题目：

Given an array of scores that are non-negative integers. Player 1 picks one of the numbers from either end of the array followed by the player 2 and then player 1 and so on. Each time a player picks a number, that number will not be available for the next player. This continues until all the scores have been chosen. The player with the maximum score wins.

Given an array of scores, predict whether player 1 is the winner. You can assume each player plays to maximize his score.

Example 1:

```
Input: [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 play
er 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: [1, 5, 233, 7]

Output: True

Explanation: Player 1 first chooses 1. Then player 2 have 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 re
turn True representing player1 can win.
```

Note:

1. 1 <= length of the array <= 20.
2. Any scores in the given array are non-negative integers and will not exceed 10,000,000.
3. If the scores of both players are equal, then player 1 is still the winner.

```
/* 参考<<程序员代码面试指南>>p233 */
1.时间：O（2^N）;空间：O（N）
class Solution {
public:
    bool PredictTheWinner(vector<int>& nums) {
        if(nums.empty()) return true;
        return first(nums, 0, nums.size() - 1) >= second(nums, 0, nums.size() - 1);
    }
private:
    int first(const std::vector<int>& nums, int start, int end){
        if (start == end) return nums[start];
        return std::max(nums[start] + second(nums, start + 1, end), nums[end] + second(nums, start, end - 1));
    }
    int second(const std::vector<int>& nums, int start, int end){
        if (start == end) return 0;
        return std::min(first(nums, start + 1, end), first(nums, start, end - 1));
    }
};
```

2.时间：O（N^2）;空间：O（N^2）

```
class Solution {
public:
    bool PredictTheWinner(vector<int>& nums) {
        if (nums.empty()) return true;
        std::vector<std::vector<int>>                    first(nums.size(),
std::vector<int>(nums.size(), 0));
        std::vector<std::vector<int>> second = first;
        for (int k = 0; k < nums.size(); ++k){
            first[k][k] = nums[k];
            for (int i = k - 1; i >= 0; --i){
                first[i][k] = std::max(nums[i] + second[i + 1][k], nums[k]
+ second[i][k - 1]);
                 second[i][k] = std::min(first[i+1][k], first[i][k-1]);
            }
        }
        return first[0][nums.size() - 1] >= second[0][nums.size() - 1];
    }
};
```