**Premier League Standings**

**Overall idea**

Using a directed graph and BFS approach.

The teams represent the vertices and the results of each match represent the edges between each two teams.

- By BFS to find out all the results till the given round or date.

**Pseudo code**

- Read in the match data from a file using a FileReader function

- Create a function to convert the date string in the match data to an integer format for easier comparison

- Create a function to update the round number of each match to ensure that all matches in a round have the same round number (Handle Postponed matches).

- Create a function to make a graph representation of the matches data, where each node represents a team, and each edge represents a match details.

- Create a function to calculate the standing of each team based on the match results

- Create a function to sort the teams based on their standing

- Create a function to print the standings table

- Prompt the user to choose whether to view the standings for a specific round or a specific date

- If the user chooses to view the standings for a specific round, prompt the user to enter the round number

- If the user chooses to view the standings for a specific date, prompt the user to enter the date in string format and convert it to integer format using the Converter function

- Use a BFS algorithm to traverse the graph and calculate the standings of the teams based on the chosen criteria.

Sort the teams based on their standing using the Sorting function

- Print the standings table using the PrintStanding function.

- Repeat steps 8 to 13 until the user chooses to exit the program.

**Overall Complexity**

O(E + V + Vlog V)

And because our graph is dense, so E ≤

So, O(E + V) =O(

Complexity = O()

**Detailed Functions Usage and Complexity**

1)

vector<MatchData> FileReaderx()

Parameters (in): None

Return value: Vector of struct MatchData

Description: Reads data from the CSV file

Complexity: O(E) as it reads all rows from file and each row corresponding to edge in graph.

2)

void UpdateRounds()

Parameters (in): None

: None Return value

Description: This Function update the postponed rounds number by the number of round that it actually played in.

Also, Give each team a number to make the vertices of graph by this numbers.

Complexity: O(E) as it iterate over all matches(edges) and update rounds numbers.

3)

void MakeGraph()

Parameters (in): None

Return value: None

Description: Function to create the graph as we consider each team as the vertex and the against team, result of match and goals scored as edge information.

Complexity: O(E)

4)

void init()

Parameters (in): None

Return value : None

Description: Initialize all standings to zero and give each team a number to print by later.

Complexity: O(V)

5)

void BFS(int x, int condition)

Parameters (in): The source node and condition shows round or date.

Return value: None

Description: make BFS to find all matches results before the given round or date.

Complexity: O(V+E)

6)

void Stand(int winner, int loser, MatchData Edge, char Result)

Parameters (in): winner and loser teams, all information on edge.

None Return value:

Description: This Function calculate the results for each match such as points and goals scored for each team

Complexity: O(1)

7)

void Stand(int winner, int loser, MatchData Edge, char Result)

Parameters (in): None

None Return value:

Description: Function to print The final standings.

Complexity: O(1)

8)

bool sortBy(TeamStats &a, TeamStats &b)

Parameters (in): Two teams to sort them

Boolean value Return value:

Description: Function to sort teams according to specific rules in premier league.

Complexity: O(1)

9)

void Sorting()

Parameters (in): None

Return value: None

Description: Sorts data according to sortby function

Complexity: O(Vlog V)

10)

int Converter(string date)

Parameters (in): Input date

Return value: integer number indicates the date

Description: Function to convert given date to unique number

Complexity: O(1)