**Figure 21.4** A passive reinforcement learning agent that learns utility estimates using temporal differences. The step-size function $\alpha(n)$ is chosen to ensure convergence, as described in the text.

```plaintext
function PASSIVE-TD-AGENT(percept) returns an action
inputs: percept, a percept indicating the current state $s'$ and reward signal $r'$
persistent: $\pi$, a fixed policy
- $U$, a table of utilities, initially empty
- $N_s$, a table of frequencies for states, initially zero
- $s, a, r$, the previous state, action, and reward, initially null

if $s'$ is new then $U[s'] \leftarrow r'$
if $s$ is not null then
  increment $N_s[s]$
  $U[s] \leftarrow U[s] + \alpha(N_s[s])(r + \gamma U[s'] - U[s])$
if $s'$.TERMINAL? then $s, a, r \leftarrow$ null
else $s, a, r \leftarrow s', \pi[s'], r'$
return $a$
```

**Figure 21.8** An exploratory Q-learning agent. It is an active learner that learns the value $Q(s, a)$ of each action in each situation. It uses the same exploration function $f$ as the exploratory ADP agent, but avoids having to learn the transition model because the Q-value of a state can be related directly to those of its neighbors.

```plaintext
function Q-LEARNING-AGENT(percept) returns an action
inputs: percept, a percept indicating the current state $s'$ and reward signal $r'$
persistent: $Q$, a table of action values indexed by state and action, initially zero
- $N_{sa}$, a table of frequencies for state–action pairs, initially zero
- $s, a, r$, the previous state, action, and reward, initially null

if TERMINAL?(s) then $Q(s, None) \leftarrow r'$
if $s$ is not null then
  increment $N_{sa}[s, a]$
  $Q[s, a] \leftarrow Q[s, a] + \alpha(N_{sa}[s, a])(r + \gamma \max_{a'} Q[s', a'] - Q[s, a])$
  $s, a, r \leftarrow s', \arg\max_{a'} f(Q[s', a'], N_{sa}[s', a']), r'$
return $a$
```