# Module 12: Playoff Elimination Problem

# JHU EP 606.206 - Introduction to Programming Using Python

## Introduction

In the late months of fall, Major League Baseball teams battle for the chance to compete in the playoffs and ultimately compete in the World Series. Some teams, however, are less fortunate, and are mathematically eliminated along the way. If we model a division of teams (say, the NL East) as a graph, it turns out we can compute a max-flow through the graph and determine whether or not a team has been mathematically eliminated from playoff contention!

**Skills**: file I/O, NetworkX, control structures, data structures, conditional logic, classes, functions

## Playoff Elimination Problem

We consider a division that consists of n teams. A given team `t` currently has `wins[t]` wins, `losses[t]` losses, `remaining[t]` games yet to be played, and `games[t][opp]` games left to play against team opp. In our formulation of the problem, only the 1st place teams will make the playoffs so we say a team is mathematically eliminated from playoff contention if they cannot finish the season in 1st place (or tied for 1st). Assumptions: all games have a winner and loser (no ties) and every remaining game is played (there are no cancellations).

## Elimination Scenarios

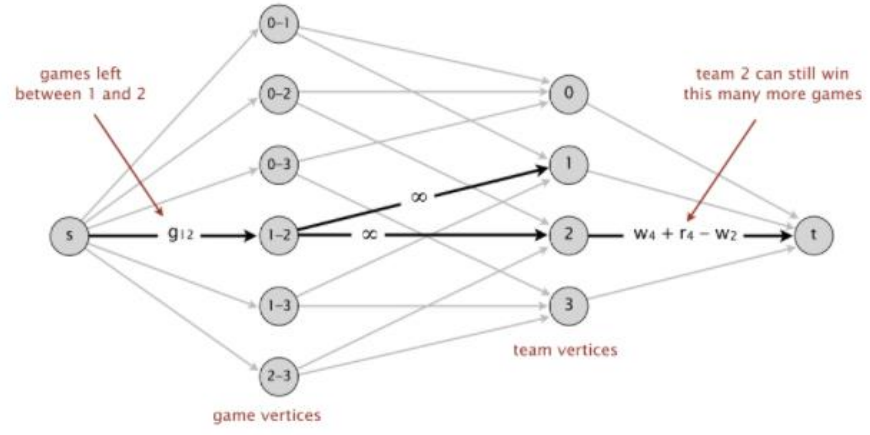
| | | w[i] | l[i] | r[i] | g[i][j] | | | |
|---|---|---|---|---|---|---|---|---|
| i | team | wins | loss | left | Atl | Phi | NY | Mon |
| 0 | Atlanta | 83 | 71 | 8 | - | 1 | 6 | 1 |
| 1 | Philadelphia | 80 | 79 | 3 | 1 | - | 0 | 2 |
| 2 | New York | 78 | 78 | 6 | 6 | 0 | - | 0 |
| 3 | Montreal | 77 | 82 | 3 | 1 | 2 | 0 | - |

## Scenario #1: Trivial Elimination

In some cases it is easy to see when a team has been eliminated. In the example above Montreal is eliminated because, if we assume they win the rest of their games (3), their total possible wins would equal 80 which is not enough to finish in 1st place. That is, because `wins[m] + remaining[m] < wins[t]` where `m` is Montreal and `t` is a team whose current total wins are more than the total number of wins possible for Montreal (Atlanta in this case), Montreal is eliminated. It is less clear, however, if either New York or Philadelphia have been eliminated because their playoff prospects hinge on not only how many games are left, but also who those games are scheduled against.

## Scenario #2:

Given a division with 5 teams, for example, if we want to find out if team x has been eliminated we can construct a flow graph representing all of the games left to be played by the other x-1 teams:



We can then compute a max-flow through the graph and examine the set of edges from `s` to each game vertex to determine if a team has been mathematically eliminated from contention. More on this later.

Intuitively, we can think of `s` (the "source") as "bucket" of all games remaining between the x-1 teams, excluding any games that are yet to be played by x (we assume that, in the best case, x wins all of their games). `t` (the "sink") can simply be thought of as a "bucket" that receives any of the games that flow through the graph.

Each node in the column containing 6 vertices can be thought of as a game played between the two teams. That is, [0-1] represents the games played between Team 0 and Team 1. Thus, the edge weight ("capacity") of each edge between s and any game node represents the number of games that are left to be played between the two teams.

Each node in the column containing 4 vertices can be thought of as representing an individual team. Thus, an edge placed between a game node and a team node can be thought of as the number of games won by that team against the other team in the game node. That is, each unit of flow that goes from [1-2] to [2] represents a win by Team 2 over Team 1. Since the max-flow is going to simulate the games played for us, we do not yet know what the capacity of these edges will be. So, we'll initially set them to infinity and let the max-flow algorithm compute the values for us.

Finally, and perhaps most importantly, is our choice of edge weights for the edges between the team vertices and the sink node. Remember our overall goal:

1. Assume team x wins all of their remaining games
2. Determine if there is any possible way for the rest of the games between the other teams to be played in such a way that no team wins more games than x. In plain English, we're just asking if there's any way for x to finish with the most wins out of any of the teams.

To achieve this would be to restrict the flow on these edges to be: ***`wins[x] + remaining[x] - team[t]`*** where `x` is the team we're checking to see if they've been eliminated and `t` is the team on the team node. This is the same as saying "The maximum capacity possible along these edges is the number of games won by team `t` if `t` loses all of their games against team `x` (necessary because we're assuming team `x` wins the rest of their games) but wins all of their other games against the other `x-1` teams." Now, the max-flow will determine the actual values for these edges; we're just establishing what the maximum possible flow could be. Setting the edge weights in this way is precisely what enables the max-flow computation to model whether team `x` has been eliminated.

Now that our graph has been constructed as described above, NetworkX makes it very easy to compute the max-flow of the graph. The remaining question is: once we've calculated the max-flow, how do we interpret it to determine if team `x`'s playoff hopes (their hopes of finishing first in their division) are still alive? There are two outcomes to consider:

1. After computing the max-flow, all of the edges from `s` to the games nodes are saturated (full). If this happens, then at least 1 scenario exists where no team in the graph wins more games than `x` which means `x` still has a way to win the division and is thus not yet eliminated. More simply,

"All simulations of the games between the other $x-1$ teams have been played (again, assuming $x$ wins all of their remaining games) and there are still games remaining to be won by $x$ that would give them a 1st place finish".

2. After computing the max-flow, all of the edges from $s$ to the games nodes are **not** saturated. In this case, there is (at least) one team who finishes with more wins than $x$ in every possible outcome of the tournament, and therefore $x$ has been mathematically eliminated. This is like saying "All simulations of the games between the other $x-1$ teams have been played and there are not enough games left for $x$ to win that would allow them to finish in 1st place".

## Programming Assignment

For this assignment you will use NetworkX to build a flow graph and solve the playoff elimination problem. Note that the below example shows the use of the playoff elimination algorithm against the MLB dataset provided, however, you should ensure your solution is generic and works across all the provided inputs.

1. Your program will read data from various text files (provided on blackboard) in the format shown below. The text file begins with the number of teams (an integer) and is followed that number of rows. Each row contains a team name followed by that team's wins, losses, remaining games, followed by the schedule of those remaining games with each of the other teams. In the example below Atlanta has 83 wins, 71 losses, 8 games left to be played with 0 of those games against themselves, 1 game against Philadelphia, 6 games against New York, and 1 game against Montreal.

```
4
Atlanta       83 71  8  0 1 6 1
Philadelphia  80 79  3  1 0 0 2
New_York      78 78  6  6 0 0 0
Montreal      77 82  3  1 2 0 0
```

2. Once you've read and stored the data in the appropriate data structures, for each team in the list determine if they've been eliminated based on the method in Scenario #1 above.

3. Next, for each team in the list build a flow-graph like the one shown in Scenario #2 above. Once constructed, compute the max-flow for each graph using the appropriate built-in algorithm provided by NetworkX. Once you've computed the flow, use rules 1 and 2 from Scenario #2 to determine if any of the teams have been eliminated.

4. Output your results using the following format. The solution shown below is from the input in Step 1 above. Montreal is eliminated by Scenario #1 because the most wins they can earn is 80 but Atlanta already has 83 wins, so there is no way for Montreal to finish in 1st place. Less obviously, Philadelphia is eliminated because they can finish with at most 83 wins, but since there are 6 games remaining between Atlanta and New York, one of them is guaranteed to finish with at least 84 wins.

teams.txt

```
1  4
2  Atlanta       83 71  8  0 1 6 1
3  Philadelphia  80 79  3  1 0 0 2
4  New_York      78 78  6  6 0 0 0
5  Montreal      77 82  3  1 2 0 0
```

```
Atlanta is not eliminated.
Philadelphia is eliminated.
New_York is not eliminated.
Montreal has been trivially eliminated by Atlanta.
```

## Deliverables

You do not have to submit a readme.txt for this assignment. Please submit your source code in a file called **elimination.py** along with screenshots of the results generated from running your code on each of the input files on Blackboard.

## Challenge (**not for credit**)

In the example above Philadelphia was eliminated because there were 6 games remaining between Atlanta and New York. Our program used the max-flow algorithm to detect the elimination but did nothing to certify what caused it. A concept closely related to the max-flow is called the min-cut. In fact, the max-flow of a graph is equal to the min-cut. By finding all the team nodes on the source-side of the min-cut we can fully deduce the cause of a team's elimination. Thus, we can compute the min-cut of our graph and use it to tell us the reason Philadelphia was eliminated:

teams.txt

```
4
Atlanta       83 71  8  0 1 6 1
Philadelphia  80 79  3  1 0 0 2
New_York      78 78  6  6 0 0 0
Montreal      77 82  3  1 2 0 0
```

```
Atlanta is not eliminated.
Philadelphia is eliminated by ['Atlanta', 'New_York']
New_York is not eliminated.
Montreal has been trivially eliminated by Atlanta.
>
```

## Hints

My solution to the problem posed in the Programming Assignment sections was about 100 lines of code. ~30 of those lines dealt with reading the inputs from the file and storing them in the appropriate data structures. Another 20 lines were dedicated to building the flow graph. You actually don't have to think too much about the max-flow algorithm. I wrote a function to compute and return the max-flow and it is 1 line of code long; NetworkX does all the hard work for you. For me, I spent the largest percentage of time correctly building the flow graph to match the image shown in Scenario #2.

For anyone interested in tackling the Challenge, this took me an additional ~15 lines of code to both compute the min-cut and then figure out which teams caused the elimination.