MCS – 253P ADVANCED PROGRAMMING AND PROBLEM SOLVING

LAB 10 Code(Stone Game V)

Aswin Sampath
saswin@uci.edu
53844684

Question)

Problem List | Run | Submit

Description | Editorial | Solutions | Submissions

1563. Stone Game V

Solved

Hard | Topics | Companies | Hint

There are several stones **arranged in a row**, and each stone has an associated value which is an integer given in the array `stoneValue`.

In each round of the game, Alice divides the row into **two non-empty rows** (i.e. left row and right row), then Bob calculates the value of each row which is the sum of the values of all the stones in this row. Bob throws away the row which has the maximum value, and Alice's score increases by the value of the remaining row. If the value of the two rows are equal, Bob lets Alice decide which row will be thrown away. The next round starts with the remaining row.

The game ends when there is only **one stone remaining**. Alice's is initially **zero**.

Return *the maximum score that Alice can obtain*.

Example 1:

```
Input: stoneValue = [6,2,3,4,5,5]
Output: 18
Explanation: In the first round, Alice divides the row to [6,2,3], [4,5,5]. The left row has the value 11 and the right row has value 14. Bob throws away the right row and Alice's score is now 11.
In the second round Alice divides the row to [6], [2,3]. This time Bob throws away the left row and Alice's score becomes 16 (11 + 5).
The last round Alice has only one choice to divide the row which is [2], [3]. Bob throws away the right row and Alice's score is now 18 (16 + 2). The game ends because only one stone is remaining in the row.
```

Example 2:

```
Input: stoneValue = [7,7,7,7,7,7,7]
Output: 28
```

Example 3:

```
Input: stoneValue = [4]
Output: 0
```

Constraints:

Output:

Solutions | Submissions | Accepted ×

Aswin submitted at Dec 11, 2023 14:42

Accepted

Editorial | + Solution

Runtime

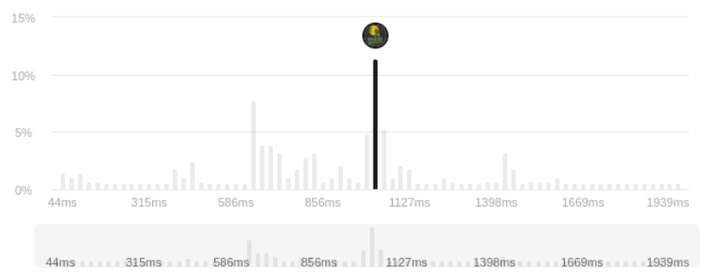
1030 ms

Beats **35.82%** of users with C++

Memory

23.67 MB

Beats **46.10%** of users with C++



C++

```
class Solution {
```

Code

```cpp
class Solution {
public:
    int stoneGameV(vector<int>& a) {
        int n = a.size();
        if (n == 1)
            return 0;

        // Calculate the prefix sum of the array for quick range sum queries.
        vector<int> prefixSum(n + 1, 0);
        for (int i = 0; i < n; ++i) {
            prefixSum[i + 1] = prefixSum[i] + a[i];
        }

        // Initialize the memoization table to store the results of subproblems.
        vector<vector<int>> memo(n, vector<int>(n, -1));

        return stoneGameHelper(a, prefixSum, memo, 0, n - 1);
    }

    int stoneGameHelper(vector<int>& a, vector<int>& prefixSum, vector<vector<int>>& memo, int left, int right) {
        if (left == right)
            return 0;

        if (memo[left][right] != -1)
            return memo[left][right];

        int result = 0;
        for (int i = left; i < right; ++i) {
            int leftSum = prefixSum[i + 1] - prefixSum[left];
            int rightSum = prefixSum[right + 1] - prefixSum[i + 1];

            if (leftSum < rightSum) {
                result = max(result, leftSum + stoneGameHelper(a, prefixSum, memo, left, i));
            } else if (leftSum > rightSum) {
                result = max(result, rightSum + stoneGameHelper(a, prefixSum, memo, i + 1, right));
            } else {
                // If leftSum and rightSum are equal, consider both divisions.
                result = max(result, leftSum + max(
                    stoneGameHelper(a, prefixSum, memo, left, i),
                    stoneGameHelper(a, prefixSum, memo, i + 1, right)
                ));
            }
        }

        memo[left][right] = result;
        return result;
    }
};
```