Problem 1406: Stone Game III

Problem Information

Difficulty: Hard
Acceptance Rate: 63.22%
Paid Only: No
Tags: Array, Math, Dynamic Programming, Game Theory

Problem Description

Alice and Bob continue their games with piles of stones. There are several stones **arranged in a row** , and each stone has an associated value which is an integer given in the array `stoneValue`.

Alice and Bob take turns, with Alice starting first. On each player's turn, that player can take `1`, `2`, or `3` stones from the **first** remaining stones in the row.

The score of each player is the sum of the values of the stones taken. The score of each player is `0` initially.

The objective of the game is to end with the highest score, and the winner is the player with the highest score and there could be a tie. The game continues until all the stones have been taken.

Assume Alice and Bob **play optimally**.

Return `"Alice"`_if Alice will win,_`"Bob"`_if Bob will win, or_`"Tie"`_if they will end the game with the same score_.

Example 1:

Input: stoneValue = [1,2,3,7] **Output:** "Bob" **Explanation:** Alice will always lose. Her best move will be to take three piles and the score become 6. Now the score of Bob is 7 and Bob wins.

Example 2:

Input: stoneValue = [1,2,3,-9] **Output:** "Alice" **Explanation:** Alice must choose all the three piles at the first move to win and leave Bob with negative score. If Alice chooses one pile her score will be 1 and the next move Bob's score becomes 5. In the next move, Alice will take the pile with value = -9 and lose. If Alice chooses two piles her score will be 3 and the next move Bob's score becomes 3. In the next move, Alice will take the pile with value = -9 and also lose. Remember that both play optimally so here Alice will choose the scenario that makes her win.

Example 3:

Input: stoneValue = [1,2,3,6] **Output:** "Tie" **Explanation:** Alice cannot win this game. She can end the game in a draw if she decided to choose all the first three piles, otherwise she will lose.

Constraints:

* `1 <= stoneValue.length <= 5 * 104` * `-1000 <= stoneValue[i] <= 1000`

Code Snippets

C++:

```
class Solution {
public:
string stoneGameIII(vector<int>& stoneValue) {

}
};
```

Java:

```
class Solution {
public String stoneGameIII(int[] stoneValue) {

}
}
```

Python3:

```
class Solution:
def stoneGameIII(self, stoneValue: List[int]) -> str:
```