CSE 101 Summer Session 2 2022
Programming Assignment 5

INSTRUCTIONS: Solutions to this assignment must be submitted through Gradescope by Saturday, September 3 2022 at 10:00 PM— no late submissions will be accepted.

Note: All necessary pseudocode may be found on the lecture slides.

1. **CVRP (Saving's Heuristic)**

   Recall the *Capacitated Vehicle Routing* problem: find a set of minimum total cost tours for a fleet of capacitated vehicles based at a single depot, to serve a set of customers under the following constraints:

   (a) each tour begins and ends at the depot,

   (b) each customer is visited exactly once, and

   (c) the total demand of each tour does not exceed the capacity of the vehicle.

   Suppose that G is a complete undirected acyclic graph with adjacency matrix $A \in \mathbb{R}^{n \times n}$. Let $S \in \mathbb{R}^{n \times n}$ be the table of shortest distances between nodes (we will provide this to you in each problem instance, although it could be found using Floyd-Warshall's algorithm from A).

   Let $Q \in \mathbb{R}n = [Q_{Total}, Q_A, Q_B, \ldots]$, where Q_{Total} is the total capacity and each subsequent Q_j is the demand of customer j.

   Implement CVRP by writing **four** different functions:

   (a) `mySavings(S)`: The matrix with the Savings-values

   (b) `myCVRP(S, Q)`: The list/vector of tours along with their respective demands and lengths. Each tour is represented as a tuple with the following three elements:

      i. The list/vector of all vertices in the tour in the correct order (except the depot).
      Each vertex in the graph corresponds to their position in the alphabet (A -> 1, B -> 2, and so on). For example, the tour O - A - E - O would correspond to [1, 5], if O is considered as the depot.

      ii. Total demand of the tour

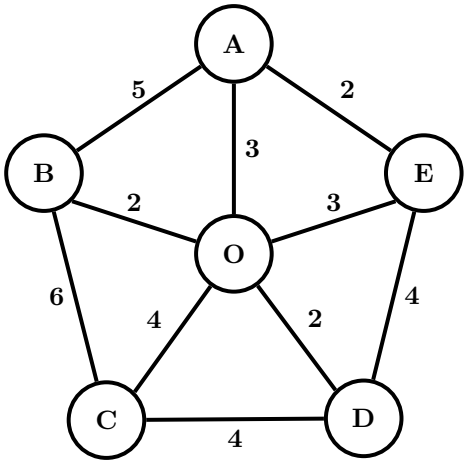
      iii. Total length of the tour

      NOTE: The order of the tours in this vector/list does not matter and you will receive full points as long as all the tours are present in this vector/list.

   (c) `myRoundTripLength(S, Q)`: Sum of the lengths of all roundtrips before CVRP.

   (d) `mySavingsLength(S, Q)`: Sum of the lengths of all tours after CVRP.

For example, suppose that we are given the following graph, where the table gives the shortest distance between nodes (where "O" denotes the depot):



	O	A	B	C	D	E
O	0	3	2	4	2	3
A		0	5	7	5	2
B			0	6	4	5
C				0	4	7
D					0	4
E						0

$$\mathrm{A} = \begin{bmatrix} 0 & 3 & 2 & 4 & 2 & 3 \\ 3 & 0 & 5 & \infty & \infty & 2 \\ 2 & 5 & 0 & 6 & \infty & \infty \\ 4 & \infty & 6 & 0 & 4 & \infty \\ 2 & \infty & \infty & 4 & 0 & 4 \\ 3 & 2 & \infty & \infty & 4 & 0 \end{bmatrix}$$

We would input shortest-paths matrix and the capacities to our procedure as

$$S = \begin{bmatrix} 0 & 3 & 2 & 4 & 2 & 3 \\ 0 & 0 & 5 & 7 & 5 & 2 \\ 0 & 0 & 0 & 6 & 4 & 5 \\ 0 & 0 & 0 & 0 & 4 & 7 \\ 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad Q = \begin{bmatrix} 90 \\ 30 \\ 10 \\ 20 \\ 10 \\ 40 \end{bmatrix}$$

Solution to the *CVRP* for the above S and Q:

```
The matrix with the Savings-values:
    0 0 0 0 0 0
    0 0 0 0 0 4
    0 0 0 0 0 0
    0 0 0 0 2 0
    0 0 0 0 0 1
    0 0 0 0 0 0

Tours:
    0  -  A  -  E  -  0 (Demand: 70, Length: 8)
    0  -  C  -  D  -  0 (Demand: 30, Length: 10)
    0 - B - 0 (Demand 10, Length: 4)

Total length of roundtrips before CVRP: 28

Total length of all tours after CVRP using the Savings's Heuristic: 22
```

`mySavings(S)` would return the savings-values matrix:

```
  0 0 0 0 0 0
  0 0 0 0 0 4
  0 0 0 0 0 0
  0 0 0 0 2 0
  0 0 0 0 0 1
  0 0 0 0 0 0
```

`myCVRP(S, Q)` would return a vector (C++)/list (Python) containing each tour, along with its demand and length.

```
[ ( [1,5], 70,  8),
  ( [3,4], 30, 10),
  ( [2],   10,  4) ]
```

`myRoundTripLength(S)` would return the total length of roundtrips before CVRP: **28**

`mySavingsLength(S)` would return the total length of all tours after CVRP: **22**

2. **(Linear Programming (Simplex))** In this problem, you will implement the Simplex algorithm to solve linear programs of the following form:

$$\text{maximize } \sum_{i=1}^{n} c_i x_i \text{ subject to } \begin{cases} \sum_{i=1}^{n} a_{ji} x_i \leq b_j & \text{for all } 1 \leq j \leq m \\ x_i \geq 0 & \text{for all } 1 \leq i \leq n \end{cases},$$

where a_{ij}'s, b_j's, and c_i's are integer coefficients with n variables and m constraints, and where variable x_i may be rational. This program can be written succinctly in matrix form:

$$\text{maximize } C \cdot X \text{ subject to } \begin{cases} AX \leq B \\ X \geq 0 \end{cases}, \text{ where}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}, \; B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{bmatrix}, \; C = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{bmatrix}, \; X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}.$$

For example, consider the "Ducks and Trains" problem from lecture.

$$\text{maximize } 3x_1 + 2x_2 \text{ subject to } \begin{cases} 2x_1 + x_2 \leq 100 \\ x_1 + x_2 \leq 80 \\ x_1 \leq 40 \\ x_1, x_2 \geq 0 \end{cases}$$

The matrix form of the problem consists of the following matrices:

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \; B = \begin{bmatrix} 100 \\ 80 \\ 40 \end{bmatrix}, \; C = \begin{bmatrix} 3 \\ 2 \end{bmatrix}.$$

Your task: Write a program called `mySimplexLP(A, B, C)` that takes as input the matrix form of a linear program:

- A, an $m \times n$ array of integers
- B, an m-item list of integers
- C, an n-item list of integers,

Your output should be in the form of a tuple (optimal, slack, value)

- optimal: array of length n with optimal values for values for $x_1, ..., x_n$
- slack: array of length m with the slack variable values for $s_1, ..., s_m$
- value: objective value of the optimal solution

We will only use feasible and bounded linear programs.

Using the "Ducks and Trains" example again, your code might return the following:

Input:

```
A = [
  [2,1],
  [1,1],
  [1,0]
]
B = [100,80,40]
C = [3,2]

mySimplexLP(A,B,C)
```

Output:

```
([20, 60], [0,0,20], 180)
```