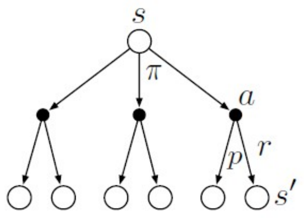
Exercise: Finite Markov Decision Process & Dynamic Programming

Task 1)
a) What does MDP abbreviate?
b) What are the properties of MDP states?
c) Why are MDPs useful for Reinforcement Learning?

Task 2)
Describe the Bellman Equation for v_π.
Note: Use the Backup Diagram.

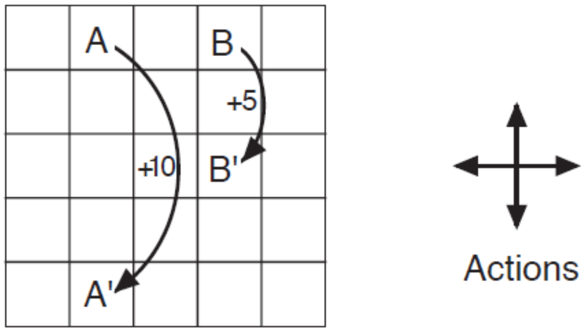


Backup diagram for v_π

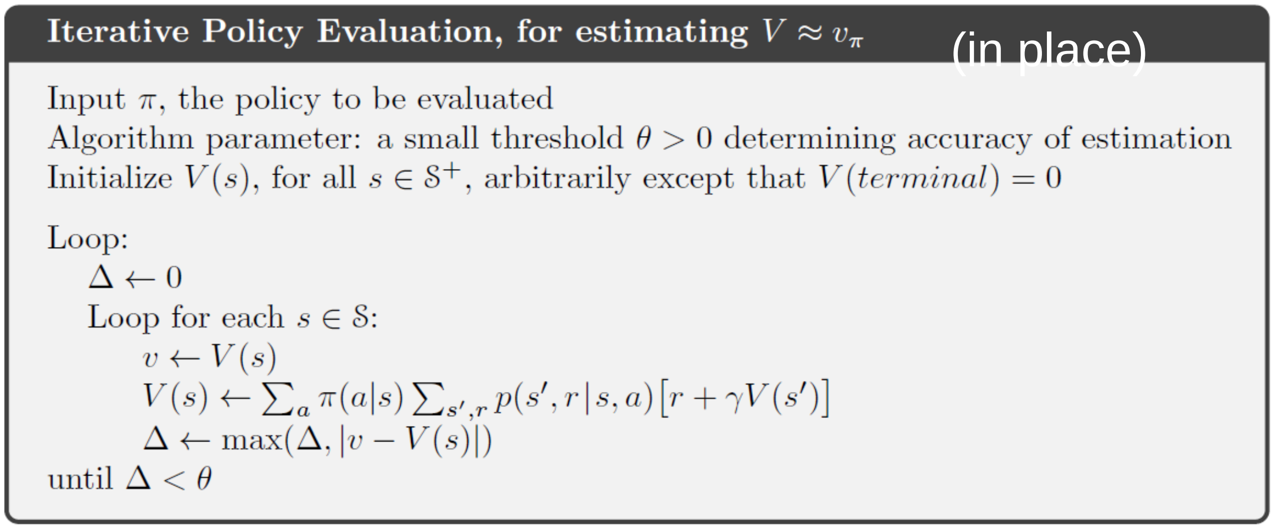
Task 3)
Suppose γ = 0.5 and the following sequence of rewards is received $R_1 = -1$,
$R_2 = 2$, $R_3 = 6$, $R_4 = 3$, and R5 = 2, with T = 5. What are G0, G1, . . . , G5?
Note: Work backwards & use

$$G_t \doteq R_{t+1} + \gamma G_{t+1}$$

Task 4)
User Iterative Policy Evaluation for estimating v_π given the following task and a policy picking a random action (left, right, up, down). The cells correspond to individual states.
Actions that would take the agent off the grid → Agent stays at this state but reward = -1.
All other cases reward=0, except any action from A brings the agents to A' and a reward of +10 is given and any action from B brings the agents to B' and a reward of +5 is given.
The discount rate is set to γ = 0.9.



a) use the in-place algorithm



```
Iterative Policy Evaluation, for estimating $V \approx v_\pi$        (in place)

Input $\pi$, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
      $v \leftarrow V(s)$
      $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$
```

b) use an alternative algorithm using two tables. One to store V_k and another to store V_{k+1}.
After time step *k+1* update tables: $V_{k+1} \rightarrow V_k$

c) which one converges faster (i. e. reaches threshold $\Theta = 0.1$ first).
Use the same initial conditions.
Make a plot: time step (or wall time) vs. Θ for both algorithms.

d) why is the value function at position *A* below 10 and the value function at position *B* above 5?

Bonus Task
Jack's car rental problem:
Jack manages two locations (A and B) for a nationwide car rental company.
The number of cars requested at a day is given by Poisson random variables, meaning that the probability that the number is *n* is given by:

$$\frac{\lambda^n}{n!} e^{-\lambda}$$

with $\lambda_A = 3$ and $\lambda_B = 4$.
Cars become available for renting the day after they are returned given also Poisson random variables with $\lambda_{A,\,return} = 3$ and $\lambda_{B,\,return} = 2$ for returns.

If Jack has a car available, he rents it out and is credited $10 by the national company.
If he is out of cars at that location, then the business is lost.

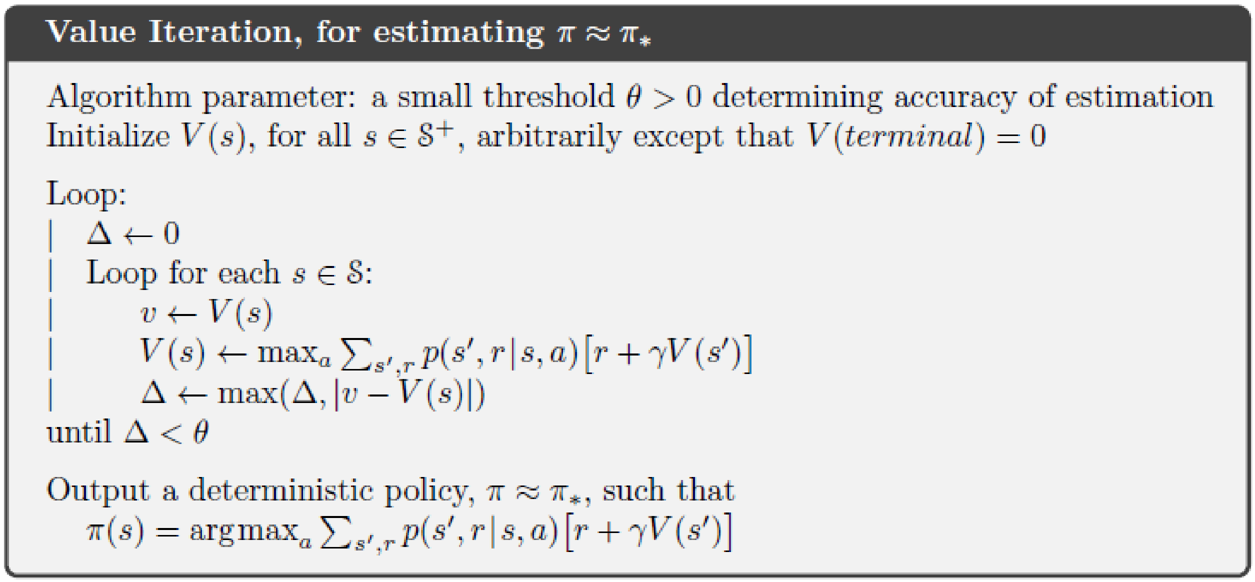
To help ensure that cars are available where they are needed, Jack can move them between the two locations overnight, at a cost of $2 per car moved.

Assume that there can be no more than 20 cars at each location (any additional cars are returned to the nationwide company, and thus disappear from the problem) and a maximum of five cars can be moved from one location to the other in one night.

We take the discount rate to be γ = 0.9 and formulate this as a continuing finite MDP, where the time steps are days, the state is the number of cars at each location at the end of the day, and the actions are the numbers of cars moved between the two locations overnight.

Use policy iteration or value iteration to solve the Jack's car rental problem of finding an optimal policy.

Note: We search for the number of cars moved between the two locations in the state space (number cars at location A, number of cars at location B).
Assume negative numbers indicate transfers from B → A and positive numbers transfers from A→ B.
Note: To speed up computation a tabular version of the Poisson distributions might be helpful.



Value Iteration, for estimating $\pi \approx \pi_*$

```
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
|  $\Delta \leftarrow 0$
|  Loop for each $s \in \mathcal{S}$:
|      $v \leftarrow V(s)$
|      $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
|      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
   $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
```