ELTE | FACULTY OF INFORMATICS

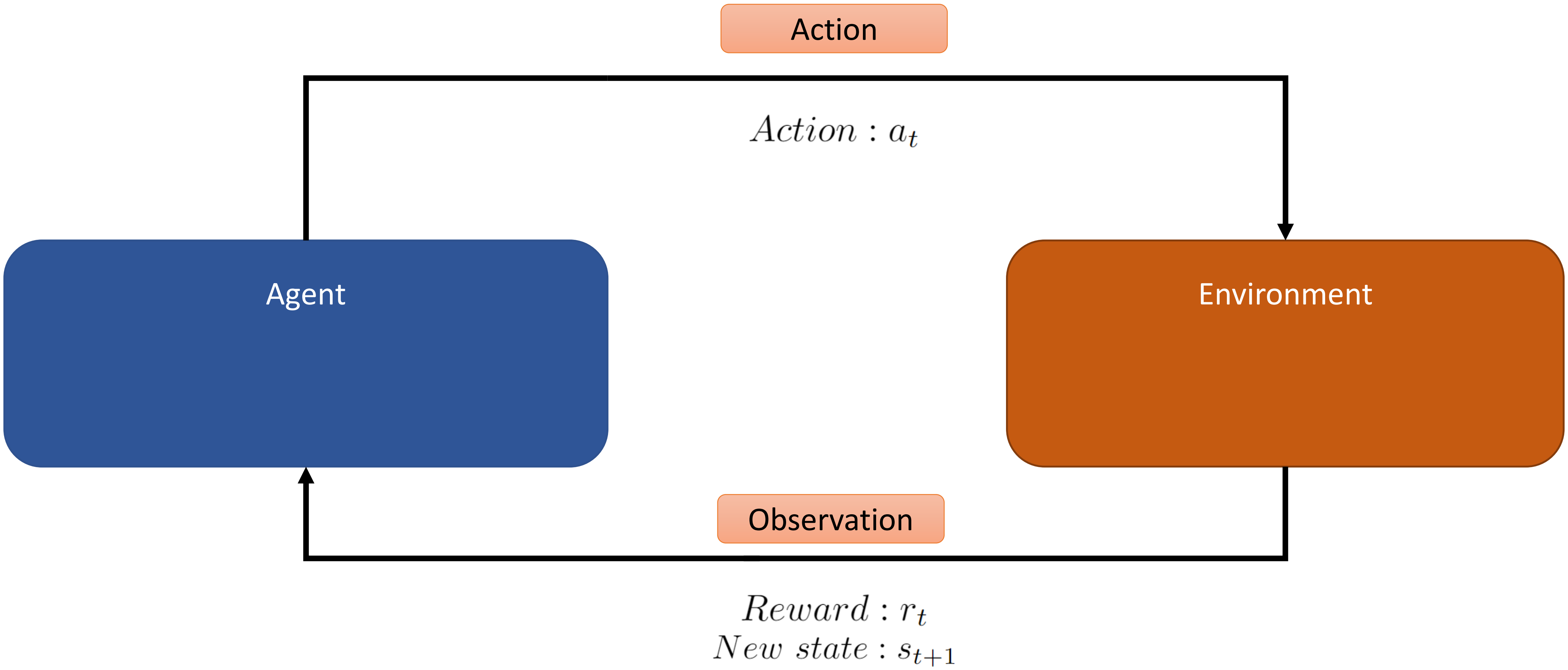
PLANNING AND LEARNING

Deep Reinforcement Learning

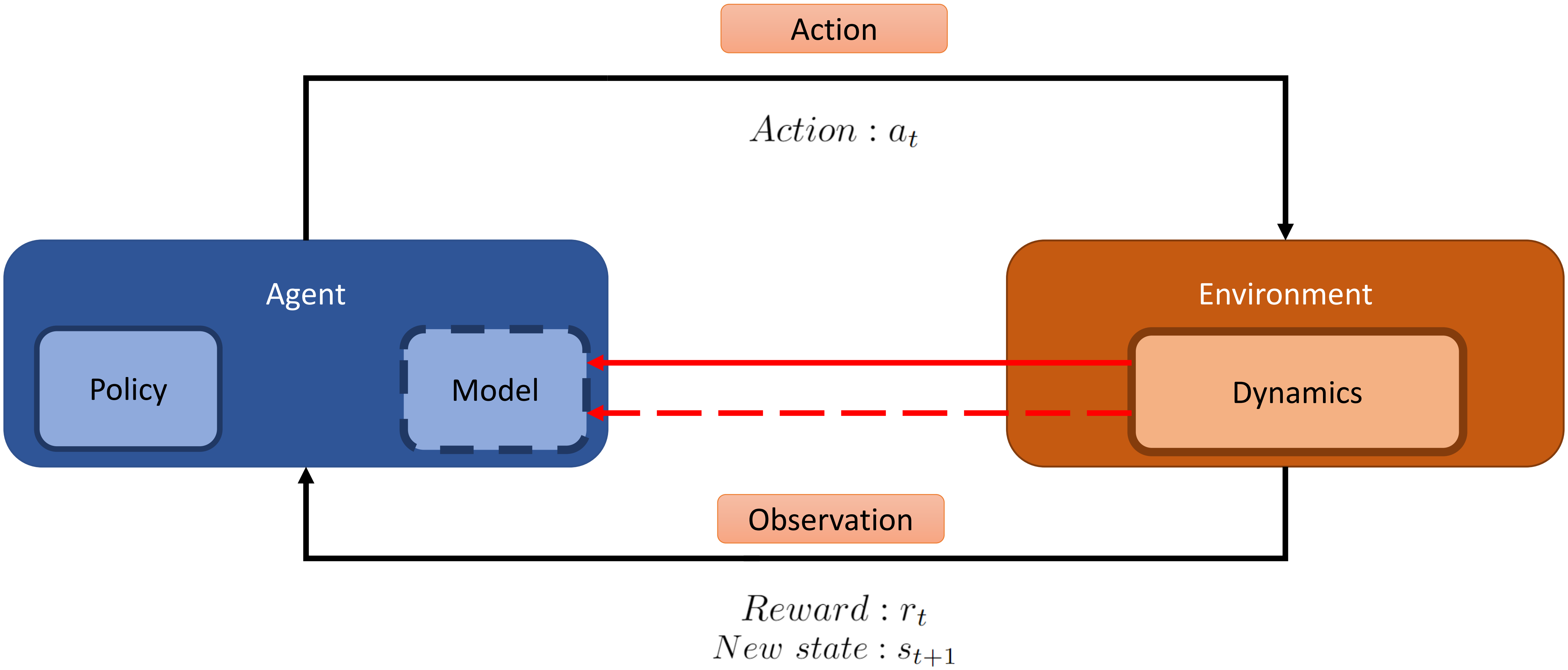
Balázs Nagy, PhD

ELTE | IK
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

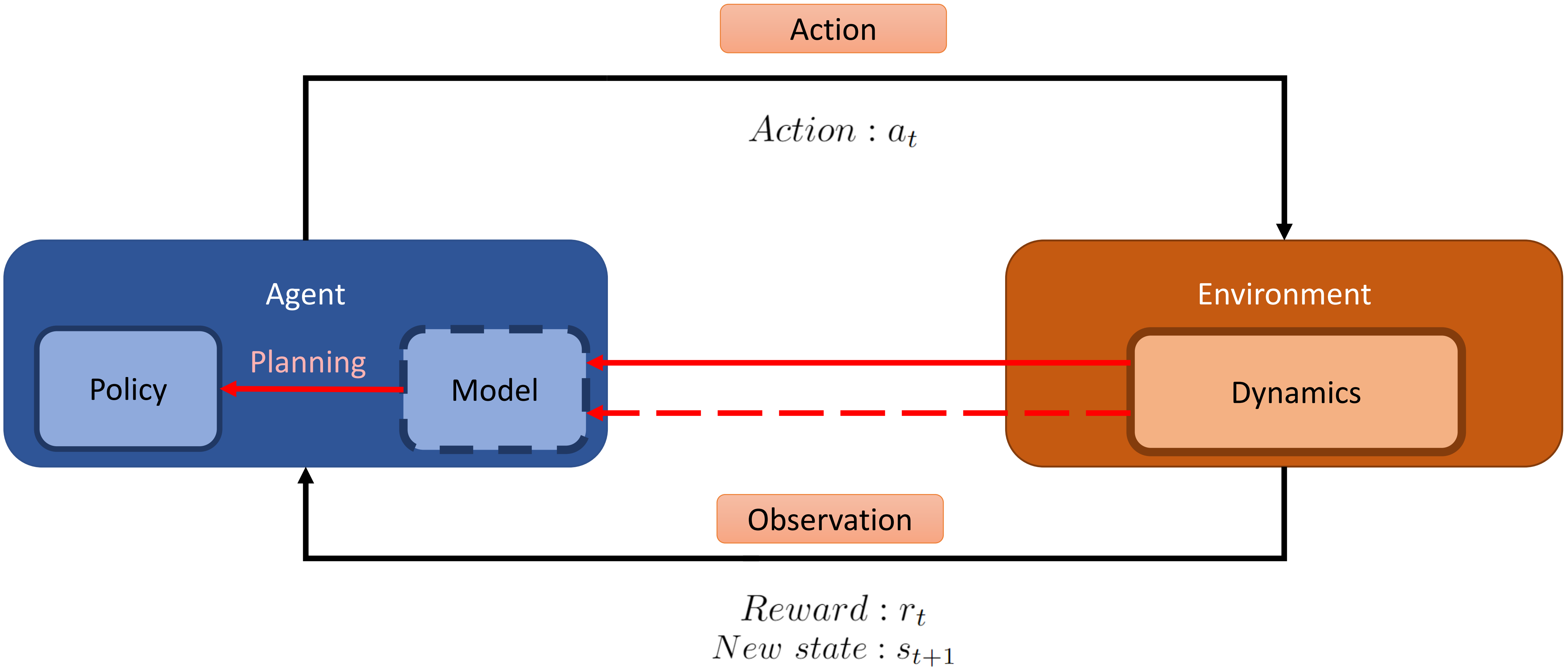
Reinforcement Learning (RL) – Key concept



Reinforcement Learning (RL) – Key concept

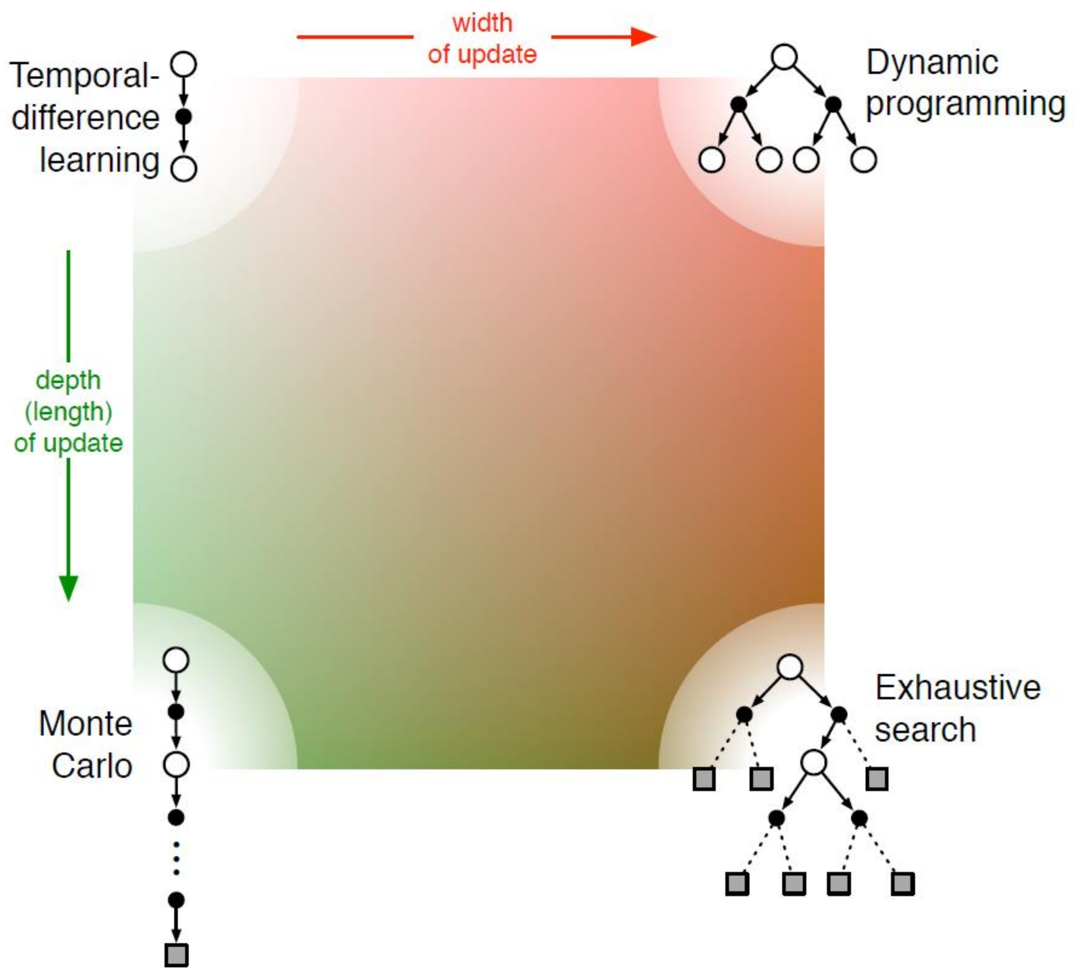


Reinforcement Learning (RL) – Key concept

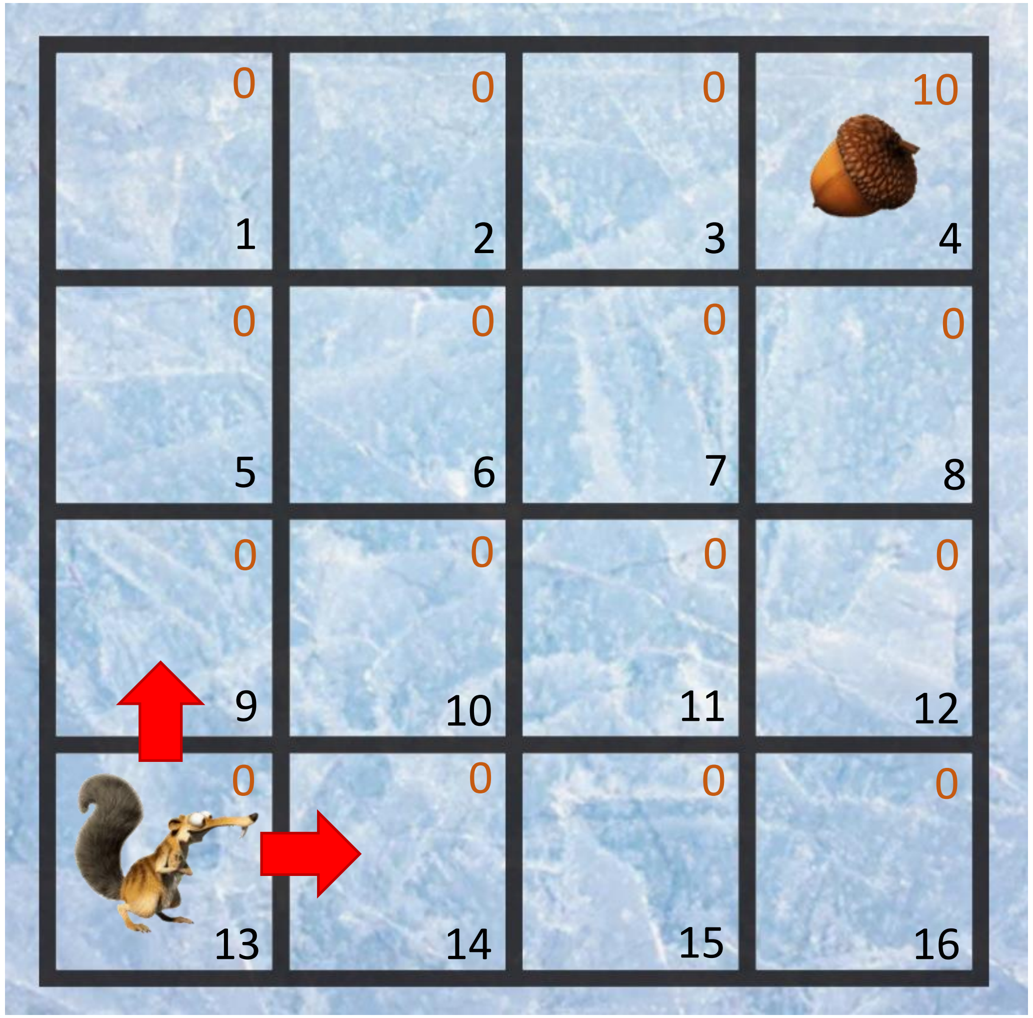


Tabular methods

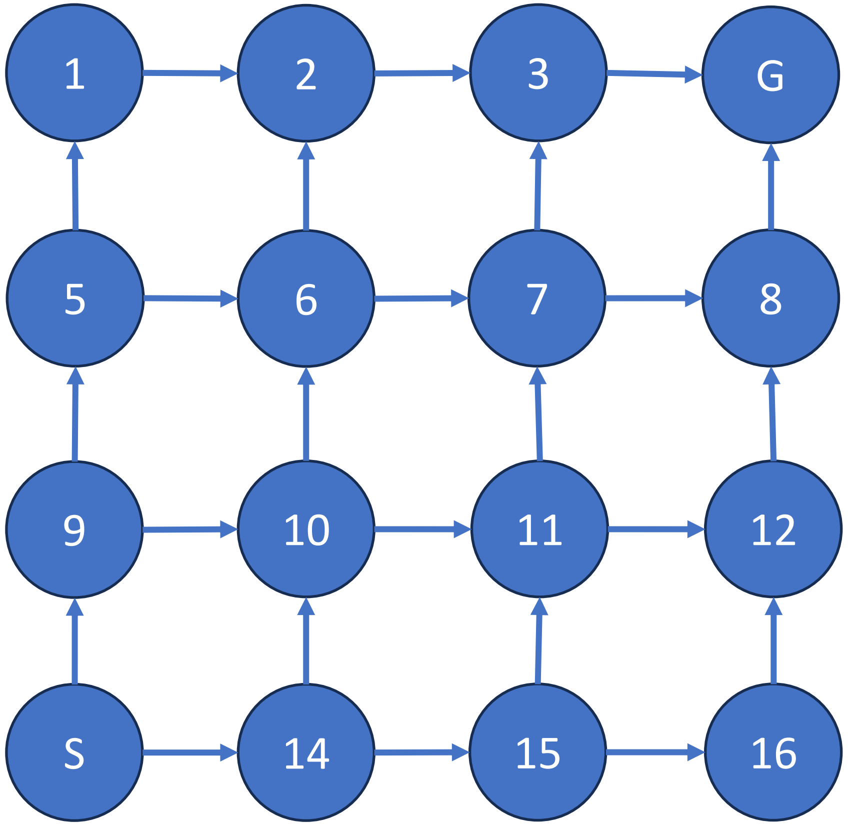
- Model-based methods
  - Dynamic programming
  - Heuristic search
  - **Planning**
- Model-free methods
  - Monte Carlo
  - Temporal Difference
  - **Learning**



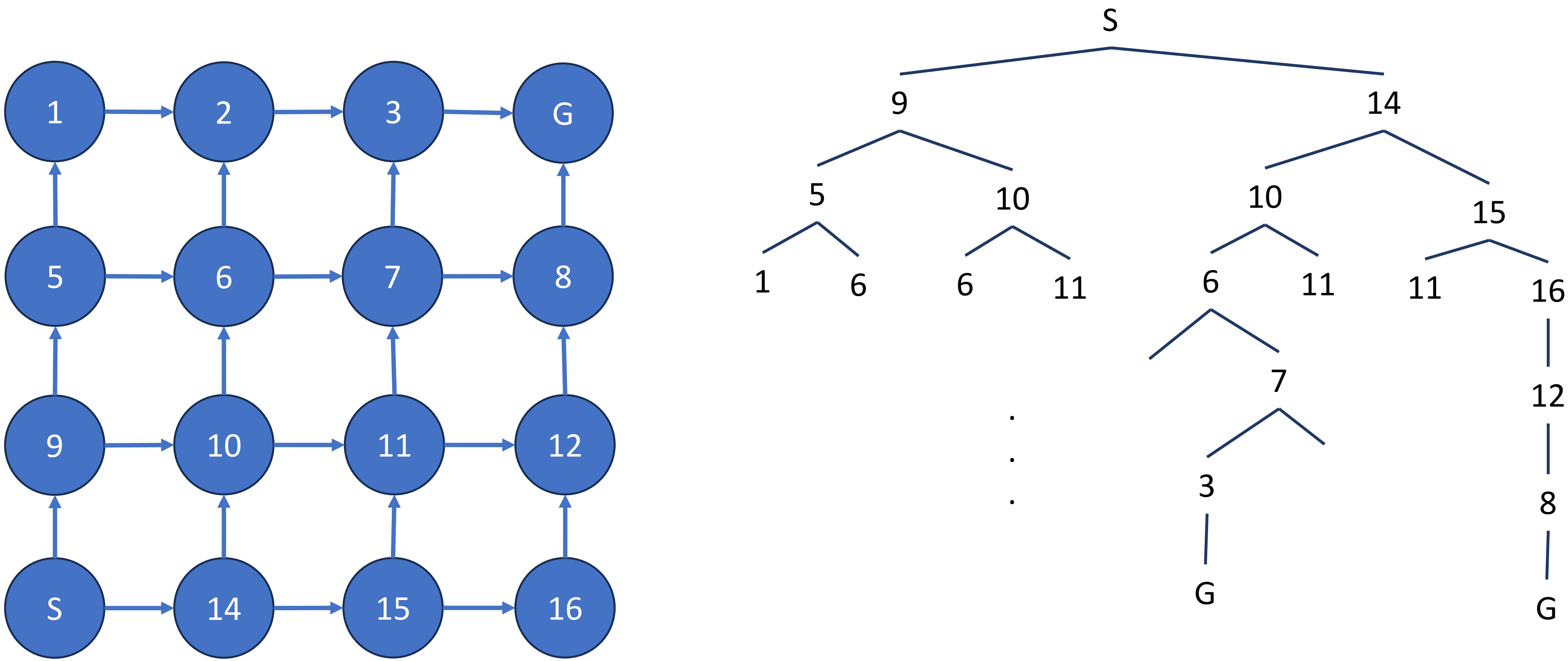
Search problems are Models



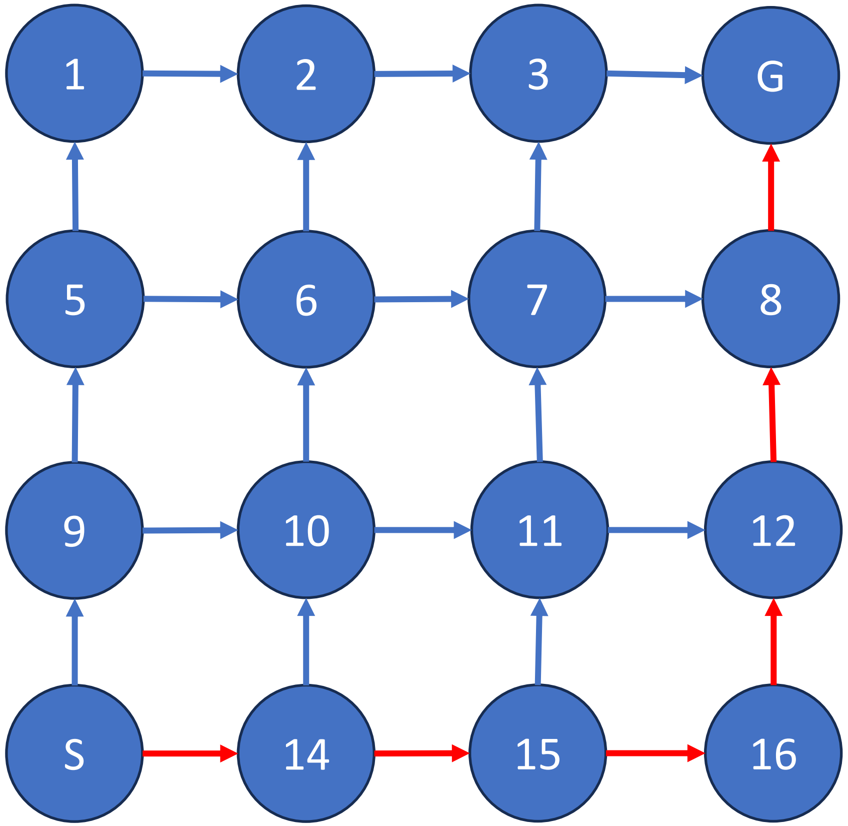
State Space Graph vs Search Trees

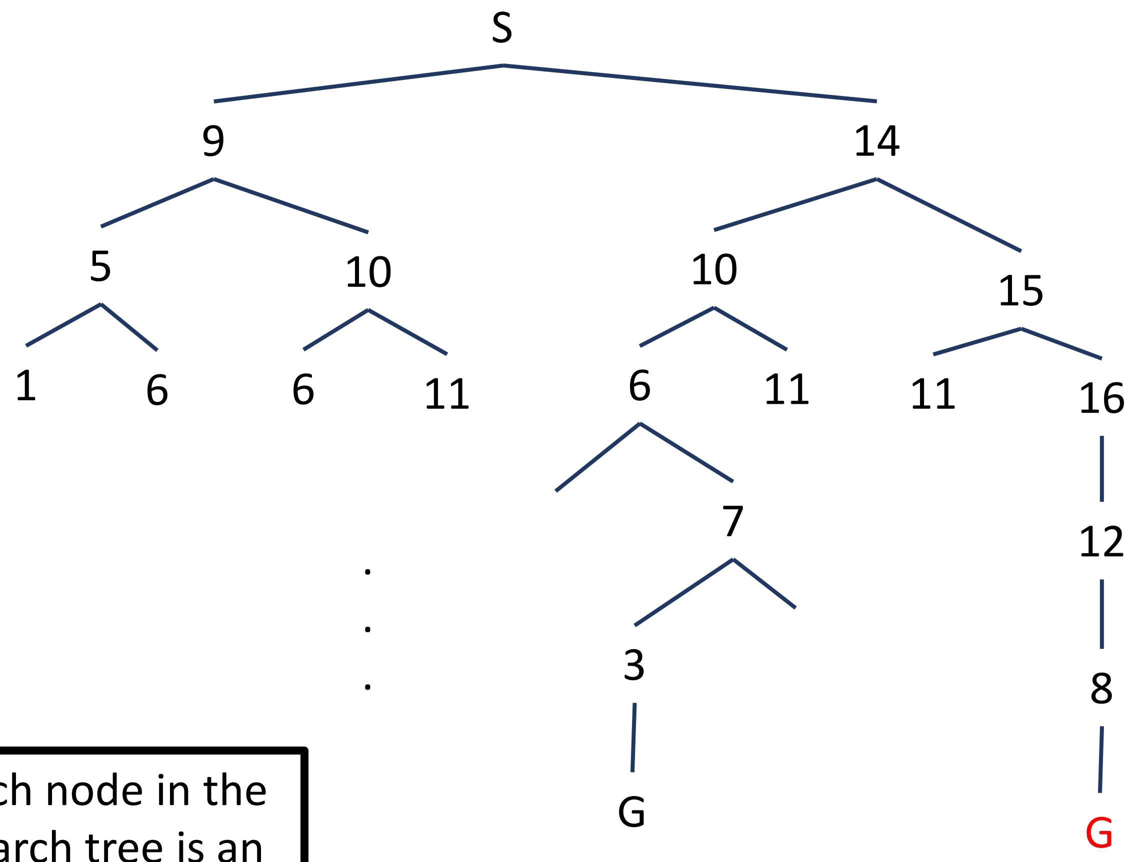


State Space Graph vs Search Trees



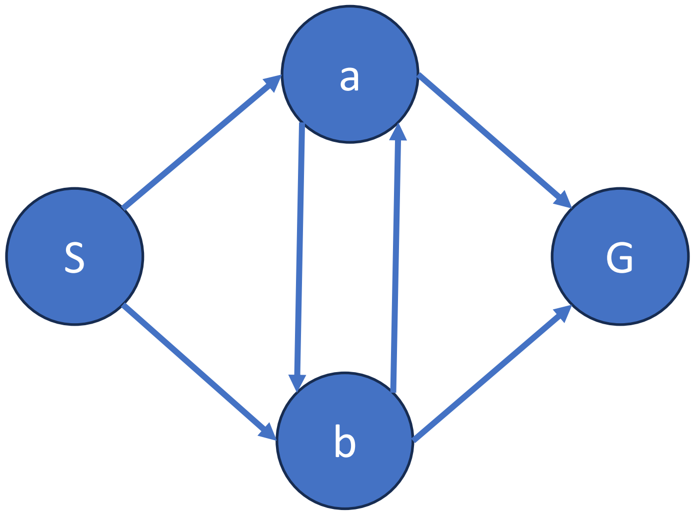
State Space Graph vs Search Trees





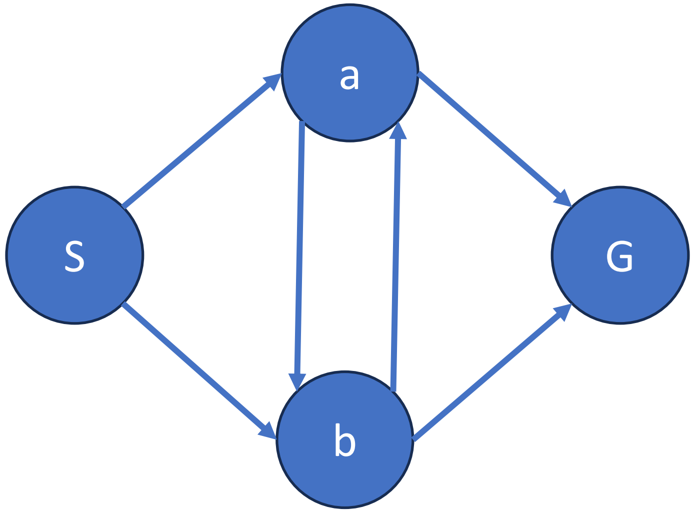
Each node in the search tree is an entire PATH in the state space graph

State Space Graph vs Search Trees



How big is the search tree? (from S)

State Space Graph vs Search Trees

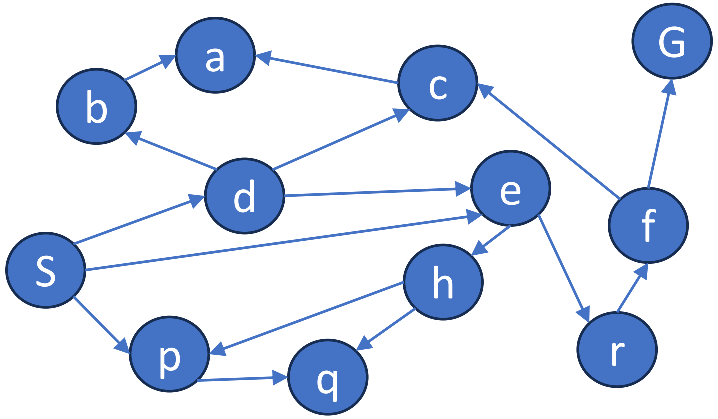


How big is the search tree? (from S)

$$\infty$$

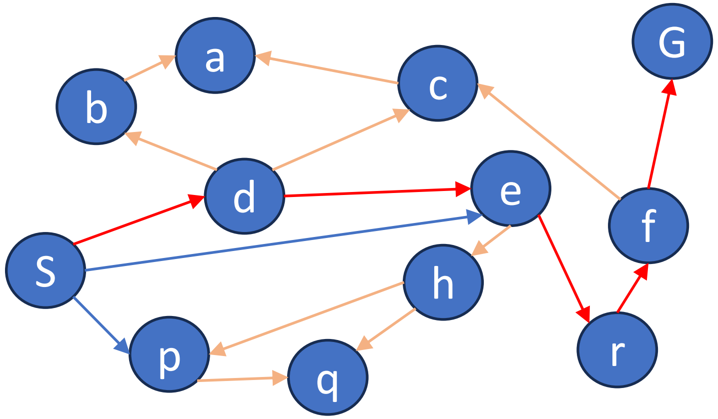
Lots of repeated structure in the search tree!

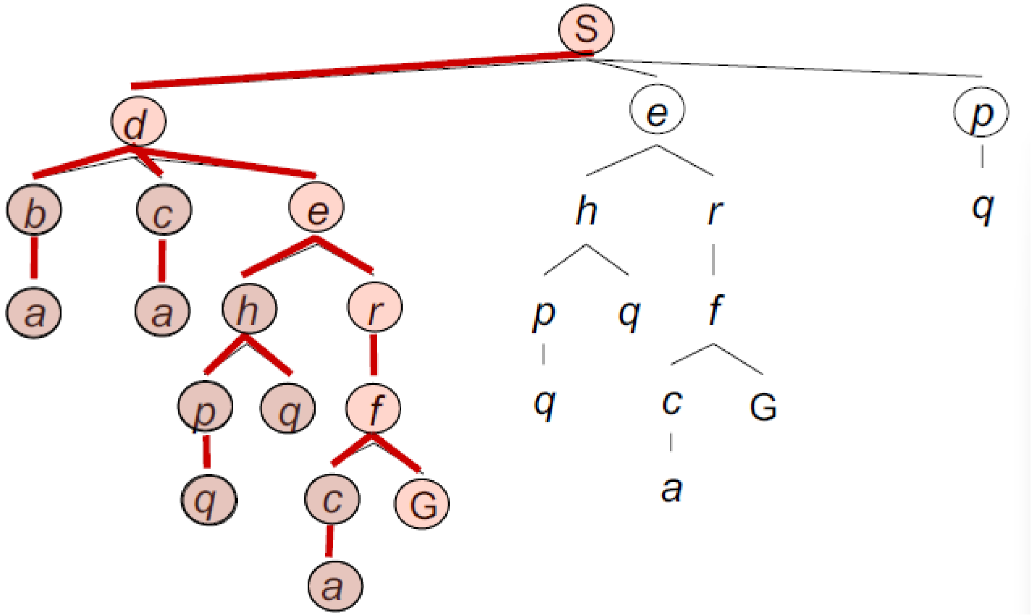
Depth-First Search



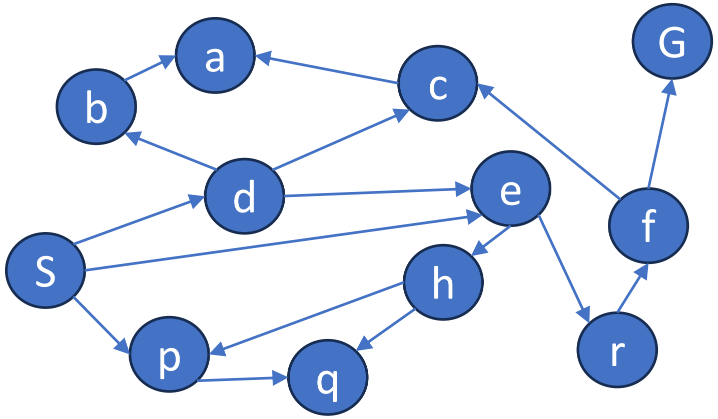
Strategy: Expand the deepest node first

Depth-First Search



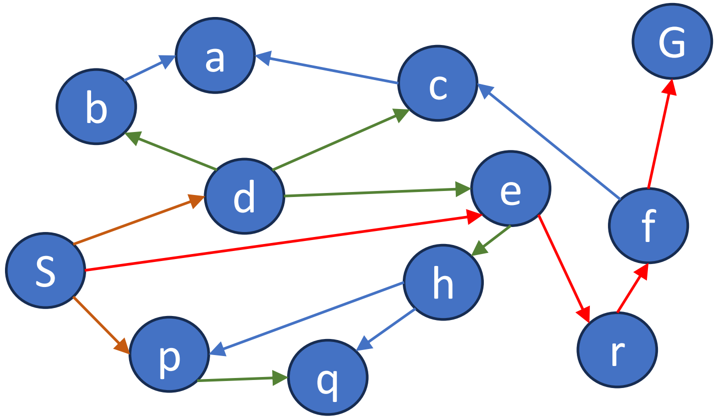


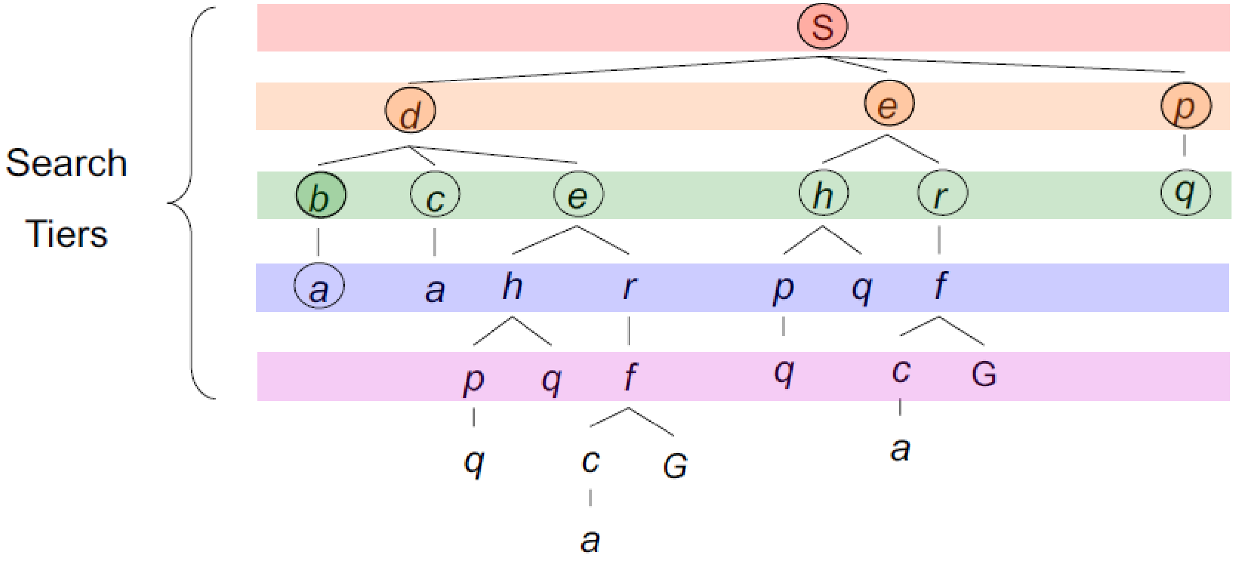
Breadth-First Search



Strategy: Expand the shallowest node first

Breadth-First Search





Tabular methods

- Model-based methods
  - Dynamic programming
  - Heuristic search
  - **Planning**
- Model-free methods
  - Monte Carlo
  - Temporal Difference
  - **Learning**

Similarities:

- computation of value functions
- looking ahead to future events
- computing a backed-up value, and then using it as an update target for an approximate value function

Reflex Agents vs Planning Agent

- Reflex Agent
  - Decision based on current perception
  - May have a model of the environment
  - Do not consider the future consequence of their action

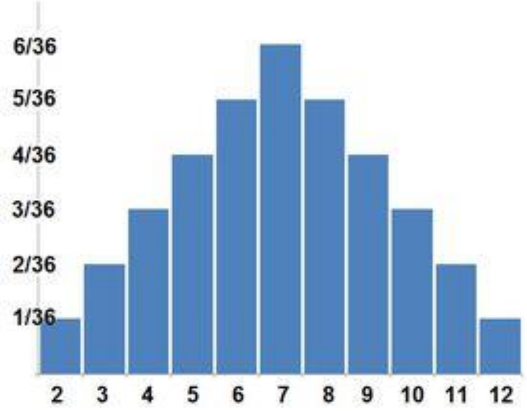
- Planning Agent
  - Decision based on consequences of actions
  - Must have a model of the environment
  - Consider the future consequence of their action

Models

- a model of the environment:
  anything that an agent can use to predict how the environment will respond to its actions
  stochastic = several possible next states
- **Distribution models**:
  produce a description of all possibilities and their probabilities
- **Sample models**:
  produce just one of the possibilities, sampled according to the probabilities

Example: 2 dice sum

- Distribution model:



Sum of 2 Dice Chart

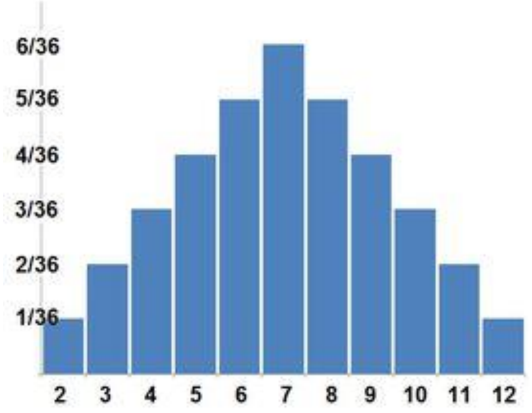
	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

- Sample model

  Randomize 2 number between
  1 and 6 then sum it up = 1 sample

Example: 2 dice sum

- Distribution model:



Sum of 2 Dice Chart

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

- Sample model

  Randomize 2 number between
  1 and 6 then sum it up = 1 sample

Models can be used to mimic or simulate experience

Planning and Learning

- **Planning:**
  Uses simulated experience generated by a model
- **Learning:**
  Use real experience generated by the environment

Planning and Learning

- **Planning:**
  Uses simulated experience generated by a model
- **Learning:**
  Use real experience generated by the environment
- **Goal:**
  Common structure for planning and learning

Pseudocode

Random-sample one-step tabular Q-planning

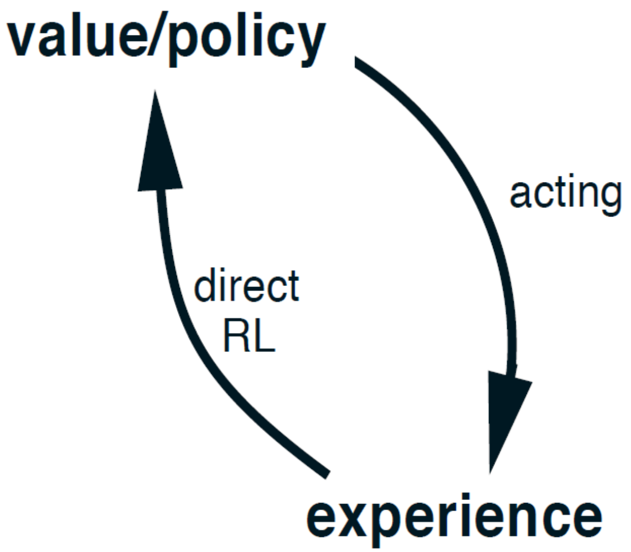
Loop forever:

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

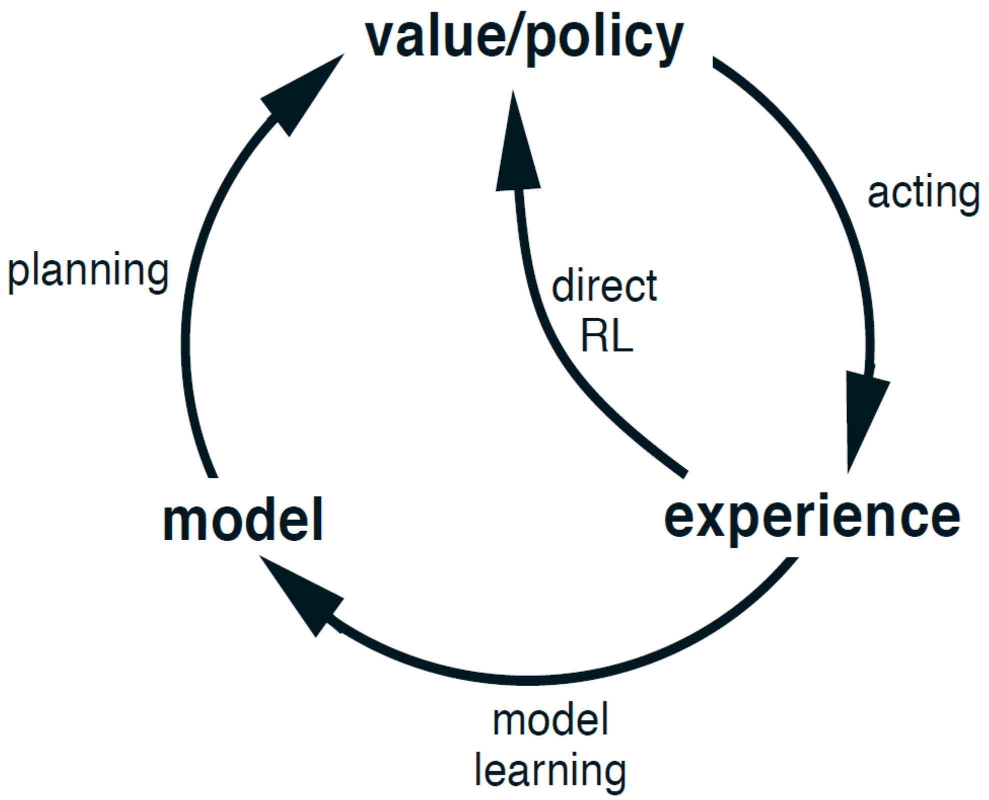
On-line planning problems

- New information gained from the interaction may change the model and thereby interact with planning
- If decision making and model learning are both computation-intensive processes, then the available computational resources may need to be divided between them.

Planning Agent

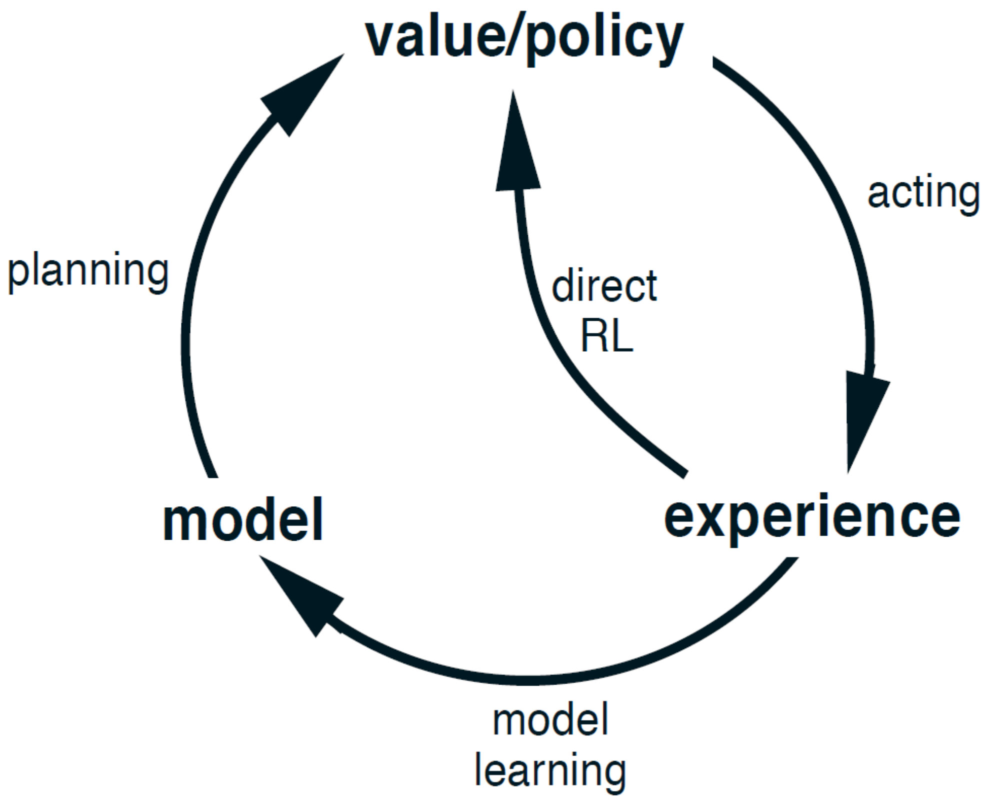


Planning Agent



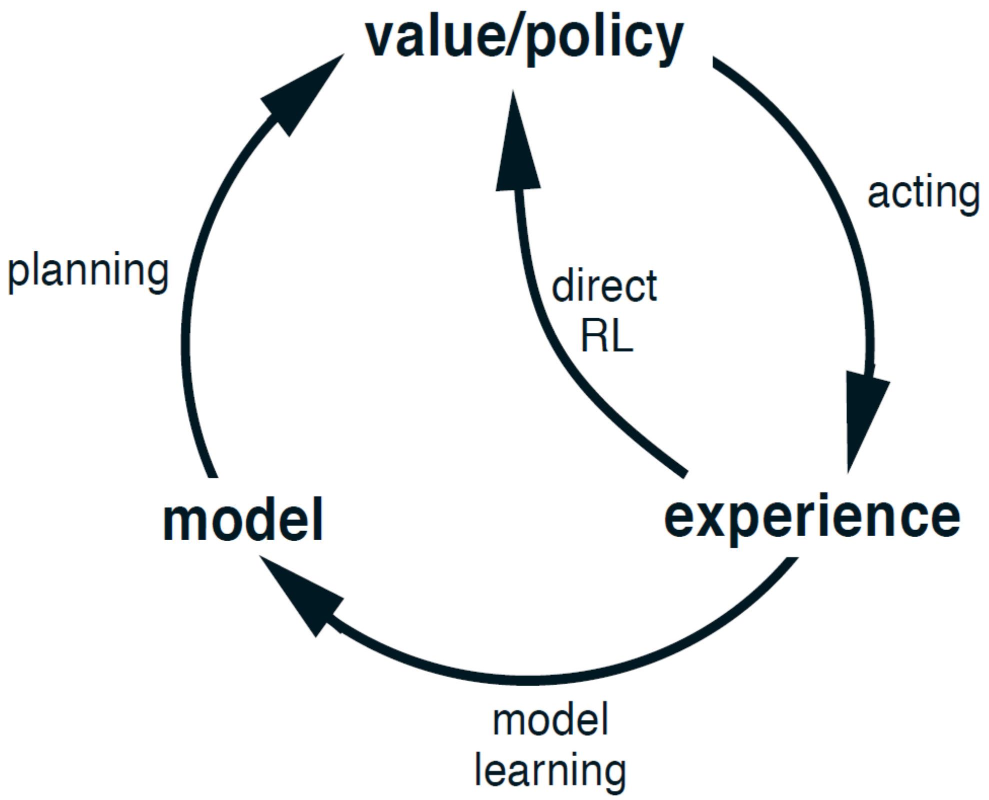
Planning Agent

- Two roles for real experience:
  - ***model-learning*** or ***indirect reinforcement learning:***
    improve the model (to make it more accurately match the real environment)
  - ***direct reinforcement learning (direct RL):***
    directly improve the value function and policy



Planning Agent

- Two roles for real experience:
  - ***model-learning*** or ***indirect reinforcement learning:***
    improve the model (to make it more accurately match the real environment)
    - Better use of limited amount of experience
    - Achieve a better policy with fewer environmental interactions
  - ***direct reinforcement learning (direct RL):***
    directly improve the value function and policy
    - Much simpler
    - Not affected by biases in the design of the model



Dyna-Q

Q-Learning

- Init Q-table
- Observe s
- Execute a, observe s’, r
- Update Q with <s,a,s’,r>

Dyna-Q

Q-Learning

- Init Q-table
- Observe s
- Execute a, observe s’, r
- Update Q with <s,a,s’,r>

Dyna-Q

- Learn model
- Hallucinate experience
- Update Q

Dyna-Q

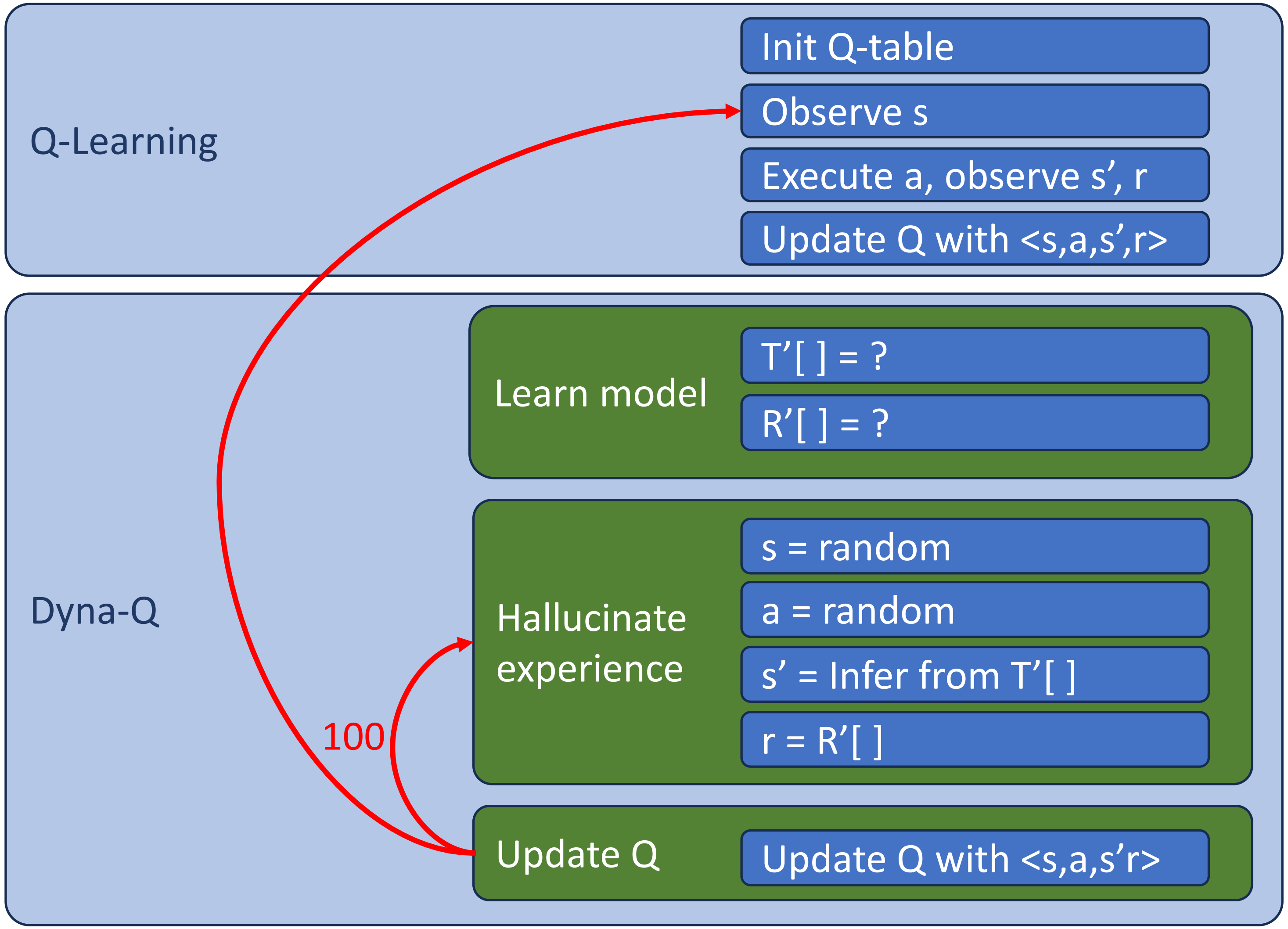
Q-Learning

- Init Q-table
- Observe s
- Execute a, observe s’, r
- Update Q with <s,a,s’,r>

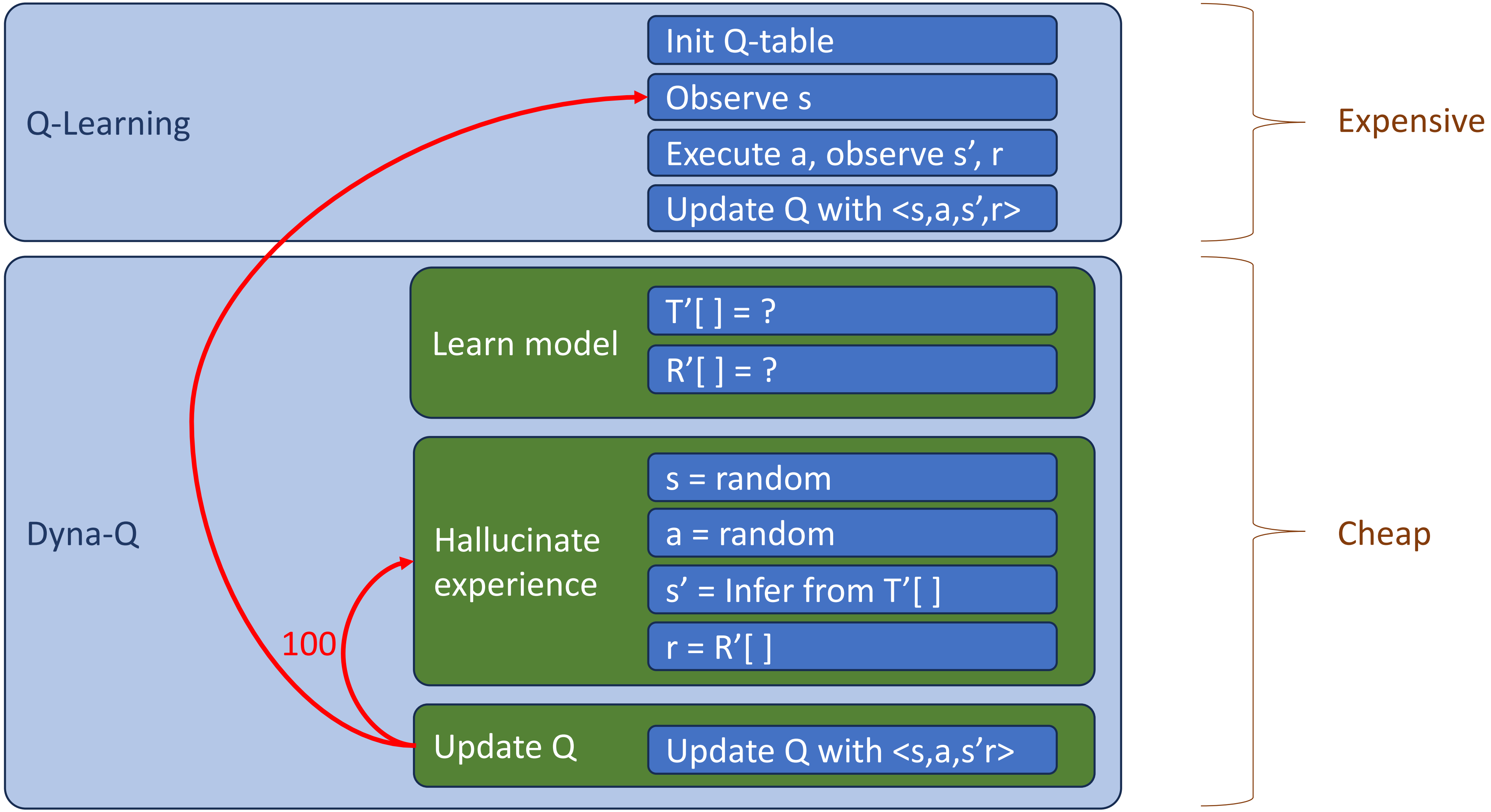
Dyna-Q

- Learn model
  - T’[ ] = ?
  - R’[ ] = ?
- Hallucinate experience
  - s = random
  - a = random
  - s’ = Infer from T’[ ]
  - r = R’[ ]
- Update Q
  - Update Q with <s,a,s’r>

Dyna-Q

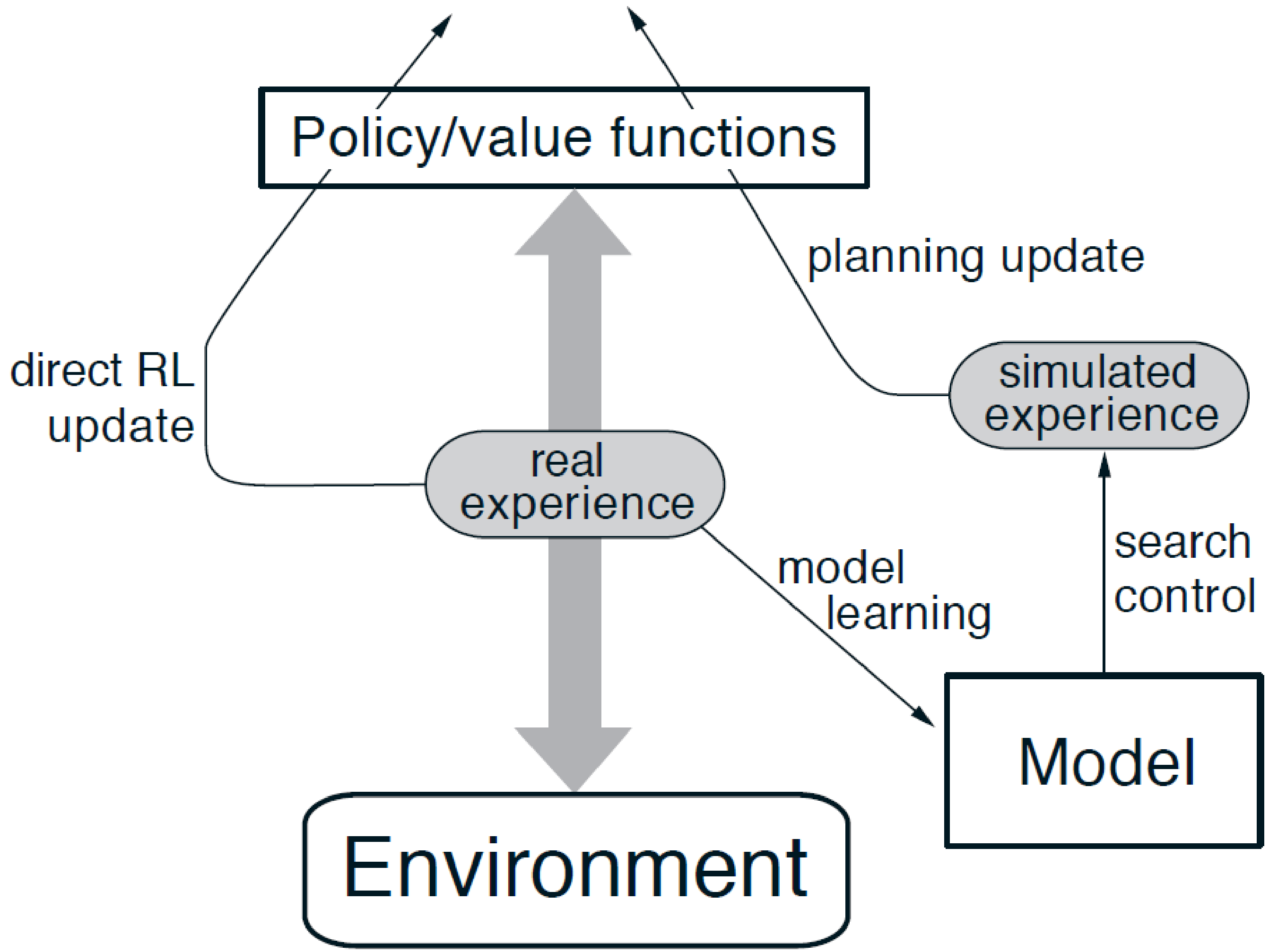

Q-Learning
Init Q-table
Observe s
Execute a, observe s’, r
Update Q with <s,a,s’,r>
Dyna-Q
Learn model
T’[ ] = ?
R’[ ] = ?
Hallucinate experience
s = random
a = random
s’ = Infer from T’[ ]
r = R’[ ]
100
Update Q
Update Q with <s,a,s’r>

Dyna-Q



Q-Learning
Init Q-table
Observe s
Execute a, observe s’, r
Update Q with <s,a,s’,r>
Expensive
Dyna-Q
Learn model
T’[ ] = ?
R’[ ] = ?
Hallucinate experience
s = random
a = random
s’ = Infer from T’[ ]
r = R’[ ]
100
Update Q
Update Q with <s,a,s’r>
Cheap

General Dyna Architecture

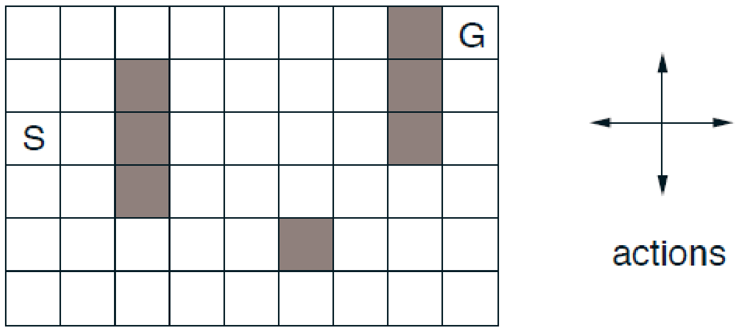


Pseudocode

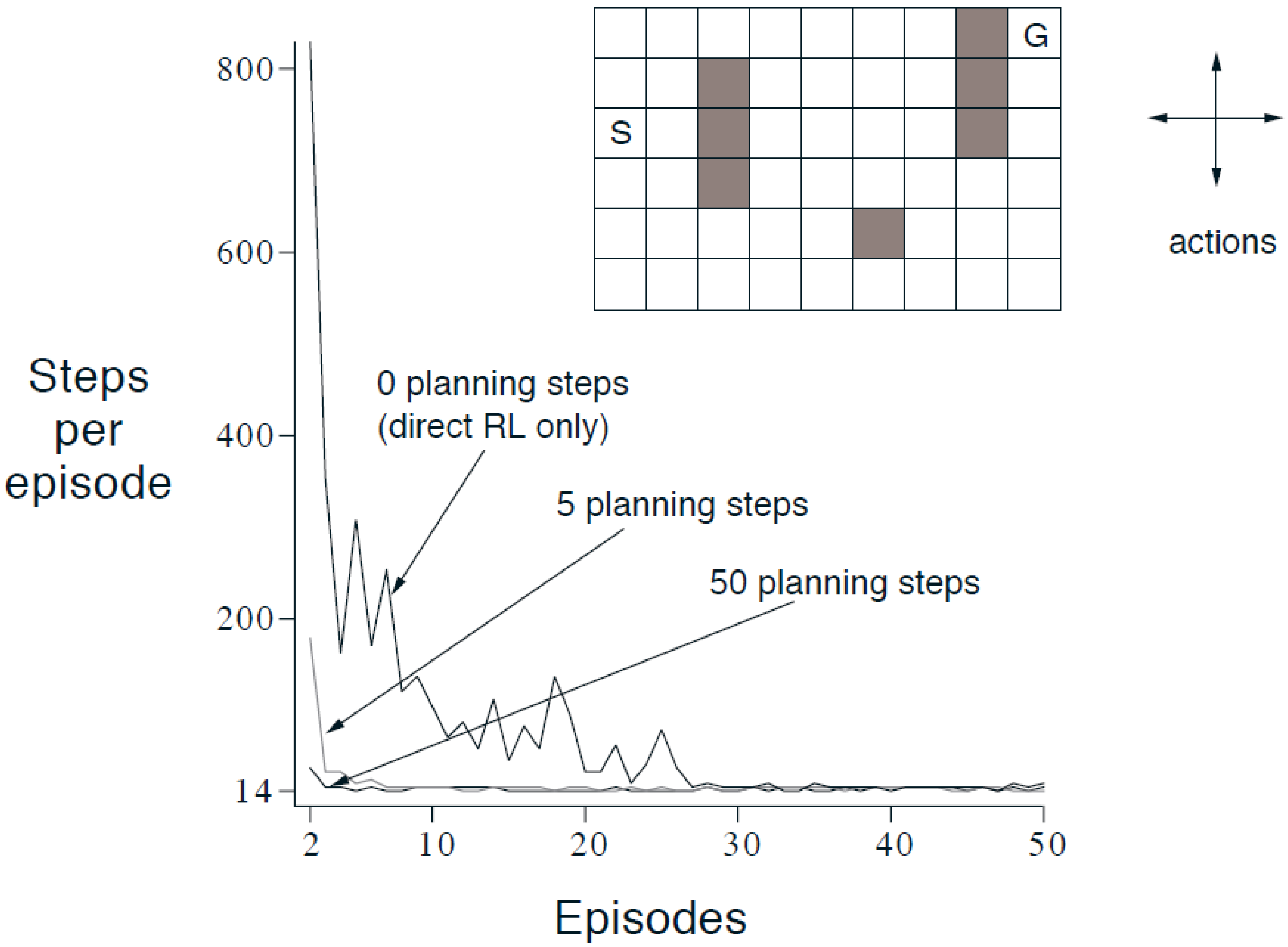
Tabular Dyna-Q

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

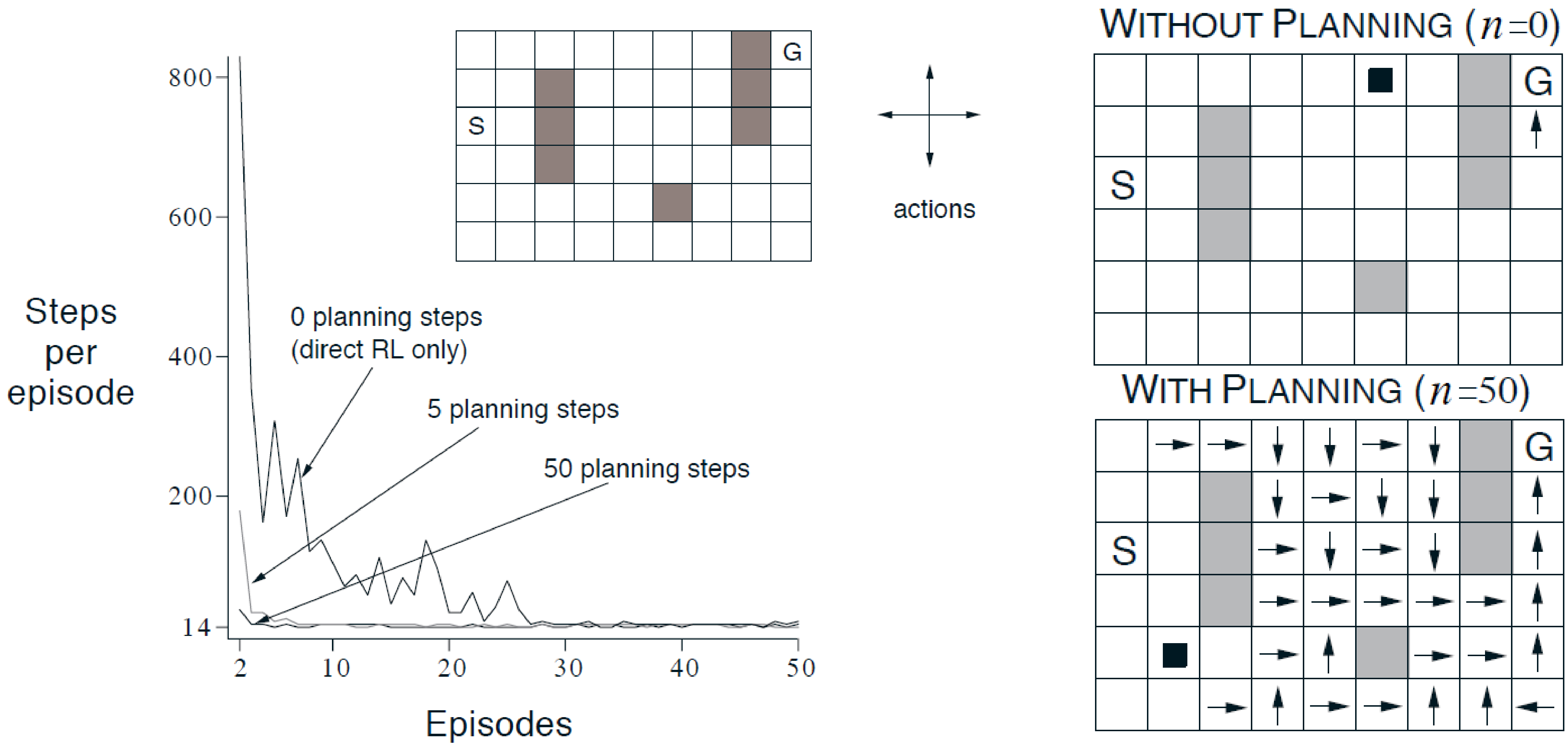
Dyna-Q agent in a maze



Dyna-Q agent in a maze



Dyna-Q agent in a maze



ELTE | FACULTY OF INFORMATICS

Thank you for your attention!