DSO-570 Mock Final

Version D

Instructions:

Complete this Jupyter notebook and submit it on Blackboard as Exercise 13.1 in Week 13 (due Thursday 12/2 by 9am).

For the exam, you can consult any textbooks, notes, or online material. However, you are **NOT ALLOWED TO:**

- ask anyone other than the professor for help.
- share your exam or any part of your work with anyone else other than the professor.
- communicate with another student during the exam in any way, including phone calls, text messages, emails, chats, or any other communication.

Please do not violate academic integrity! A documented violation will result in a failing grade of F and leave an inerasable mark on your record. Employers will not hire a cheater. It's not worth it!

Please type your name below to certify that you have adhered to all university policies regarding ethical behavior in preparing for and completing this exam.

Name:

Q1. Airport Transportation Service (Concrete Formulation; 10 points)

Pedro owns a family business providing rides to and from the airport. His primary vehicle is a large van, which he operates himself. He owns a second vehicle, a small sedan, which is operated by his wife. Ride requests that are not fulfilled by him or his wife are passed on to his network of business partners. He has many business partners and you can assume that he can find available business partners to pick up every customer if needed.

Pedro would like to compute an optimal plan for fulfilling ride requests so as to maximize the total profit earned by him and his wife. (His business does not profit from requests passed on to his business partners.) The following table summarizes the ride requests his business received for a given morning.

Customer	Van Needed	Pickup Time	Pickup Location	Dropoff Location	Profit (dollars)
Anne	Yes	09:00	Suburb	Airport	60
Beth	No	10:00	Downtown	Airport	30
Cui	No	10:30	Airport	Suburb	50
Dwane	No	11:00	Airport	Downtown	35
Edna	No	11:30	Downtown	Airport	30
Frank	No	12:00	Suburb	Airport	55

Below are the travel times (in minutes) between each pair of locations. You can assume that these estimates are conservative, so that regardless of traffic conditions, one can always travel between any spot in the Airport and any spot in Downtown within 30 minutes, and between any spot in the Airport and any spot in the Surburb within 45 minutes, and so on.

Origin/Destination	Airport	Downtown	Suburb
Airport	0	30	45
Downtown	30	0	30
Suburb	45	30	0

Each customer is only requesting one car, so if Pedro picks up a customer, then his wife can be simultaneously fulfilling another request. Moreover, a ride needs to be completed before the same driver can pickup another customer. A plan for fulfilling ride requests also needs to satisfy the following:

- A request with a "Yes" in the "Van Needed" column can be assigned to Pedro, but it cannot be assigned to his wife.
- Each assigned driver needs to be able to arrive on time to each pickup location, based on the travel times above. For example, the same driver cannot pickup both Anne and Beth, because if the driver departs from the Suburb at 09:00, he/she would drop off Anne at the Airport at 09:45. If the driver immediately proceed to pick up Beth, he/she would arrive at Downtown at 10:15, which is later than the 10:00 pickup time requested by Beth. On the other hand, the same driver can pick up both Beth and Cui, because if the driver pickups Beth at Downtown at 10:00, he/she would drop her off at the airport at exactly 10:30, which is the perfect time to pickup Cui. You should examine the tables above to infer all such scheduling constraints. When doing so, you just have to make sure that after dropping off the previous customer, the driver can arrive on time at the next pickup location assigned to him or her, according to the driving time estimates.

Moreover, you should not plan for extra margins: the driving times can be thought of as already incorporating everything needed for a trip, such as finding the passenger within the given location, loading or unloading luggage, and collecting payments.

A final consideration is as follows: that morning, his wife has other things she would like to attend to, so there is a large opportunity cost on her time. Pedro estimates that having his wife drive that morning is equivalent to incurring a fixed cost of 90 dollars, whereas if he does not assign any rides to her, he can save this cost. **This cost needs to be accounted for in the total profit being maximized.**

Write a linear optimization formulation for the above problem. You only need to write a concrete formulation, so you don't need any data variables but can simply plug in the numbers above. However, you cannot just solve the problem by hand and write down the optimal solution. In particular, if the "Profit (dollars)" column changes to other numbers, then one should be able to simply update the corresponding numbers in your formulation, and it should still be correct.

Decision Variables:

Objective:

Constraints:

Q2. Pooling in Ride-Hailing (Abstract Formulation; 10 points)

This problem asks you to write a linear optimization formulation to solve a simplified version of the pooling problem faced by ride-hailing companies such as Uber and Lyft. These companies provide discounts to riders who are willing to let their trips be "pooled" with another trip, so that a driver would pick up both riders before dropping off each of them at their respective destinations. Through pooling, the travel time of each rider would increase, but driver capacity is more efficiently utilized. This practice is only profitable if the company finds good matches between trips so as to maximize the total benefit of pooling.

For simplicity, assume that each trip can be pooled with at most one other trip. The ride-hailing company has estimated the benefit of pooling for each pair of trips, which accounts for the potential cost savings from pooling and the potential inconveniences incurred for the pooled customers. As an illustration, suppose there are six trips, labelled A through F. The following table shows what the benefit values may look like.

Benefit of Pooling	A	B	C	D	E	F
A	0	6	4	3	1	1
B	6	0	5	5	2	3
C	4	5	0	1	4	3
D	3	5	1	0	2	1
E	1	2	4	2	0	4
F	1	3	3	1	4	0

Note that the table is symmetric, which means that it remains the same if it is transposed. In other words, pooling A-B is the same as pooling B-A; their order does not matter. As an illustration of how to use the table, suppose that trips B and C are pooled together, and trips E and F are pooled together, while trip A and trip D are not pooled; then the total benefit is $5 + 4 = 9$. As another example, suppose that trips A and B are pooled together, and none of the other trips are pooled; then the total benefit is 6.

The ride-hailing company would like to pool trips so as to maximize the total benefit of pooled trips subject to the following constraints:

- Each trip can be pooled with at most one other trip. (It is also possible for a trip to be not pooled, as in the examples above.)
- For a pair of trips, if the benefit of pooling is strictly less than a threshold t, then the trips cannot be pooled. For example, if $t = 3$, then the above table implies that A-D is a valid pooling, but B-E is not.
- The total number of pooled pairs is at most k. In the above example, if $t = 0$ and $k = 3$, then the optimal solution is to pool A-C, B-D, and E-F, which yields a total benefit of $4 + 5 + 4 = 13$. However, if $t = 0$ and $k = 1$, then the optimal solution is to only pool A-B, which yields a benefit of 6.

Your abstract formulation needs to be able to correctly handle arbitrarily many trips, arbitrary benefits of pooling, and arbitrary values of t and k. You may assume that the table is always symmetric as above, with zeros on the diagonal and non-negative entries everywhere else. Moreover, you may assume that t is always a non-negative number, and k is always a positive integer.

Data:

Decision Variables:

Objective:

Constraints:

Q3. Strategic Driving (Gurobi coding; 8 points)

A driver in a ride-hailing platform may increase earnings by strategizing about where to go next, rather than simply following the App. For example, at certain times in the day, waiting at Downtown for ride requests may lead to short trips that don't pay well. An alternative strategy might be to move to the Airport in hope of getting assigned to a long trip from there. Of course, whether this strategy is actually better depends on the expected earnings of each trip, the demand patterns at each location, and the amount of time left before the driver intends to go home.

The following linear optimization formulation solves a simplified version of the above optimization problem for a single driver. The formulation assumes that the time needed to travel between each pair of locations is one time period. (It is possible to modify the formulation to account for realistic driving times, but that would be outside the scope of this exam.)

Data:

- L: the set of locations.
- s: the driver's initial location at time 0.
- n: the number of time periods to plan. The driver needs to go home at time n.
- $T = \{0, 1, 2, \cdots, n\}$: the set of time periods.
- $r_{i,j}$: the expected earnings of a trip from location $i \in L$ to $j \in L$. Note that it is possible that $i = j$, as this can correspond to short trips within a local region.
- $p_{i,j}$: the probability that if a driver turns on the App at location $i \in L$ in a given time period, he/she will receive a ride request from location i to location j in the same time period.

Decision Variables:

- $x_{i,t}$: the earnings-to-go if the driver is at location $i \in L$ at time $t \in T$. Earnings-to-go is defined as the total expected earning from time t until time n, assuming optimal behavior by the driver.

Objective and Constraints:

$$\begin{aligned}
\text{Minimize:} \quad & x_{s,0} \\
\text{s.t.:} \quad & \\
& x_{i,t} \geq x_{j,t+1} && \text{for each } i \in L, j \in L, \text{ and } t \in \{0, 1, \cdots, n-1\}. \\
& x_{i,t} \geq \sum_{j \in L} p_{i,j}(r_{i,j} + x_{j,t+1}) && \text{for each } i \in L \text{ and } t \in \{0, 1, \cdots, n-1\}. \\
& x_{i,t} \geq 0 && \text{for each } i \in L \text{ and } t \in T.
\end{aligned}$$

An explanation of the above formulation is outside the scope of this exam and is not necessary for solving this problem. You should simply trust that the formulation works and implement it using Gurobi.

Write a function called "plan_route" with three input arguments:

- **inputFile**: the filename to the input file. You can assume that the input file is an Excel file of the same format as the "Q3-input1.xlsx" and "Q3-input2.xlsx" files provided.
- **s**: a string representing the parameter s in the formulation above.
- **n**: a positive integer representing the parameter n in the formulation above.

As you can see in the two sample input files, each input file has two sheets. The first encodes the earnings $r_{i,j}$. The second encodes the probabilities $p_{i,j}$. In each table, the row corresponds to the first index, which is the origin, and the column corresponds to the second index, which is the destination. These tables are not symmetric. For example, in "Q3-input1.xlsx", $r_{Downtown,Airport} = 60$ while $r_{Airport,Downtown} = 70$. In other words, a ride from Downtown to Airport earns 60, whereas a trip from the Airport to Downtown earns 70. Similarly, in "Q3-input1.xlsx", $p_{Downtown,Airport} = 0.05$ while $p_{Airport,Downtown} = 1$. This reflects the situation in which ride requests from Downtown are primarily local, and only 5 percent are going to the Airport. On the other hand, ride requests from the Airport are all travelling long distances to Downtown. "Q3-input2.xlsx" contains a different dataset with four locations instead of two. Your code needs to be able to handle arbitrary input data of the same format.

Your function should return two objects:

- **total_earnings**: a float corresponding to the optimal objective value of the above formulation, **rounded to two decimal places.**
- **earnings_table**: a DataFrame where the rows are the locations $i \in S$, and the columns are $t \in \{1, 2, \cdots, n\}$. Each entry contains the corresponding optimal value of $x_{i,t}$, **rounded to two decimal places.**

See sample runs 1, 2 and 3 for illustration of the format of the returned objects. After you get these to work, you can run sample runs 4, which processes the results of each of the other sample runs and show the implied optimal strategy of the driver at every location and time period. For example, in sample run 1, the optimal strategy is to begin at time 0 by following the App. Then, if the driver finds himself/herself at Downtown at time periods 1, 3, or 5, the driver should turn the App off and move to the airport. Under all other circumstances, the driver should follow the App. At the last period, the driver goes home.

Write all of your final code in the following cell, so that if one restarts Kernel and only runs the following cell as well as the cell containing "obtain_movement," all of the sample runs would work.

```
[3]: # Write all of your final code here
```

```
[4]: # Sample run 1
     total_earnings,earnings_table=plan_route('Q3-input1.xlsx','Airport',7)
     print('Optimal earnings starting at Airport:',total_earnings)
     earnings_table

Optimal earnings starting at Airport: 280.0
```

```
                1       2       3      4     5     6  7
Downtown      210  153.16     140  83.12    70  12.5  0
Airport    223.16     210  153.12    140  82.5    70  0
```

```
[5]: # Sample run 2
     total_earnings,earnings_table=plan_route('Q3-input1.xlsx','Downtown',7)
     print('Optimal earnings starting at Downtown:',total_earnings)
     earnings_table

Optimal earnings starting at Downtown: 223.16
```

```
                1       2       3      4     5     6  7
Downtown      210  153.16     140  83.12    70  12.5  0
Airport    223.16     210  153.12    140  82.5    70  0
```

```
[6]: # Sample run 3
     total_earnings,earnings_table=plan_route('Q3-input2.xlsx','Suburb A',6)
     print('Optimal earnings starting at Suburb A:',total_earnings)
     earnings_table

Optimal earnings starting at Suburb A: 206.22
```

```
                1       2       3      4     5  6
Downtown   169.86  134.48    96.4     64    16  0
Airport    206.22  169.86  134.48   96.4    64  0
Suburb A   169.86  134.48   97.98     64  27.5  0
Suburb B   173.22   136.7  101.48     64    31  0
```

```
[7]: # Sample run 4
    def obtain_movement(total_earnings,earnings_table):
        found=False
        for i in earnings_table.index:
            t=earnings_table.columns[0]
            if total_earnings==earnings_table.loc[i,t]:
                print(f'Start by leaving App off and moving to {i}')
                found=True
                break
        if not found:
            print(f'Start by following the App')
        movements=earnings_table.copy()
        for i in earnings_table.index:
            for t in earnings_table.columns:
                if t+1 in earnings_table.columns:
                    movements.loc[i,t]='Follow App'
                    for j in earnings_table.index:
                        if earnings_table.loc[i,t]==earnings_table.loc[j,t+1]:
                            movements.loc[i,t]=f'Turn App off and move to {j}'
                            break
                else:
                    movements.loc[i,t]='Go home'
        return movements
    print('Optimal Strategy for Sample Run 1: ',end='')
    total_earnings,earnings_table=plan_route('Q3-input1.xlsx','Airport',7)
    display(obtain_movement(total_earnings,earnings_table))
    print('\n\nOptimal Strategy for Sample Run 2: ',end='')
    total_earnings,earnings_table=plan_route('Q3-input1.xlsx','Downtown',7)
    display(obtain_movement(total_earnings,earnings_table))
    print('\n\nOptimal Strategy for Sample Run 3: ',end='')
    total_earnings,earnings_table=plan_route('Q3-input2.xlsx','Suburb A',6)
    display(obtain_movement(total_earnings,earnings_table))
```

```
Optimal Strategy for Sample Run 1: Start by following the App

                                         1           2  \
Downtown  Turn App off and move to Airport  Follow App
Airport                         Follow App  Follow App

                                         3           4  \
Downtown  Turn App off and move to Airport  Follow App
Airport                         Follow App  Follow App

                                         5           6        7
Downtown  Turn App off and move to Airport  Follow App  Go home
Airport                         Follow App  Follow App  Go home



Optimal Strategy for Sample Run 2: Start by leaving App off and moving to Airport

                                         1           2  \
Downtown  Turn App off and move to Airport  Follow App
Airport                         Follow App  Follow App

                                         3           4  \
Downtown  Turn App off and move to Airport  Follow App
Airport                         Follow App  Follow App

                                         5           6        7
Downtown  Turn App off and move to Airport  Follow App  Go home
Airport                         Follow App  Follow App  Go home



Optimal Strategy for Sample Run 3: Start by leaving App off and moving to Airport

                                         1                                 2  \
Downtown  Turn App off and move to Airport  Turn App off and move to Airport
Airport                         Follow App                        Follow App
Suburb A  Turn App off and move to Airport  Turn App off and move to Airport
Suburb B                        Follow App                        Follow App

                                         3                                 4  \
Downtown  Turn App off and move to Airport  Turn App off and move to Airport
Airport                         Follow App                        Follow App
Suburb A                        Follow App  Turn App off and move to Airport
Suburb B                        Follow App  Turn App off and move to Airport

                   5        6
Downtown  Follow App  Go home
Airport   Follow App  Go home
Suburb A  Follow App  Go home
Suburb B  Follow App  Go home
```