|  |  |
| --- | --- |
|  |  |
| Premier League’s Project |  |
|  |  |
|  | Team members Rana Tarek Abd El Halim  Menna Allah Mohamed Kamel  Mai Adel Darwish Ahmed  Yassmin Attia Ebrahim |

Pseudocode :

|  |
| --- |
| DEFINE MAX\_TEAMS **20**  DEFINE MAX\_MATCHES **600**  STRUCT TeamVertix:  name[**20**]  edges  num\_edges  matches\_played  wins  draws  losses  goals\_for  goals\_against  goal\_difference  points  next  visited  STRUCT Edge:  opponent  round  date[**11**]  home\_goals  away\_goals  next  FUNCTION findTeamVertix(teams[], num\_teams, team\_name):  FOR i = **0** TO num\_teams - **1**:  IF teams[i].name EQUALS team\_name:  RETURN teams[i]  RETURN NULL  FUNCTION addTeamVertix(teams[], num\_teams, team\_name):  CREATE a new team vertex  SET the properties of the team vertex  ADD the team vertex to the teams array  INCREMENT num\_teams  FUNCTION addEdge(team, opponent, round, date, home\_goals, away\_goals):  CREATE a new edge  SET the properties of the edge  ADD the edge to the team vertex's edges list  INCREMENT team.num\_edges  FUNCTION parseMatchResults(file, teams[], num\_teams):  READ the first line of the file  WHILE NOT end of file:  READ a line from the file  PARSE the fields of the line using commas  GET round number  GET match date  GET home team name  GET away team name  GET home goals  IF home goals is "-":  CONTINUE (skip processing)  GET away goals  IF away goals is "-":  CONTINUE (skip processing)  FIND or ADD home team vertex in teams  FIND or ADD away team vertex in teams  ADD an edge between home and away teams  FUNCTION calculateStandingsBFS(teams[], num\_teams, round\_number):  RESET the visited flags of teams  INITIALIZE the BFS queue  ENQUEUE the first team vertex  MARK the first team vertex as visited  WHILE the queue is not empty:  DEQUEUE a team vertex  RESET the team statistics  FOR EACH edge IN the team vertex's edges:  IF the edge's round is less than or equal to the specified round number:  UPDATE team statistics based on the match result  IF the opponent team vertex has not been visited:  ENQUEUE the opponent team vertex and MARK it as visited  FUNCTION printStandings(teams[], num\_teams, round\_number):  SORT the teams array in descending order of points and goal difference  FOR i = **0** TO num\_teams - **2**:  FOR j = i + **1** TO num\_teams - **1**:  IF teams[i].points < teams[j].points OR (teams[i].points EQUALS teams[j].points AND teams[i].goal\_difference < teams[j].goal\_difference):  SWAP the positions of teams[i] and teams[j]  PRINT the standings **for** each team vertex within the specified round  **MAIN:**  OPEN the match results file  INITIALIZE the teams array and num\_teams counter  PARSE and PROCESS the match results from the file  CLOSE the file  REPEAT UNTIL the user chooses to exit:  READ the round number input  IF the round number is -**1**:  EXIT the program  IF the round number is invalid:  DISPLAY an error message and prompt again  CALCULATE the standings up to the specified round using BFS  PRINT the standings up to the specified round  EXIT the program |

##### Complexity analysis :

##### (findTeamVertix) function:

##### Complexity: O(N).

##### N = num\_teams.

##### This function performs a linear search through the teams array.

##### (addTeamVertix) function:

##### Complexity: O(1).

##### This function simply adds a team node to the teams array.

##### (addEdge) function:

##### Complexity: O(1).

##### This function adds an edge to the team node's edges list.

##### (parseMatchResults) function:

##### Complexity: O(M).

##### M = number of matches.

##### This function reads the match results from the file and creates the team vertices and edges accordingly. It loops through each line of the file.

##### (calculateStandingsBFS) function:

##### Complexity: O(N + E).

##### N = number of teams.

##### E = number of edges.

##### This function performs a breadth-first search (BFS) traversal of the teams' edges. The BFS traversal visits each team once and each edge once.

##### (printStandings) function:

##### Complexity: O(N log N).

##### N = number of teams.

##### This function sorts the teams array based on points and goal difference, which takes O(N log N) time using a comparison-based sorting algorithm.

##### Overall Complexity:

##### Considering the complexities of individual functions, the overall complexity depends on the number of matches (M), teams (N), and edges (E) as follows: O(M + N + E + N log N).

##### since the number of matches and teams is relatively small (600 matches and 20 teams), the complexity can be considered as approximately O(N log N).