Trapping Rain Water

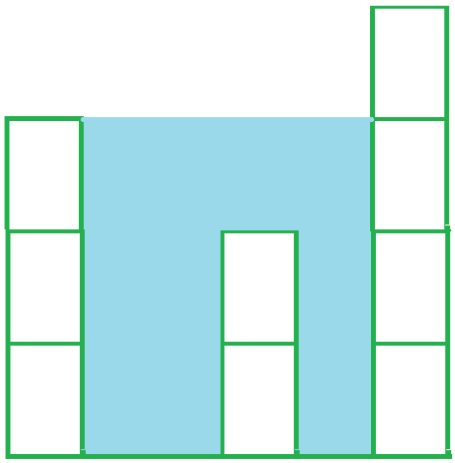
Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

Examples:

```
Input: arr[]   = {2, 0, 2}
Output: 2
Structure is like below
| |
|_|
We can trap 2 units of water in the middle gap.

Input: arr[]   = {3, 0, 0, 2, 0, 4}
Output: 10
Structure is like below
     |
|    |
|  | |
|__|_|
We can trap "3*2 units" of water between 3 an 2,
"1 unit" on top of bar 2 and "3 units" between 2
and 4.  See below diagram also.

Input: arr[] = [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]
Output: 6
       |
   |   || |
_|_||_||||||
Trap "1 unit" between first 1 and 2, "4 units" between
first 2 and 3 and "1 unit" between second last 1 and last 2
```



Bars for input {3, 0, 0, 2, 0, 4}
Total trapped water = 3 + 3 + 1 + 3 = 10

We strongly recommend that you click here and practice it, before moving on to the solution.

An element of array can store water if there are higher bars on left and right. We can find amount of water to be stored in every element by finding the heights of bars on left and right sides. The idea is to compute amount of water that can be stored in every element of array. For example, consider the array {3, 0, 0, 2, 0, 4}, we can store two units of water at indexes 1 and 2, and one unit of water at index 2.

A **Simple Solution** is to traverse every array element and find the highest bars on left and right sides. Take the smaller of two heights. The difference between smaller height and height of current element is the amount of water that can be stored in this array element. Time complexity of this solution is $O(n^2)$.

An **Efficient Solution** is to prre-compute highest bar on left and right of every bar in O(n) time. Then use these pre-computed values to find the amount of water in every array element. Below is C++ implementation of this solution.

C++

```
// C++ program to find maximum amount of water that can
// be trapped within given set of bars.
#include<bits/stdc++.h>
using namespace std;

int findWater(int arr[], int n)
{
    // left[i] contains height of tallest bar to the
    // left of i'th bar including itself
    int left[n];

    // Right [i] contains height of tallest bar to
    // the right of ith bar including itself
    int right[n];

    // Initialize result
    int water = 0;
```

```
    // Fill left array
    left[0] = arr[0];
    for (int i = 1; i < n; i++)
       left[i] = max(left[i-1], arr[i]);

    // Fill right array
    right[n-1] = arr[n-1];
    for (int i = n-2; i >= 0; i--)
       right[i] = max(right[i+1], arr[i]);

    // Calculate the accumulated water element by element
    // consider the amount of water on i'th bar, the
    // amount of water accumulated on this particular
    // bar will be equal to min(left[i], right[i]) - arr[i] .
    for (int i = 0; i < n; i++)
       water += min(left[i],right[i]) - arr[i];

    return water;
}

// Driver program
int main()
{
    int arr[] = {0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "Maximum water that can be accumulated is "
         << findWater(arr, n);
    return 0;
}
```

Java

```
// Java program to find maximum amount of water that can
// be trapped within given set of bars.

class Test
{
    static int arr[] = new int[]{0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1};

    // Method for maximum amount of water
    static int findWater(int n)
    {
        // left[i] contains height of tallest bar to the
        // left of i'th bar including itself
        int left[] = new int[n];

        // Right [i] contains height of tallest bar to
        // the right of ith bar including itself
        int right[] = new int[n];

        // Initialize result
        int water = 0;

        // Fill left array
        left[0] = arr[0];
        for (int i = 1; i < n; i++)
           left[i] = Math.max(left[i-1], arr[i]);

        // Fill right array
        right[n-1] = arr[n-1];
        for (int i = n-2; i >= 0; i--)
           right[i] = Math.max(right[i+1], arr[i]);

        // Calculate the accumulated water element by element
        // consider the amount of water on i'th bar, the
        // amount of water accumulated on this particular
        // bar will be equal to min(left[i], right[i]) - arr[i] .
        for (int i = 0; i < n; i++)
           water += Math.min(left[i],right[i]) - arr[i];
```

```
        return water;
    }

    // Driver method to test the above function
    public static void main(String[] args)
    {

        System.out.println("Maximum water that can be accumulated is " +
                                        findWater(arr.length));
    }
}
```

Python3

```
# Python program to find maximum amount of water that can
# be trapped within given set of bars.

def findWater(arr, n):

    # left[i] contains height of tallest bar to the
    # left of i'th bar including itself
    left = [0]*n

    # Right [i] contains height of tallest bar to
    # the right of ith bar including itself
    right = [0]*n

    # Initialize result
    water = 0

    # Fill left array
    left[0] = arr[0]
    for i in range( 1, n):
        left[i] = max(left[i-1], arr[i])

    # Fill right array
    right[n-1] = arr[n-1]
    for i in range(n-2, -1, -1):
        right[i] = max(right[i+1], arr[i]);

    # Calculate the accumulated water element by element
    # consider the amount of water on i'th bar, the
    # amount of water accumulated on this particular
    # bar will be equal to min(left[i], right[i]) - arr[i] .
    for i in range(0, n):
        water += min(left[i],right[i]) - arr[i]

    return water


# Driver program

arr = [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]
n = len(arr)
print("Maximum water that can be accumulated is",findWater(arr, n))

# This code is contributed by
# Smitha Dinesh Semwal
```

C#

```
// C# program to find maximum amount of water that can
// be trapped within given set of bars.
using System;
```

```
class Test
{
    static int []arr = new int[]{0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1};

    // Method for maximum amount of water
    static int findWater(int n)
    {
        // left[i] contains height of tallest bar to the
        // left of i'th bar including itself
        int []left = new int[n];

        // Right [i] contains height of tallest bar to
        // the right of ith bar including itself
        int []right = new int[n];

        // Initialize result
        int water = 0;

        // Fill left array
        left[0] = arr[0];
        for (int i = 1; i < n; i++)
        left[i] = Math.Max(left[i - 1], arr[i]);

        // Fill right array
        right[n - 1] = arr[n - 1];
        for (int i = n-2; i >= 0; i--)
        right[i] = Math.Max(right[i + 1], arr[i]);

        // Calculate the accumulated water element by element
        // consider the amount of water on i'th bar, the
        // amount of water accumulated on this particular
        // bar will be equal to min(left[i], right[i]) - arr[i] .
        for (int i = 0; i < n; i++)
        water += Math.Min(left[i],right[i]) - arr[i];

        return water;
    }

    // Driver method to test the above function
    public static void Main()
    {

        Console.WriteLine("Maximum water that can be accumulated is " +
                          findWater(arr.Length));
    }
}

// This code is contributed by vt_m.
```

PHP

```
<?php
// PHP program to find maximum
// amount of water that can
// be trapped within given set of bars.

function findWater($arr, $n)
{

    // left[i] contains height of
    // tallest bar to the
    // left of i'th bar including
    // itself

    // Right [i] contains height
    // of tallest bar to the right
    // of ith bar including itself
    // $right[$n];
```

```
    // Initialize result
    $water = 0;

    // Fill left array
    $left[0] = $arr[0];
    for ($i = 1; $i < $n; $i++)
    $left[$i] = max($left[$i - 1],
                    $arr[$i]);

    // Fill right array
    $right[$n - 1] = $arr[$n - 1];
    for ($i = $n - 2; $i >= 0; $i--)
    $right[$i] = max($right[$i + 1],
                     $arr[$i]);

    // Calculate the accumulated
    // water element by element
    // consider the amount of
    // water on i'th bar, the
    // amount of water accumulated
    // on this particular
    // bar will be equal to min(left[i],
    // right[i]) - arr[i] .
    for ($i = 0; $i < $n; $i++)
    $water += min($left[$i], $right[$i])
                          - $arr[$i];

    return $water;
}

    // Driver program
    $arr = array(0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1);
    $n = sizeof($arr);
    echo "Maximum water that can be accumulated is ",
        findWater($arr, $n);

// This code is contributed by ajit
?>
```

Output:

```
Maximum water that can be accumulated is 6
```

Time Complexity: O(n)

Auxiliary Space: O(n)

Space Optimization in above solution :

Instead of maintaing two arrays of size n for storing left and right max of each element, we will just maintain two variables to store the maximum till that point. Since water trapped at any element = min(max_left, max_right) – arr[i] we will calculate water trapped on smaller element out of A[lo] and A[hi] first and move the pointers till *lo* doesn't cross *hi*.

C++

```
// C++ program to find maximum amount of water that can
// be trapped within given set of bars.
// Space Complexity : O(1)

#include<iostream>
using namespace std;

int findWater(int arr[], int n)
{
    // initialize output
    int result = 0;
```

```
    // maximum element on left and right
    int left_max = 0, right_max = 0;

    // indices to traverse the array
    int lo = 0, hi = n-1;

    while(lo <= hi)
    {
        if(arr[lo] < arr[hi])
        {
            if(arr[lo] > left_max)
            // update max in left
            left_max = arr[lo];
            else
            // water on curr element = max - curr
            result += left_max - arr[lo];
            lo++;
        }
        else
        {
            if(arr[hi] > right_max)
            // update right maximum
            right_max = arr[hi];
            else
            result += right_max - arr[hi];
            hi--;
        }
    }

    return result;
}

int main()
{
    int arr[] = {0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "Maximum water that can be accumulated is "
        << findWater(arr, n);
}

// This code is contributed by Aditi Sharma
```

Java

```
// JAVA Code For Trapping Rain Water
import java.util.*;

class GFG {

    static int findWater(int arr[], int n)
    {
        // initialize output
        int result = 0;

        // maximum element on left and right
        int left_max = 0, right_max = 0;

        // indices to traverse the array
        int lo = 0, hi = n-1;

        while(lo <= hi)
        {
            if(arr[lo] < arr[hi])
            {
                if(arr[lo] > left_max)

                // update max in left
                left_max = arr[lo];
                else
```

```
                // water on curr element =
                // max - curr
                result += left_max - arr[lo];
                lo++;
            }
            else
            {
                if(arr[hi] > right_max)

                // update right maximum
                right_max = arr[hi];

                else
                result += right_max - arr[hi];
                hi--;
            }
        }

        return result;
    }

    /* Driver program to test above function */
    public static void main(String[] args)
    {
        int arr[] = {0, 1, 0, 2, 1, 0, 1,
                     3, 2, 1, 2, 1};
        int n = arr.length;

        System.out.println("Maximum water that "
                       + "can be accumulated is "
                       + findWater(arr, n));

    }
}
// This code is contributed by Arnav Kr. Mandal.
```

Python3

```
# Python program to find
# maximum amount of water that can
# be trapped within given set of bars.
# Space Complexity : O(1)

def findWater(arr,n):

    # initialize output
    result = 0

    # maximum element on left and right
    left_max = 0
    right_max = 0

    # indices to traverse the array
    lo = 0
    hi = n-1

    while(lo <= hi):

        if(arr[lo] < arr[hi]):

            if(arr[lo] > left_max):

                # update max in left
                left_max = arr[lo]
            else:

                # water on curr element = max - curr
                result += left_max - arr[lo]
```

```
            lo+=1

        else:

            if(arr[hi] > right_max):
                # update right maximum
                right_max = arr[hi]
            else:
                result += right_max - arr[hi]
            hi-=1

    return result

# Driver program

arr = [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]
n = len(arr)

print("Maximum water that can be accumulated is ",
        findWater(arr, n))

# This code is contributed
# by Anant Agarwal.
```

C#

```
// C# Code For Trapping Rain Water
using System;

class GFG {

    static int findWater(int []arr, int n)
    {
        // initialize output
        int result = 0;

        // maximum element on left and right
        int left_max = 0, right_max = 0;

        // indices to traverse the array
        int lo = 0, hi = n - 1;

        while(lo <= hi)
        {
            if(arr[lo] < arr[hi])
            {
                if(arr[lo] > left_max)

                // update max in left
                left_max = arr[lo];
                else

                // water on curr element =
                // max - curr
                result += left_max - arr[lo];
                lo++;
            }
            else
            {
                if(arr[hi] > right_max)

                // update right maximum
                right_max = arr[hi];

                else
                result += right_max - arr[hi];
                hi--;
            }
        }
```

```
        return result;
    }

    // Driver program
    public static void Main()
    {
        int []arr = {0, 1, 0, 2, 1, 0, 1,
                     3, 2, 1, 2, 1};
        int n = arr.Length;

        Console.WriteLine("Maximum water that "
                          + "can be accumulated is "
                          + findWater(arr, n));

    }
}

// This code is contributed by vt_m.
```

PHP

```
<?php
// PHP program to find maximum amount
// of water that can be trapped within
// given set of bars.

// Method to find maximum amount
// of water that can be trapped within
// given set of bars.
function findWater($arr, $n)
{

    // initialize output
    $result = 0;

    // maximum element on
    // left and right
    $left_max = 0;
    $right_max = 0;

    // indices to traverse
    // the array
    $lo = 0; $hi = $n - 1;

    while($lo <= $hi)
    {
        if($arr[$lo] < $arr[$hi])
        {
            if($arr[$lo] > $left_max)

                // update max in left
                $left_max = $arr[$lo];

            else

                // water on curr
                // element = max - curr
                $result += $left_max - $arr[$lo];
                $lo++;
        }
        else
        {
            if($arr[$hi] > $right_max)

                // update right maximum
                $right_max = $arr[$hi];
            else
                $result += $right_max - $arr[$hi];
```

```
                $hi--;
            }
        }

        return $result;
    }

        // Driver Code
        $arr = array(0, 1, 0, 2, 1, 0,
                     1, 3, 2, 1, 2, 1);
        $n = count($arr);
        echo "Maximum water that can be accumulated is "
                                , findWater($arr, $n);

    // This code is contributed by anuj_67.
    ?>
```

Output:

```
Maximum water that can be accumulated is 6
```

Thanks to Gaurav Ahirwar and **Aditi Sharma** for above solution.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

Maximum litres of water that can be bought with N Rupees

Program to find amount of water in a given glass

Minimizing array sum by applying XOR operation on all elements of the array

Arrange array elements such that last digit of an element is equal to first digit of the next element

Amazon SDE-2 interview experience

Find element with the maximum set bits in an array

Count pairs with average present in the same array

Microsoft Interview experience for full time position of software engineer at Microsoft Ireland Research

Given two arrays count all pairs whose sum is an odd number

Maximum length of subarray such that sum of the subarray is even

Reorder the position of the words in alphabetical order

Print the final string when minimum value strings get concatenated in every operation

Sliding Window Maximum (Maximum of all subarrays of size k) using stack in O(n) time

Maximize the subarray sum after multiplying all elements of any subarray with X

Improved By : jit_t, vt_m

Dearborn, MI: Residents Who Drive A LEXUS RX 350 Should Check This Out

Search Rates Now

Article Tags : Arrays Accolite Adobe Amazon D-E-Shaw Microsoft Payu

Practice Tags : Accolite Microsoft Amazon D-E-Shaw Payu Adobe Arrays

31

To-do Done

3.5

Based on **310** vote(s)

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments | Share this post!