```
% DYNAQPLUS_MAZE_SCRIPT - Implements the DynaQ Algorithm on the simple maze example found
in Chapter 9
%
% Written by:
% --
% John L. Weatherwax                2007-12-03
%
% email: wax@alum.mit.edu
%
% Please send comments and especially bug reports to the
% above email address.
%
%-----

close all;
clc;

% sample_discrete.m
addpath( genpath( '../../../FullBNT-1.0.4/KPMstats/' ) );

% the learning rate:
%alpha = 1e-1;
alpha = 2e-1;

% the probability of a random action (non-greedy):
epsilon = 0.1;

% the discount factor:
gamma = 0.95;
%gamma = 1.0;

% get our initial maze (the blocking maze):
MZ = mk_ex_9_2_mz(0); [sideII,sideJJ] = size(MZ);

% the beginning and terminal states (in matrix notation):
s_start = [ 6, 4 ];
s_end   = [ 1, 9 ];

MAX_N_STEPS=30;
MAX_N_STEPS=3e3;
MAX_N_STEPS=3e4;

% the number of steps to do in planning:
nPlanningSteps = 0;
nPlanningSteps = 5;
nPlanningSteps = 50;

% a factor relating how important revisiting old states is, relative to
% the past recieved reward coming from these states/action pairs ...
%kappa = 0.02;
kappa = 2/sqrt(MAX_N_STEPS);

[Q,ets,cr] =
dynaQplus_maze(alpha,epsilon,gamma,kappa,nPlanningSteps,@mk_ex_9_2_mz,s_start,s_end,MAX_N
_STEPS);

% compute the (negative) cost to go and the optimal (greedy with respect to the state-
value function) policy:
pol_pi = zeros(sideII,sideJJ); V = zeros(sideII,sideJJ);
```

```
for ii=1:sideII,
  for jj=1:sideJJ,
    sti = sub2ind( [sideII,sideJJ], ii, jj );
    [V(ii,jj),pol_pi(ii,jj)] = max( Q(sti,:) );
  end
end

plot_mz_policy(pol_pi,mk_ex_9_2_mz(MAX_N_STEPS),s_start,s_end);
title( 'policy (1=>up,2=>down,3=>right,4=>left); start=green; stop=red' );
fn = sprintf('dynaQplus_maze_policy_nE_%d',MAX_N_STEPS); saveas( gcf, fn, 'png' );

figure; imagesc( V ); colormap(flipud(jet)); colorbar;
title( 'state value function' );
fn = sprintf('dynaQplus_maze_state_value_fn_nE_%d',MAX_N_STEPS); saveas( gcf, fn, 'png'
);

figure; plot( 1:length(ets), ets, '-x' );
title( 'number of timesteps to reach goal' ); grid on; drawnow;
fn = sprintf('dynaQplus_q_learning_rate_nPS_%d',nPlanningSteps); saveas( gcf, fn, 'png'
);

figure; plot( (1:MAX_N_STEPS), cr(2:end), '-x' );
title( 'cummulative reward' ); grid on; axis( [ 0 3000, 0, cr(3001)] );
xlabel('timestep index'); ylabel('cum. reward'); drawnow;
fn = sprintf('dynaQplus_q_cum_reward_nPS_%d',nPlanningSteps); saveas( gcf, fn, 'png' );

clear functions;
%close all;
return;
```