Leetcode Questions

Willy Zuo

October 21, 2024

Q1: 486: Predict the Winner

You are given an integer array nums. Two players are playing a game with this array: player 1 and player 2.

Player 1 and player 2 take turns, with player 1 starting first. Both players start the game with a score of 0. At each turn, the player takes one of the numbers from either end of the array (i.e., nums[0] or nums[nums.length - 1]) which reduces the size of the array by 1. The player adds the chosen number to their score. The game ends when there are no more elements in the array.

Return true if Player 1 can win the game. If the scores of both players are equal, then player 1 is still the winner, and you should also return true. You may assume that both players are playing **optimally**.

Example 1:

Input: nums = [1,5,2]

Output: false

Explanation: Initially, player 1 can choose between 1 and 2. If he chooses 2 (or 1), then player 2 can choose from 1 (or 2) and 5. If player 2 chooses 5, then player 1 will be left with 1 (or 2). So, final score of player 1 is $1 + 2 = 3$, and player 2 is 5. Hence, player 1 will never be the winner and you need to return false.

Example 2:

Input: nums = [1,5,233,7]

Output: true

Explanation: Player 1 first chooses 1. Then player 2 has to choose between 5 and 7. No matter which number player 2 choose, player 1 can choose 233. Finally, player 1 has more score (234) than player 2 (12), so you need to return True representing player 1 can win.

Constraints:

$1 \leq$ nums.length ≤ 20

$0 \leq$ nums[i] $\leq 10^7$

Solution

1 Intuition

I came up with Dynamic Programming when I noticed that each player can have flexible choices about numbers in the array because you are given the assumption that both players are playing optimally.

2 Approach

Define a two-dimension array $dp[i][j]$ as

$$dp[i][j] = \max(p_1.score - p_2.score) \tag{1}$$

in a subarray $nums[i, ..., j]$.

Considering the case when the number of entries in the nums array is even, each player gets the half of the turns, since the number of elements is even. Player 1 has positional advantage by doing first, which gives disadvantageous position to player 2. For a subarray of size 2, Player 1 can always pick the larger element.

By induction, for larger arrays of even size, Player 1 can always ensure that the "optimal" sequence of plays leads to a tie or win, thanks to their ability to make the first choice and force Player 2 into weaker positions.

Denote the left and right end of elements of the array by $nums[i]$ and $nums[j]$, respectively.

If the left end one is picked by player 1, the remaining subarray is $nums[i + 1, ..., j]$. In terms of this subarray, player 2 will try to play optimally. Thus,

$$d_1 = nums[i] - dp[i + 1][j] \tag{2}$$

.

Likewise, if the right end one is picked by player 1, the

$$d_2 = nums[j] - dp[i][j - 1] \tag{3}$$

.

Hence, we can obtain the equation about

$$\begin{aligned} dp[i][j] &= \max(d_1, d_2) \\ &= \max(nums[i] - dp[i + 1][j], nums[j] - dp[i][j - 1]) \end{aligned}$$

Considering that if there is only one element in the array, we have $dp[i][i] = nums[i]$.

Hence, if player 1 can win the game, we must ensure that dp[0][nums.length$-$ 1] ≥ 0 (p1.score $\geq$ p2.score).

3 Accepted Code

```cpp
class Solution {
public:
    bool predictTheWinner(vector<int>& nums) {
        int n = nums.size();

        vector<vector<int>> dp(n, vector<int>(n, 0));
        for(int i = 0; i <= n - 1; ++i)
            dp[i][i] = nums[i];

        for(int sub_length = 2; sub_length <= n; ++sub_length){
            for(int i = 0; i <= n - sub_length; ++i){
                int j = sub_length + i - 1;
                dp[i][j] = max(nums[i] - dp[i+1][j], nums[j] -
                ↪  dp[i][j - 1]);
            }
        }

        return dp[0][n - 1] >= 0;
    }
};
```