model based — ex: DP — [Planning]
- model-free — ex: MC, TD. — [learning]

Model : $P(S_{t+1}, R_{t+1} | S_t, R_t)$

Stochastic models.
- Distribution model
  [Complete distrib. of S_{t+1}, R_{t+1}]
- Sample model.
  [produce just one of the possibilities sampled according to the prob distrib]

→ ex: modelling the sum of a dozen dice.

State/action
- Sample model → Produce transitions.
- distrb. → generate all possible transitions

Policy + model
$S_0, A_0, R_1, S_1, A_1, \ldots$

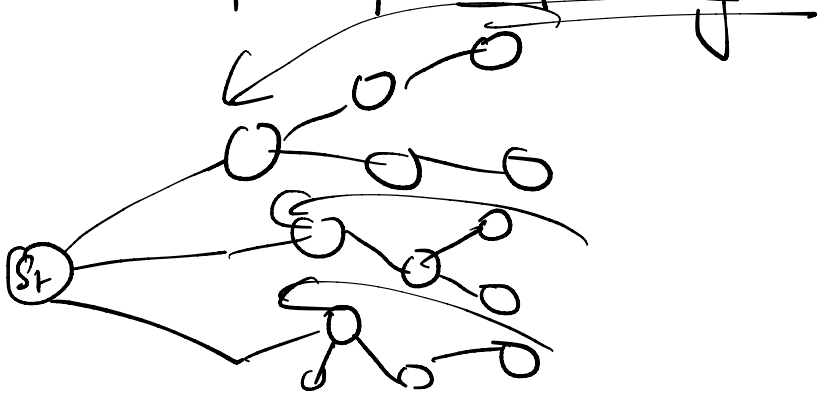
Start state
Policy
- Sample → Can generate one episode.
- distrb. → Can generate all possible episodes

How to use the model?

—Simulate the env.

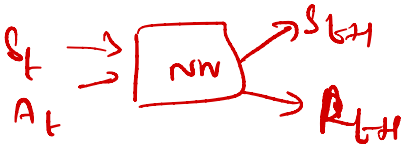
model —planning→ Policy.

Planning
- State-space planning (RL)
- Plan-space planning.



Learning the model :-

- tabular model.
- Linear model.
- Neural networks.

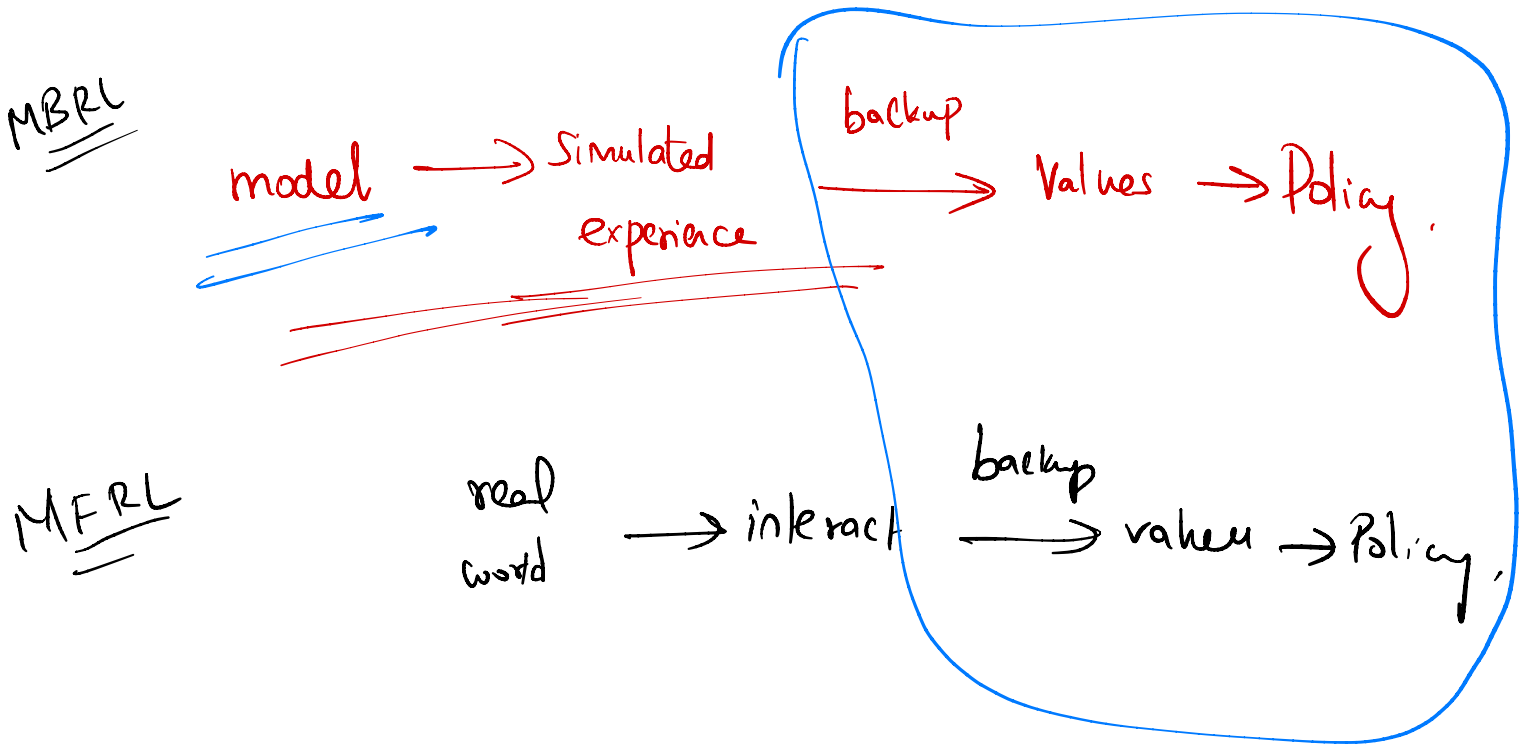


Planning and Learning :

→ model → learn better value fn, Policy

→ model → to improve the policy during decision time.

(Decision time planning)


MBRL
model
Simulated
experience
backup
Values
Policy
MFRL
real
world
interact
backup
values
Policy

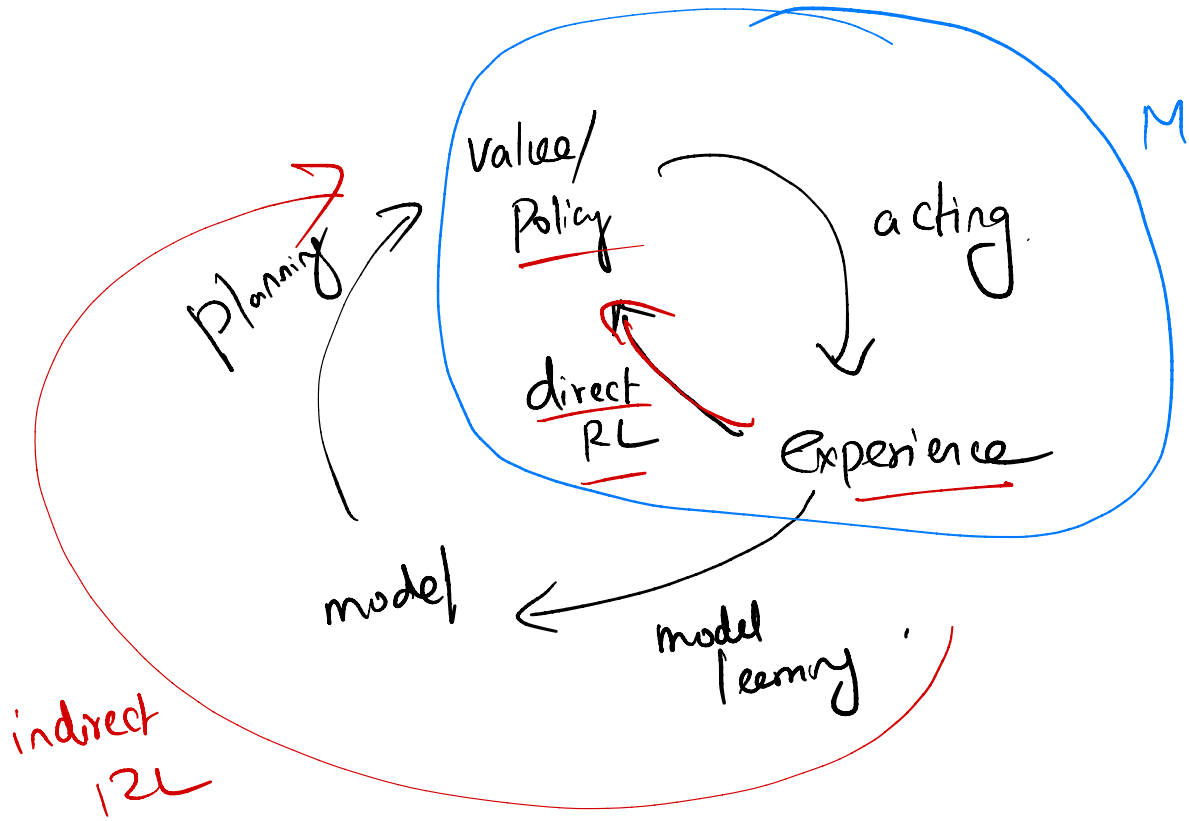
Random-sample one-step tabular Q-planning

Loop forever:

1. Select a state, $S \in \mathcal{S}$, and an action, $A \in \mathcal{A}(S)$, at random
2. Send S, A to a sample model, and obtain
   a sample next reward, R, and a sample next state, S'
3. Apply one-step tabular Q-learning to S, A, R, S':
   $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

Dyna : Integrated planning, acting, and learning

Dyna-Q : simple framework for online planning agent.

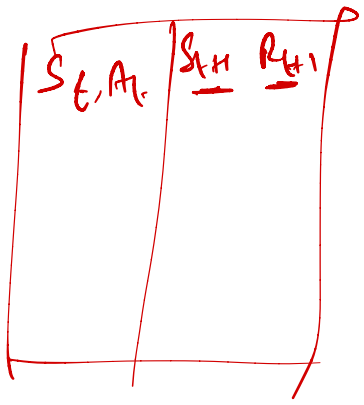


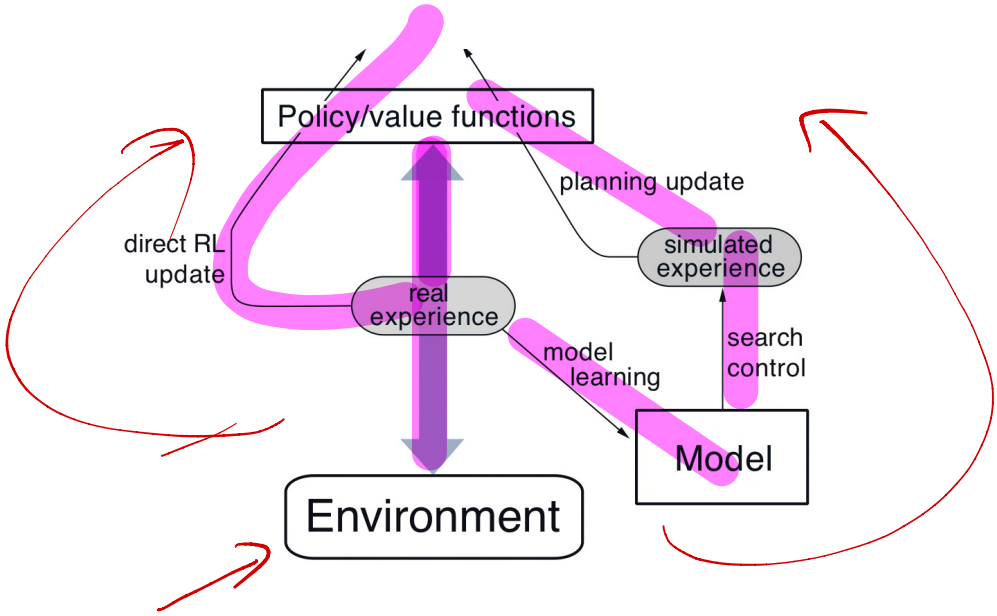
Dyna-Q :

planning — random Sample one-step tabular Q-Planning.

direct RL - one-step tabular Q-learning.

model-learning - tabular model.




Policy/value functions
planning update
direct RL
update
simulated
experience
real
experience
search
control
model
learning
Model
Environment

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Loop forever:

(a) $S \leftarrow$ current (nonterminal) state

(b) $A \leftarrow \varepsilon$-greedy(S,Q)

(c) Take action A; observe resultant reward, R, and state, S'

(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)

(f) Loop repeat n times:

$S \leftarrow$ random previously observed state

$A \leftarrow$ random action previously taken in S

$R, S' \leftarrow Model(S,A)$

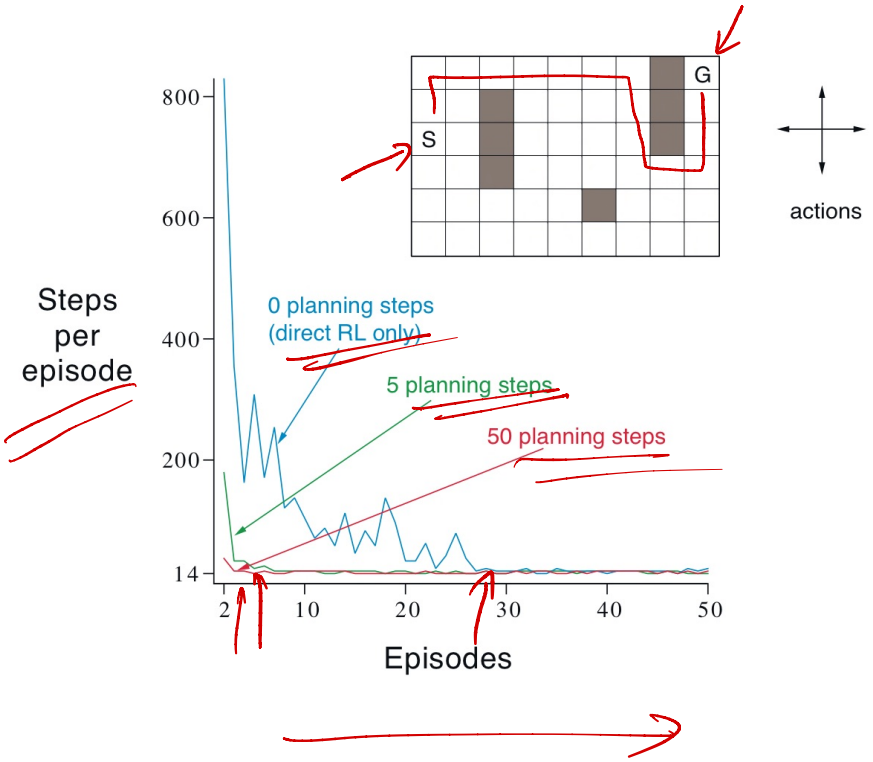
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

direct RL update

model learning

update for NN model.

indirect RL update



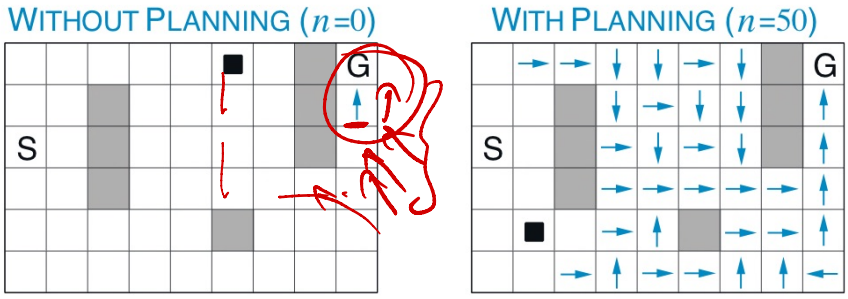
reward

↓

0 everywhere

goal: +1

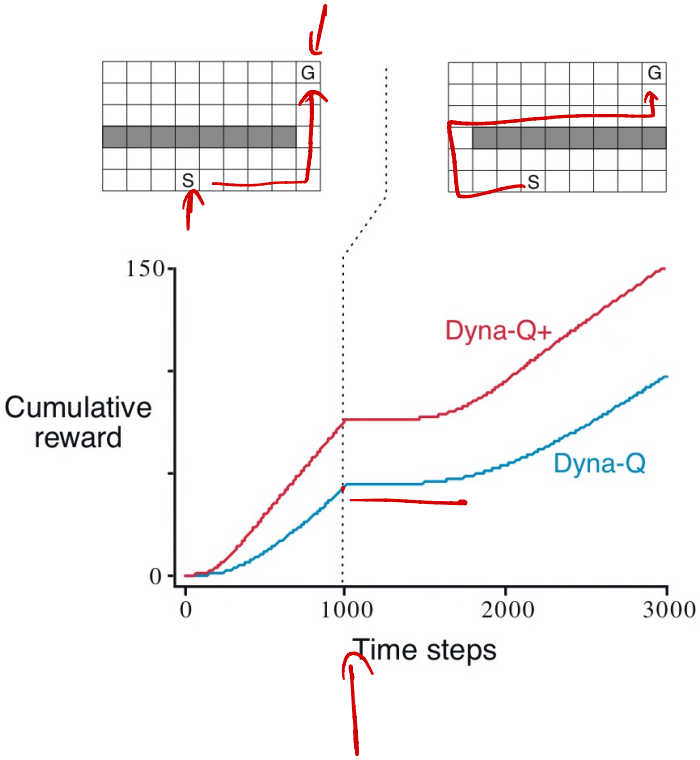
$\gamma = 0.95$

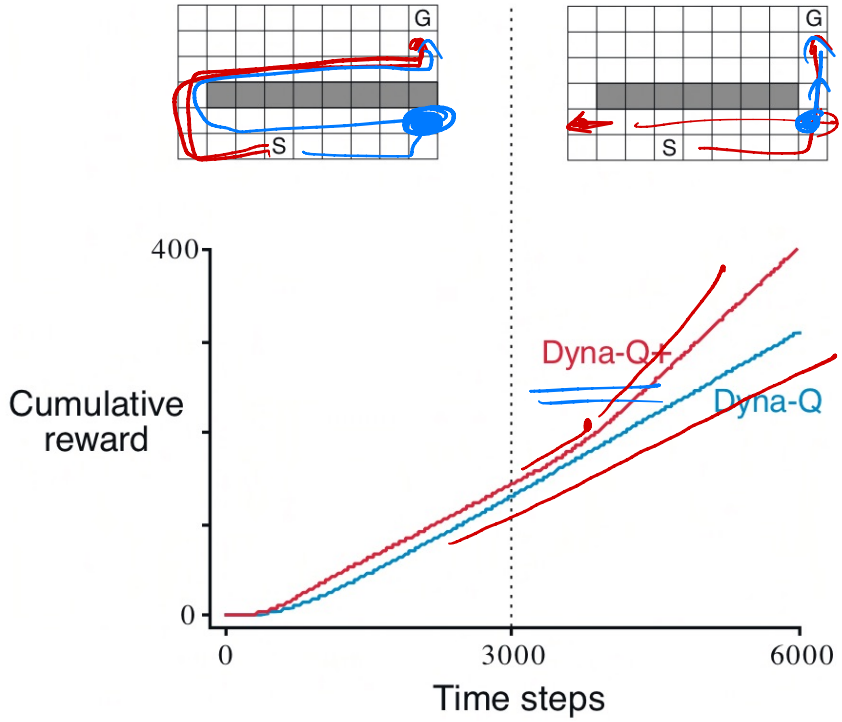


What if the model is wrong?

(1) world is Stochastic.

(2) Fnctun approximatn


G
S
G
S
150
0
Cumulative reward
Dyna-Q+
Dyna-Q
0
1000
2000
3000
Time steps



Exploration vs. Exploitation

time lapse.

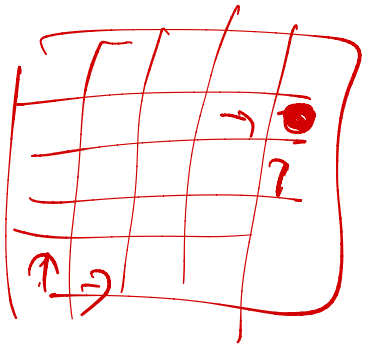
S_t, A_t	S_{t+1}, R_{t+1}	τ
⋮	⋮	⋮

$r + \kappa\sqrt{\tau}$

Prioritized Sweeping:

backward focusing

Priority queue



Prioritized sweeping for a deterministic environment

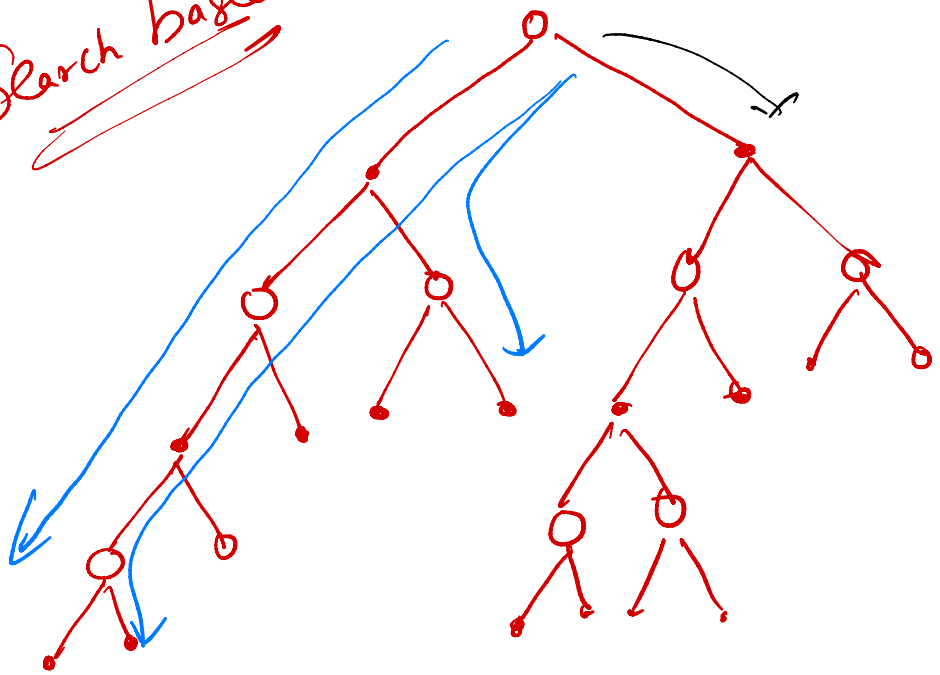
Initialize $Q(s,a)$, $Model(s,a)$, for all s,a, and $PQueue$ to empty
Loop forever:

(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow policy(S,Q)$
(c) Take action A; observe resultant reward, R, and state, S'
(d) $Model(S,A) \leftarrow R, S'$
(e) $P \leftarrow |R + \gamma \max_a Q(S',a) - Q(S,A)|$.
(f) if $P > \theta$, then insert S,A into $PQueue$ with priority P
(g) Loop repeat n times, while $PQueue$ is not empty:
    $S,A \leftarrow first(PQueue)$
    $R,S' \leftarrow Model(S,A)$
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    Loop for all $\bar{S},\bar{A}$ predicted to lead to S:
        $\bar{R} \leftarrow$ predicted reward for $\bar{S},\bar{A},S$
        $P \leftarrow |\bar{R} + \gamma \max_a Q(S,a) - Q(\bar{S},\bar{A})|$.
        if $P > \theta$ then insert $\bar{S},\bar{A}$ into $PQueue$ with priority P

Planning at decision time :-

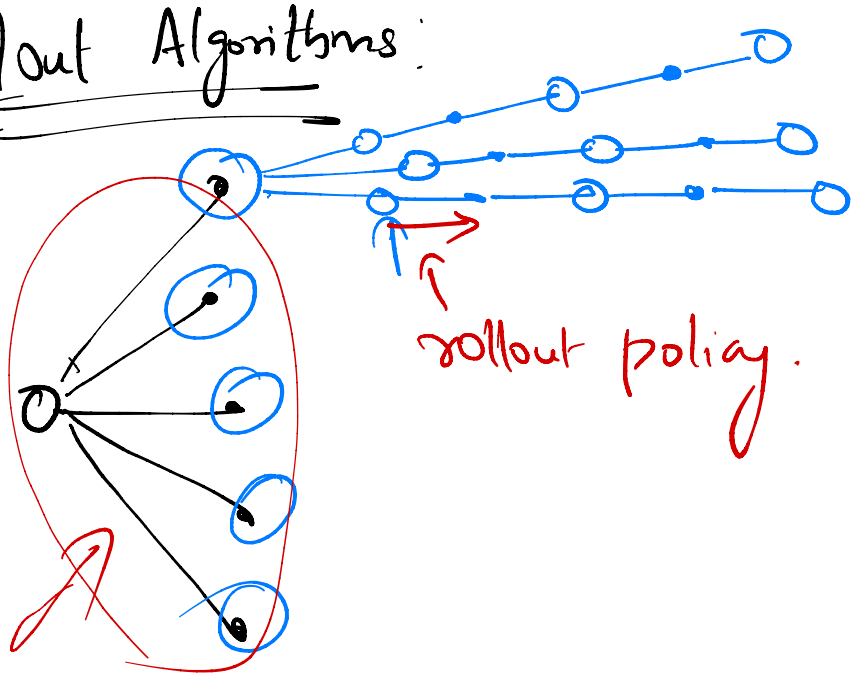
- background planning
- decision-time planning
  - discrete actions → search over a tree
  - Cont. actions

Search based



- Policy
- model

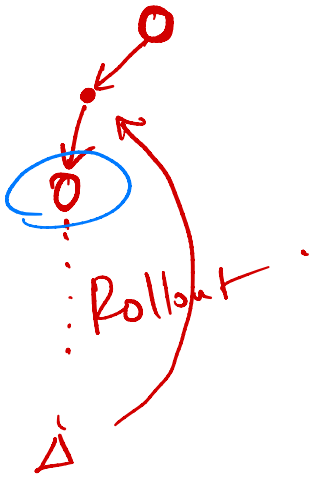
- Rollout Algorithms :

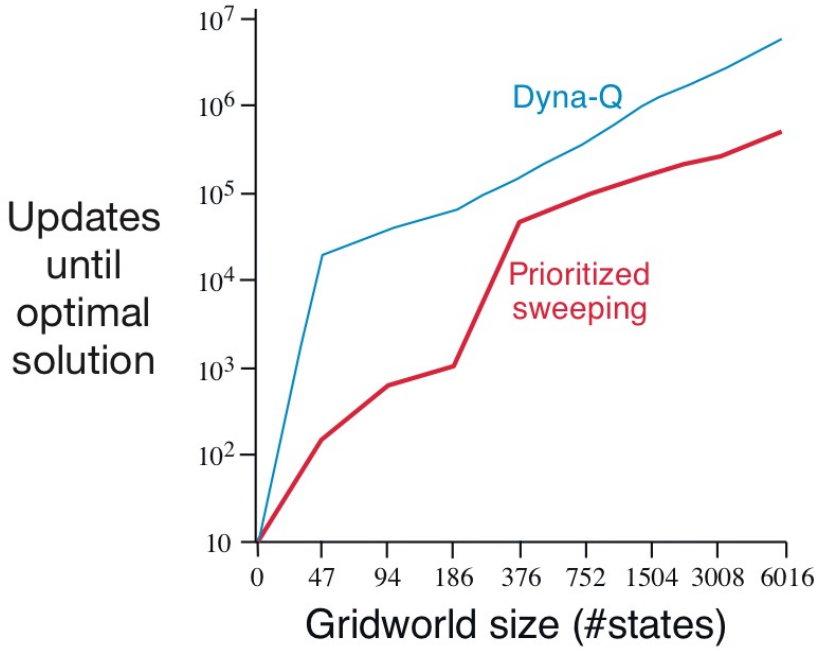


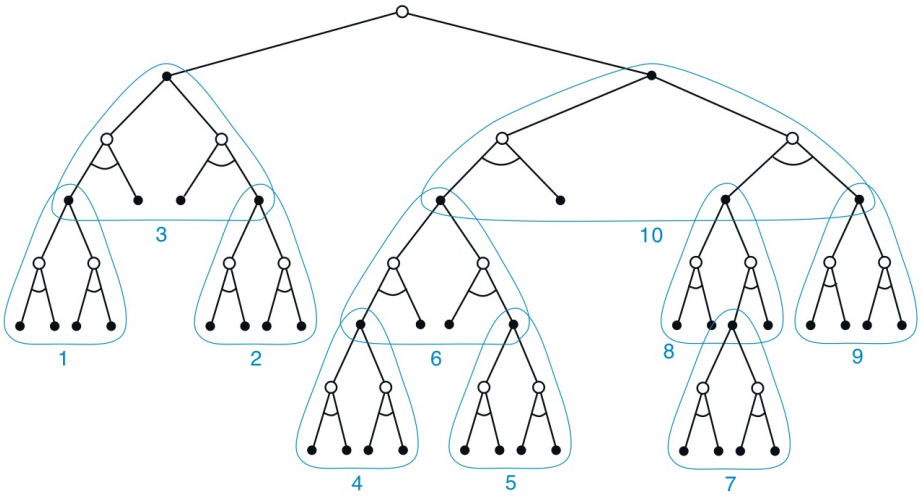
Monte-Carlo Tree Search (MCTS)

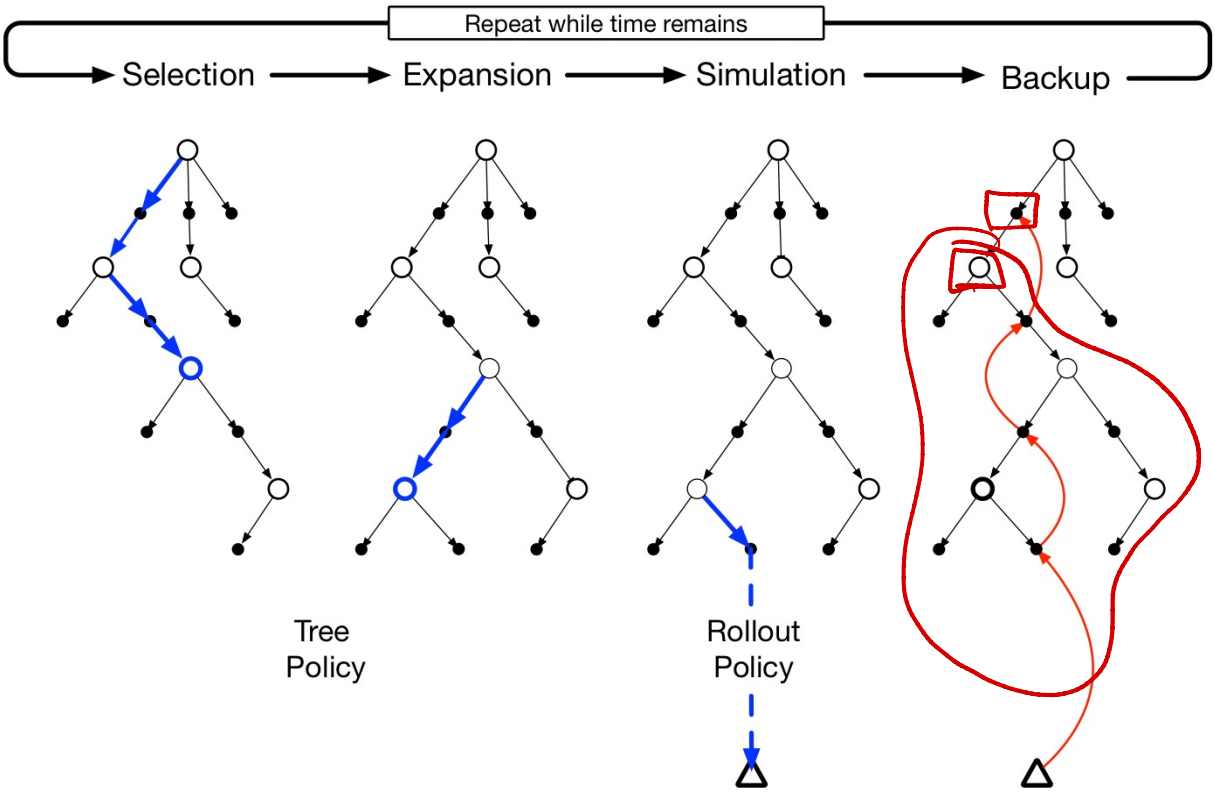
- Tree Policy
- Rollout policy

Selection, expansion, Simulation, backup.




Updates until optimal solution
10^7
10^6
10^5
10^4
10^3
10^2
10
Dyna-Q
Prioritized sweeping
0
47
94
186
376
752
1504
3008
6016
Gridworld size (#states)


1
2
3
4
5
6
7
8
9
10



RL → model free / model based

RL → MC / TD.

RL → value-based / Policy based.