Lab 0

Due February 22, 2021 before class

Cassandra, Shashank, Sherrie, and Manu were starting to get bored because they didn't have enough Advanced Algorithms work, so they decided they needed to find themselves a hobby. They found a local badminton league and each joined separate teams. Everyone on the team with the most wins at the end of the season will win a Dunkin Donuts gift card. Shashank has stated that once his team can no longer win the most games, he is going to quit. Therefore he wants to keep track of when his team has been mathematically eliminated from the competition.

Part 1 - Setting up the problem

The current standings are as follows:

				Against			
Team	**Wins**	**Losses**	**Left**	**Sherrie**	**Shashank**	**Manu**	**Cassandra**
Sherrie	83	71	8	-	1	6	1
Shashank	80	79	3	1	-	0	2
Manu	78	78	6	6	0	-	0
Cassandra	77	82	3	1	2	0	-

1) Which teams have been eliminated from getting the Dunkin Donuts prize? Which teams have not been eliminated? Why or why not?

Cassandra has been eliminated; even if she wins the rest of her games, she cannot catch up to Sherrie. Shashank has also been eliminated – he would need to win all three of his games to match Sherrie's total at 83 wins, and would need Sherrie to lose all of her games. However, if she does lose all her games, Manu's win total goes to 84, still putting him above Shashank.

Sherrie and Manu have not been eliminated, because they have a higher current number of wins than Cassandra and have 8 and 6 games left, respectively. Because they have more games left, there is the possibility of them increasing their number of wins past 84, giving them the highest number of wins and getting the Dunkin' Donuts prize.

2) Sherrie decides that she's going to be smarter than the rest of the Advanced Algorithms team and create an easy way to tell who's been eliminated. Due to bad record-keeping, Sherrie has access to none of the scores. However, she does have a 5 minute window to look at the standings to quickly determine what teams are eliminated. She decides she is going to set this up as a network flow problem.

The games won will be represented by w_{name} and the games remaining will be represented by r_{name}. For instance, Sherrie has won $w_{Sherrie}$ games and has $r_{Sherrie}$ games remaining. The teaching team trusts you can figure out the variable representation for the other players.

She sets it up as the following network flow:

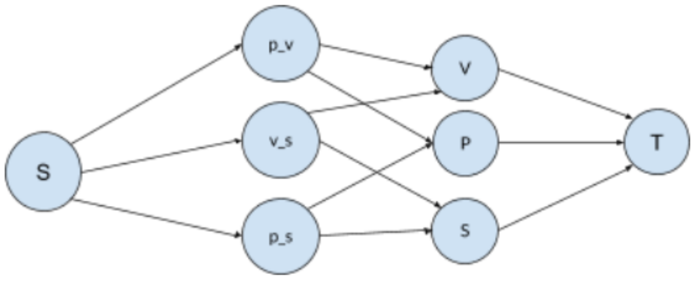
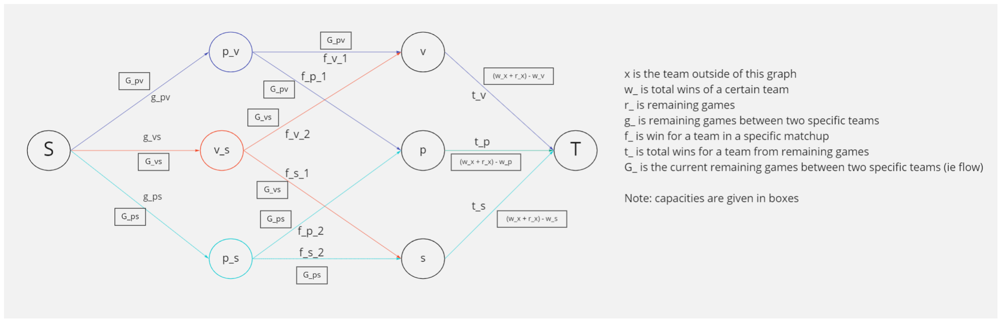


Figure 1: Sherrie's network flow

Figure 1 denotes the network flow diagram that Sherrie constructed to figure out if she was eliminated (note: the flow diagram is specific to her), without any of the capacities. The first node is a source node. The second series of nodes represent the matches between each of the other teams (i.e. Cassandra vs Shashank, Cassandra vs Manu). The third series of nodes represent the other teams (i.e. Cassandra, Shashank). The final node is a sink node.

Construct a procedure for determining if teams are eliminated. Note that to do this, you will have to determine what the capacities of the graph depicted in Figure 1 are. For this question, you must:

1. Draw out the graph with the capacities represented in variable form (explain what the variables represent).



x is the team outside of this graph
w_ is total wins of a certain team
r_ is remaining games
g_ is remaining games between two specific teams
f_ is win for a team in a specific matchup
t_ is total wins for a team from remaining games
G_ is the current remaining games between two specific teams (ie flow)

Note: capacities are given in boxes

2. Identify what the values of the variables would be for this specific problem

v	Maru's Team
p	Cassandra's Team
s	Shashank's Team
x	Sherrie's Team
G_pv	0
G_vs	0
G_ps	2
r_x	8
w_x	83
w_v	78
w_p	77
w_s	80

3. Write out the strategy for solving the problem.

The strategy to answering the question "If Sherrie won every game, is there still a chance she could lose?" consists of analyzing the network graph from above in two steps:

(a) We check to see if the edge capacities leading into the sink are positive or negative. If they are negative, then we know there is no chance that the team we are evaluating could overtake the team that the flow is associated.

(b) We find the maximum flow and we determine if the maximum flow is less than the source output capacities (in this case $G_vp+G_vs+G_ps$). If it is less than that value, that means that there are some games which, if won, would exceed the difference between Sherrie's best possible standing and the other player's current standing.

4. Explain why this strategy works.

   To reiterate, the question that we're answering is "If a player (Sherrie) wins all their upcoming matches, is there still a possibility that they will lose?" The way we have constrained our network flow diagram, if maximum flow is not reached, it means that we have "leftover" flow and that an edge in any augmenting path in the network has reached its capacity. The three edges in each augmenting path consist of the edge leaving the source (the amount of matches left), the wins from the given match ups and the last edge which represents how many less wins a given player has, if Sherrie wins all her upcoming matches. Since the middle edge always has the same capacity as the edge leaving the source (amount of matches left), maximum flow not being reached must happen when the rightmost edges reach their capacity. For a rightmost edge to reach its capacity means that a given player has "caught up" to Sherrie's wins, that the team has the ability to tie. This means if there is still this "leftover" flow, that means there are more games that are not yet counted that would put one of the other teams ahead over Sherrie's maximum possible win total, showing that she has been mathematically eliminated. Furthermore, since the rightmost edges' capacities are given as the difference between a players current wins and Sherrie's maximum wins, if a given edge has negative capacity, that means that that player has more wins than Sherrie's maximum and that Sherrie has definitely been mathematically eliminated.

3) For the network flow diagram you finished above:

1. Convert it into a linear program (using variables, not the values). If you aren't sure how to do this, check out this link: http://www.mathcs.emory.edu/ cheung/Courses/323/Syllabus/NetFlow/max-flow-lp.html.

   We want to maximize using the objective function: $z = g_{pv} + g_{vs} + g_{ps}$. We are operating under the following conditions:

$$\begin{aligned} f_{v1} + f_{p1} &= g_{pv}, & f_{v1} + f_{v2} &= t_v, \\ f_{v2} + f_{s1} &= g_{vs}, & f_{p1} + f_{p2} &= t_p, \\ f_{p2} + f_{s2} &= g_{ps}, & f_{s1} + f_{s2} &= t_s \end{aligned}$$

   The variables in these conditions are bounded by their own constraints which are defined by values given:

$$\begin{aligned} g_{pv}, f_{v1}, f_{p1} &\le G_{pv}, & t_v &\le w_x + r_x - w_v \\ g_{vs}, f_{v2}, f_{s1} &\le G_{vs}, & t_p &\le w_x + r_x - w_p \\ g_{ps}, f_{p2}, f_{s2} &\le G_{ps}, & t_s &\le w_x + r_x - w_s \end{aligned}$$

2. Provide an explanation of why this formulation makes sense, given the original context.

   - First Set: These represent each node as for any given node that is not the source or sink, the flow coming in should equal the flow coming out. If this is not the case, than the flow is invalid.
   - Second Set: If a team is to not be eliminated than we do not want any of the flows into the sink (t variables) to exceed the difference between our evaluated teams best wins and wins of the team associated with the flow. So with these constraints, when the maximum flow is found none of these flows will be overflowing. But if the maximum flow doesn't equal number of games being played by the other teams, that means the flows coming out of the sink are not maxed out. That means one of the flows going into the sink will have to be overflowed causing the linear programming problem to fall apart if the analyzed team is eliminated.

Part 2 - Implementation

Implement the network flows and the linear programming approach to the problem in Python (we are providing input files and starter code).

Make a fork of this github repo: https://github.com/AdvancedAlgorithms/Lab0.

Use "pip install -r requirements.txt" to install the requirements for the right libraries (you might want to use pip3 to use python3).

We have also provided a test file (test_badminton_elimination.py). At a minimum, your code should pass all of the tests in that file. Feel free to add your own additional test cases if you would like to more robustly test your code. If you think the test cases we have given you are sufficient, please explain how either in a comment or in your answer to this question. We aren't evaluating you on this test cases portion, but it's a good exercise to go through. To run your code on a specific input table (defined in a txt file, see teams2.txt and teams4.txt for examples), you can simply run "python badminton_elimination.py teams2.txt"

We recommend using the networkx function to solve the problem using network flows (documentation can be found here: https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms.flow.maximum_flow.html) and using the picos solver to solve the problem using linear programming (documentation can be found here: https://picos-api.gitlab.io/picos/graphs.htmlmax-flow-min-cut-lp).

Your program should be able to answer the following question: Who is eliminated given a table of the current standings? You should be able to do this using a network flows approach and a linear programming approach.

Example input (the 4 at the top represents the # of teams in the division and the remainder of the rows and columns correspond to the same rows and columns as were specified in the table above):

```
4
Sherrie 83 71 8 0 1 6 1
Shashank 80 79 3 1 0 0 2
Manu 78 78 6 6 0 0 0
Cassandra 77 82 3 1 2 0 0
```

Corresponding output:

```
Sherrie: Eliminated? False
Shashank: Eliminated? True
Manu: Eliminated? False
Cassandra: Eliminated? True
```

To submit this lab - submit a link on Canvas to your Github repository with code and answers to questions 1-3.

Happy coding!

Link to repo: https://github.com/JulianStone5/Lab0

We believe the tests provided are sufficient because they cover elimination cases that are not necessarily obvious from just looking at the score totals, but require looking at the matchup details to see whether elimination is mathematically guaranteed. Additionally, we went through a few of the tests by hand to ensure that we came to the same conclusions, and agree with the outcomes.