EECS 278: Advanced Algorithms | Due: 11:59pm, 12-06-2019
Project Report- Baseball Elimination | Submitted by: Puneet Soni and Shubham Rohal

---

Introduction:

In the baseball elimination problem, there is a league consisting of N teams. At some point during the season, team i has $w[i]$ wins and $r[i][j]$ remaining games to play against team j. A team is eliminated if it cannot possibly finish the season in first place or tied for first place. The goal is to determine exactly which teams are eliminated.

Total Number of teams	4	Wins	Losses	Remaining games	Against			
					Team 0	Team 1	Team 2	Team 3
Team 0	**0**	83	71	8	0	1	6	1
Team 1	**1**	80	79	3	1	0	0	2
Team 2	**2**	78	78	6	6	0	0	0
Team 3	**3**	77	82	3	1	2	0	0

Figure 1: Sample team data for Baseball Elimination

Team 3 is eliminated as it can only finish at most 80 wins and Team 0 is already having 83 wins. Team 1 is also eliminated which can finish the season with at most 83 wins in the season, which appears to get a tie with team 0, but for that to happen team 0 must lose all the remaining games, including the 6 games against the team 2. Then, team 2 would finish with 84 wins. We can see that team 2 will not be eliminated despite the fact that it has fewer game wins than team 1.

Goal:
To find that how many and which teams are eliminated in the baseball tournament.

Code Explaination:
1. Input files will be read and checked with the base condition if ($teams == 1$) output will be returned as -1.

2. In second case, if there are no remaining games left then the decision will be made on the basis on the maximum number of wins by a team. In that case, everyone except the leading team will be eliminated.

3. $Layer1\_capacities$ and $Layer2\_capacities$ are the list in which the edge values are pushed. Starting from Source S to sink T. The weight of edges going from Source S to $Layer1$ nodes will be defined by the number of remaining games between two teams and edges going from $layer1$

to $layer2$ can have the maximum flow = total number of remaining games between those teams (where $Layer1$: consists of remaining games and $Layer2$: number of teams).

4. In $fordFulkerson$ function we are passing the graph, with source and sink with the index of team whose possibility of winning we are verifying. (In which we have used the $depth\_first\_search()$ function). Here, we have used the python library networkx to generate the graph from the input file which helps us to compute the network flow by using the number of wins, losses and remaining games.

After completing the computation of network flow, the eliminated teams index value will be displayed.

Input:

Input consists of Multiple lines. First line consists of the total number of teams which is defined with the variable named *teams*.

Following N lines consists of teams index values labeled/indexed as $0, 1...N-1$, number of wins, number of loses, remaining games to be played and distribution of remaining games against each teams. Here, the $Figure1$. describes a sample input(2.txt file) for required program.

Output:

If the teams are eliminated output for the above input file(2.txt) will be displayed as, where team 1,3 are eliminated.

References:

- https://catcourses.ucmerced.edu/courses/14910/files/folder/Lecture
- https://networkx.github.io/documentation/stable/tutorial.html
- https://www.cs.princeton.edu/courses/archive/spr03/cs226/assignments/baseball.html
- https://github.com/nastra/AlgorithmsPartII-Princeton/tree/master/baseball