Review for Exam 2 (4/19)

As written in the syllabus, Exam 2 will be on Tue. 4/24 during class. The format is similar to the two practice exams given homework 8:

Question 1. (10 points) Two **multiple choice** questions testing the following points:

- LP Geometry: graphical solution in 2-D, binding constraints, non-binding constraints.
- Shadow prices, allowable regions, and interpretation of shadow prices.
- Conceptual questions involving LP or MIP.

Question 2. (30 points) A word problem asking you to write a **concrete formulation** of a LP or MIP. (It's okay to write an abstract formulation instead but make sure you define the input data and write using the correct notation.)

- By concrete formulation, we mean a formulation where the data are numbers, and the objective and constraints are fixed.
- By abstract formulation, we mean a formulation in which the input data are contained in input variables, and the objective and constraints can be generalized to arbitrarily large data sets with different input data. For examples, see the course notes to session 21, which contains the concrete and abstract formulations for 3 in-class exercises.

Question 2. (35 points) A word problem asking you to write parts of a **concrete formulation** of a LP or MIP (with particular numbers), followed by a complete **abstract formulation** (that is not dependant on particular numbers but can be generalized to arbitrarily large data sets when the values of the input variables change).

Question 3. (25 points) A problem in which you are given an abstract formulation of a LP or MIP, and are asked to **write code on paper** to implement it using Python and Gurobi. The question will provide the format of the input data, as well as the format of the output. For inputs and outputs, you should be comfortable working with `DataFrame`, `Series`, `list`, `dict`, and `float` data types.

Submit Questions for Review Session

Please submit your most urgent question to be answered during the review session on Thursday 4/19 using this survey. If you desire your question to be answered during class, you should submit your question at least 1 hour before the start of class.

Resources for Studying

Practice Exam Solutions (Most Relevant):

- Solution to Homework 8 (Practice Exams)

General resources:

- Folder with all in-class handouts
- Folder with all online notes

LP Geometry and Shadow Prices:

- Shadow prices: 16-LP Duality.
- Allowable range: 17-LP Modeling.
- Solution to Homework 6 (Basic LP and shadow prices)

- Solution to Quiz 2.

LP/MIP Modeling:

- Explanation of in-class examples: 21-Optimization Modeling I, 22-Optimization Modeling II, and 24-Modeling with Auxiliary Decision Variables.
- Addional practice problems: see end of course notes to session 22. Solutions here.
- The exam will not include linearizing absolute values or modeling with quadratic constraints. However, you may need to use auxiliary variables
- to link multiple periods as in Homework 8 Q2 and in course notes to session 24.
- to link binary auxiliary variables with continuous variables, as appeared in Homework 8 Q7, as explained in course notes to session 24, and illustrated further in 25-Lab 7.
- to apply the tricks in 17-LP Modeling with U, L, and soft constraints. You should also be able to apply the trick described here to simplify a constraint on a ratio by multiplying out the denominator.
- (Optional) additional practice from the book: DMD Exercise 7.7, 7.11, 7.13, 7.14, 7.15, 9.2, 9.3, 9.4, 9.5, 9.6, 9.7, 9.8. (For each question, focus on writing the formulation: decision variables, objective, and constraints.)

Relevant Python Coding:

- Basic Gurobi Commands for Linear Programming (end of notes for 16-LP Duality)
- Solution to Homework 7 (Labs 5 and 6)
- DataFrames and Excel (beginning of notes for 19-Mixed Integer Programming)
- Additional practice: code up the abstract formulation for Exercise 1 and 2 from Session 21, as well as Exercise 4 from Session 24. If you want additional practice, then generalize the concrete formulations to the additional exercises here to abstract formulations, create an Excel file containing the data, and code up the formulations in Gurobi (on paper), outputing the solutions to another Excel file. To check, you can type up your code and see if it works.
- (Optional) Easy to read tutorial of DataFrame and Series
- (Optional) Additional explanation on list comprehension

Tips for Coding on Paper

Saving Time:

- No need to write `name=` for any variable or constraints. (This is optional and you don't have time to write optional code in the Exam.)
- No need to write `vtype=grb.GRB.CONTINUOUS` as this is the default option when creating decision variables. The code `x=mod.addVar()` by itself creates a continuous decision variable. If you want it to be non-negative, then do `x=mod.addVar(lb=0)`. No need to supply any other arguments.
- No need to write `mod.setParam('OutputFlag',False)`.
- Don't spend time changing the format of input data. If it is in a DataFrame already, you don't need to load it into a dictionary or a Series using a for loop. Directly access each element of the data when writing objectives or constraints. (If the input data is in a Excel file, you can load it with one line such as `data=pd.read_excel(filename,index_col=0)`. Don't write many unnecessary lines of code manipulating the input data; the most you will be asked to do will be to obtain the index sets.)
- Use `mod.addVars` to save time when defining variables. Instead of

```
x={}
for i in I:
    for j in J:
        for k in K:
            x[i,j,k]=mod.addVar(vtype=grb.GRB.BINARY)
```

Write the equivalent one-liner:

```
x=mod.addVars(I,J,K,vtype=grb.GRB.BINARY)
```

See the solution to Homework 8 for further explanation.
Avoiding Common mistakes:

- Circle on the exam paper which are the input variables and write their types, so you can access it correctly.
- Use `a[XXX]` for dictionaries. (XXX is placeholder for the the key) (This is the way to access decision variables.)
- Use `a.loc[XXX]` for Series. (XXX is placeholder for the particular row you want)
- Use `a.loc[XXX, XXX]` for DataFrame. (XXX's are placeholders for the particular row and column you want)
- You only need to use `mod.addVar` for decision variables (not input variables).
- When expressing a constraint, for example

$$\sum_{i \in S} x_{ij} = 1$$

You should write what set you are summing over within the sum construct (this is the same syntax as list comprehension):

```
mod.addConstr(sum(x[i,j] for i in S)==1)
```

Do not do something like `sum(x[i,j])` without specifying what you are summing over. Make sure to put the `for i in S` inside the sum, and not in a for loop around the `mod.addConstr`. You are creating one constraint with a sum of many things, not many constraints (each being the sum of one thing).