**Problem 1.**

*Solution:*

1. State space: - your advantage at the moment

Control space:

Transition probabilities:

For all other states , we have ;

Reward function for one period:.

Dynamic programming equations: The value function expresses the probability to win at time if the state (score) is.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 0 |  |  |  |  |  | any |  |  |  |  |
| 0 | 0 | 0 |  |  |  |  | any | any |  |  |  |
| 0 | 0 | 0.1013 | 0.104 |  |  |  | any | bold | timid |  |  |
| 0 | 0.225 | 0.23 | 0.3038 | 0.3089 |  |  | bold | timid | bold | timid |  |
| 0.50 | 0.50 | 0.5513 | 0.555 | 0.5643 | 0.5684 | timid | timid | bold | timid | bold | timid |
| 1 | 0.95 | 0.91 | 0.8826 | 0.8603 |  |  | timid | timid | timid | timid |  |
| 1 | 1 | 0.995 | 0.987 |  |  |  | any | timid | timid |  |  |
| 1 | 1 | 1 |  |  |  |  | any | any |  |  |  |
| 1 | 1 |  |  |  |  |  | any |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |

Code:



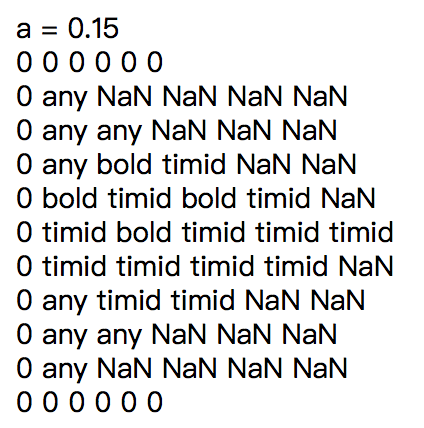
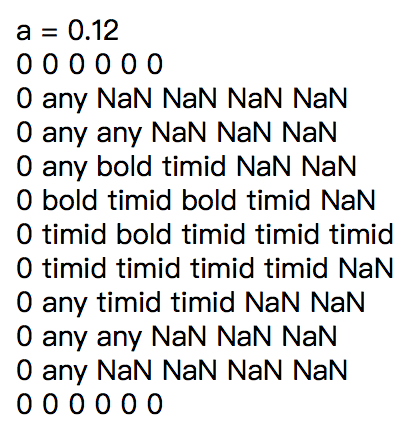
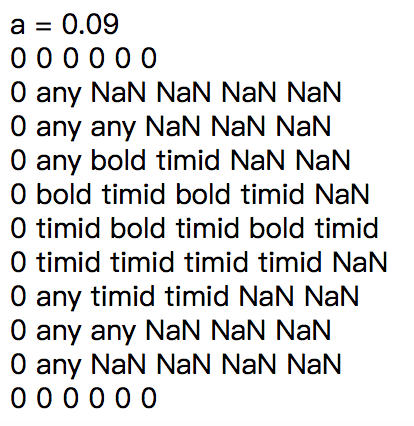
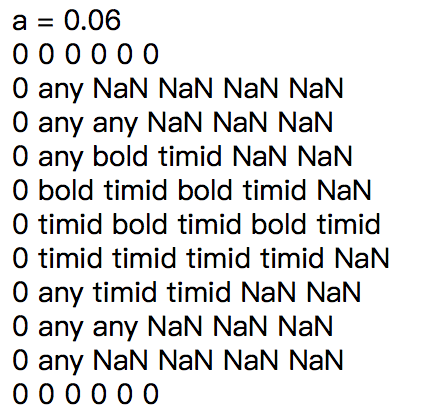
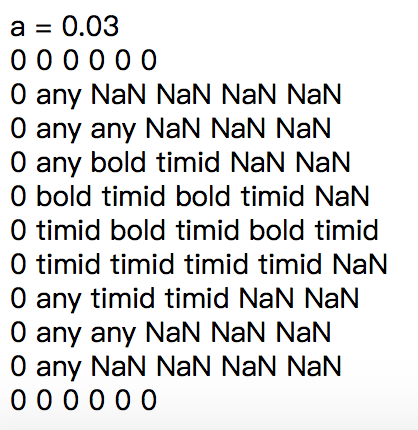
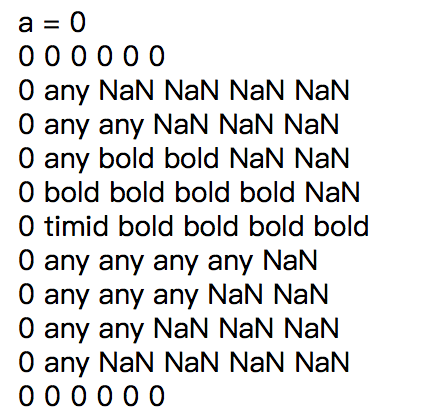
 

1. Let the probability of wining in the case of a timid play be 0, 0.3, 0.6, 0.9, 0.12, 0.15, we can see the higher probability of wining, the more we choose to use timid way.

Code:

Output:



**Problem 2.**

*Solution:*

1. State space: – the size of tree at that moment

Control space:

Transition probabilities:

For all other states , we have ;

Reward function for one period:

Dynamic programming equations: The value function expresses the biggest profit at time if the state (score) is.

1. I choose value iteration and policy iterations to solve this problem.

**Value iterations:**

I got in 4 iterations.

MATLAB Code:

% [v\_lo,n\_it] = dne1\_value\_iteration\_revised (0.9,10000);

function [v\_lo,n\_it] = dne1\_value\_iteration\_revised (alpha,max\_it)

i = 0;

n\_it = max\_it;

v=[0,0,0,0,0];

vv=[0,0,0,0,0];

v\_lo=[0,0,0,0,0];

v\_up=[0,0,0,0,0];

while (i < n\_it)

vv(1) = max(-10+alpha\*v(1),-30);

vv(2) = max(-11+alpha\*(0.05\*v(1)+0.15\*v(2)+0.7\*v(3)+0.1\*v(4)),115);

vv(3) = max(-12+alpha\*(0.05\*v(1)+0.2\*v(3)+0.7\*v(4)+0.05\*v(5)),140);

vv(4) = max(-13+alpha\*(0.05\*v(1)+0.5\*v(4)+0.45\*v(5)),165);

vv(5) = max(-14+alpha\*(0.05\*v(1)+0.95\*v(5)),210);

v\_lo = vv + min(vv-v)\*alpha/(1-alpha);

v\_up = vv + max(vv-v)\*alpha/(1-alpha);

if (isequal(v,vv))

n\_it=i;

end

i=i+1;

v(1)=vv(1);

v(2)=vv(2);

v(3)=vv(3);

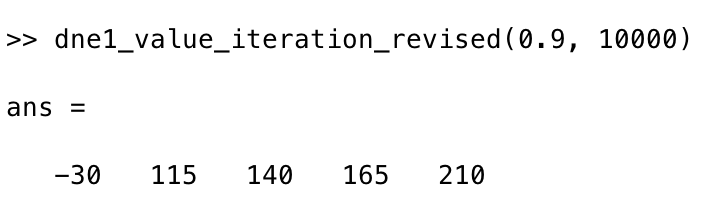
v(4)=vv(4);

v(5)=vv(5);

end

end

Output:



**Policy iterations:**

, is the solution of the system of equations

So

The policy maximize is , the is the solution of the system of equations. So the policy is , we have

, .