Divide Players Into Teams Of Equal Skill

solved by	Senan
Platform	LeetCode
difficulty	Medium
Serial	2491
tags	Hash Map, Sorting
language	C++
solved on	@04/10/2024
link	https://leetcode.com/problems/divide-players-into-teams-of-equal-skill/
Completion	✓

Intuition

The problem requires dividing players into teams such that the sum of their skill levels is balanced across all teams. Specifically, if there are `n` players, we need to form `n / 2` teams, where the total sum of skills in each team is the same. If this is not possible, we return `-1`. We also need to maximize the total skill-product across all teams.

Approach

1. **Check if division is possible**: First, calculate the total sum of skill levels and check if it is divisible by the number of teams (`n / 2`). If not, return `-1` because equal distribution isn't possible.
2. **Pairing strategy**: To ensure equal distribution, we aim to pair players whose skill levels sum up to a predefined value, `skillsPerTeam = totalSkills / numTeams`. Using a hash map, we count the occurrences of each skill level and try to match a player with another whose skill level, when added to the first, equals `skillsPerTeam`.
3. **Calculate total skill-product**: For each valid pair, calculate the product of their skill levels and accumulate it into the answer.
4. **Early termination**: If at any point a valid pairing isn't found, return `-1`.

Complexity

Time Complexity:

- **Sorting**: Not needed here.
- **Hash map operations**: Each lookup and update operation in the hash map is `O(1)` on average.
- **Iteration**: We iterate through the skill array once, and each player is processed twice (once for themselves and once for their pair), so the time complexity is `O(n)`.

Thus, the overall time complexity is **O(n)**, where `n` is the number of players.

Space Complexity:

We are using an unordered map to store the frequency of skill levels, which takes up space proportional to the number of unique skill levels. In the worst case, the number of unique skills can be `O(n)`, so the space complexity is **O(n)**.

Code

```
class Solution {
public:
    long long dividePlayers(vector<int>& skill) {
        int n = skill.size();

        int numTeams = n / 2;
        int totalSkills = accumulate(skill.begin(), skill.end(), 0);

        if (totalSkills % numTeams != 0) return -1;
        int skillsPerTeam = totalSkills / numTeams;

        long long answer = 0;
        unordered_map<int, int> mpp;

        for (auto elem : skill) mpp[elem]++;

        for (auto elem : skill) {
            int remainder = skillsPerTeam - elem;

            if (mpp[elem]) {
                if (mpp[remainder]) {
                    answer += (long long)(elem * remainder);
                    mpp[remainder]--;
                    mpp[elem]--;
                } else {
                    return -1;                  }
            }
        }

        return answer;
    }
};
```