1140. Stone Game II

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

Code:

```
//A trick that simplifies implementation is realising that Alice's, say, total number of stones is equal to the sum of remaining stones minus Bob's optimal number of stones after Alice's move.


class Solution

{

public:

   int stoneGameII(vector<int>& piles)

   {
        int n = int(piles.size());

        vector<vector<int>> dp(n+1,vector<int>(n+1));

        for(int i=n-1,sum=0;i>=0;--i)
        {
            sum+=piles[i];

            for(int m=1;m<=n;++m)
            {
                for(int x=1;x<=min(n,2*m);++x)
                {
                    dp[i][m]=max(dp[i][m], sum-dp[min(i+x,n)][max(m,x)]);
                }
            }
        }
        return dp[0][1];
    }
};
```