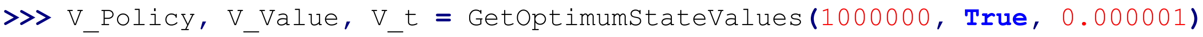
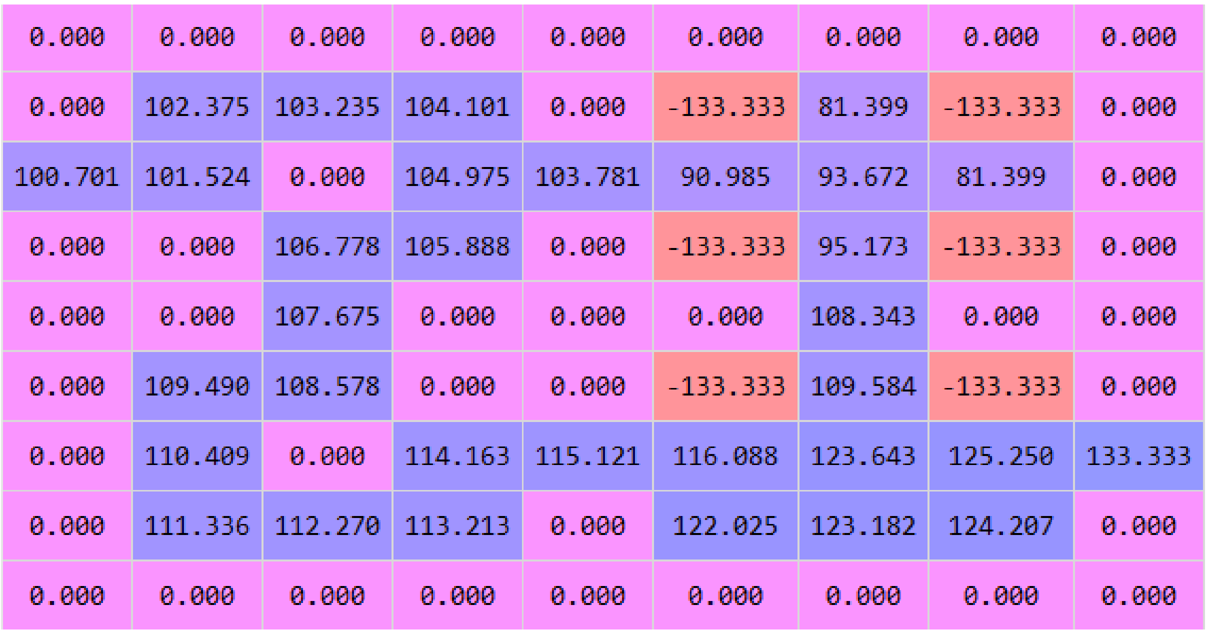
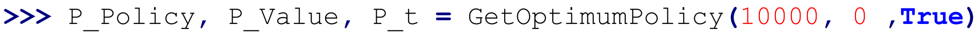


```
>>> V_Policy, V_Value, V_t = GetOptimumStateValues(1000000, True, 0.000001)
```

Grid's State Values using Value iteration. (epsilon = .000001, Initial Value = 0, Converged after 1836 iteration)

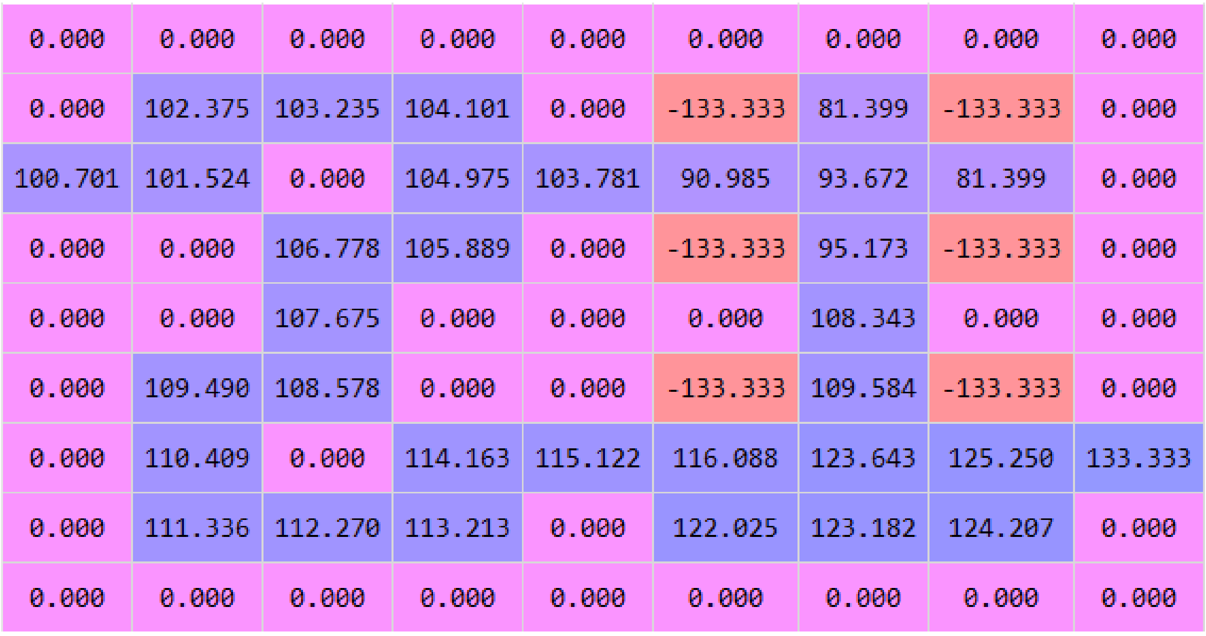


0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	102.375	103.235	104.101	0.000	-133.333	81.399	-133.333	0.000
100.701	101.524	0.000	104.975	103.781	90.985	93.672	81.399	0.000
0.000	0.000	106.778	105.888	0.000	-133.333	95.173	-133.333	0.000
0.000	0.000	107.675	0.000	0.000	0.000	108.343	0.000	0.000
0.000	109.490	108.578	0.000	0.000	-133.333	109.584	-133.333	0.000
0.000	110.409	0.000	114.163	115.121	116.088	123.643	125.250	133.333
0.000	111.336	112.270	113.213	0.000	122.025	123.182	124.207	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000



```
>>> P_Policy, P_Value, P_t = GetOptimumPolicy(10000, 0 ,True)
```

Grid's State Values using Policy iteration (epsilon = .000001, Initial Policy = 0 i.e. 'WEST' for every state, Converged after 5 iteration)



0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	102.375	103.235	104.101	0.000	-133.333	81.399	-133.333	0.000
100.701	101.524	0.000	104.975	103.781	90.985	93.672	81.399	0.000
0.000	0.000	106.778	105.889	0.000	-133.333	95.173	-133.333	0.000
0.000	0.000	107.675	0.000	0.000	0.000	108.343	0.000	0.000
0.000	109.490	108.578	0.000	0.000	-133.333	109.584	-133.333	0.000
0.000	110.409	0.000	114.163	115.122	116.088	123.643	125.250	133.333
0.000	111.336	112.270	113.213	0.000	122.025	123.182	124.207	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

The State Values computed in policy iteration and value iterations converged to same value for each block.

c) i) Starting with the initial policy that points EAST in every state, it takes 5 iterations for convergence. (epsilon = 0.000001)

ii) Starting with the initial policy that points SOUTH in every state, it takes 11 iterations for convergence. (epsilon = 0.000001)

```
#Source
"""
Created on Sat Nov 19 21:47:01 2016
@author: gopal
"""
import numpy as np
#P(S'={1..81} |S={1..81},a={1,2,3,4})
states = 81; actions = 4; gamma = 0.9925
#P[action][PreviousState][NextState]
def LoadInput():
    P = np.zeros(shape=(actions, states, states))
    for i in range(actions):
        Pa_raw = np.loadtxt('prob_a' + str(i+1) + '.txt')
        for item in Pa_raw:
            P[i][item[0]-1][item[1]-1] = item[2]
    R = np.loadtxt('rewards.txt')
    return P, R

P, R = LoadInput()
#StateValue(S) = R(S) + gamma* SumOverS_dash(P[policy(S)][S][S_dash] *
StateValue(S_dash))
def EvaluatePolicy(Policy):
    I = np.identity(states, dtype=int)
    P_policy = np.zeros(shape = (states, states));
    for S in range(states):
        for S_dash in range(states):
            P_policy[S][S_dash] = P[Policy[S]][S][S_dash]
    return np.linalg.solve(I - gamma*P_policy, R)

def GetBetterPolicy(Policy):
    V = EvaluatePolicy(Policy);
    Policy_better = np.argmax([np.inner(P[i][:][:], V) for i in range(actions)],
axis=0)
    return Policy_better

def GetOptimumPolicy(T, startPolicy = 0, stopAtConvergence = False, epsilon=0):
    Policy = np.zeros(states) + startPolicy
    for t in range(T):
        betterPolicy = GetBetterPolicy(Policy)
        if stopAtConvergence and np.count_nonzero(betterPolicy - Policy) == epsilon:
            Policy = betterPolicy
            break;
        Policy = betterPolicy
    return Policy, EvaluatePolicy(Policy), t

def GetBetterStateValues(V):
    V_better = R + gamma*np.max([np.inner(P[i][:][:], V) for i in range(actions)],
axis=0)
    return V_better

def GetOptimumStateValues(T, stopAtConvergence = False, epsilon=0):
    Value = np.zeros(states)
    for t in range(T):
        betterValue = GetBetterStateValues(Value)
        if stopAtConvergence and np.max(np.abs(betterValue - Value)) <= epsilon:
            Value = betterValue
            break;
        Value = betterValue
    Policy_star = np.argmax([np.inner(P[i][:][:], Value) for i in range(actions)],
axis=0)
    return Policy_star, Value, t
```