

```

function [V] = rw_online_w_replacing_traces(lambda,alpha,n_rw_states,nEpisodes)
% RW_ONLINE_W_REPLACING_TRACES - Performs td(lambda) learning with {\em replacing}
% eligibility traces
%
% Written by:
% --
% John L. Weatherwax          2007-12-07
%
% email: wax@alum.mit.edu
%
% Please send comments and especially bug reports to the
% above email address.
%
%-----

% the number of episodes (in general this would need to be much larger to gaurentee
% complete learning):
%nEpisodes = 1000;

% the number of non terminal states plus two terminal states:
nStates=n_rw_states+2;

% initialize the states ... terminal states are defined to be ZERO
V = zeros(1,nStates);

% initialize the eligibility trace (to zero):
et = zeros(1,nStates);

gamma      = 1.0;
%gamma     = 0.9;

for ei=1:nEpisodes,

    % perform a random walk getting the sequence of rewards recieved and the
    % sequence of states we transition through ... this is the surragate of
    % performing actions with some policy
    %
    [rewseen,stateseen] = rw_episode(n_rw_states);

    % compute the temporal difference and elibility traces update for each
    % state s_t for t=0,1,2,\cdots,T-1 (the states visited)
    %
    % and use these to update V(\cdots):
    %
    T = length(rewseen);
    for t=0:(T-1),      % <- STATE index ...

        st      = stateseen(t+1);
        rew     = rewseen(t+1);
        stp     = stateseen(t+2);
        delta   = rew + gamma*V(stp) - V(st); % the temporal difference
        % implement REPLACING elagability traces:
        et(st) = 1;

        % update V(\cdots) with this information:
        V = V + alpha*delta*et;
        % update our elibility trace:
        et = gamma*lambda*et;
    end
end

```

```
    end % end state sequence loop ...  
end % end episode loop ...  
  
% remove the terminal states:  
V = V(2:end-1);
```