```
import numpy as np
from collections import defaultdict
from PIL import Image, ImageDraw

def get_reward(fname):
    reward_list = np.zeros((81, 1), dtype = int)
    i = 0
    for l in open(fname):
        reward_list[i][0] = int(l)
        i += 1
    return reward_list

def get_transistion_matrix(fname):
    action = np.zeros((81, 81), dtype = float)
    for l in open(fname):
        l = l.split()
        action[int(l[0]) - 1, int(l[1]) - 1] = float(l[2])
    return action

def get_max(action, state, v_k):
    max_value = -10000.00
    max_action = -1
    for i in range(4):
        temp = 0.0
        for k in range(81):
            temp += action[i][state][k] * v_k[k]
        if temp > max_value:
            max_value = temp
            max_action = i
    return max_value, max_action

def get_identity_matrix(n):
    identity = np.zeros((n, n), dtype = int)
    for i in range(n):
        identity[i][i] = 1
    return identity

def get_transion_matrix_policy(action, pi_):
    transition_matrix = np.zeros((81, 81), dtype = float)
    #s = i, pi(s) = pi_[i],
    for i in range(81):
        a = pi_[i]
        transition_matrix[i] = action[a][i]
    return transition_matrix

def policy_optimization(action, v, type, rewards):
    pi_ = [0] * 81
    identity_matrix = get_identity_matrix(81)
    for k in range(30):
        if type == 'value_iteration':
            old_v = list(v)
        else:
            transition_matrix = get_transion_matrix_policy(action, pi_)
            old_v = np.matrix(identity_matrix - 0.99 * transition_matrix).I *
                rewards
        for i in range(81):
```

```
                #print i, get_max(action, i, old_v)
                max_value, max_action = get_max(action, i, old_v)
                v[i] = rewards[i][0] + 0.99 * max_value
                pi_[i] = max_action
        return v, pi_

    def draw_arrow(policy, value):
        im = Image.open('mazePlain-page-001.jpg')
        draw = ImageDraw.Draw(im)
        for i in range(81):
            row, col = i % 9, i / 9
            center = [length + col * length * 2, length + row * length * 2]
            if value[i][0][0] > np.float(0) and i != 78:
                line, arrow1, arrow2 = get_coordinate(policy[i], center)
                draw.line(line, fill = 128)
                draw.line(arrow1, fill = 128)
                draw.line(arrow2, fill = 128)
        im.save('mazePlain-page-002.jpg')

    def get_coordinate(policy, center):
        line = []
        arrow1 = []
        arrow2 = []
        if policy == 0:
            left, right = center[0] - length + 20, center[0] + length - 20
            line = [left, center[1], right, center[1]]
            arrow1 = [left, center[1], left + 20, center[1] - 20]
            arrow2 = [left, center[1], left + 20, center[1] + 20]
        elif policy == 2:
            left, right = center[0] - length + 20, center[0] + length - 20
            line = [left, center[1], right, center[1]]
            arrow1 = [right, center[1], right - 20, center[1] - 20]
            arrow2 = [right, center[1], right - 20, center[1] + 20]
        elif policy == 1:
            top, bottom = center[1] - length + 20, center[1] + length - 20
            line = [center[0], top, center[0], bottom]
            arrow1 = [center[0], top, center[0] - 20, top + 20]
            arrow2 = [center[0], top, center[0] + 20, top + 20]
        elif policy == 3:
            top, bottom = center[1] - length + 20, center[1] + length - 20
            line = [center[0], top, center[0], bottom]
            arrow1 = [center[0], bottom, center[0] - 20, bottom - 20]
            arrow2 = [center[0], bottom, center[0] + 20, bottom - 20]
        return line, arrow1, arrow2

    rewards = get_reward('rewards.txt')
    action1 = get_transistion_matrix('prob_a1.txt')
    action2 = get_transistion_matrix('prob_a2.txt')
    action3 = get_transistion_matrix('prob_a3.txt')
    action4 = get_transistion_matrix('prob_a4.txt')
    action = [action1, action2, action3, action4]
    # v stores v_k+1
    v = [0.0] * 81
    value, pi_policy_iteration = policy_optimization(action, v, 'policy_iteration',
        rewards)
    value, pi_value_iteration = policy_optimization(action, v, 'value_iteration',
```

```
    rewards)
print pi_value_iteration, value

length = 70
draw_arrow(pi_policy_iteration, value)
```