Google Developers

Experts

표에 기반한 방법을 이용한 계획 및 학습

강화학습

송호연 @chris_loves_ai
sjhshy@gmail.com

모델 기반 & 모델 없는
model-based & model-free

모델 기반 & 모델이 없는 강화학습

model-based & model-free RL

- **모델 기반(model-based)** 강화학습: **계획(planning)**이 중요
- **모델 없는(model-free)** 강화학습: **학습(learning)**이 중요

모델

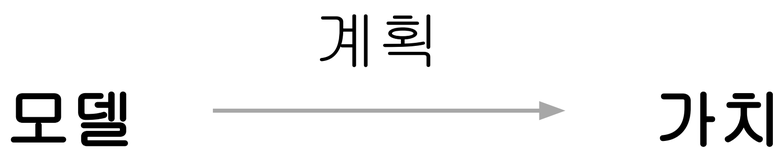
model

- **모델(model)**이란 환경이 행동에 어떻게 반응할 것인지를 예측하기 위해 학습자가 사용할 수 있는 모든 것을 의미한다.

계획

planning

- **계획(planning)**은 모델링된 환경과의 상호작용을 위해
  모델을 입력으로 하여 정책을 만들어 내거나 향상시키는 모든 계산 과정을 지칭



분포 모델 & 표본 모델

distributional model & sample model

- **분포 모델(distributional model):**
  모든 가능성을 제공하고 각 가능성에 해당하는 확률을 제공하는 모델
  예) 주사위 12개 던져서 나올 합의 확률 분포

- **표본 모델(sample model):**
  모든 가능성 중에서 확률에 따라 추출된 하나의 가능성만을 제공하는 모델
  예) 주사위 12개 던져서 나온 하나의 합계

시뮬레이션
simulation

시뮬레이션

simulation

- **시뮬레이션(simulation)**

- **표본 모델(sample model)**의 경우에는 환경을 시뮬레이션하기 위해 모델을 사용
- **분포 모델(distributional model)**의 경우에는 시뮬레이션된 경험을 만들기 위해 모델을 사용

상태 공간 계획 & 계획 공간 계획

state-space planning & plan-space planning

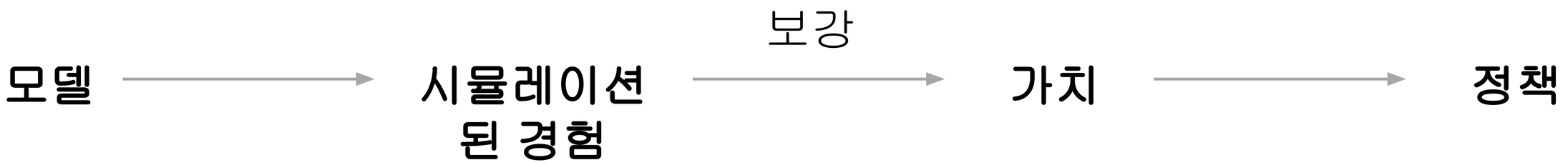
- **상태 공간 계획(state-space planning)**
  최적 정책이나 목표를 향한 최적 경로를 찾기 위해 상태 공간을 탐색하는 것

- **계획 공간 계획(plan-space planning)**
  계획 공간에 대한 탐색을 통해 계획이 수행됨
  학습자는 한 계획에서 다른 계획으로 전이하고,
  가치 함수가 존재한다면 그것은 계획 공간에서 정의됨

시뮬레이션된 경험

simulated experience

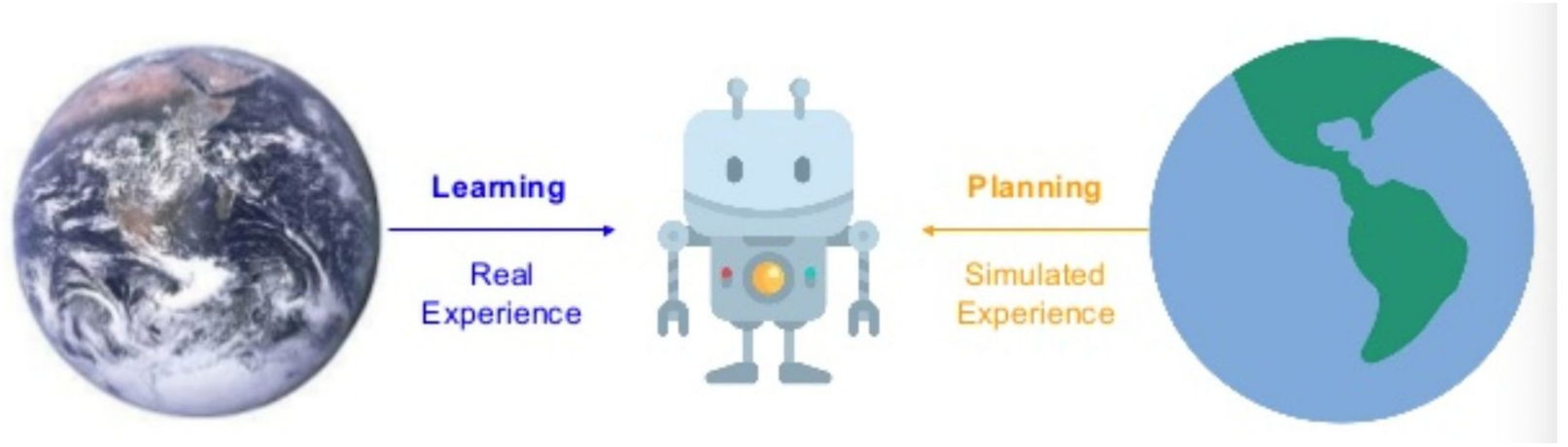
- **모델모든 상태 공간 계획 방법은 하나의 공통된 구조를 갖는다.**
  - 모든 상태 공간 계획 방법은 가치 함수 계산을 정책을 향상시키기 위한 중간 단계로 포함
  - 이 방법은 시뮬레이션된 경험에 적용된 갱신 또는 보강 과정에 의해 가치 함수를 계산



계획과 학습의 관계

Relationship between planning and learning

- 둘 다 가치 함수를 계산하기 위해 사용된다
  - **계획(Planning)**은 모델로부터 시뮬레이션된 경험을 생성해 학습한다
  - **학습(Learning)**은 실제 경험으로부터 학습한다



Experts

그림 출처: https://www.slideshare.net/SeungJaeLee17/reinforcement-learning-an-introduction-chapter-8

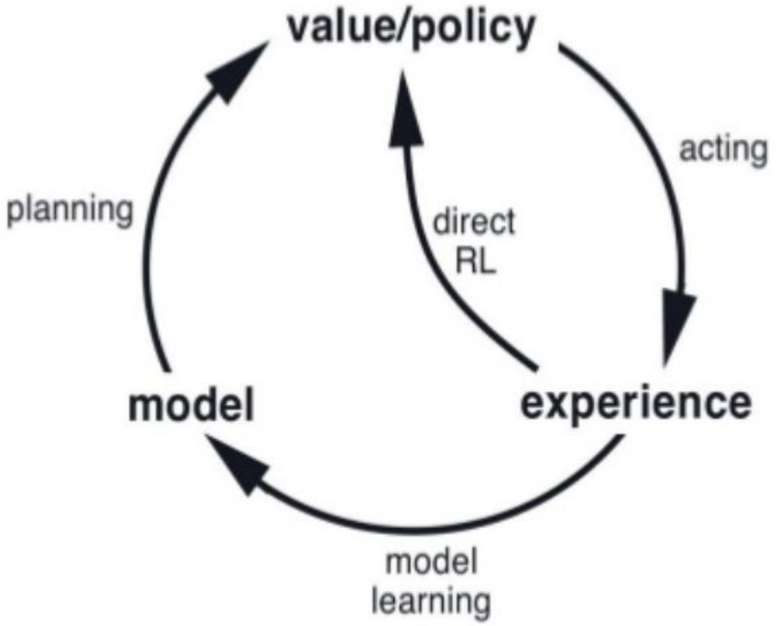
다이나 Q
Dyna-Q

모델 학습 & 강화학습

model-learning & reinforcement-learning

- **모델 학습(model learning):**
  실제 경험을 모델을 향상시키기 위해 사용

- **강화 학습(reinforcement learning):**
  실제 경험을 가치 함수와 정책을 직접 향상시키기 위해 사

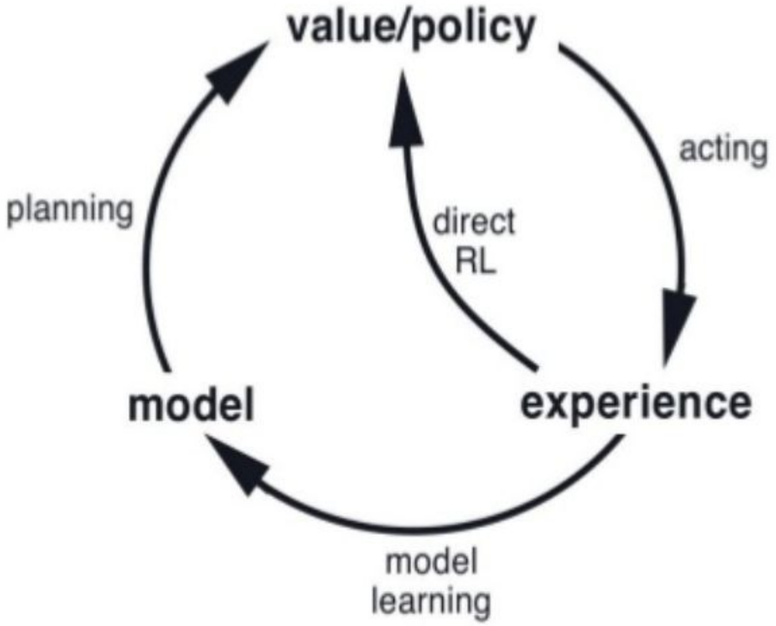


직/간접적 강화학습

direct/indirect reinforcement-learning

- **간접적 강화학습(indirect reinforcement learning):**
  모델에서 시뮬레이션된 경험을 생성하여 가치 함수와 정

- **직접적 강화 학습(direct reinforcement learning):**
  실제 경험을 가치 함수와 정책을 직접 향상시키기 위해 사

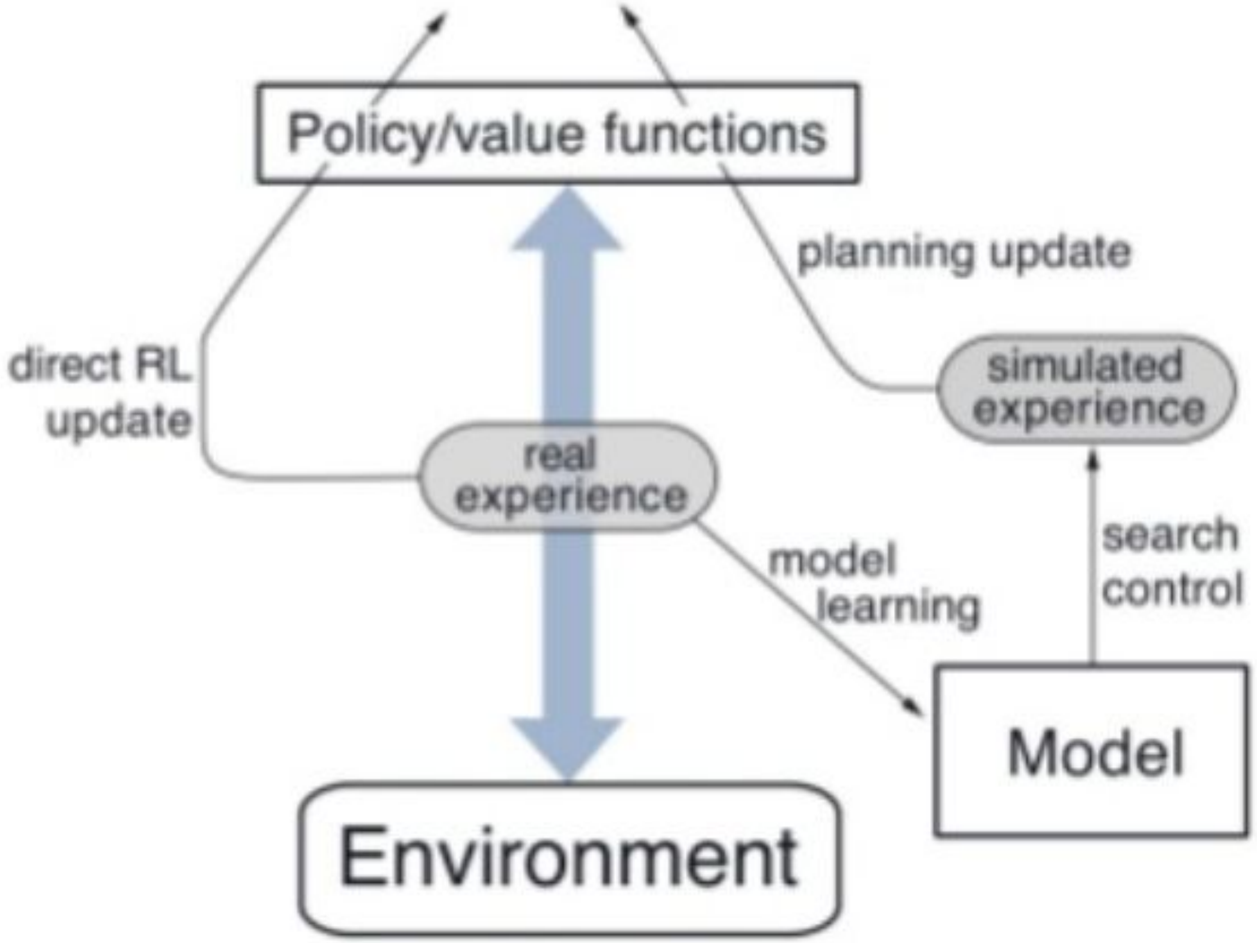


다이나-Q 알고리즘

Dyna-Q Algorithm

- 왼쪽에 있는 화살표는 직접적 강화학습
- 오른쪽은 모델 기반으로 간접적 강화학습

- **탐색 제어(search control):**
  모델로부터 시뮬레이션된 경험을
  생성하기 위해 시작 상태와 행동을
  선택하는 과정



Experts

다이나-Q : 알고리즘

Dyna-Q: Algorithm

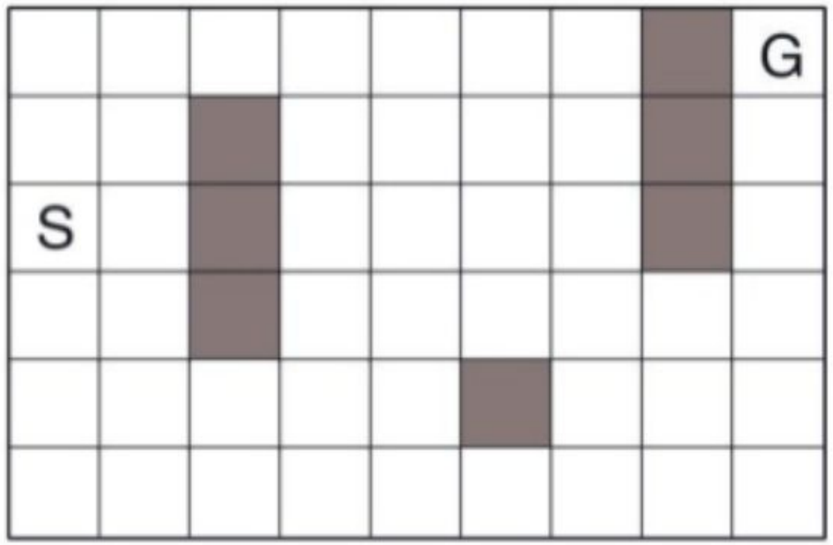
Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \epsilon$-greedy(S,Q)
(c) Execute action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Repeat n times:
$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

행동
직접적 강화학습 Q 러닝

모델 학습
계획

Dyna Maze 예제

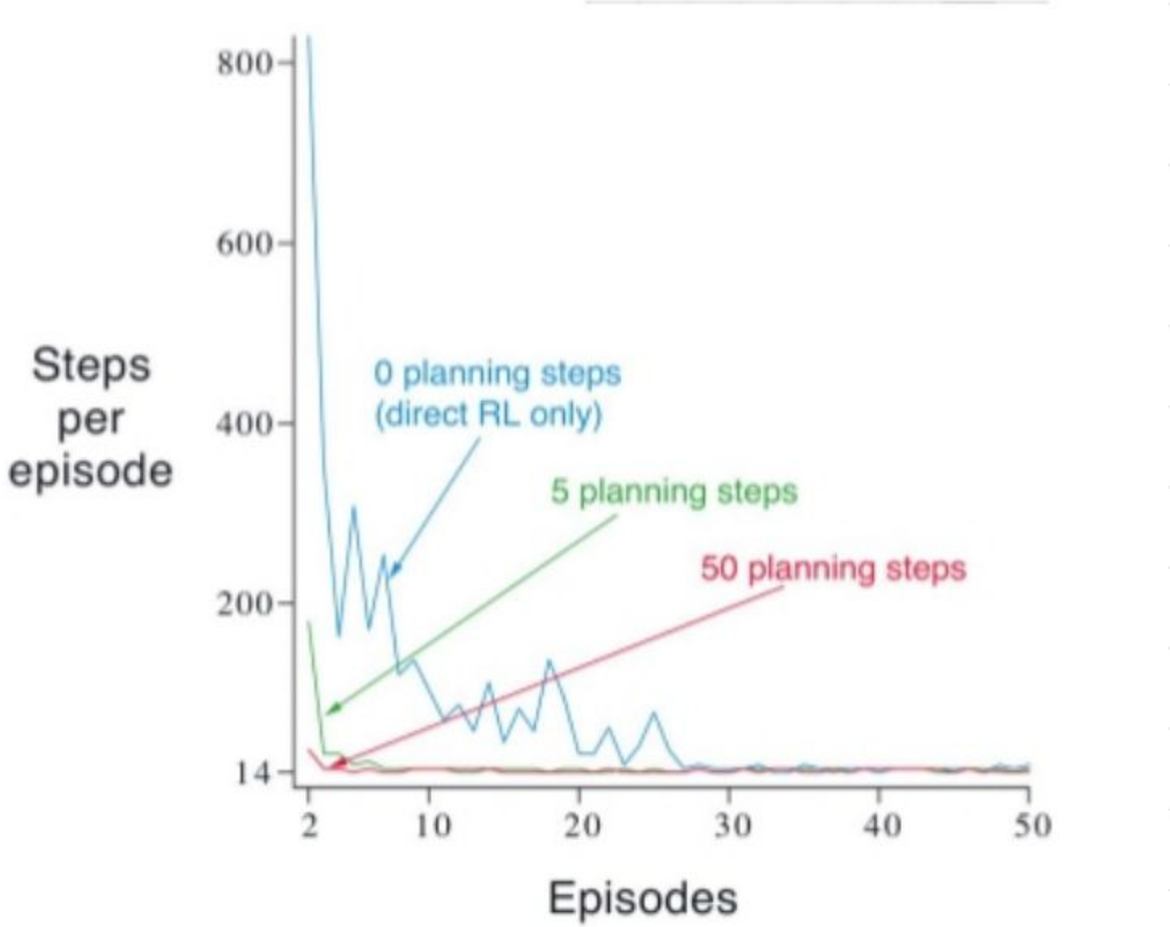
Dyna Maze Example



actions

다이나-Q : Maze 실험 결과

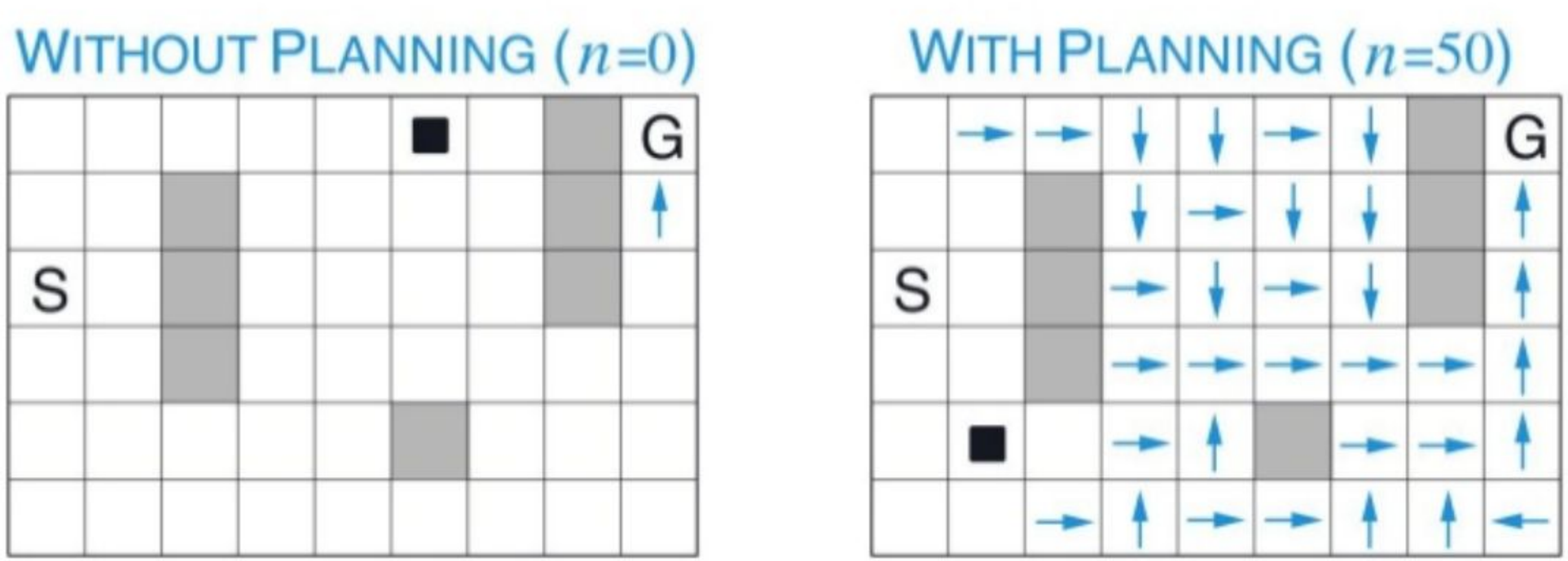
Dyna-Q: Maze Experiment



Experts

다이나 미로: 직관

Dyna Maze: Intuition



Halfway through second episode

Black square ■ : location of the agent

잘못된 모델의 확률

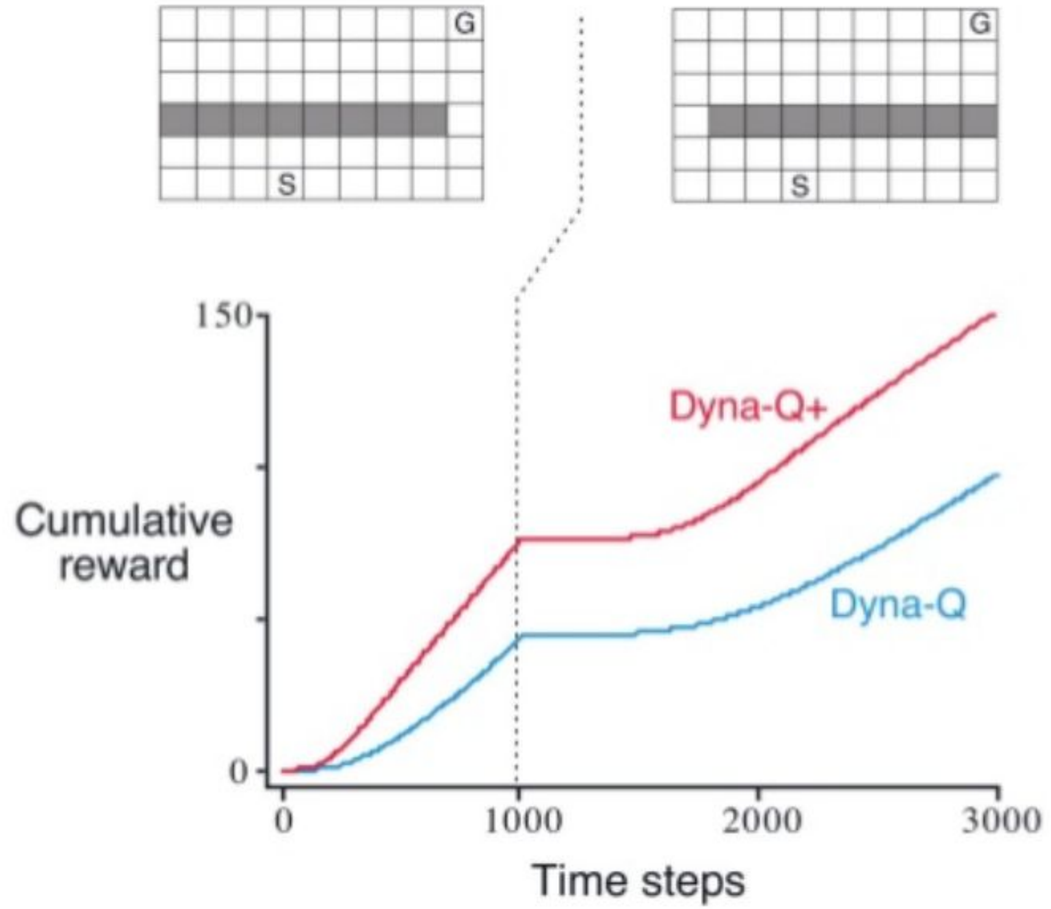
Possibility of a wrong model

- 모델은 다양한 이유로 잘못될 수 있음
  - 확률적 환경 & 제한된 샘플의 수
  - 근사 함수
  - 환경의 변화

- 이로 인해 최적해에 도달하지 못할 수 있음:
  - 준최적 정책(sub-optimal) policy

환경 변화 예시

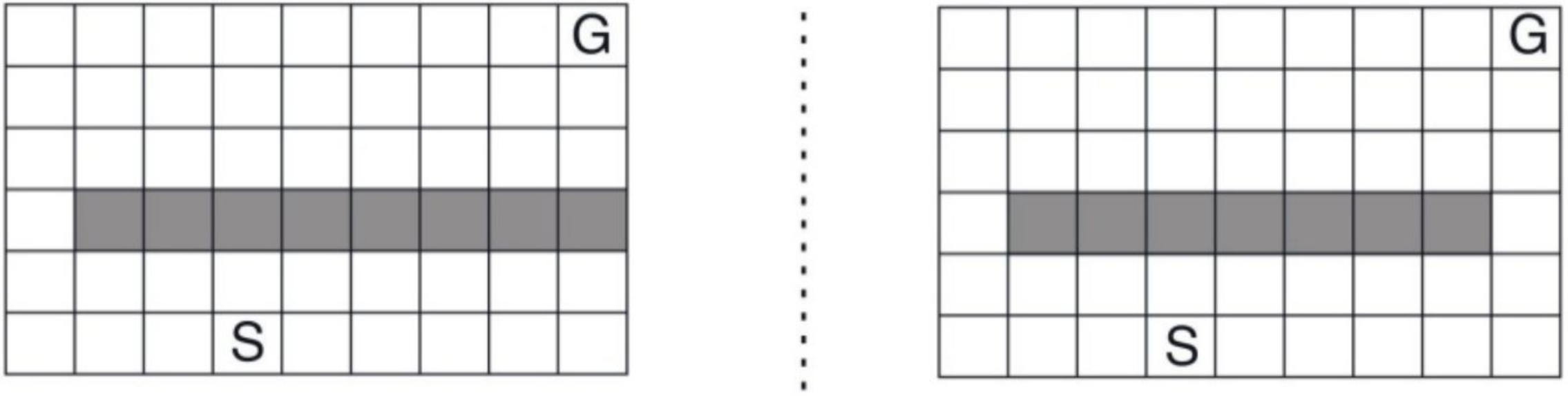
An example of environment change



Experts

환경 변화 예시

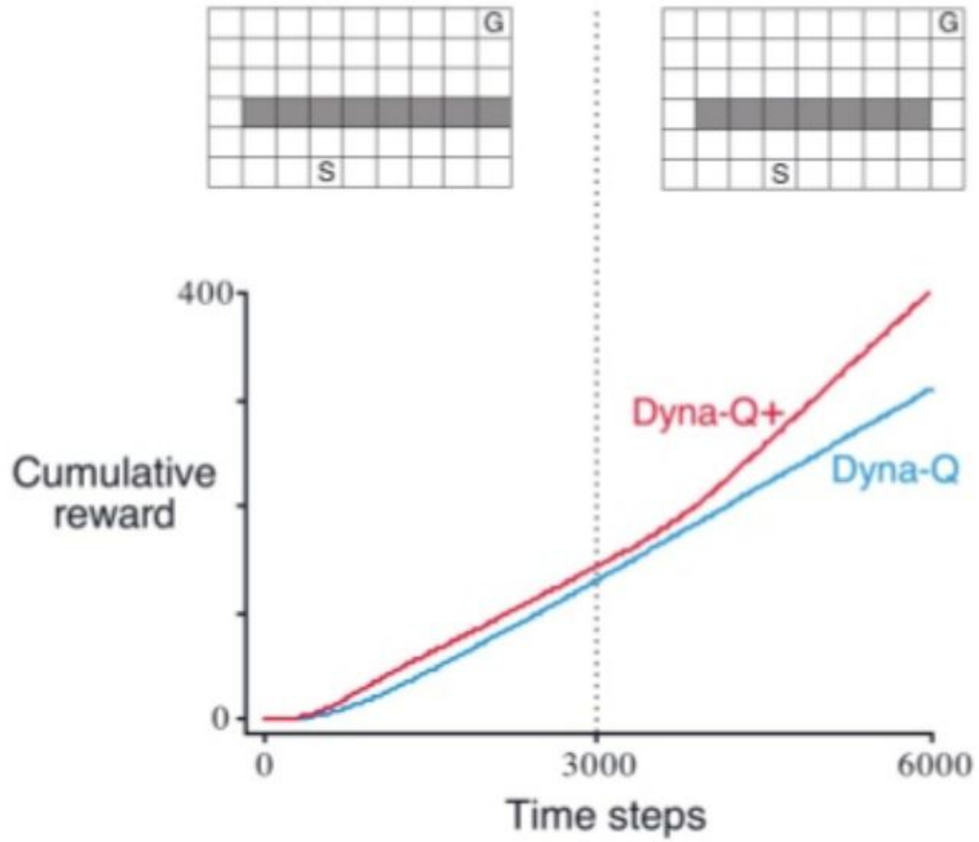
An example of environment change



Environment changes after 1000 timesteps

환경 변화 예시

An example of environment change



다이나-Q+

Dyna-Q+

- 탐험과 착취의 밸런스를 맞춰야 함
- (상태, 행동) 페어에 대해서 시간이 얼마나 지났는 지를 기록
- 계획(planning) 단계에서 보너스 보상을 추가
- 오랫동안 시도되지 않은 행동을 시도하도록 장려

- 이런 추가적인 탐험에는 비용이 발생하지만, 호기심은 비용을 지불할 가치가 있음

우선순위가 있는 일괄처리
Prioritized Sweeping

우선순위가 있는 일괄처리

Prioritized Sweeping

- 모든 변화가 동일하게 유용한 것은 아님
  - 변화의 양
  - 전이 확률
- 우선순위 큐를 활용해 업데이트의 우선순위를 매김
  - 높은 우선순위의 페어를 뽑은 후 업데이트
  - 상태 S이후 일정 기간동안 지나는 상태들을 업데이트 대상에 넣음

우선순위가 있는 일괄처리

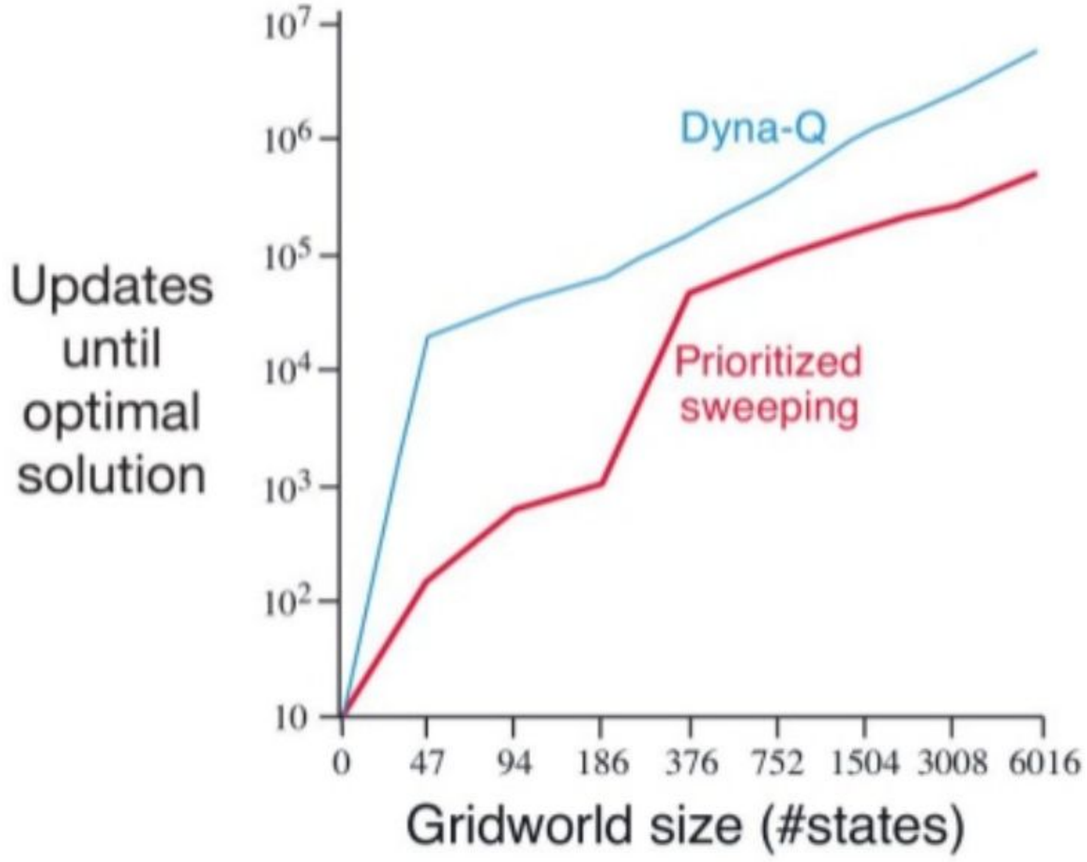
Prioritized sweeping for a deterministic environment

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:
  - $S,A \leftarrow first(PQueue)$
  - $R,S' \leftarrow Model(S,A)$
  - $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
  - 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

Expert

우선순위가 있는 일괄처리

Decisive advantage over uprioritized Dyna-Q



주사위 던지기 알고리즘
Rollout Algorithm

주사위 던지기 정책

Rollout Policy

- 게임이 끝날 때까지 위치를 여러 번 이동시킴으로써 즉 주사위를 던짐으로써 백개몬 위치의 가치를 추정한 데서 유래한다.

- **주사위 던지기 알고리즘**의 목적은 주어진 정책 pi에 대해 완전한 최적 행동 가치 함수 q* 또는 완전한 q_pi를 추정하는 것이 아니다.

- 대신, 주사위 던지기 알고리즘은 각각의 현재 상태에 대해서만, 보통 **주사위 던지기 정책**이라고 불리는 정책에 대해서만 행동의 가치를 몬테카를로 추정값을 계산한다. 이렇게 상태값을 추정한 다음엔 바로 폐기해버린다.

몬테카를로 트리 탐색
Monte-Carlo Tree Search

몬테카를로 트리 탐색

Monte-Carlo Tree Search

- 눈에 띄게 성공적인 결정 시각 계획의 사례

- 근본적으로는 주사위 던지기 알고리즘이지만, 시뮬레이션이 더욱 큰 보상을 주는 궤적을 향하게 하는 연속적인 방향 설정을 위해 몬테카를로 시뮬레이션으로부터 얻은 가치 추정값을 축적하는 수단을 추가하면서 주사위 던지기 알고리즘보다 향상되었음

- 2016년에 알파고 프로그램의 기본 알고리즘

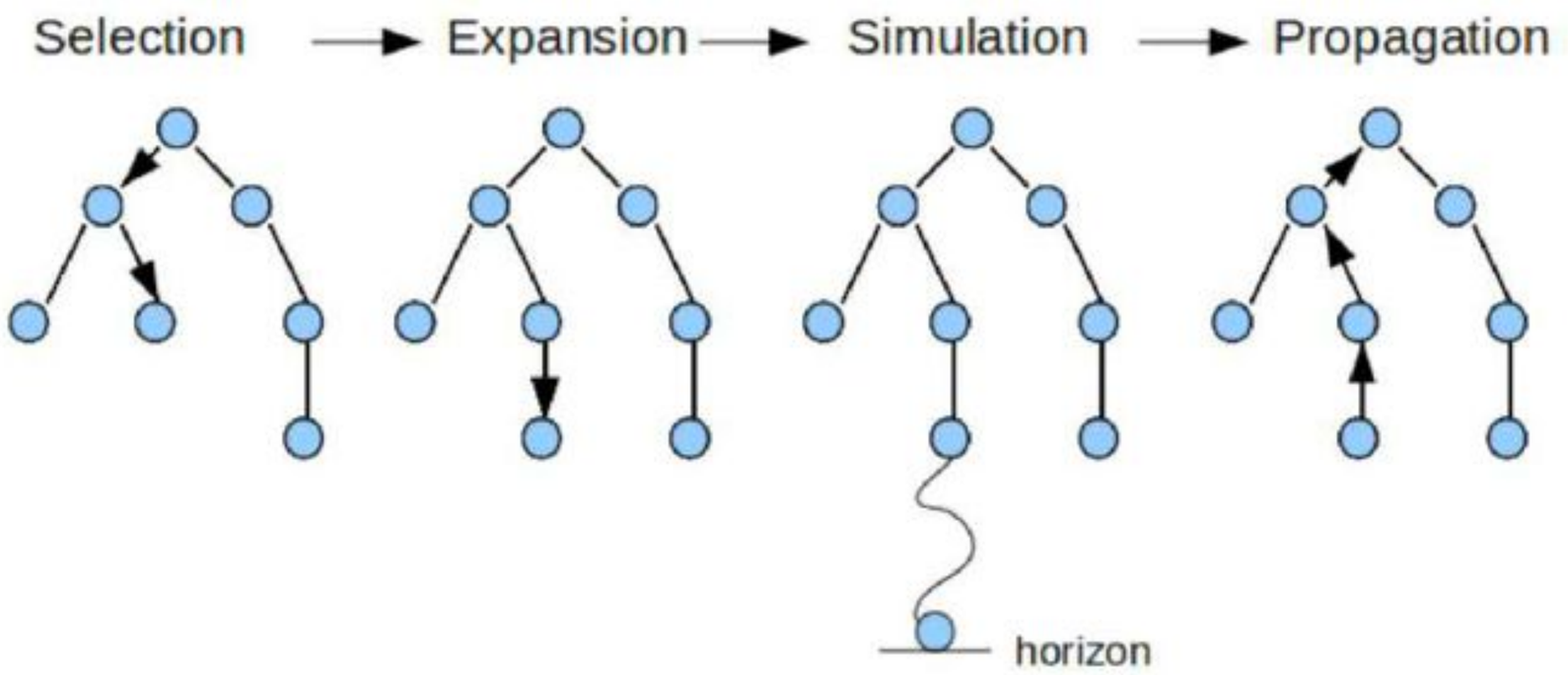
몬테카를로 트리 탐색: 4단계

Monte-Carlo Tree Search: 4 Steps

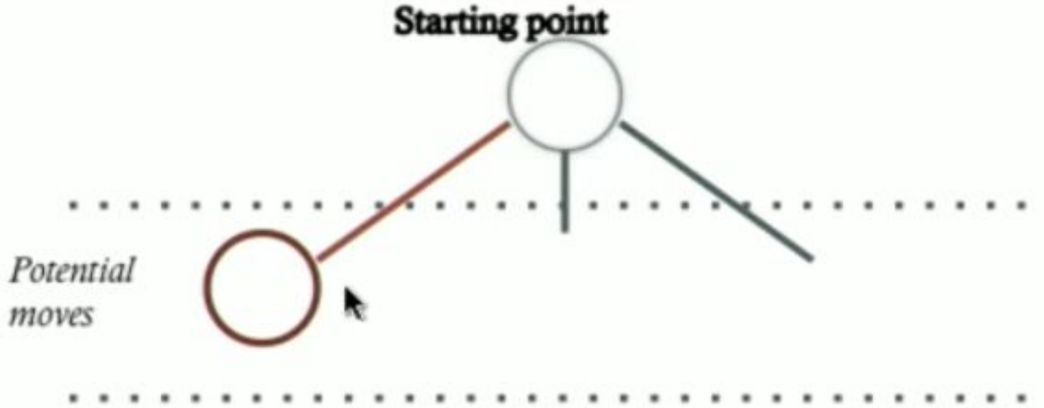
- **선택(Selection):**
  루트 노드에서 시작하여, 트리 구조의 모서리에 할당된 행동 가치를 기반으로 하는
  트리 정책이 트리 구조를 따라 이동하여 리프 노드를 선택
- **확장(Expansion):**
  몇 번의 반복 실행에서 선택된 노드로부터 탐험되지 않은 행동을 통해 도달한 하나
  또는 그 이상의 자식 노드를 추가함으로써 트리 구조는 선택된 리프 노드로부터 확장됨
- **시뮬레이션(Simulation):**
  새롭게 추가된 자식 노드들 중 하나로부터 완전한 에피소드에 대한 시뮬레이션이 실행되고
  주사위 던지기 정책에 따라 행동이 선택됨.
- **보강(Backup):**
  시뮬레이션된 에피소드에 의해 생성된 이득은 이 MCTS의 반복 과정에서
  트리 정책에 따라 형성되는 트리 구조의 모서리에 할당된 행동 가치를 갱신하거나 초기화함

몬테카를로 트리 탐색: 4단계

Monte-Carlo Tree Search: 4 Steps



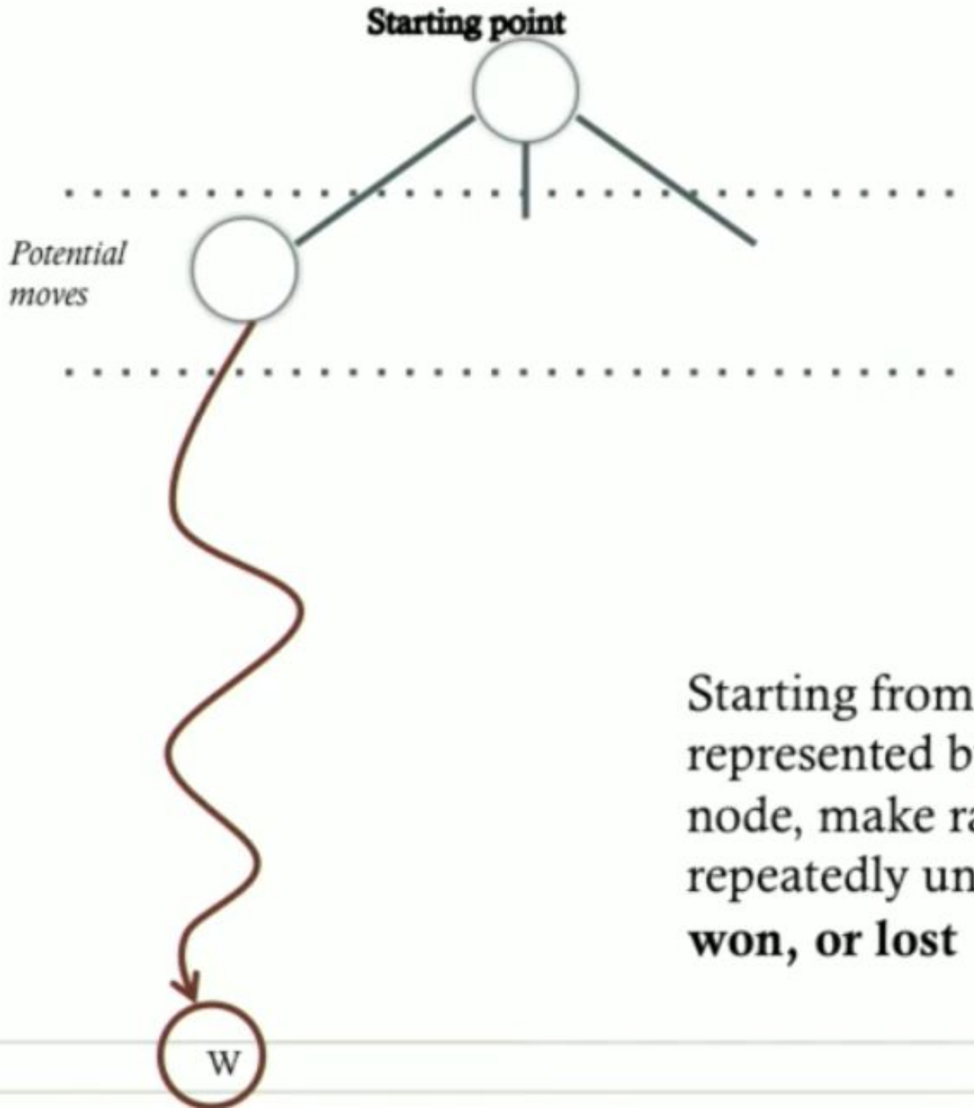
Selection, Expansion, Simulation, Update



EXPANSION

Add a new node to the stats tree, representing a position in the game tree that the AI will "investigate" (how good the move is) next.

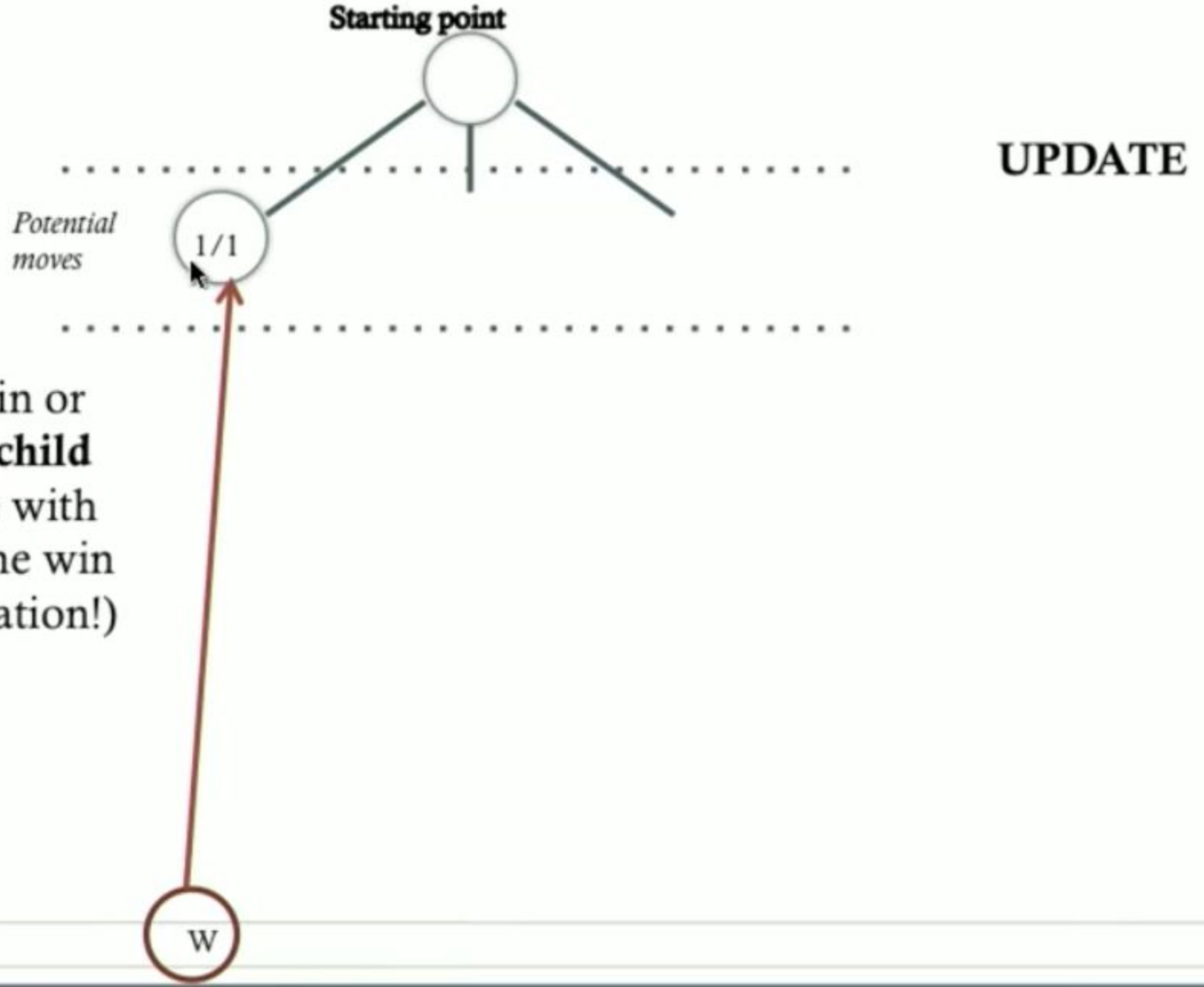
Selection, Expansion, Simulation, Update



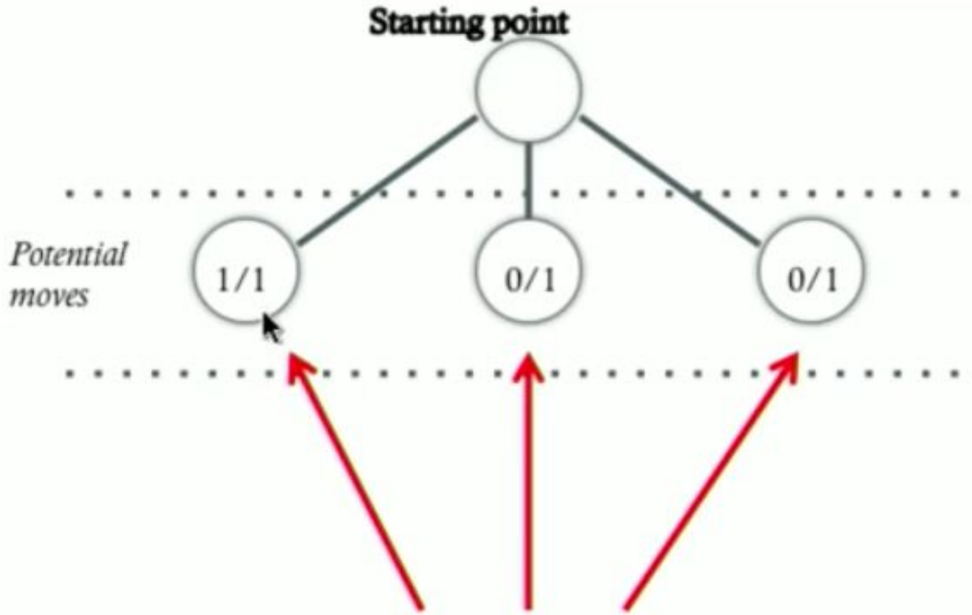
SIMULATION

Starting from the position represented by the left child node, make random moves repeatedly until the game is **won, or lost**

Selection, Expansion, Simulation, Update



Selection, Expansion, Simulation, Update

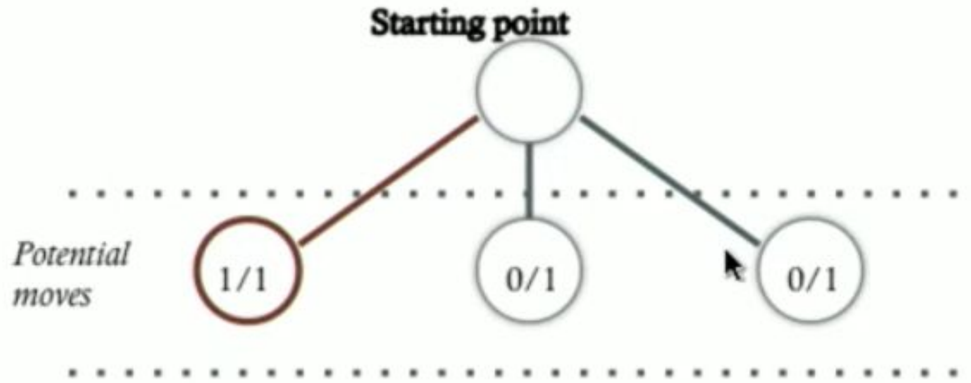


UPDATE

#'s say **left node** is the *better move,* but chances are, these numbers (produced by a single random simulation) are probably not a good indicator of how good any of the moves are!

More simulations will make them more accurate!

Selection, Expansion, Simulation, Update

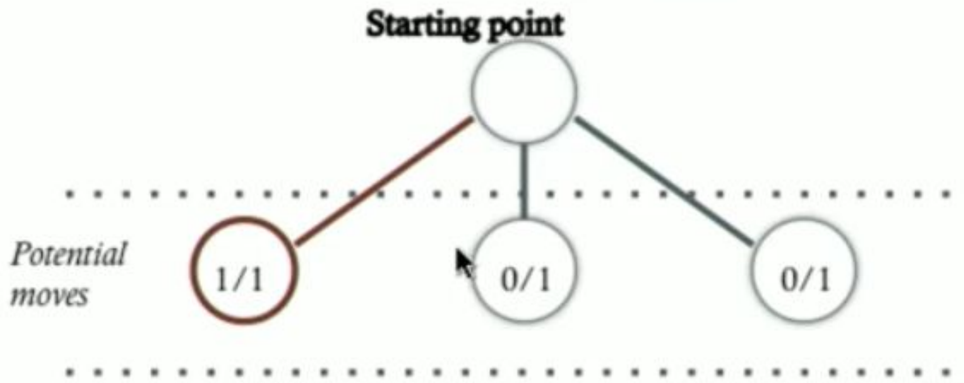


SELECTION

All child nodes have now been visited at least once. Now AI can **select** which child node to be **investigated further.**

Remember the higher the value, the "better" the move is (for producing a win)

Selection, Expansion, Simulation, Update

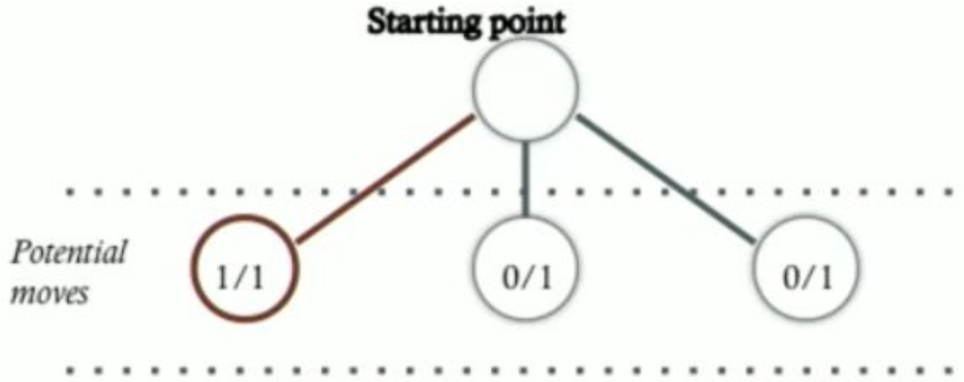


SELECTION

Selection based on 2 things:

- How good are the stats?
- How much has child node been “ignored”

Selection, Expansion, Simulation, Update



SELECTION

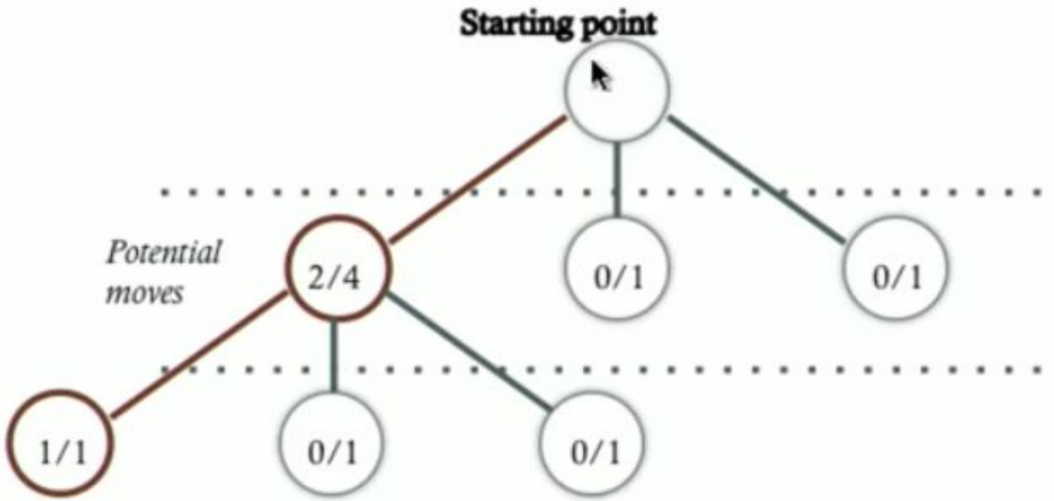
$$\bar{x}_i \pm \sqrt{\frac{2 \ln n}{n_i}}$$

$\bar{x}_i$: Mean "value" of node

n_i: Number of simulations done for child node (i)

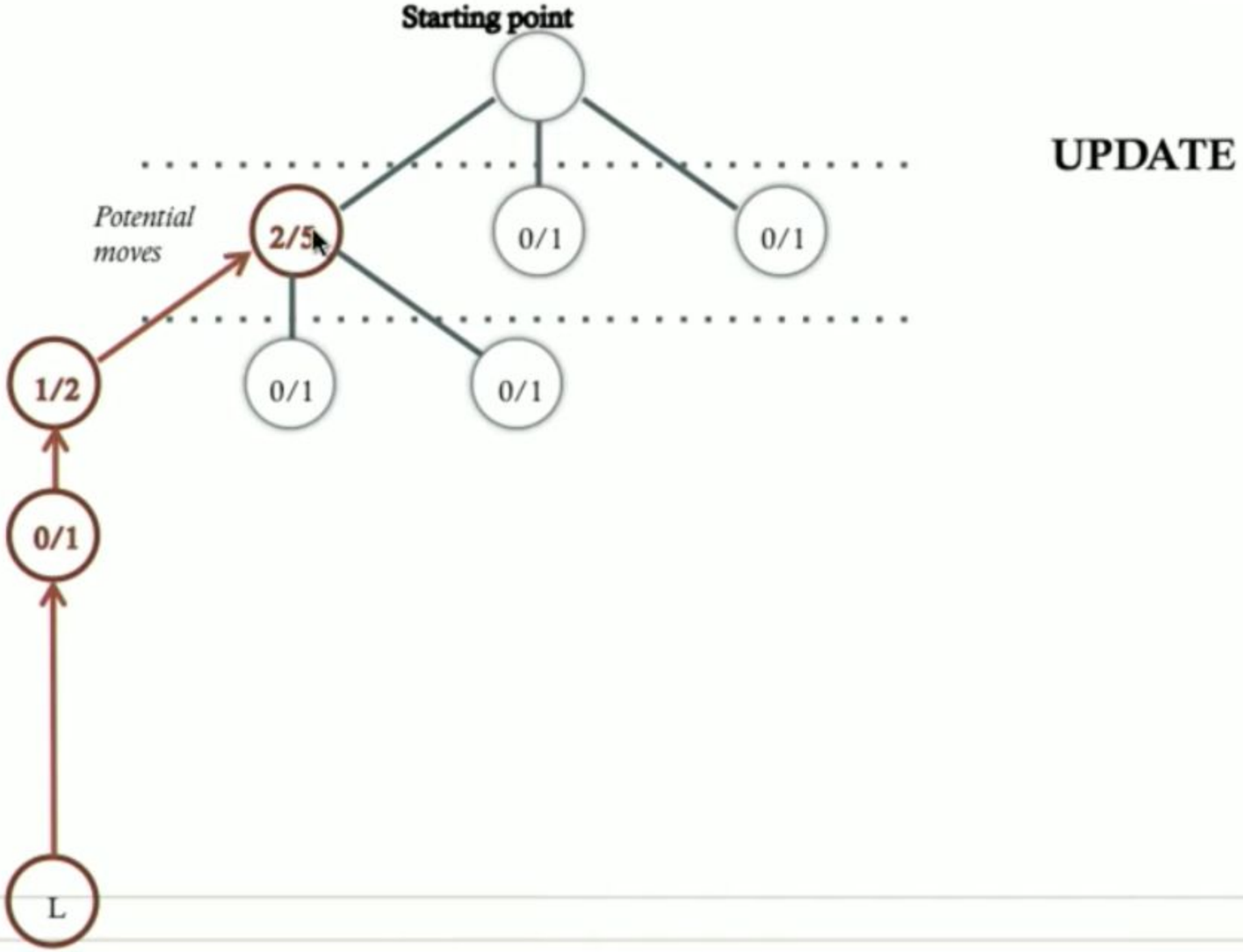
n: Number of total simulations done for all nodes

Selection, Expansion, Simulation, Update

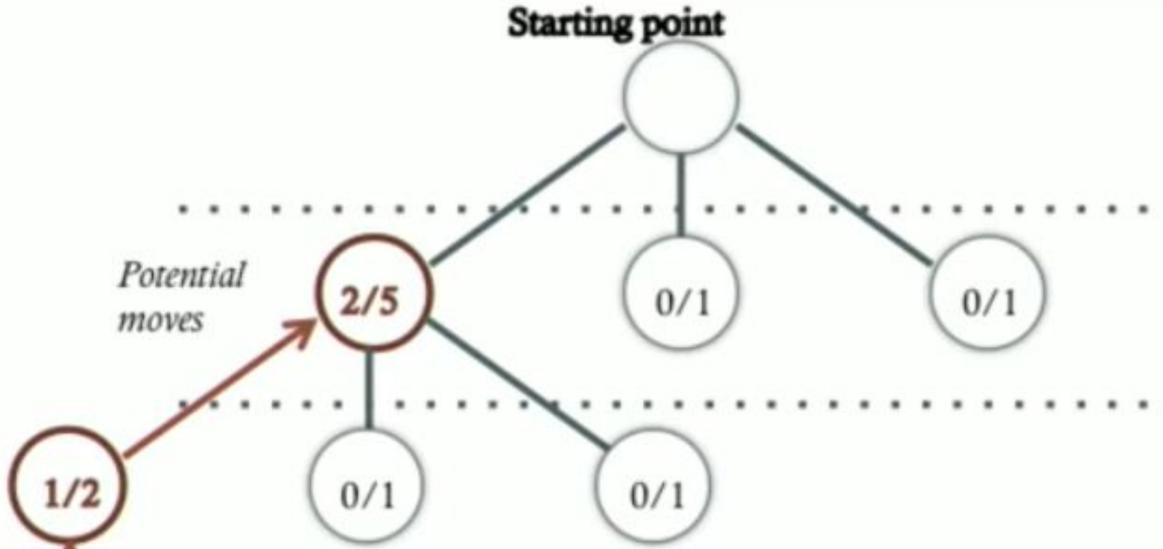


SELECTION

Selection, Expansion, Simulation, Update



Selection, Expansion, Simulation, Update



UPDATE

E.O.D.