

Assignment #2

Sudhanshu Pathak

Q1:

NOTE: I have used Java for programming. Please run program in following format:

```
java MDPPlanningAlgo filePath
```

Assumed values :

```
epsilon = 0.0001;
```

Q2:

MDP1.txt

```
Epsilon          :: 0.00010
```

```
Discount factor :: 0.1
```

```
-----
```

```
-----
```

Value Function Matrix

```
-----
```

```
[0.10003 ]
```

```
[0.00894 ]
```

```
[0.00874 ]
```

```
[0.00894 ]
```

```
[1.00100 ]
```

```
[0.00682 ]
```

```
[0.06826 ]
```

```
[0.00856 ]
```

```
[0.01000 ]
```

```
[0.08956 ]
```

```
Policy Matrix
```

```
-----
```

```
-----
```

[3]

[3]

[2]

[0]

[0]

[0]

[1]

[2]

[1]

[3]

MDP2.txt:

Epsilon :: 0.00010
Discount factor :: 0.1

Value Function Matrix

[0.01144]

[0.01003]

[0.57325]

[0.00147]

[0.06040]

[0.10100]

[1.01010]

[0.00796]

[0.00604]

[0.10100]

Policy Matrix

[3]

[0]

[0]

[1]

[2]

[2]

[2]

[3]

[1]

[2]

MDP1.txt

Epsilon :: 0.00010

Discount factor :: 0.9

Value Function Matrix

[3.32100]

[2.92332]

[2.89132]

[2.92332]

[3.69001]

[2.84071]

[3.15635]

[2.90711]

[2.98890]

[3.24813]

Policy Matrix

[3]

[3]

```

[2 ]
[0 ]
[0 ]
[0 ]
[1 ]
[2 ]
[1 ]
[3 ]

```

MDP2.txt:

```

Epsilon      :: 0.00010
Discount factor :: 0.9

```

```

-----
Value Function Matrix
-----

```

```

[4.26311 ]
[4.25760 ]
[5.18847 ]
[3.84398 ]
[4.41682 ]
[4.73680 ]
[5.26311 ]
[3.83506 ]
[3.97513 ]
[4.73679 ]

```

```

Policy Matrix
-----

```

```

[0 ]
[0 ]
[0 ]

```

[1]

[2]

[2]

[2]

[1]

[1]

[2]

Q3:

For Parking MDP I have considered following assumption:

Implementation is divided into following parts:

```
.populateDriveMatrix();
.populateParkMatrix();
.populateExitMatrix();
.calculateReward();
```

Firstly, the no of states are considered as 81 x 81 due to Location {A,B}, O {T,F}, P {T,F} combinations. Each states is represented as “A1,T,F” similarly there would be 80 states and one last for EXIT state.

The objective of the implementation is to calculate three matrices mainly Drive, Park and Exit.

Drive:

Assumptions: Towards A1 or B1 the probability increases. Towards Bn probability decreases.

The cyclic order is considered to generate matrix as given in question. There is a specific set of transition from A1 to B1 and B10 to A10.

Park:

The action Park on incoming state if Park is false then probability is set to 1.

Exit:

The action Exit on incoming state if Park is true then Probability is set to 1.

Reward:

Reward is calculated in the following rule as mentioned in question:

```
if (park.equals("F")){
    reward.add(-1.0);
}else if (park.equals("T")){

    if(ocup.equals("T")){
        reward.add(-100.0);
    } else if (no == 1){
        reward.add(50.0);
    } else{
        reward.add(1000.0/no);
    }
}
```

Output MDP:

81 3

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible][illegible]

[illegible]

[illegible]

[illegible][illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
-100.0 -1.0 50.0 -1.0 -100.0 -1.0 50.0 -1.0 -100.0 -1.0 500.0 -1.0 -
100.0 -1.0 500.0 -1.0 -100.0 -1.0 333.33333333333333 -1.0 -100.0 -1.0
333.33333333333333 -1.0 -100.0 -1.0 250.0 -1.0 -100.0 -1.0 250.0 -1.0 -
100.0 -1.0 200.0 -1.0 -100.0 -1.0 200.0 -1.0 -100.0 -1.0
166.66666666666666 -1.0 -100.0 -1.0 166.66666666666666 -1.0 -100.0 -1.0
142.85714285714286 -1.0 -100.0 -1.0 142.85714285714286 -1.0 -100.0 -1.0
125.0 -1.0 -100.0 -1.0 125.0 -1.0 -100.0 -1.0 111.11111111111111 -1.0 -
100.0 -1.0 111.11111111111111 -1.0 -100.0 -1.0 100.0 -1.0 -100.0 -1.0
100.0 -1.0
```