```
% Achyuth Nandikotkur
% V00975928
% ECE-559B
% October 30, 2021

% Question 2

clear;
clc;

global returns qvector searchrewards waitrewards startState
 actionsAtHigh actionsAtLow;

% To store average returns
% high - low
statevalues = [0 0];

searchrewards = [3, 4, 5, 6];
waitrewards = [0, 1, 2];

prob.high.search = 1/2;
prob.high.wait = 1/2;
prob.low.search = 1/4;
prob.low.wait = 1/2;
prob.low.recharge = 1/4;

loop = 1;
counter = [0 0 0 0 0];
% returns{0} high
% returns{1} low
returns = [0 0 0 0 0];

qvaluehighsearch = [0];
qvaluehighwait = [0];

qvaluelowsearch = [0];
qvaluelowwait = [0];
qvaluelowrecharge = [0];

while(loop < 1200)
    %since a four step episodic task,
    % example: high -> search -> high -> search ->
    %low -> wait -> low -> recharge -> high

    G = 0;
    sequence = generateEpisode(prob);
    for i = (length(sequence)-1): -1: 1
        G = G + sequence{i}.reward;

        skip = 0;
        for k = i-1: -1: 1
            if((sequence{i}.state == sequence{k}.state))
```

```
               skip = 1;
               break;
           end
       end

       % if the pair isn't seen in the values
       if(~skip)
           localreturns = [0, 0, 0, 0, 0];

           if(sequence{i}.state == 1)
               localreturns(sequence{i}.action) = G;
               returns = [returns; localreturns];
               counter(sequence{i}.action) =
counter(sequence{i}.action) + 1;

               if(sequence{i}.action == 1)
                   qvaluehighsearch = [qvaluehighsearch;
sum(returns(:, 1))/counter(1)];
                   qvaluehighwait = [qvaluehighwait;
qvaluehighwait(end, :)];
               else
                   qvaluehighwait = [qvaluehighwait; sum(returns(:,
2))/counter(2)];
                   qvaluehighsearch = [qvaluehighsearch;
qvaluehighsearch(end, :)];
               end

               A = [qvaluehighsearch(end) qvaluehighwait(end)];
               maxval = max(A);
               lia = ismember(A,maxval);
               idx = find(lia);

               probability = 1;
               pihigh = {0, 0};
               for i = 1:numel(idx)
                   pihigh{idx(i)} = (probability/numel(idx));
                   if(size(idx) == 1)
                       break;
                   end
               end

               prob.high.search = pihigh{1};
               prob.high.wait = pihigh{2};

           else
               localreturns(sequence{i}.action + 2) = G;
               returns = [returns; localreturns];
               counter(sequence{i}.action + 2) =
counter(sequence{i}.action + 2) + 1;

               if(sequence{i}.action == 1)
                   qvaluelowsearch = [qvaluelowsearch; sum(returns(:,
3))/counter(3)];
```

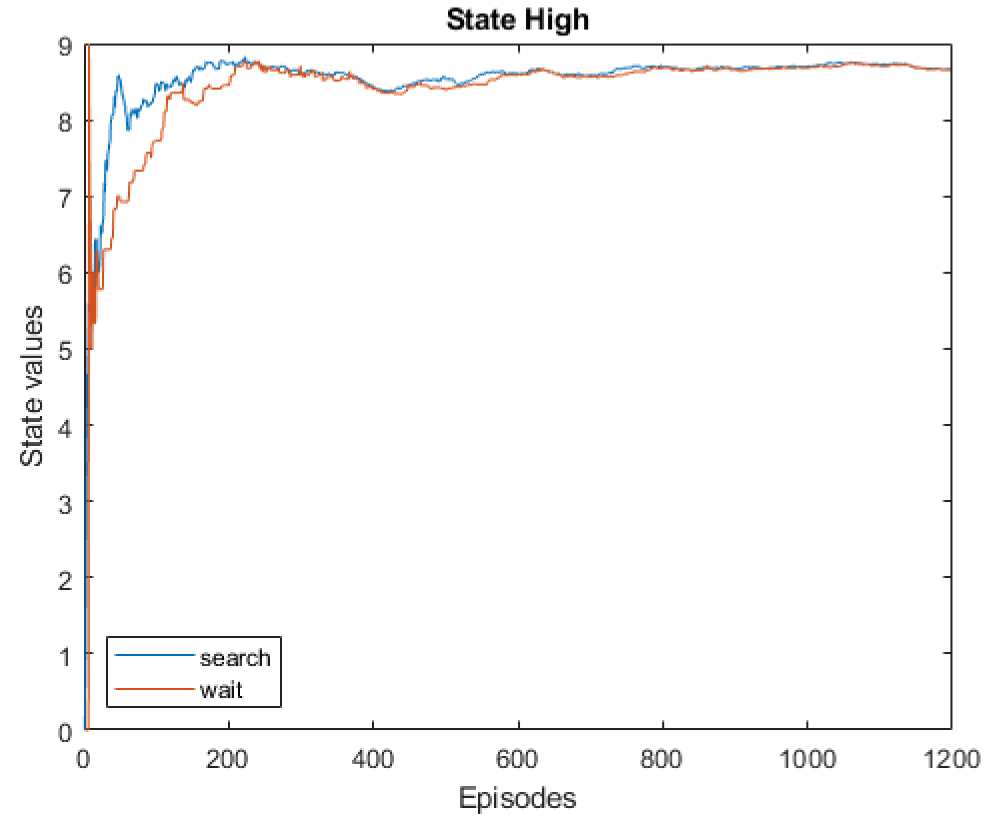
```
                    qvaluelowwait = [qvaluelowwait;
 qvaluelowwait(end, :)];
                    qvaluelowrecharge = [qvaluelowrecharge;
 qvaluelowrecharge(end, :)];
                elseif(sequence{i}.action == 2)
                    qvaluelowwait = [qvaluelowwait; sum(returns(:,
 4))/counter(4)];
                    qvaluelowsearch = [qvaluelowsearch;
 qvaluelowsearch(end, :)];
                    qvaluelowrecharge = [qvaluelowrecharge;
 qvaluelowrecharge(end, :)];
                else
                    qvaluelowrecharge = [qvaluelowrecharge;
 sum(returns(:, 5))/counter(5)];
                    qvaluelowsearch = [qvaluelowsearch;
 qvaluelowsearch(end, :)];
                    qvaluelowwait = [qvaluelowwait;
 qvaluelowwait(end, :)];
                end

                A = [qvaluelowsearch(end) qvaluelowwait(end)
 qvaluelowrecharge(end)];
                maxval = max(A);
                lia = ismember(A,maxval);
                idx = find(lia);

                probability = 1;
                pilow = {0, 0, 0};
                for i = 1:numel(idx)
                    pilow{idx(i)} = (probability/numel(idx));
                    if(size(idx) == 1)
                        break;
                    end
                end

                prob.low.search = pilow{1};
                prob.low.wait = pilow{2};
                prob.low.recharge = pilow{3};
            end
        end
    end

    loop = loop + 1;
end

celldisp(sequence);

t1=1:length(qvaluehighsearch);
t2=1:length(qvaluehighwait);
t3=1:length(qvaluelowsearch);
t4=1:length(qvaluelowwait);
t5=1:length(qvaluelowrecharge);

figure(1)
```

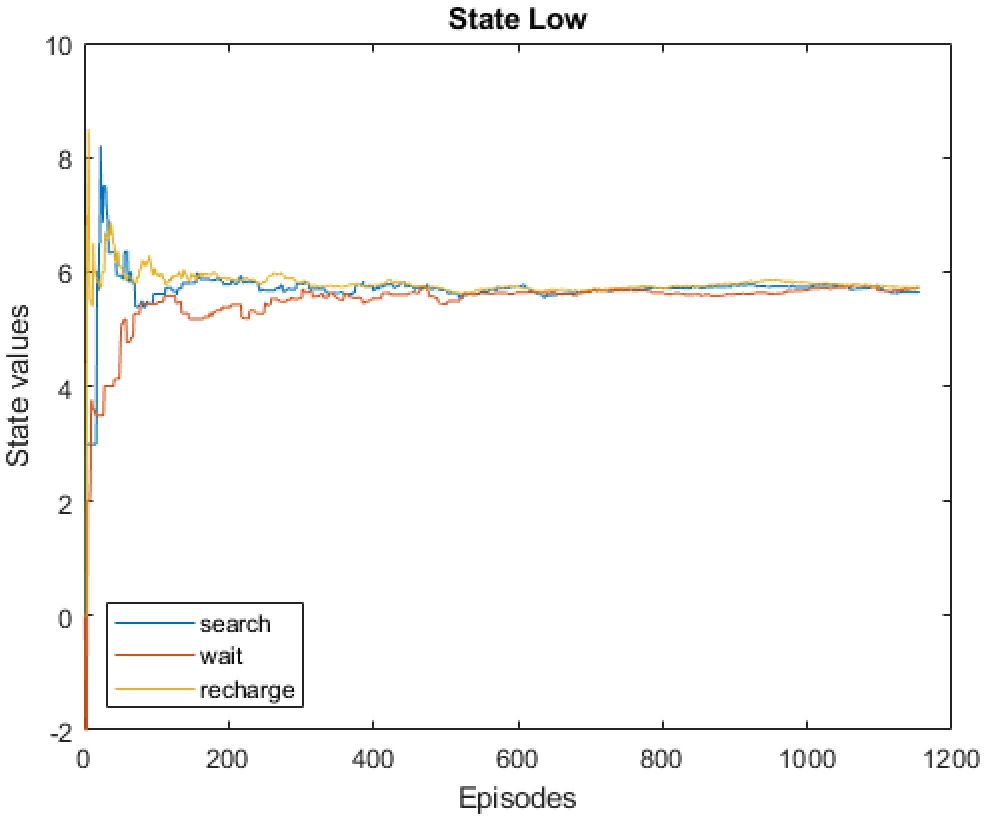
```
plot(t1,qvaluehighsearch, t2,qvaluehighwait);
xlabel('Episodes')
ylabel('State values')
legend({'search','wait'},'Location','southwest')
title('State High');

figure(2)
plot(t3, qvaluelowsearch, t4, qvaluelowwait, t5, qvaluelowrecharge);
xlabel('Episodes')
ylabel('State values')
legend({'search','wait', 'recharge'},'Location','southwest')
title('State Low');


function resp = generateEpisode(probability)
    global waitrewards searchrewards;
    sequence = cell(1, 4);

    sequence{1}.state = 0;
    sequence{1}.action = 0;
    sequence{1}.reward = 0;

    sequence{2}.state = 0;
    sequence{2}.action = 0;
    sequence{2}.reward = 0;

    sequence{3}.state = 0;
    sequence{3}.action = 0;
    sequence{3}.reward = 0;

    sequence{4}.state = 0;
    sequence{4}.action = 0;
    sequence{4}.reward = 0;

    sequence{5}.state = 0;
    sequence{5}.action = 0;
    sequence{5}.reward = 0;

    % selecting initial state as high = 1 or low = 2 with equal
 probability
    initialstate = randsample([1, 2], 1, true, [0.5, 0.5]);
    sequence{1}.state = initialstate;
    % check if state is high or low
    if(sequence{1}.state == 1)
        sequence{1}.action = randsample([1, 2], 1, true, [0.5, 0.5]);
        if sequence{1}.action == 1
          sequence{2}.state = randsample([1, 2], 1, true, [0.25,
 0.75]);
          sequence{1}.reward =  randsample(searchrewards,1);
        else
          sequence{1}.reward  =  randsample(waitrewards,1);
          sequence{2}.state = 1;
        end
    else
```

```
       % state is low
       sequence{1}.action = randsample([1, 2, 3], 1, true, [1/3, 1/3,
1/3]);
       if sequence{1}.action == 1
         sequence{2}.state = randsample([2, 1], 1, true, [0.25,
0.75]);
         if(sequence{2}.state == 2)
             sequence{1}.reward =  randsample(searchrewards,1);
         else
             sequence{1}.reward = -3;
         end
       elseif(sequence{1}.action == 2)
         sequence{1}.reward =  randsample(waitrewards,1);
         sequence{2}.state = 2;
       else
          sequence{1}.reward = 0;
          sequence{2}.state = 1;
       end
   end

   for i=2:4
       % check if state is high or low
       if(sequence{i}.state == 1)
           % action can be search = 1, wait = 2;
           action = randsample([1, 2], 1, true,
[probability.high.search, probability.high.wait]);
           if action == 1
             nextstate = randsample([1, 2], 1, true, [0.25, 0.75]);
             reward =  randsample(searchrewards,1);
           else
             reward =  randsample(waitrewards,1);
             nextstate = 1;
           end
       else
           % state is low
           % generate action with input probabilities
           action = randsample([1, 2, 3], 1, true,
[probability.low.search, probability.low.wait,
probability.low.recharge]);

           if action == 1
             nextstate = randsample([2, 1], 1, true, [0.25, 0.75]);
             if(nextstate == 2)
                 reward =  randsample(searchrewards,1);
             else
                 reward = -3;
             end
           elseif(action == 2)
             reward =  randsample(waitrewards,1);
             nextstate = 2;
           else
              reward = 0;
              nextstate = 1;
           end
```

```
        end
        sequence{i}.action = action;
        sequence{i}.reward = reward;
        sequence{i+1}.state = nextstate;
    end
    resp = sequence;
end
```

```
sequence{1} =

     state: 1
    action: 1
    reward: 4



sequence{2} =

     state: 2
    action: 3
    reward: 0



sequence{3} =

     state: 1
    action: 1
    reward: 6



sequence{4} =

     state: 1
    action: 1
    reward: 4



sequence{5} =

     state: 2
    action: 0
    reward: 0
```


State High
State values
Episodes
0
1
2
3
4
5
6
7
8
9
0
200
400
600
800
1000
1200
search
wait


State Low
State values
Episodes
-2
0
2
4
6
8
10
0
200
400
600
800
1000
1200
search
wait
recharge

Published with MATLAB® R2021a