Justine John "JJ" A. Serdoncillo **EE 5571: Reinforcement Learning and Dynamic Programming**
December 4, 2023 **Monday**

Salesman Rollout!

Code

To implement the Rollout algorithm with a base heuristic of nearest neighbors, at each iteration, a set of paths to each possible city and then a nearest neighbors until the very end is simulated. From all of these paths calculated, the initial next city chosen with the shortest distance is added to the actual path. The code is seen in the appendix but here is the output of the total distance so far and the final path.

```
Current totDist = 2
Current totDist = 7
Current totDist = 8
Current totDist = 11
Current totDist = 17
Current totDist = 18
Current totDist = 27
Current totDist = 34
Current totDist = 40
Current totDist = 47
Current totDist = 59
Current totDist = 70
Current totDist = 83
Current totDist = 86
Current totDist = 90
Current totDist = 94
Current totDist = 96
Current totDist = 108
Current totDist = 122
Current totDist = 129
This is the distance traveled: 129
This is the salesman path: [0, 7, 20, 12, 8, 16, 17, 13, 11, 9, 19,
    10, 1, 15, 4, 3, 6, 14, 18, 5, 2]
```

This is interesting because if we used just the nearest neighbor method, we get a total distance of 139 which is not as good as the rollout method.

Appendix

Python

```python
'''
Justine Serdoncillo
IE 5571 - Dynamic Programming
HW 4 Exercise 6.10
December 5, 2023
'''

"""
Consider the distance matrix provided in the file HW6Data.
Your goal is to find a route of the Traveling Salesperson for this matrix assuming you start and finish your tour at node 1.
In a tour you can only visit each city once, and the tour is complete once you have visited all cities.
Your goal is to minimize your total distance traveled.
You will use rollout with a base policy of nearest neighbor to find your route.
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import math

# %% Inputs and Functions
filePath = 'HW6Data.csv'

def evalTravel(evalTravel):
    evalTravel.append(0)
    totDist = 0
    for i in range(21):
        totDist += dist_matrix[evalTravel[i], evalTravel[i+1]]
    return totDist

def nearNeigh(numNodes, posNodes, currNode, currList, totDist):

    for j in range(numNodes):
        currMin = 1E5 # current minimum
        currNodeMin = 0
        for i in posNodes:
            if dist_matrix[currNode, i] <= currMin:
                currMin = dist_matrix[currNode, i]
                currNodeMin = i
        currList.append(currNodeMin)
        totDist += currMin
        posNodes = np.setdiff1d(np.copy(posNodes),[currNodeMin])
        currNode = np.copy(currNodeMin)

    return totDist

def cycle(numNodes):
    global posNodes, currNode, currList, totDist

    for j in range(numNodes):
        currMin = 1E5 # current minimum
        currNodeMin = 0
        for i in posNodes:
            dist = dist_matrix[currNode, i] + dist_matrix[i, 0]
            if dist <= currMin:
                currMin = dist_matrix[currNode, i]
                currNodeMin = i
        currList.append(currNodeMin)
        totDist += currMin
        posNodes = np.setdiff1d(np.copy(posNodes),[currNodeMin])
        currNode = np.copy(currNodeMin)

def rollOut(numNodes, posNodes, currNode, currList, totDist):

    for j in range(numNodes):
        score = np.zeros(len(posNodes))
        num = 0
        for i in posNodes:
            posCopy = np.setdiff1d(np.copy(posNodes),[i])
            currCopy = currList + [i]
            totCopy = totDist + dist_matrix[currNode, i]
            score[num] = nearNeigh(numNodes-1, posCopy, i, currCopy, totCopy)
            num += 1

        currNodeMin = posNodes[np.argmin(score)]
        currList.append(currNodeMin)
        totDist += dist_matrix[currNode, currNodeMin]
        posNodes = np.setdiff1d(np.copy(posNodes),[currNodeMin])
        currNode = np.copy(currNodeMin)
        print(f"Current totDist = {totDist}")

    return totDist

# %%
df = pd.read_csv(filePath)
dist_matrix = df.values
baseScores = np.zeros(20)
posNodes = np.arange(21) #list of possible nodes initially available
posNodes = np.setdiff1d(posNodes,[0]) #remove 0
currNode = 0 # updating node location
currList = [0] # updating list
totDist = 0 # total distance
base = 0

#cycle(20) # totDist = 487
#nearNeigh(20) # totDist = 157
totDist += rollOut(20, posNodes, currNode, currList, totDist)

for i in range(21):
    currList[i] += 1

totDist += dist_matrix[currNode, 0]

print(f"This is the distance traveled: {totDist}")
print(f"This is the salesman path: {currList}")
```