IEM 5013 Introduction to Optimization
Mini-project 2
Check Canvas for due date/submission information.

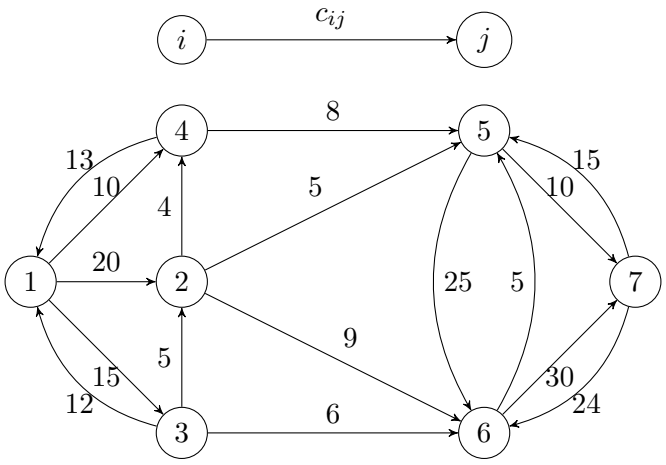
Assignment Policy: This assignment can be done individually or with a partner i.e., maximum team size is **three**. A student can be part of exactly one submission. In order to include your name in the team's submission you must have made a substantial contribution by actively and continuously participating in the development of your team's assignment submission.

Permitted Resources: Course textbook and any other textbook, even if it is not listed among the references on the course syllabus. All materials provided on the Canvas course website. All materials available at Gurobi.com and at other links provided on the Canvas course website as learning resources for the Python programming language.

Prohibited Activities: Collaboration, using any means, with anyone other than your teammate. Searching the internet for answers. Accessing past assignment solutions with the help of former students or using websites that facilitate this access. Use of any resource outside the resources explicitly noted above as permissible. Anything else I haven't enumerated, but violates the spirit of academic integrity.

The assignment must be completed using only your individual/team effort and no external assistance beyond the permissible resources.

The Single Source Shortest Path Problem



Suppose you are given a network $G = (N, A)$ with arc weights c_{ij}, and a distinguished source node $s \in N$. The single source shortest path problem (SS-SPP) is to find the shortest path from s to every other node in $N \setminus \{s\}$. We can solve this problem by formulating it as a minimum cost network flow problem (given below), which is a valid approach as long as the network does not contain a negative total weight directed cycle. We let x_{ij} denote the flow on arc $(i, j) \in A$.

$$\textbf{(SS-SPP)} \quad \min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\text{subject to:} \quad \sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = \begin{cases} |N|-1, & i = s \\ -1, & \forall i \in N \setminus \{s\} \end{cases}$$

$$x_{ij} \geq 0 \quad \forall (i,j) \in A$$

Assignment. Develop a Python code that implements and solves the foregoing formulation using Gurobi. Your implementation must meet the following requirements.

1. Your code must accept the source ID and input filename as inputs (keyboard input is fine); the input files contain the network information (nodes, arcs, costs) in a standardized format. Two input files are provided (`sample.gr` and `biginstance.gr`) on Canvas and the sample network represents the network in the figure above. Use the sample network to understand the format; it is self explanatory.

2. Do not modify the instances in anyway; modify your code to read the instances as is, given the filename as input. We will validate your code using these two instances and for our choice of a source node. Make sure you are producing the right output for different source node choices on the small network given above. Test your implementation on both instances before submission.

3. A Python file named `LastName1_LastName2_LastName3.py`, clearly commented and formatted should be uploaded to Canvas; the last names are of the team members. List the team member names at the start of the code file as comments.

4. Your code should produce a formatted output file (`.csv` is preferable) that lists two columns: "Node ID" , "Predecessor ID". If $x_{i,j} > 0$ in the optimal solution, we say i is the predecessor of j in the shortest path tree. You would then list j under node ID and i under predecessor ID. (FYI: With this information you can reconstruct all the shortest paths if you are so inclined; this output is essentially the optimal solution you found, represented more compactly. Because the solution will only use $|N|-1$ arcs, which is much smaller than $|A|$, the number of variables in the model.)

5. Check the termination status of the algorithm; if optimal, produce the output file as required above. If infeasible or unbounded, report an appropriate output message.

Implementations that do not meet all of the above requirements will be **heavily penalized.**

Hint: A directed network can be represented using a forward-star (and/or reverse-star) representation(s) where you have a list for every node in the network, and the members of the list for node i are the heads of the arcs leaving i, i.e., the out-neighbors of i; as our arcs are also associated with costs, each entry of list i can be a tuple (j, c_{ij}). The best implementation for such a data structure is using a dictionary of lists; each entry of each list can be a two-element tuple or list.

`Fstar`	:	
1	:	$(2,20),(3,15),(4,10)$
2	:	$(4,4),(5,5),(6,9)$
3	:	$(1,12),(2,5),(6,6)$
4	:	$(1,13),(5,8)$
5	:	$(6,25),(7,10)$
6	:	$(5,5),(7,30)$
7	:	$(5,15),(6,24)$

Alternately, you can use the NetworkX package available for network algorithms in Python. Learning this package will likely be more advantageous in the long run. For instance, if you wish to

visualize your optimal solution, it would be easier with this package. This package uses a nested dictionaries data structure to capture all the network elements and their attributes.

In my personal implementation, I create the Gurobi variables one by one as I read each arc in the input file, assigning the cost immediately as the objective coefficient. I then have no further use for the costs, except as objective coefficients. Then I create a set of nodes and a list of arcs (tuples) that I use in implementing the flow conservation constraints.