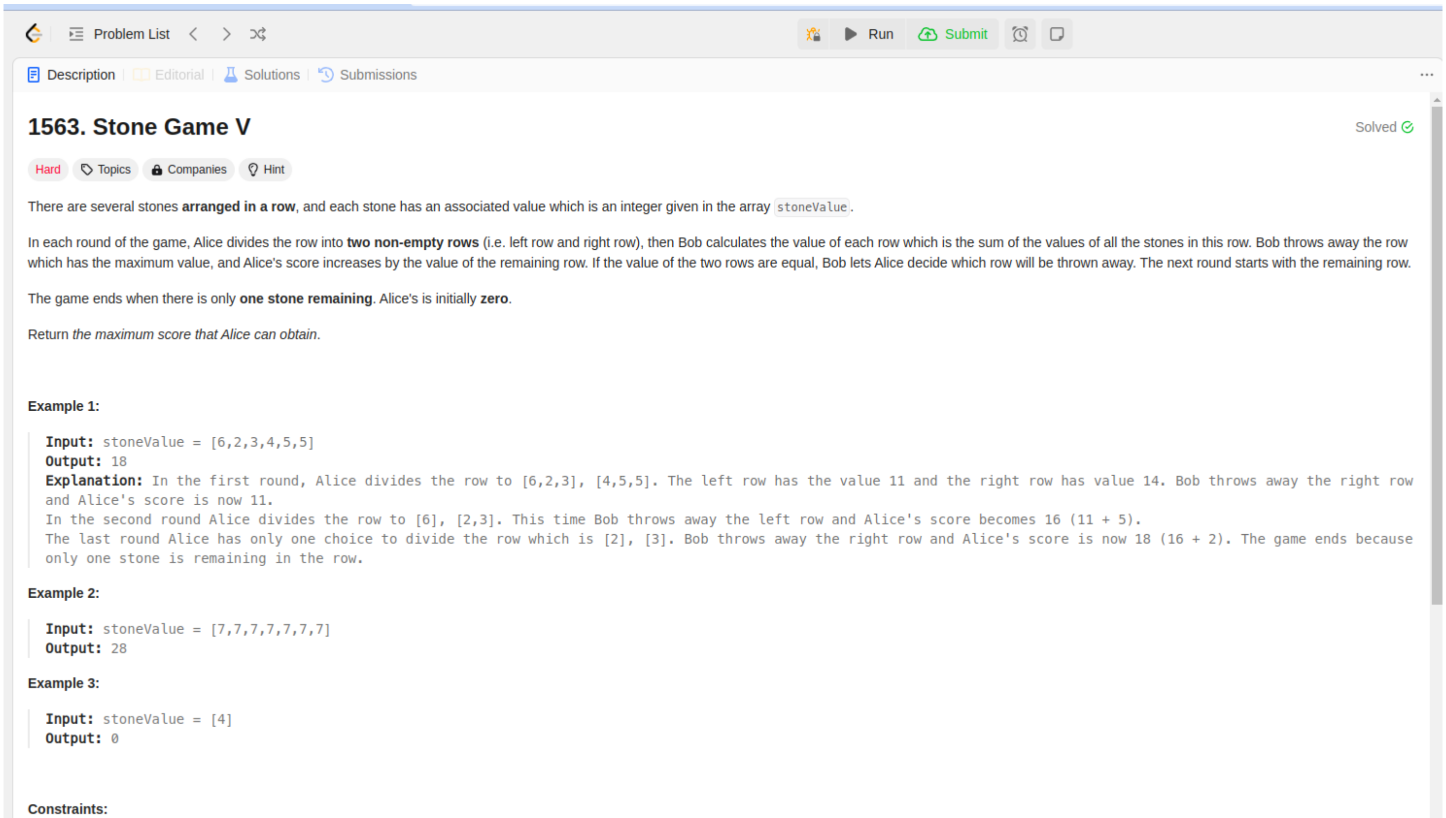
MCS – 253P ADVANCED PROGRAMMING AND PROBLEM SOLVING

LAB 10 Writeup(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:

Understanding the Problem

The problem involves a game where Alice and Bob take turns dividing a row of stones into two separate rows until there's only one stone left. Each stone has an associated value. Alice aims to maximize her score by choosing how to divide the rows, and Bob strategically discards the row with the higher value, thereby increasing Alice's score. The goal is to determine the maximum score Alice can achieve at the end of the game.

Effective Test Cases

Case 1:

Input: stoneValue = [6,2,3,4,5,5]

Expected Output: 18

Explanation:

The iterative division process results in the optimal sequence leading to Alice's maximum score of 18.

Case 2:

Input: stoneValue = [7,7,7,7,7,7,7]

Expected Output: 28

Explanation:

All stones have equal values, presenting a scenario where Alice continually chooses the optimal division to achieve the maximum score of 28.

Case 3:

Input: stoneValue = [4]

Expected Output: 0

Explanation:

With only one stone, there are no divisions possible, resulting in Alice's score remaining at 0.

Identifying Edge Cases

- There's only one stone.
- All stones have equal values.
- The number of stones is minimal or close to the upper constraint of 500.

Algorithm

- Utilizing prefix sums for efficient range sum queries.
- Employing a helper function (stoneGameHelper) to recursively determine the maximum score for each possible division.
- Memoization is applied to store and reuse the results of subproblems to enhance computational efficiency.

Time and Space Complexity

Time Complexity: The time complexity of the algorithm is O(n^3), where 'n' is the number of stones. This arises due to the nested loops and recursive calls in the memoization process.

Space Complexity: The space complexity is O(n^2) as the memoization table is of size 'n x n', storing results for all subproblems.