ME491 : Programming Exercise 1

Due Date : Wednesday, 15 September 2021, 5 p.m.

1 Introduction

The goal of this programming exercise is to obtain an optimal policy and the corresponding value function using dynamic programming (DP). You will use value iteration and policy iteration to solve two problems. Here are some general rules of the programming and results which have to be submitted:

- You have to use Python.
- The 19 states are numbered as shown in the Figure 1. The output numpy arrays (i.e., the value and

policy functions, path) must follow this convention. For example, using default settings(from Daejeon to Seoul)

– Optimal value, 19 values for state: [0.0, -22.0, -55.0, -79.0, -139.0, -183.0, -88.5, -130.1, -209.09, -197.65, -281.88, -224.09, -238.18, -296.36, -308.7, -293.81, -258.68, -310.36, -290.36]

– Optimal policy, next state to go: [7, 0, 0, 0, 0, 0, 2, 3, 7, 6, 9, 7, 8, 12, 10, 16, 11, 12, 12]

– Optimal path: [7, 3, 0]

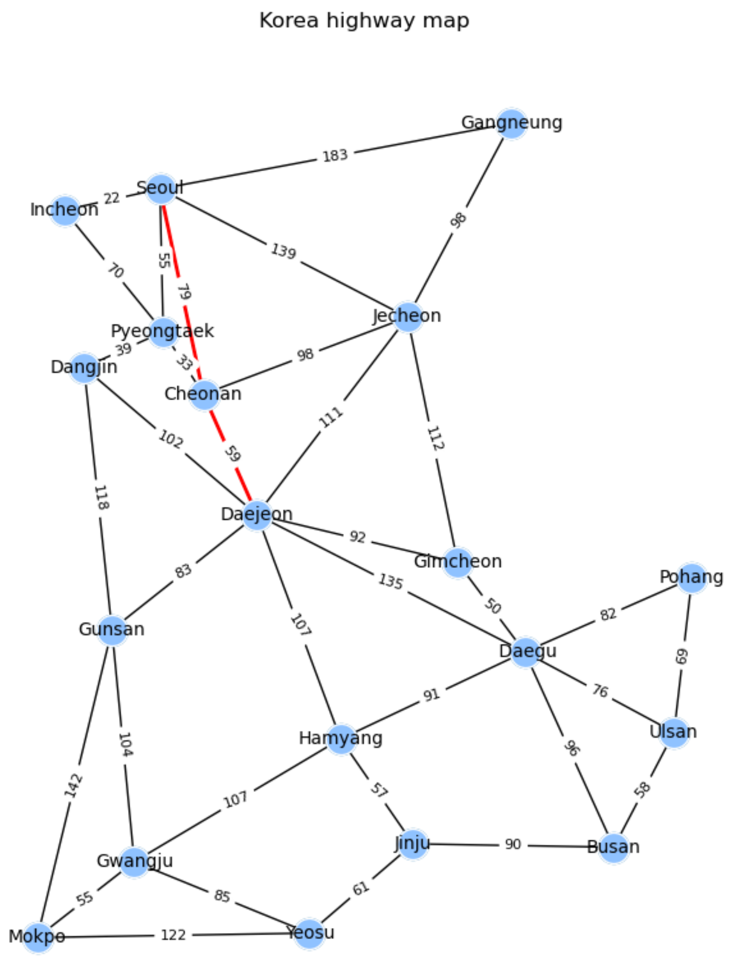


Figure1. Example path using value iteration (Daejeon to Seoul)

2 Instruction

A code template of a Python file is shown in Figure 2

• The given python script gets arguments from user input you can use -d and -t to set the departure node and destination node (You can check the list of cities using prob1_value_iter.py -h)

• The given .csv file has an adjacency matrix of weighted graph which is the distance from a node to node and can be a negative reward to go through it.

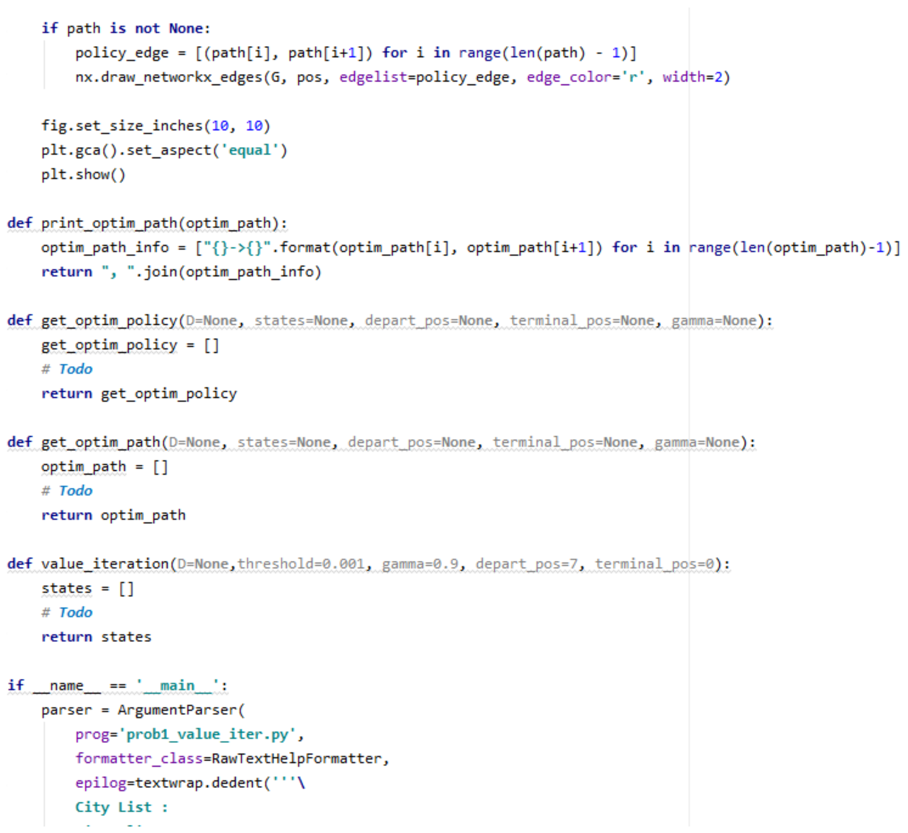
• The path should start from the user-defined departure node and end at the terminal node.

• A discount factor is 0.9 and threshold is 0.001

• You should write code of 3 functions; 'get_optim_value' which gives value functions of all states using value iteration, 'get_ optim_policy' which gives the optimal policy of calculated value function, and 'get_optim_path' which gives optimal path using optimal policy. You should not change the arguments for those 3 functions

• You will get scores for optimal value, optimal policy, and optimal path.

• Evaluation will use this kind of command, 'prob1_value_iter.py -d 1 -t 17' so check that command works.



```
    if path is not None:
        policy_edge = [(path[i], path[i+1]) for i in range(len(path) - 1)]
        nx.draw_networkx_edges(G, pos, edgelist=policy_edge, edge_color='r', width=2)

    fig.set_size_inches(10, 10)
    plt.gca().set_aspect('equal')
    plt.show()

def print_optim_path(optim_path):
    optim_path_info = ["{}->{}".format(optim_path[i], optim_path[i+1]) for i in range(len(optim_path)-1)]
    return ", ".join(optim_path_info)

def get_optim_policy(D=None, states=None, depart_pos=None, terminal_pos=None, gamma=None):
    get_optim_policy = []
    # Todo
    return get_optim_policy

def get_optim_path(D=None, states=None, depart_pos=None, terminal_pos=None, gamma=None):
    optim_path = []
    # Todo
    return optim_path

def value_iteration(D=None,threshold=0.001, gamma=0.9, depart_pos=7, terminal_pos=0):
    states = []
    # Todo
    return states

if __name__ == '__main__':
    parser = ArgumentParser(
        prog='prob1_value_iter.py',
        formatter_class=RawTextHelpFormatter,
        epilog=textwrap.dedent('''\
        City List :
```

Figure2. Code Template

Submit a zip file, consisting of 1 file

• The name of the zip file should be '(student number)_(name).zip', and the 1 Python file name should be the same as below. An example of the final file is:

20200000_yourname.zip

└ prob1_value_iter.py ('value_iteration 'get_optim_policy' 'get_optim_path')