Approach for Team Pairings and Match Simulation

Problem Statement

The task is to design a program to simulate a hypothetical Codewars utilizing the Swiss pairing system. The tournament consists of teams with varying ratings, each participating in 10 rounds of matches. The objective is to create a program that properly pairs teams for matches, updates the leaderboard after each round, and adheres to certain constraints and rules.

Solution Approach

1.Initial Setup:

- Import necessary libraries (`pandas` and `random`).
- Define functions to handle team pairings, winner probability calculation, score updates, rating updates and result simulation.

2.Data Loading:

- Load the team data from a CSV file (`data.csv`) into a Pandas DataFrame (`df`).

3. Pairing Logic (`pairings` function):

- Input: List of teams (`teams`) and set of previous pairings (`previous_pairings`).
- Randomly select a team to have a bye in each iteration.
- Create pairings between teams that haven't played before and have ratings closest to each other.
- Update the set of previous pairings to avoid duplicate pairings.

4. Match Simulation (`results` function):

- Simulate matches between paired teams using a random probability model based on their ratings.
- Update scores and ratings of winning and losing teams accordingly.

5. Bye Team Handling(`bye_update` function):

- Adjust ratings and scores for the team with a bye to maintain balance in the tournament.

6. Main Loop:

- Iterate through multiple rounds of pairings and match simulations.

- Apply pair teams, simulate matches, and update scores and ratings.

Implementation Steps

1. Library Imports:

```python
import pandas as pd
import random
```

2. Function Definitions:

- `pairings(teams, previous_pairings)`: Pair teams ensuring fair matches and avoid duplicate pairings.
- `calculate_winner_probability(rating_difference)`: Determine the probability of winning based on rating difference.
- `update_scores(winning_team, losing_team, scoreTable)`: Update scores after a match.
- `update_rating(winning_team, losing_team, df)`: Update ratings based on match results.
- `results(pairing, df, scoreTable)`: Simulate matches and update scores/ratings.
- `bye_update(team, df, scoreTable)`: Handle bye teams by adjusting ratings and scores.

3. Data Loading:

```python
df = pd.read_csv('data.csv')
```

4. Main Logic:

- Initialize an empty set for previous pairings (`previous_pairings`).
- Loop through multiple rounds of pairings and matches.
- Apply pair teams, simulate matches, and update scores and ratings.

Limitations of algorithm

During team pairings, there's a trade-off between ensuring unique matches and teams playing against opponents with similar ratings. This trade-off means that while the algorithm prioritizes creating new and diverse matchups to maintain excitement and variety in the tournament, there's a possibility of teams facing each other again in subsequent rounds. Specifically, out of a total of 240 matches played, approximately 8-9 matches may involve the same teams playing against each other again.

- Achieving a balance between offering diverse matchups and ensuring fair play among teams with similar ratings is a challenge.
- The algorithm attempts to strike this balance but may result in occasional repeat matchups.

Conclusion

This approach ensures fair and balanced pairings for teams, simulates matches realistically based on ratings, and maintains tournament integrity through constraints and bye team handling. However, efforts can be made to optimize the algorithm and manage expectations effectively.