Solution for the valley flood problem:

Author: Sean Smith

This solution iterates through the array of height data and establishes left and right boundaries that mark valleys. Then, it computes the maximum height for each valley and calculates the volume within, accumulating this in a total volume variable which is returned.

This algorithm will operate in **linear time**, O(n), and requires **constant memory**, O(1).

Test Cases for Algorithm:

```
var example = [2, 4, 5, 2, 3, 4, 6, 6, 5]; // should return 6 units
var valley1 = [2, 4, 5, 2, 3, 4, 6, 6, 4, 5]; // should return 7 units
var valley2 = [9, 8, 7, 6, 5, 5, 6, 7, 8, 9]; // should return 20 units
var valley3 = [3, 2, 1, 2]; // should return 1 unit
var valley4 = [5, 4, 3, 2, 1, 5]; // should return 10 units
var valley5 = [5, 1, 2, 3, 4, 5]; // should return 10 units

// additional valleys for testing (mostly edge cases):
var valley6 = [5]; // should return 0 units
var valley7 = [5, 3]; // should return 0 units
var valley8 = [5, 4, 7]; // should return 1 unit
var valley9 = [5, 1, 7]; // should return 4 units
var valley10 = [9, 8, 7, 6, 4, 4, 6, 7, 8, 9]; // should return 22 units

// run tests:
console.log('Example:',  valleyFlood(example)); // 6
console.log('Valley 1:', valleyFlood(valley1)); // 7
console.log('Valley 2:', valleyFlood(valley2)); // 20
console.log('Valley 3:', valleyFlood(valley3)); // 1
console.log('Valley 4:', valleyFlood(valley4)); // 10
console.log('Valley 5:', valleyFlood(valley5)); // 10
console.log('Valley 6:', valleyFlood(valley6)); // 0
console.log('Valley 7:', valleyFlood(valley7)); // 0
console.log('Valley 8:', valleyFlood(valley8)); // 1
console.log('Valley 9:', valleyFlood(valley9)); // 4
console.log('Valley 10:', valleyFlood(valley10)); // 22
```

```
function valleyFlood(array) {
    // this will track the boundaries for one (or more) valleys we encounter
    let bounds = [];

    // variables to track left and right valley bounds
    let leftBound = null;
    let rightBound = null;

    // iterate through the array, establish left/right boundaries for valleys
    for (let i = 0; i < array.length; i++) {
        if (array[i + 1] < array[i] && leftBound === null) {
            leftBound = i;
        }
        if (array[i + 1] > array[i]) {
            rightBound = i;
            if (array[i + 1] >= array[leftBound]) {
                // multiple valleys could exist, so once we have found one valley
                // we will reset the bounds in case we find another
                bounds.push([leftBound, rightBound + 1]);
                leftBound = null;
                rightBound = null;
            }
        }
    }

    // handle edge case for a partial valley on the right edge of array
    if (rightBound != null) bounds.push([leftBound, rightBound + 1]);

    // establish variable to track total volume
    let volume = 0;

    // helper function to calculate volume based on left and right bounds
    // this determines maximum height and then calculates the enclosed volume
    function calculateVolume(left, right) {
        let height = (array[left] < array[right]) ? array[left] : array[right];
        for (let i = left; i < right; i++) {
            if (array[i] < height) {
                volume += height - array[i];
            }
        }
    }

    // calculate the total volume for each valley we found
    bounds.forEach(bounds => calculateVolume(bounds[0], bounds[1]));

    return volume;
}
```