Problem 3180: Maximum Total Reward Using Operations I

Problem Information

Difficulty: Medium
Acceptance Rate: 30.42%
Paid Only: No
Tags: Array, Dynamic Programming

Problem Description

You are given an integer array `rewardValues` of length `n`, representing the values of rewards.

Initially, your total reward `x` is 0, and all indices are **unmarked**. You are allowed to perform the following operation **any** number of times:

* Choose an **unmarked** index `i` from the range `[0, n - 1]`. * If `rewardValues[i]` is **greater** than your current total reward `x`, then add `rewardValues[i]` to `x` (i.e., `x = x + rewardValues[i]`), and **mark** the index `i`.

Return an integer denoting the **maximum** _total reward_ you can collect by performing the operations optimally.

Example 1:

Input: rewardValues = [1,1,3,3]

Output: 4

Explanation:

During the operations, we can choose to mark the indices 0 and 2 in order, and the total reward will be 4, which is the maximum.

Example 2:

Input: rewardValues = [1,6,4,3,2]

Output: 11

Explanation:

Mark the indices 0, 2, and 1 in order. The total reward will then be 11, which is the maximum.

Constraints:

* `1 <= rewardValues.length <= 2000` * `1 <= rewardValues[i] <= 2000`

Code Snippets

C++:

```
class Solution {
public:
int maxTotalReward(vector<int>& rewardValues) {

}
};
```

Java:

```
class Solution {
public int maxTotalReward(int[] rewardValues) {

}
}
```

Python3:

```
class Solution:
def maxTotalReward(self, rewardValues: List[int]) -> int:
```