Lab 0

Colin Snow, Sander Miller, Nathan Shuster

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?

Sherrie: No, if she wins all remaining games she will win.
Shashank: Yes, Shashank would need Sherrie to lose all her games against Manu, in which case Manu would win.
Manu: No, Manu would need to win all 6 of his remaining games.
Cassandra: Yes, even if she wins all her games she can not win the tournament.

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:

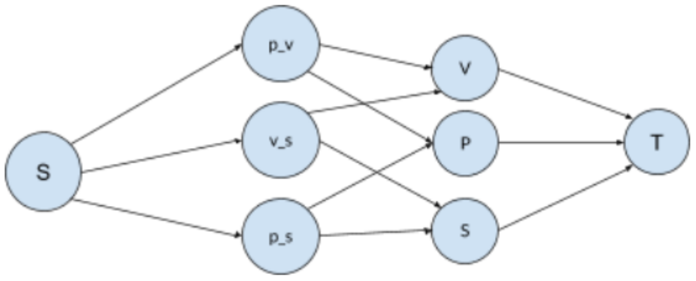

Figure 1: Sherrie's network flow

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). The first column of nodes (after Source) are games remaining between rival teams, so the first set of capacities are the number of games remaining. The second column of nodes represents the score of each of the rival teams. The capacities between the first and second columns are infinite but are bounded by the prior inputs from source. The capacities from the second column to the sink represent the max number of wins that team can get and not beat the max score of our target player (Sherrie original score + score from Sherrie winning the rest of her games - other player original score).

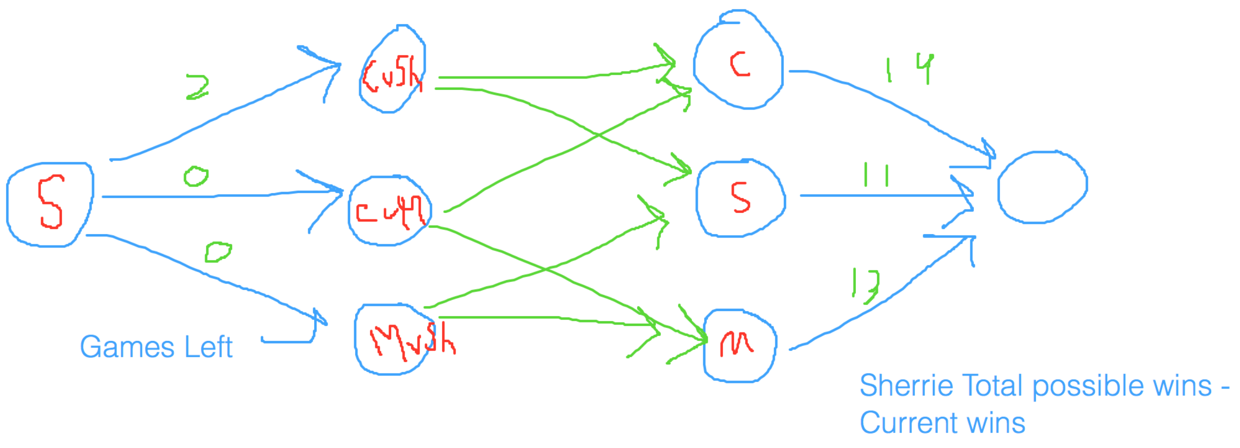

Figure 2:

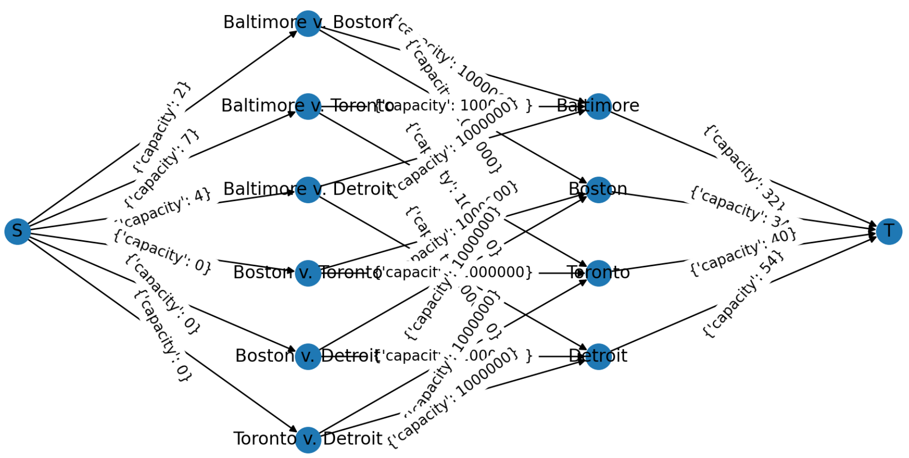

Figure 3: NetworkX Implementation

See graph image above.

2. Write out the strategy for solving the problem.

   In order to determine whether a given player is still eligible to win the tournament you must first generate the associated network flow. The first set of nodes represent the possible game matchups that don't include the given player. The capacity into those nodes from the source is equal to the number of games left for the given matchup. The next set of nodes represents each of the other players in the tournament. Each matchup node is connected to the associated player nodes. The capacity of the connection is equal to or greater than the capacity into the matchup node, accounting for the possibility that one player wins all of the given matchup games. Finally all of the player nodes are connected to the sink. The capacity of these connections is equal to the number of possible points the given player can score assuming they win their remaining games, minus the current number of points the node player has. In order to determine whether the given player is eligible we must find whether the maxflow of the graph is equal to the total capacity leaving the source node. In other words, does the max flow fully saturate all edges leaving the source node. If this is true, then the player is still eligible. If it is not, the player has been mathematically eliminated.

3. Explain why this strategy works.

   The maximum possible score someone can achieve is their current score plus the number of games they have left, assuming they win them all. Therefore, if there exists a scenario where the remaining games can be split up between players such that all the other player's scores are less than the current player's maximum score they have not yet been eliminated. The graph created above essentially emulates this process. The first edges represent the number of games left between the other players, the second edges represent those games being divided up between the two players, and the last set of edges check if that combination makes the player win. The algorithm is essentially trying to figure out if there is a way to split up the wins so that no player gets enough wins to go beyond the current player. If there is a way to do this, the person has not lost, otherwise there is no path to winning.

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.

   Maximize: $x_{S->pv} + x_{S->v_s} + x_{S->p_s}$ subject to:

$$x_{p_v->v} + x_{p_v->p} - x_{S->p_v} = 0$$
$$x_{v_s->v} + x_{v_s->s} - x_{S->v_s} = 0$$
$$x_{p_s->p} + x_{p_s->s} - x_{S->p_s} = 0$$

$$x_{p_v->v} + x_{v_s->v} - x_{v->T} = 0$$
$$x_{p_v->p} + x_{p_s->p} - x_{p->T} = 0$$
$$x_{p_s->s} + x_{v_s->s} - x_{s->T} = 0$$

$$x_{S->p_v} \leq \text{games remaining Cassandra v Shashank},$$
$$x_{S->v_s} \leq \text{games remaining Cassandra v Manu},$$
$$x_{S->p_s} \leq \text{games remaining Manu v Shashank},$$
$$x_{v->T} \leq \text{Sherrie total possible wins - Cassandra current wins},$$
$$x_{p->T} \leq \text{Sherrie total possible wins - Shashank current wins},$$
$$x_{s->T} \leq \text{Sherrie total possible wins - Manu current wins}$$

$$x_{S->p_v}, x_{S->v_s}, x_{S->p_s}, x_{v->T}, x_{p->T}, x_{s->T}, \geq 0$$

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

   The equations are a formalization of rules governing capacity and network flows. Each edge has a maximum capacity, and the flow into all non-source/sink nodes must be equal to the flow out.

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 think that the given test cases are a good test for normal operating scenarios, but it might be wise to check for specific edge cases that might not arise in the amount of tests provided.

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!