BASEBALL ELIMINATOR

Compilation and execution on terminal:

- First, open up the terminal in linux or mac. Then change the directory to the location of the unzipped(submitted) folder.
- After reaching the desired folder, run the command -
  g++ baseballeliminator.cpp

- This command will compile the program and create an a.out file at the folder.Do not worry if it shows some warnings.
- Then, run the program by passing the following command after the previous command.
  ./a.out < input file with location
  Eg.
  ./a.out < /Users/aakashgarg/Desktop/input.txt
- The output will be shown on the terminal.

Compilation and execution on online compiler:

- Go to the folder and copy the text of baseballeliminator.cpp and paste it on an online compiler like onlinegdb and select the language as C++.
- Copy the input text and paste it in the input of the compiler and run the program.
- The output will be displayed.

Source Files:

I have made the assignment using the C++ language and created only the main file "**baseballeliminator.cpp**". My implementation is based on vectors and arrays for storing data and using adjacency list representation for the max flow calculation. I didn't require any object or classes in my implementation and hence have a single source file.

Overview:

In this assignment, I have implemented a max flow network to calculate the baseball eliminations as desired in the problem. First, the input is taken and stored in the vectors and arrays and then the max flow is implemented using the Ford-fulkerson algorithm using the Residual Graph and Breadth first Search. The code is implemented very efficiently both in time and space complexity through the ideas of parametric max flow and with slight modifications, this code can be used in various max flow applications.

Explanation:

- Problem Statement:
  In this problem, we are given the data of teams about their wins, losses and remaining matches and asked to predict the teams which are mathematically eliminated. A team is eliminated if it cannot possibly finish the season in first place or tied for first place.. For eg : A team with 70 wins and 5 remaining matches is mathematically eliminated if there is a team with 80 wins.

- Algorithm proposed:
  The result cannot be predicted just by seeing the total wins and remaining matches as the sequence and distribution of matches is also important. Hence, a maxflow algorithm is proposed to check whether the ith team is eliminated or not. In this, an artificial source and sink node is created and a network is formed by connecting the source node with match nodes with capacity equal to the matches remaining representing that this amount of matches are left to be played and connected the match nodes further to team nodes of that match with infinite capacity representing that any of the team can win the match and then team nodes are connected with the sink node with capacity equal to (w[i] + rem[i] - w[j]) representing that number of matches the jth team can still win. In this network, team node [i] and matches of ith team are not included. So if the maxflow is satisfied, ie maxflow is equal to the total remaining matches then the team is not eliminated as it has the possibility to finish at top otherwise it will be eliminated.

- Findings and Observations:

  - When a team is mathematically eliminated there always exists a convincing certificate of elimination i.e. we can find a subset of non eliminated teams whose average total wins is greater than the max possible wins of the eliminated team. The subset can be found by finding the min cut and taking the team nodes which are on the source side after the maxflow is computed.

  - If two teams have equal total matches(wins + remaining), then either both teams are eliminated or neither. It can be seen by above point that if a team is eliminated, we can find a subset whose average total wins is greater so if one team is eliminated, the other will also be eliminated by the subset as the subset will contain at least one team whose total wins is greater than the maximum possible wins of either team. So we can infer from it that there exists a team above which all teams whose total matches(wins + remaining) are greater than that of the team found are not eliminated and rest teams are eliminated.

Algorithm Implemented:

With the help of above observations, I have modified the proposed algorithm to make it more time efficient and easy to implement. The findings suggest that we can find a team above which all teams will not get eliminated whose total matches(wins + remaining) are greater than that of the team found. So I have imagined an artificial team having W wins and 0 remaining matches and we will vary the parameter W such that we can find all the eliminated teams with the help of this team. We will implement the proposed algorithm on this team to find whether it is eliminated or not and vary the parameter W such that we can find the W* which will tell us that below this point all teams are eliminated and above this all teams are not eliminated. As for checking a team with x wins and y remaining matches, we can set W = x+y and check for this team if it is eliminated or not as our findings ensure that either both teams get eliminated or neither. This have various benefits on the previous algorithm as :

1. All teams having equal total matches(wins + rem) get checked in a single check and hence saves the time.
2. The graph will not get changed for every team as we have imagined an extra node and we are applying the previous algorithm on it. So it will include all the nodes except the imagined node and hence connections in the graph remain intact and we only need to change the variable W.
3. Once we got the W* by iterating from lower values to higher, we don't need to check the teams having total matches(wins+rem) greater than W* and hence saves the time.
4. Once we got W*, we can find the subset of teams whose average total wins is greater than W* and this subset is valid for all the teams whose maximum possible wins are less than W*. So we don't need to find the subset for each team separately.

Conclusion:

We have seen that slight modification in the proposed algorithm makes it more efficient and easy to implement. The algorithm produces correct output on all inputs(checked by me) and performs it efficiently as it is doing less number of maxflow calculations.The output contains all the teams eliminated and an easy explanation for the elimination. The overall worst case time complexity of the algorithm is - **O(n*(max_flow complexity))** where **max_flow** complexity is **O(max_flow*Edges)** as the maxflow is implemented through **Ford-Fulkerson Algorithm** and **n is the number of teams in the input**. We have used adjacency list representation for the BFS and hence it saves the space and has **O(n^2)** space complexity.

Name - Aakash Garg
Entry No. - 2018MT60776