1140. Stone Game II

Medium

Topics

Companies

Hint

Alice and Bob continue their games with piles of stones. There are a number of piles **arranged in a row**, and each pile has a positive integer number of stones `piles[i]`. The objective of the game is to end with the most stones.

Alice and Bob take turns, with Alice starting first. Initially, `M = 1`.

On each player's turn, that player can take **all the stones** in the **first** `X` remaining piles, where `1 <= X <= 2M`. Then, we set `M = max(M, X)`.

The game continues until all the stones have been taken.

Assuming Alice and Bob play optimally, return the maximum number of stones Alice can get.

Example 1:

Input: piles = [2,7,9,4,4]
Output: 10
Explanation: If Alice takes one pile at the beginning, Bob takes two piles, then Alice takes 2 piles again. Alice can get 2 + 4 + 4 = 10 piles in total. If Alice takes two piles at the beginning, then Bob can take all three piles left. In this case, Alice get 2 + 7 = 9 piles in total. So we return 10 since it's larger.

Example 2:

Input: piles = [1,2,3,4,5,100]
Output: 104

Constraints:

- `1 <= piles.length <= 100`
- `1 <= piles[i] <= 104`

Solution:

```
class Solution {
    public int stoneGameII(int[] piles) {
```

```
        int n = piles.length;
        int[][] dp = new int[n][n + 1];
        int[] suffixSum = new int[n];
        suffixSum[n - 1] = piles[n - 1];
        for (int i = n - 2; i >= 0; i--) {
            suffixSum[i] = suffixSum[i + 1] + piles[i];
        }
        for (int i = n - 1; i >= 0; i--) {
            for (int m = 1; m <= n; m++) {
                if (i + 2 * m >= n) {
                    dp[i][m] = suffixSum[i];
                } else {
                    for (int x = 1; x <= 2 * m; x++) {
                        dp[i][m] = Math.max(dp[i][m], suffixSum[i] - dp[i +
x][Math.max(m, x)]);
                    }
                }
            }
        }
        return dp[0][1];
    }
}
```