Optimization Method HW7

Student: Ang Zhou

Coding Part:

```
from gurobipy import *
import pandas as pd

# loading data
features = ['FromNode', 'ToNode', 'cost']
df = pd.read_csv('shortest_path_data.txt', names = features, delimiter = " ", skiprows = 1)

# create a new model
myModel = Model("HW7_Q1")

# create decision vats and integrate them into the model
fromNodes = df['FromNode']
toNodes = df['ToNode']
arcs = df['cost']
cost = [[0.0 for i in range (9)] for j in range (9)]
myVars = [[0 for i in range (9)] for j in range (9)]
for index in range (len(arcs)):
   i = fromNodes[index]
   j = toNodes[index]
   cost[i][j] = arcs[index]
   curVar = myModel.addVar( vtype = GRB.CONTINUOUS, name = "x" + str(i) + str(j))
   myVars[i][j] = curVar
myModel.update()

# create a linear expression for the objective
objExpr = LinExpr()
for i in range (1,9):
   for j in range (1,9):
      curVar = myVars[i][j]
      objExpr += cost[i][j] * curVar
myModel.setObjective(objExpr, GRB.MINIMIZE)

# Constraint for Node 1 and Node 8
constExpr = LinExpr()
for j in range (1, 9):
   curVar = myVars[1][j]
   constExpr += 1 * curVar
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = 1, name = "Node_1")
constExpr = LinExpr()
```

```
for j in range (1, 9):
   curVar = myVars[j][8]
   constExpr += -1 * curVar
myModel.addConstr(lhs = constExpr, sense = GRB.EQUAL, rhs = -1, name = "Node_8")

# Constraint for Node 2 ~ Node 7
for i in range (2,8):
   constExpr = LinExpr()
   for j in range (1,9):
      constExpr += myVars[i][j] - myVars[j][i]
   myModel.addConstr( lhs = constExpr , sense = GRB.EQUAL , rhs = 0 , name = "Node_"+str(i))

# integrate objective and constraints into the model
myModel.update()

# write the model in a file to make sure it is constructed correctly
myModel.write(filename = "HW7_Q1.lp")

# optimize the model
myModel.optimize()

# print optimal objective and optimal solution
print("\nOptimal Objective: " + str(myModel.ObjVal))
print("\nOptimal Solution:")
allVars = myModel.getVars()
for curVar in allVars:
   print (curVar.varName + " " + str(curVar.x))
```

The Output Result:

```
Optimize a model with 8 rows, 16 columns and 32 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 8e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 2 rows and 2 columns
Presolve time: 0.01s
Presolved: 6 rows, 14 columns, 28 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0   3.9920000e+00   2.004000e+00   0.000000e+00      0s
       3   8.0000000e+00   0.000000e+00   0.000000e+00      0s
```

Solved in 3 iterations and 0.03 seconds
Optimal objective 8.000000000e+00

Optimal Objective: 8.0

Optimal Solution:
x12 0.0
x13 1.0
x23 0.0
x24 0.0
x25 0.0
x34 0.0
x35 1.0
x36 0.0
x45 0.0
x46 0.0
x47 0.0
x56 1.0
x57 0.0
x67 0.0
x68 1.0
x78 0.0