


 **pinglu85** / **algoExpert** Public[<> Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)[main](#) [algoExpert](#) / [Easy](#) / **tournament-winner.md**[Go to file](#)[...](#)**pinglu85** update solutions and approachLatest commit 7c620ea on 14 Nov 2021 [History](#) 1 contributor 127 lines (89 sloc) | 5.39 KB[<>](#) [File](#) [Raw](#) [Blame](#) [View](#) [Copy](#) [Edit](#) [Delete](#)

Tournament Winner

Understanding the problem

We're asked to imagine there is an algorithms tournament taking place in which multiple teams compete against each other. In each competition there will be two teams that compete and there will be one winner and one loser out of all of these competitions; there are no ties. Each team will compete against all other teams exactly once. Every time a team wins a competition, it gets 3 points; when it loses, it gets 0 points. It's guaranteed that the tournament always has at least two teams and there will be only one tournament winner.

We are given two inputs, the `competitions` array and the `results` array, and we need to write a function that returns the winner of the tournament, or more specifically, the name of the team that has the most number of points. The `competitions` array is an array of pairs, representing all of the competitions in the tournament. Inside of these pairs, we have two strings. The first string is the name of the home team, the second string is the name of the away team. The `results` array represents the winner of each of these competitions. Inside the `results` array, a `1` means that the home team won and a `0` means the away team won. The `results` array is the same length as the `competitions` array, and the indices in the `results` array correspond with the indices in the `competitions` array.

Approach

I can use a hash table to keep track of each team's points, because a hash table is used to store key/value pairs, here each key is a team's name and the values are each team's points. Given a team name, I can retrieve its points. Moreover, hash tables provide constant-time lookup and insertion on average. Once I know all of the points for all of the teams, I can go through the hash table and figure out which team has the most amount of points and then return the name of that team.

- Initialize an empty Map to store the teams and their scores.
- Iterate through the `competitions` array. At each iteration, access the index that current competition is at in the `results` array to find out the winning team of current competition. Get the current score of the winning team from the Map. If the team is not already a key in the Map, set the current score to 0, and then add 3 points to the current score. Add the team and its current score to the Map or update its score in the Map.
- After the end of the `competitions` array is reached, iterate through the Map to find the team that has the highest score. Use a variable to keep track of the best score that I've currently seen, and use another variable to keep track of the current best team. At each iteration, if I get to a better score, set it as the current best score and set the team as the current best team. After the loop completes, return the current best team.

Time & Space Complexity

$O(n)$ time | $O(k)$ space, where n is the number of competitions and k is the number of teams.

Solution

```
// Store 1 in a constant variable, so we can use the constant variable later
// rather than having to write 1 in our program. This will make our code a
// lot more readable.
const HOME_TEAM_WON = 1;

function tournamentWinner(competitions, results) {
  const scores = new Map();

  for (let i = 0; i < competitions.length; i++) {
    const [homeTeam, awayTeam] = competitions[i];
    const result = results[i];
    const winningTeam = result === HOME_TEAM_WON ? homeTeam : awayTeam;
    const currentScore = scores.get(winningTeam) || 0;
    scores.set(winningTeam, currentScore + 3);
  }

  let currBestScore = 0;
  let currBestTeam = '';
  scores.forEach((score, team) => {
    if (score > currBestScore) {
      currBestScore = score;
      currBestTeam = team;
    }
  });

  return currBestTeam;
}
```

Improved Approach

I can solve the problem in one pass. Instead of iterating through the hash table to find the team that has the most number of points, I can keep track of the current best team while traversing the `competitions` array. To retrieve the current best team's score, I can simply look it up in the hash table.

Time & Space Complexity

$O(n)$ time | $O(k)$ space, where n is the number of competitions and k is the number of teams.

Improved Solution

```
const HOME_TEAM_WON = 1;
const POINTS = 3;

function tournamentWinner(competitions, results) {
  let currBestTeam = '';
  const scores = { [currBestTeam]: 0 };

  for (let i = 0; i < competitions.length; i++) {
    const [homeTeam, awayTeam] = competitions[i];
    const result = results[i];

    const winningTeam = result === HOME_TEAM_WON ? homeTeam : awayTeam;

    updateScores(scores, winningTeam);

    if (scores[winningTeam] > scores[currBestTeam]) {
      currBestTeam = winningTeam;
    }
  }

  return currBestTeam;
}

function updateScores(scores, team) {
```

```
    const prevScore = scores[team] || 0;
    scores[team] = prevScore + POINTS;
}
```

Improved Solution in Go

```
package main

const HOME_TEAM_WON = 1
const POINTS = 3

func TournamentWinner(competitions [][]string, results []int) string {
    currBestTeam := ""
    scores := map[string]int{currBestTeam: 0}

    for i, competition := range competitions {
        homeTeam, awayTeam := competition[0], competition[1]
        result := results[i]

        winningTeam := awayTeam
        if result == HOME_TEAM_WON {
            winningTeam = homeTeam
        }

        scores[winningTeam] += POINTS

        if scores[winningTeam] > scores[currBestTeam] {
            currBestTeam = winningTeam
        }
    }

    return currBestTeam
}
```

