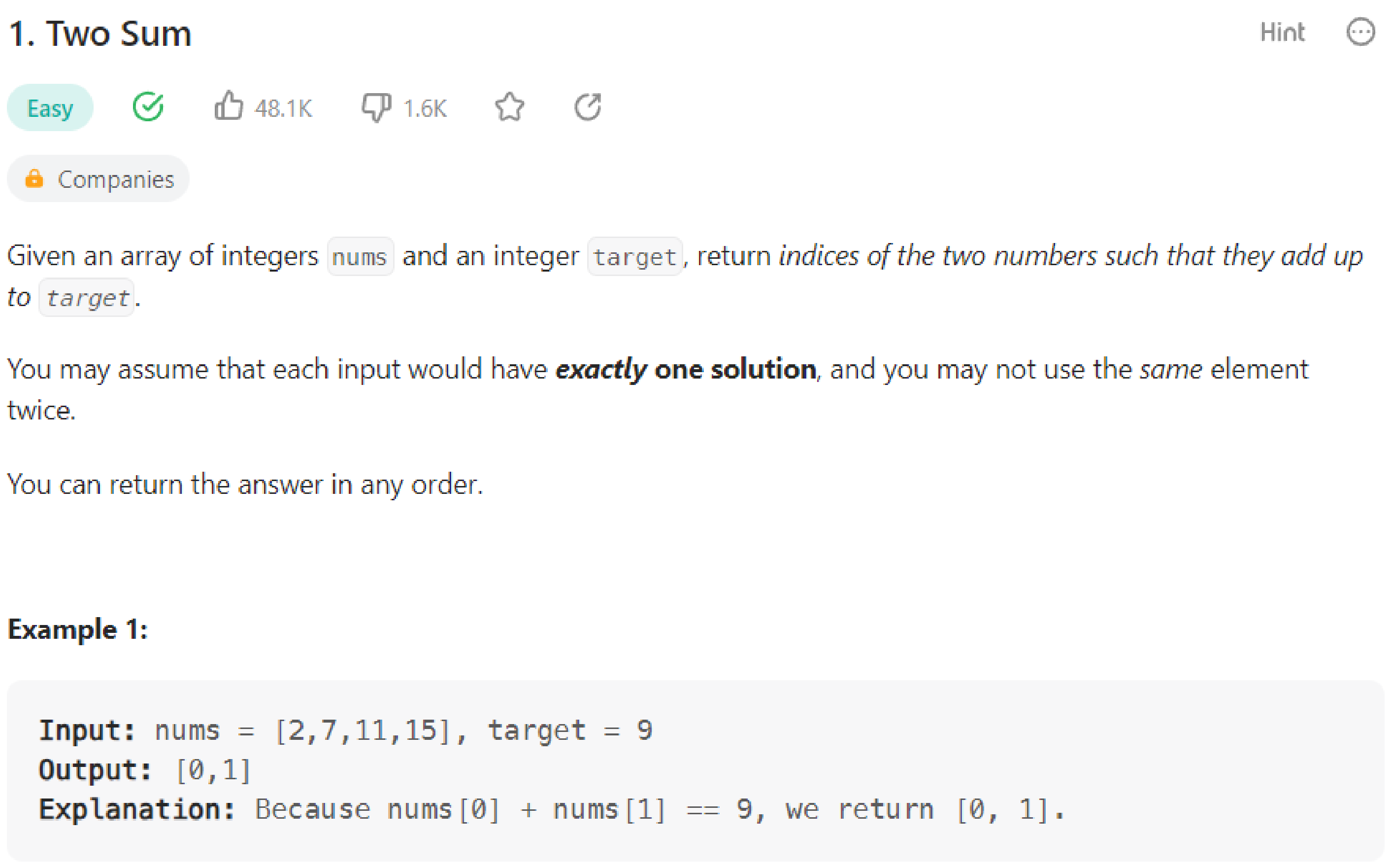
Leetcode 刷題

部分節選

簡介：

- 從入學至今我練習了大約140題Leetcode，主要由easy和medium難度組成。在此節選一些題目，包含各題的解題思路與輸出結果。

1. Two sum



1. Two Sum

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

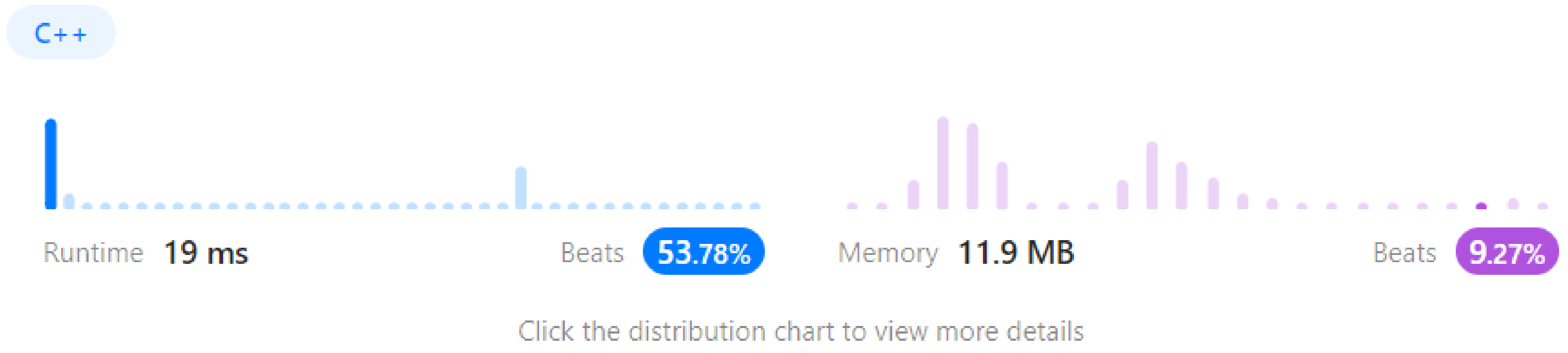
Think:

- 想法很簡單：target由兩個整數構成，而傳入的容器元素中只有唯一一個解，可以用for迴圈逐項檢查。但應該有改進的空間。
- Unordered_map：類似dictionary的概念，一個key對一個value。Unordered_map因為記憶體的不連續使其搜尋效率很高，相對地空間複雜度也很大。這種做法可以有效提高執行效率。

Solution : 雜湊表

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int> T;
        for(int i = 0; i < nums.size(); i++){
            T[nums[i]] = i;
        }
        for(int n = 0; n < nums.size(); n++){
            int temp = target - nums[n];
            if(T.count(temp) && T[temp] != n){
                return {n, T[temp]};
            }
        }
        return {};
        }
};
```

Output


C++

Runtime 19 ms

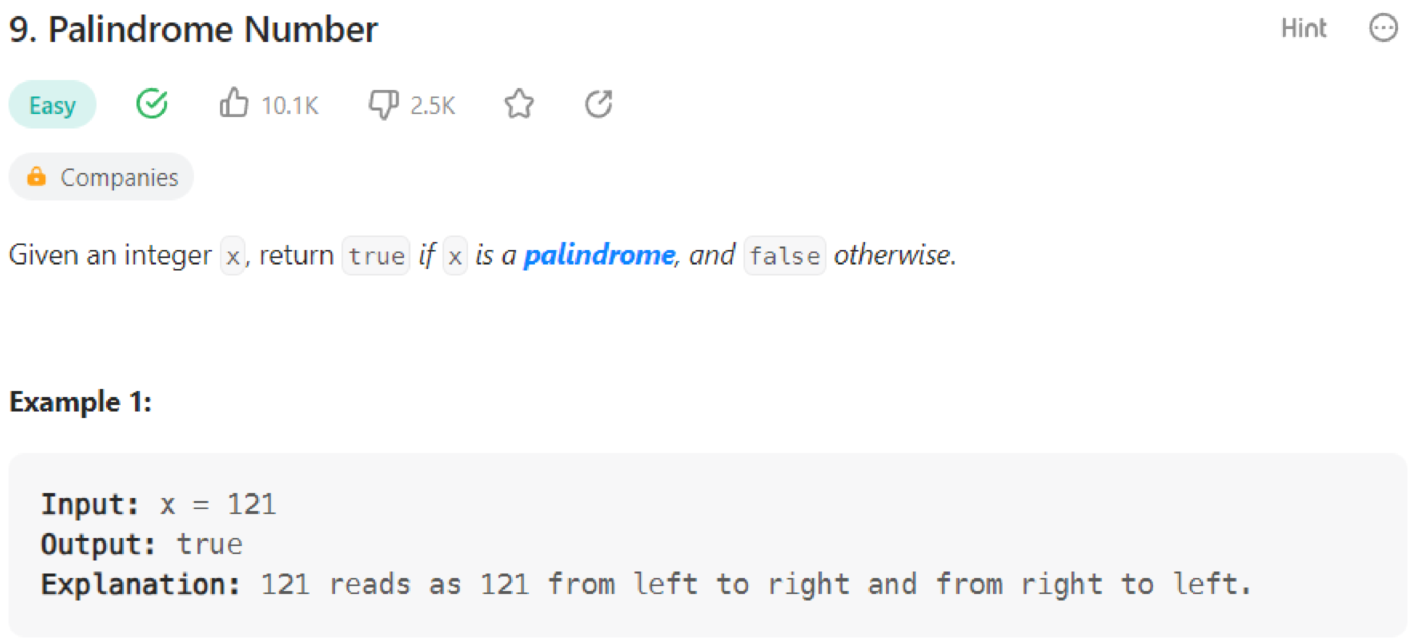
Beats 53.78%

Memory 11.9 MB

Beats 9.27%

Click the distribution chart to view more details

9. Palindrome number



9. Palindrome Number

Hint

Easy 10.1K 2.5K

Companies

Given an integer `x`, return `true` *if* `x` *is a* ***palindrome****, and* `false` *otherwise.*

Example 1:

```
Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
```

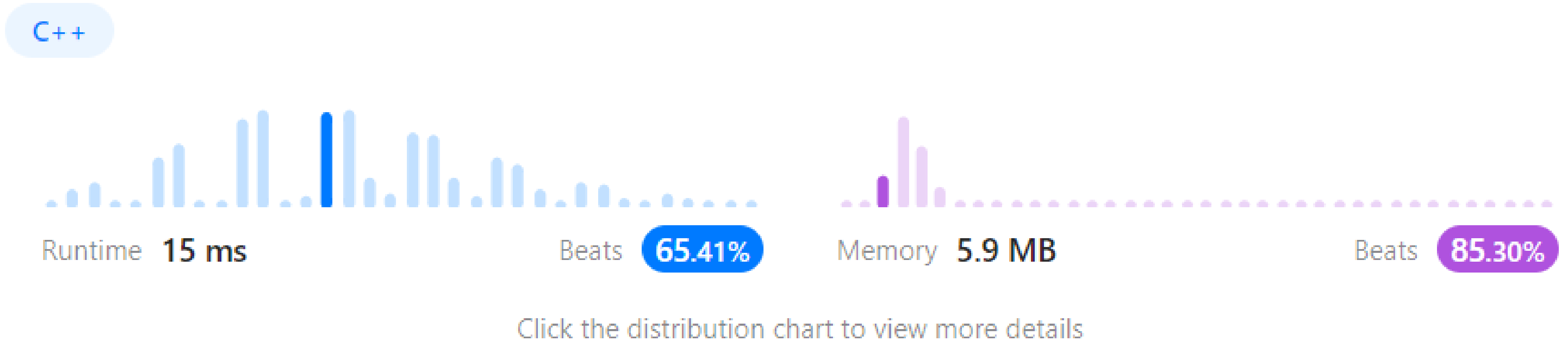
Think:

- 題目中的回文數有以下限制：不得為負數或小數，且不論從頭至尾或是相反的閱讀順序都能得出相同的整數。
- 將輸入整數逐位數切個並進行比對。
- 優先排除負數與小數

Solution:

```
class Solution {
public:
    bool isPalindrome(int x) {
        if(x < 0){//排除負數
            return 0;
        }
        if(x >= 0 && x < 10){//一位數算是回文數
            return 1;
        }
        if(x%10 == 0 && x != 0){//排除尾數為零且二位數以上的數字
            return 0;
        }

        int num = 0;
        while(num < x){
            num = num * 10 + x % 10;
            x /= 10;
            if(num == x/10 || num == x){
            return 1;
            break;
        }
        }

        return 0;
    }
};
```

Output



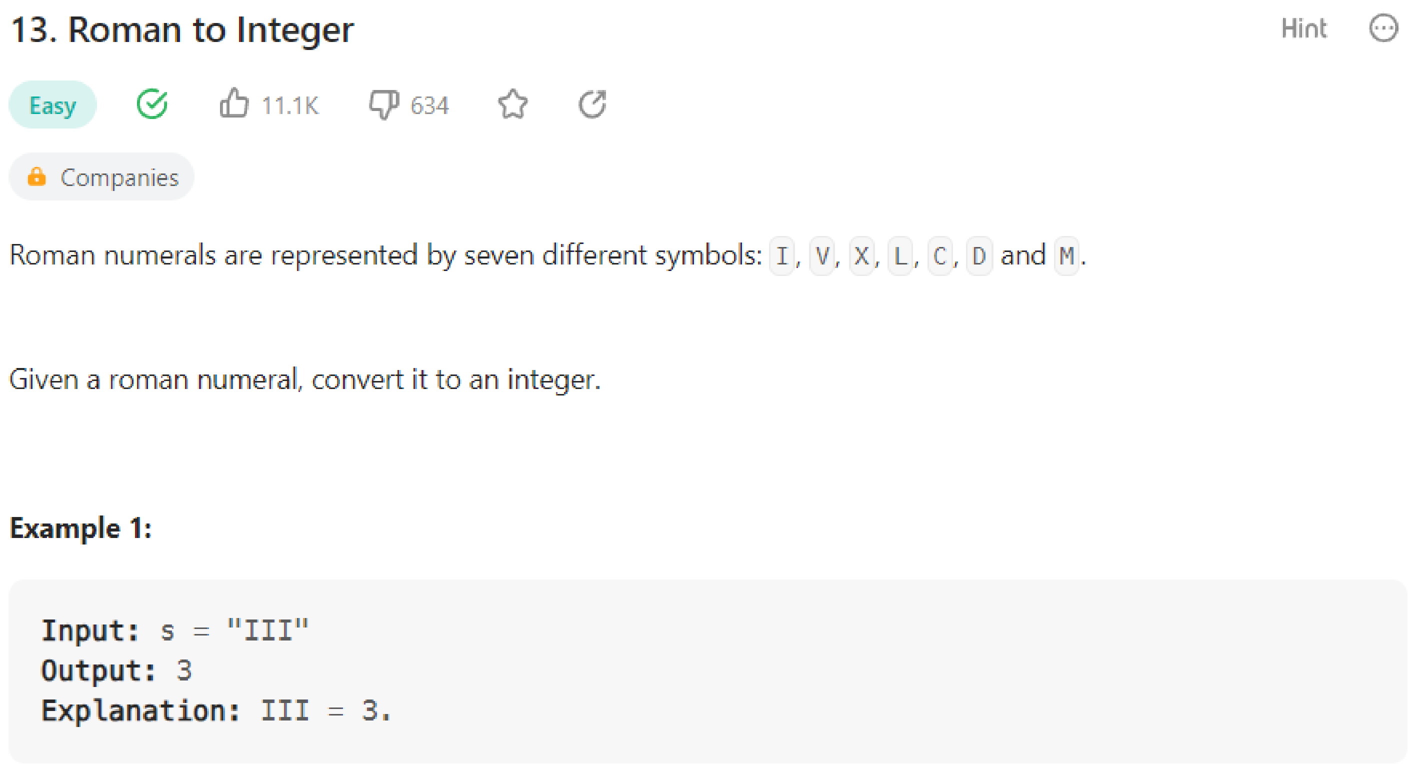
C++

Runtime 15 ms Beats 65.41%

Memory 5.9 MB Beats 85.30%

Click the distribution chart to view more details

13. Roman to Integer



13. Roman to Integer

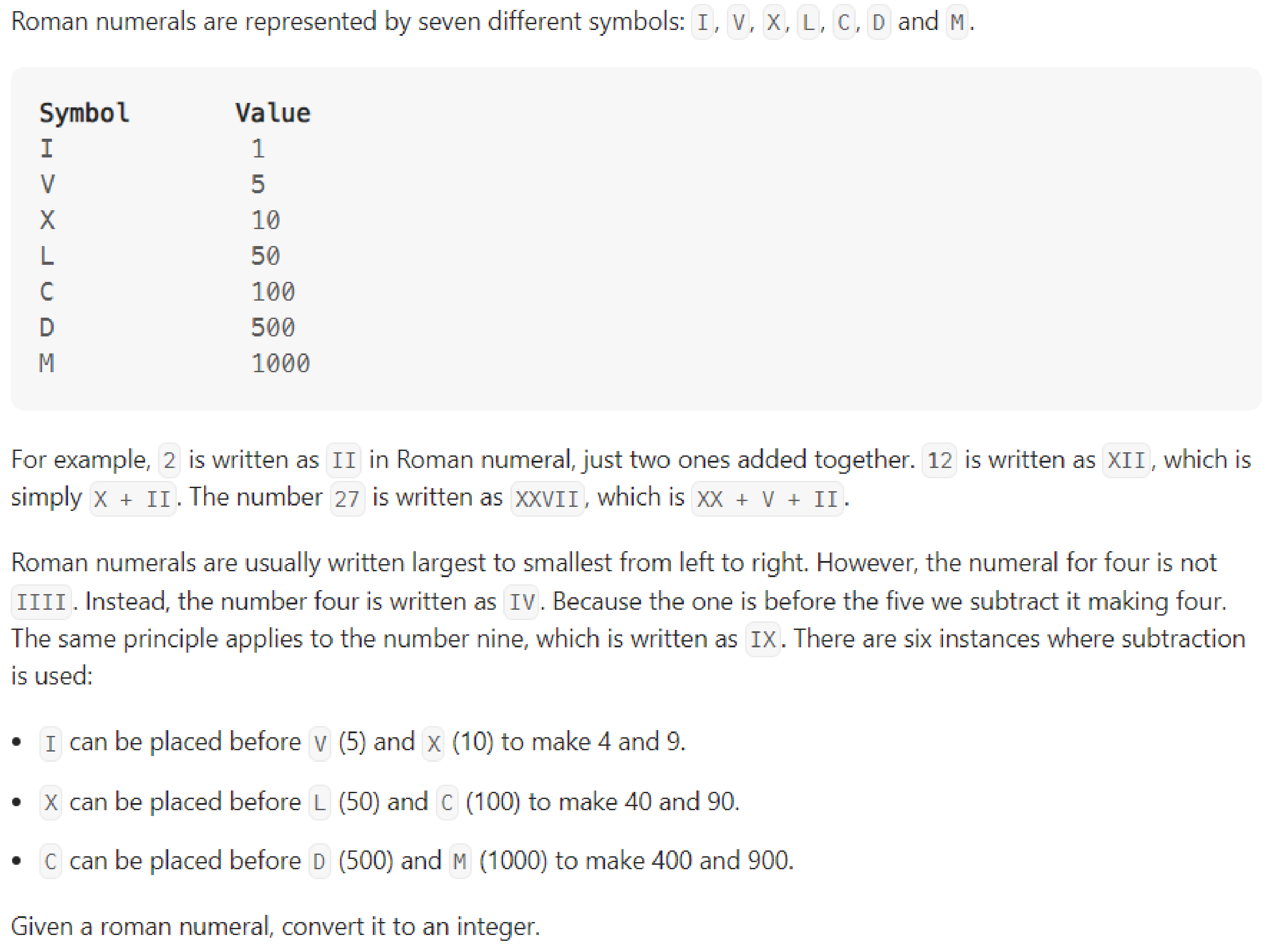
Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

Given a roman numeral, convert it to an integer.

Example 1:

```
Input: s = "III"
Output: 3
Explanation: III = 3.
```

Think:



Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, `2` is written as `II` in Roman numeral, just two ones added together. `12` is written as `XII`, which is simply `X + II`. The number `27` is written as `XXVII`, which is `XX + V + II`.

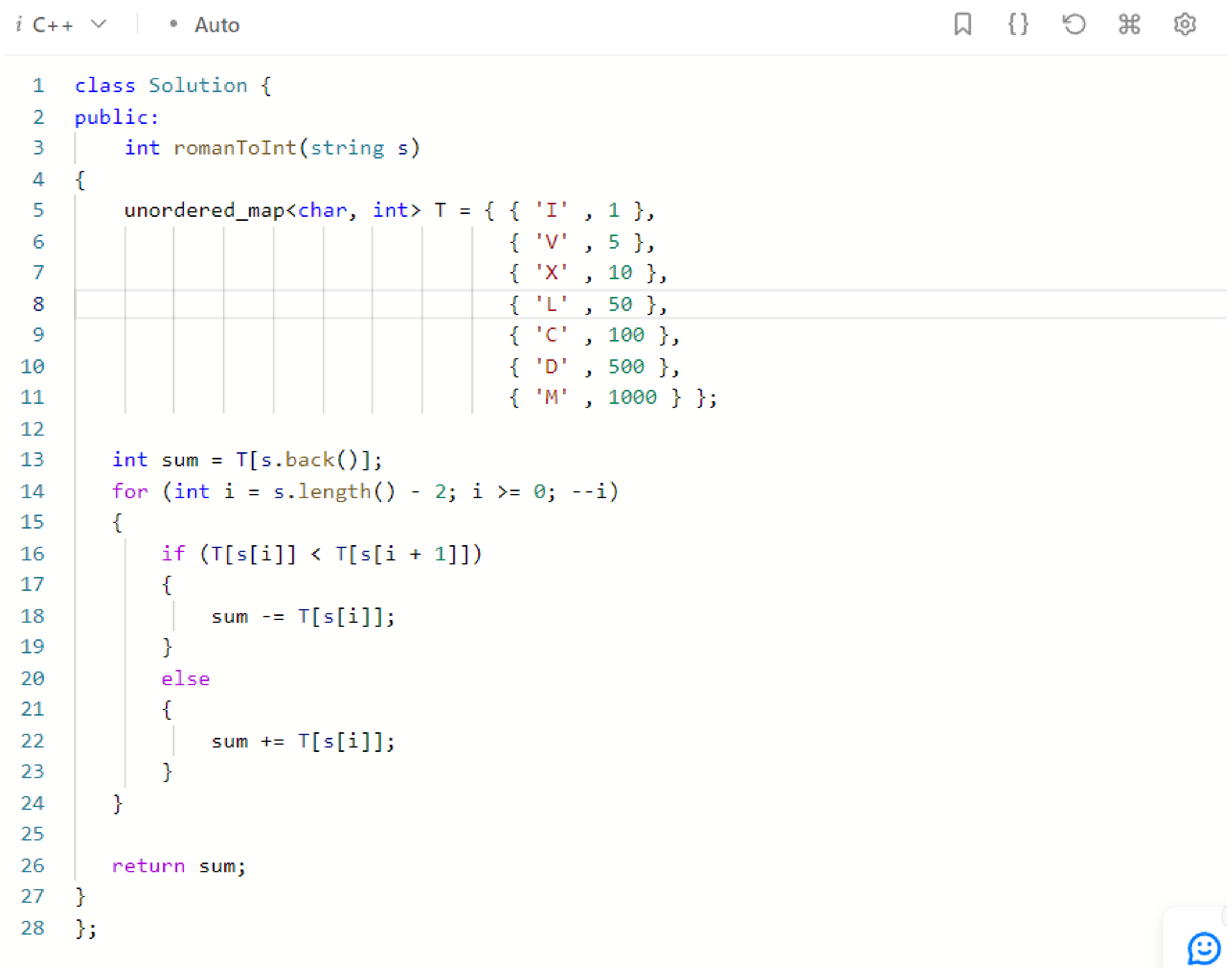
Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not `IIII`. Instead, the number four is written as `IV`. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as `IX`. There are six instances where subtraction is used:

- `I` can be placed before `V` (5) and `X` (10) to make 4 and 9.
- `X` can be placed before `L` (50) and `C` (100) to make 40 and 90.
- `C` can be placed before `D` (500) and `M` (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

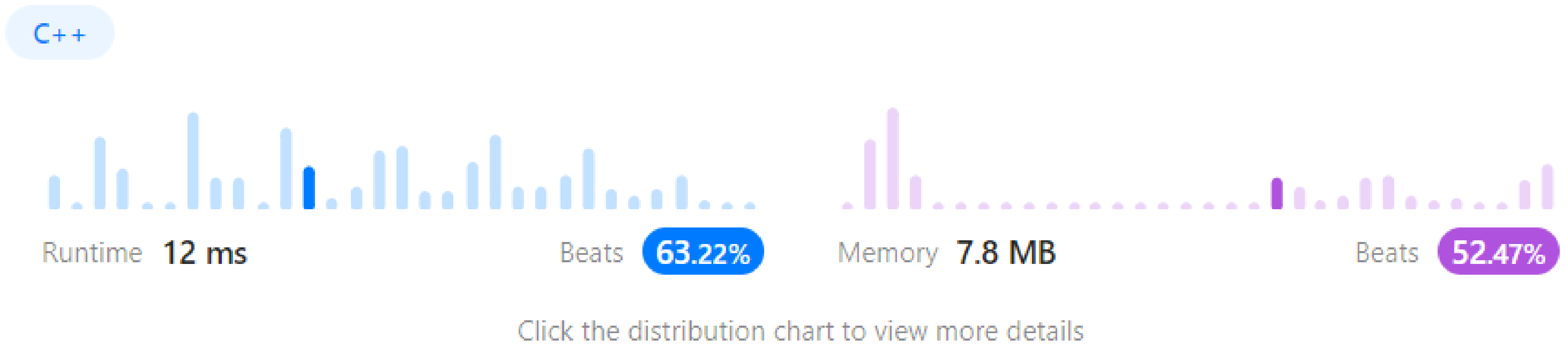
- Ex：VI = 6，但IV = 4
- 右加左減：當右側為較小的數字時，兩者相加；但若左側較小，則右側減去左側數字
- 可由左至右讀
- 同樣可以透過建立Unordered_map來加快執行效率

Solution:
雜湊表



```cpp
class Solution {
public:
    int romanToInt(string s)
{
    unordered_map<char, int> T = { { 'I' , 1 },
                                   { 'V' , 5 },
                                   { 'X' , 10 },
                                   { 'L' , 50 },
                                   { 'C' , 100 },
                                   { 'D' , 500 },
                                   { 'M' , 1000 } };

   int sum = T[s.back()];
   for (int i = s.length() - 2; i >= 0; --i)
   {
       if (T[s[i]] < T[s[i + 1]])
       {
           sum -= T[s[i]];
       }
       else
       {
           sum += T[s[i]];
       }
   }

   return sum;
}
};
```

Output



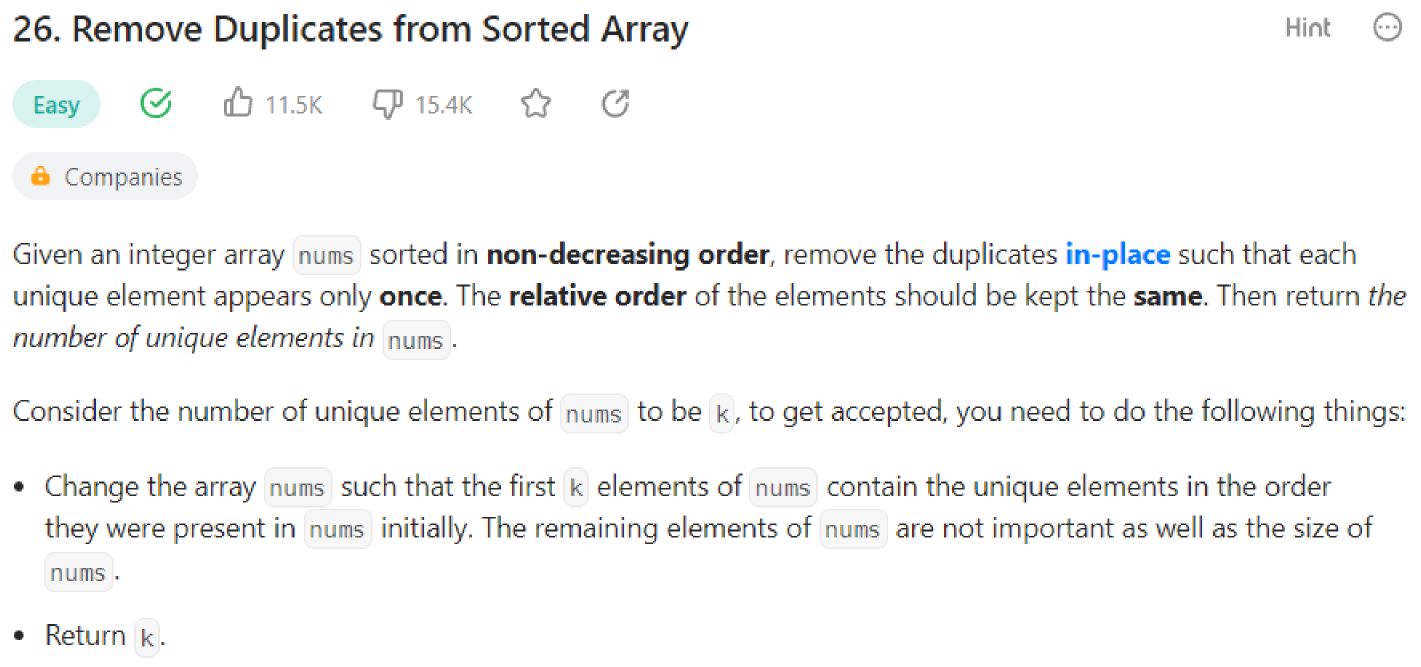
C++

Runtime 12 ms Beats 63.22%

Memory 7.8 MB Beats 52.47%

Click the distribution chart to view more details

26. Remove Duplicates from Sorted Array



26. Remove Duplicates from Sorted Array

Hint

Easy 11.5K 15.4K

Companies

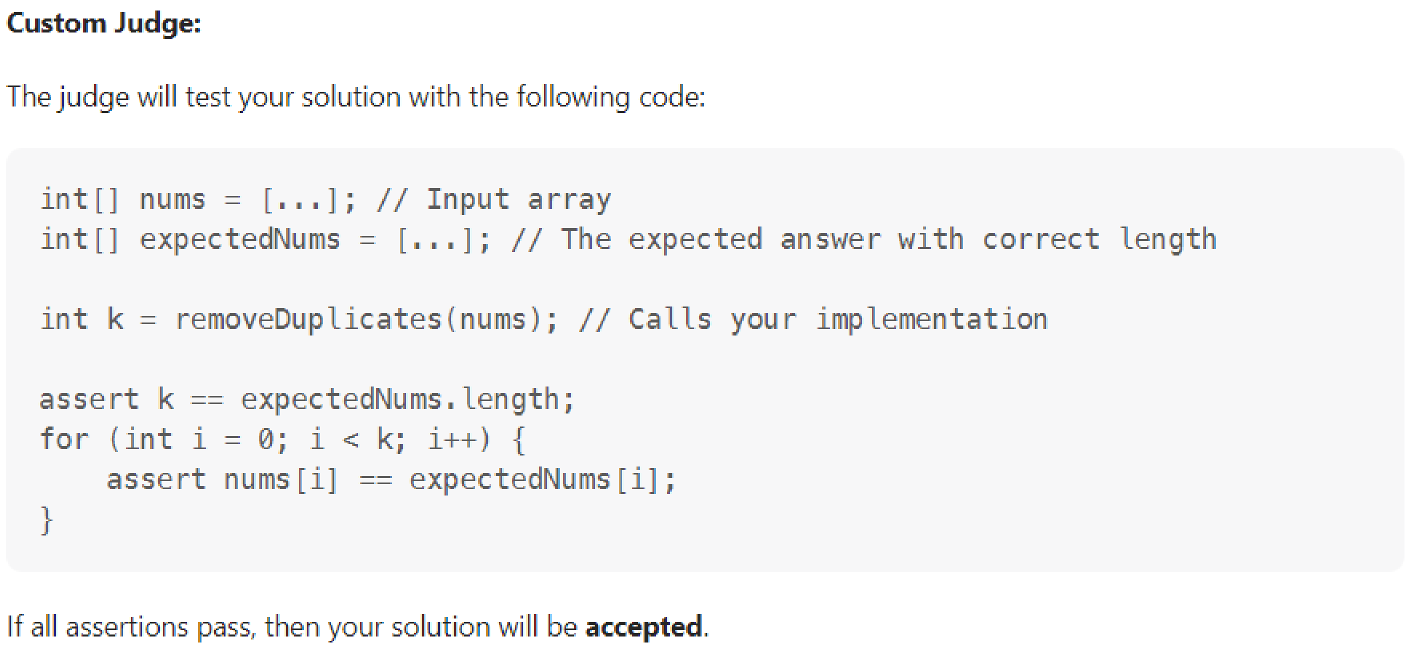
Given an integer array `nums` sorted in **non-decreasing order**, remove the duplicates **in-place** such that each unique element appears only **once**. The **relative order** of the elements should be kept the **same**. Then return *the number of unique elements in* `nums`.

Consider the number of unique elements of `nums` to be `k`, to get accepted, you need to do the following things:

- Change the array `nums` such that the first `k` elements of `nums` contain the unique elements in the order they were present in `nums` initially. The remaining elements of `nums` are not important as well as the size of `nums`.
- Return `k`.

Think:

- 針對各個位置代換數字，最後回傳代換次數(也等於容器長度)
- 會檢測原容器



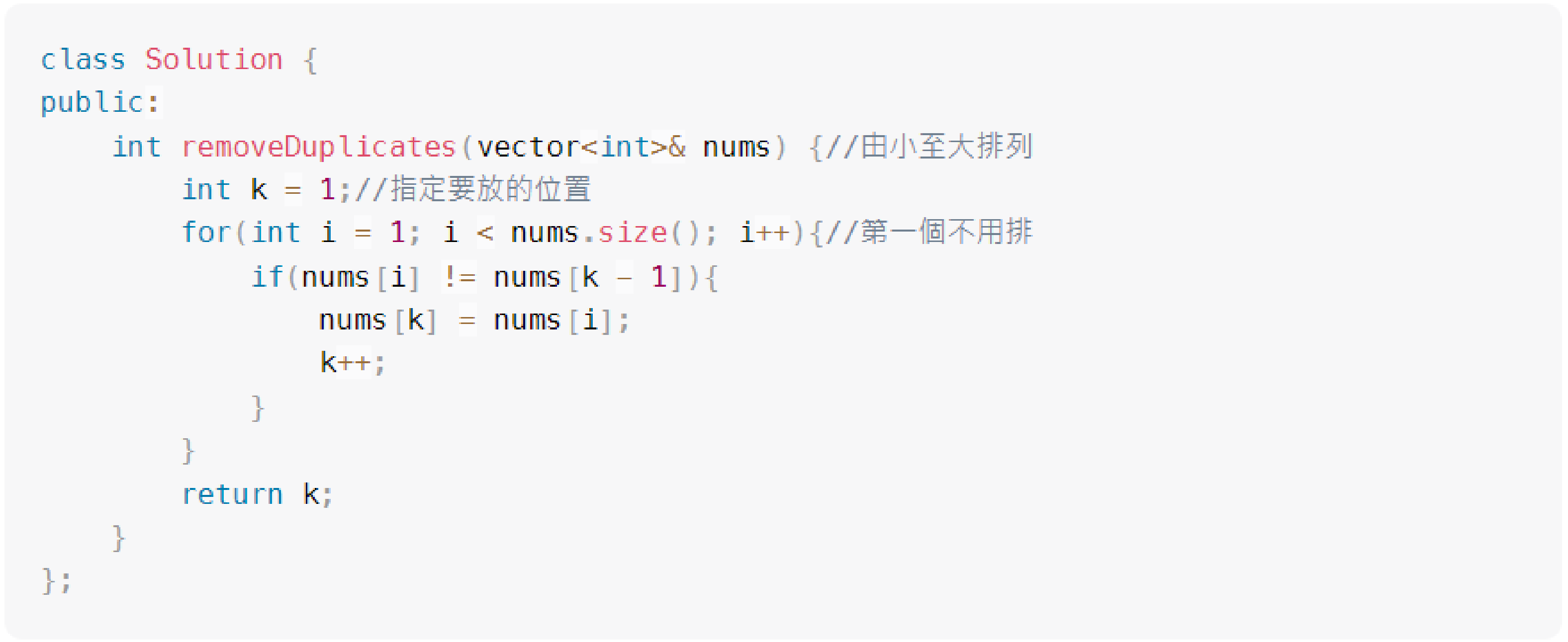
Custom Judge:

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length

int k = removeDuplicates(nums); // Calls your implementation

assert k == expectedNums.length;
for (int i = 0; i < k; i++) {
    assert nums[i] == expectedNums[i];
}
```

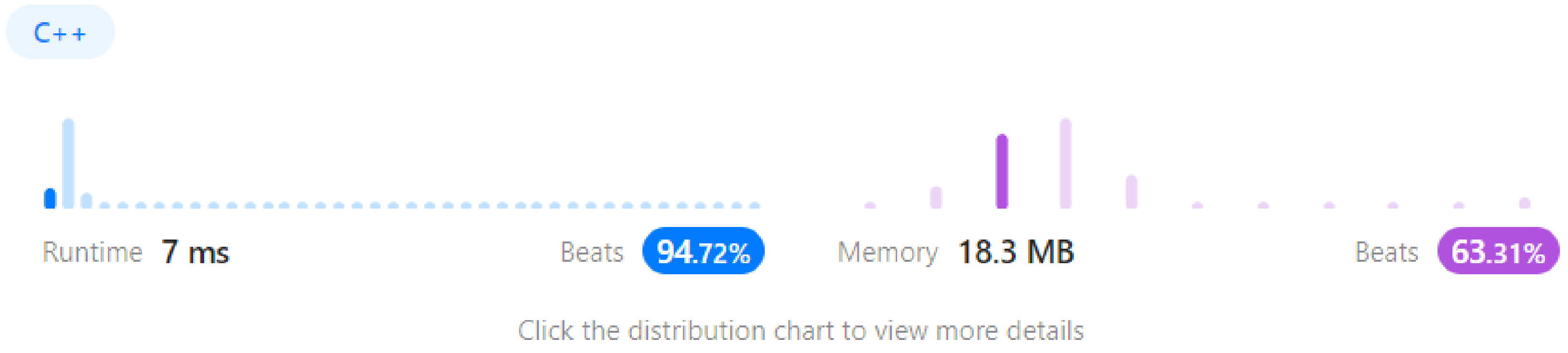
If all assertions pass, then your solution will be **accepted**.

Solution:



```
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {//由小至大排列
        int k = 1;//指定要放的位置
        for(int i = 1; i < nums.size(); i++){//第一個不用排
            if(nums[i] != nums[k - 1]){
                nums[k] = nums[i];
                k++;
            }
        }
        return k;
    }
};
```

Output



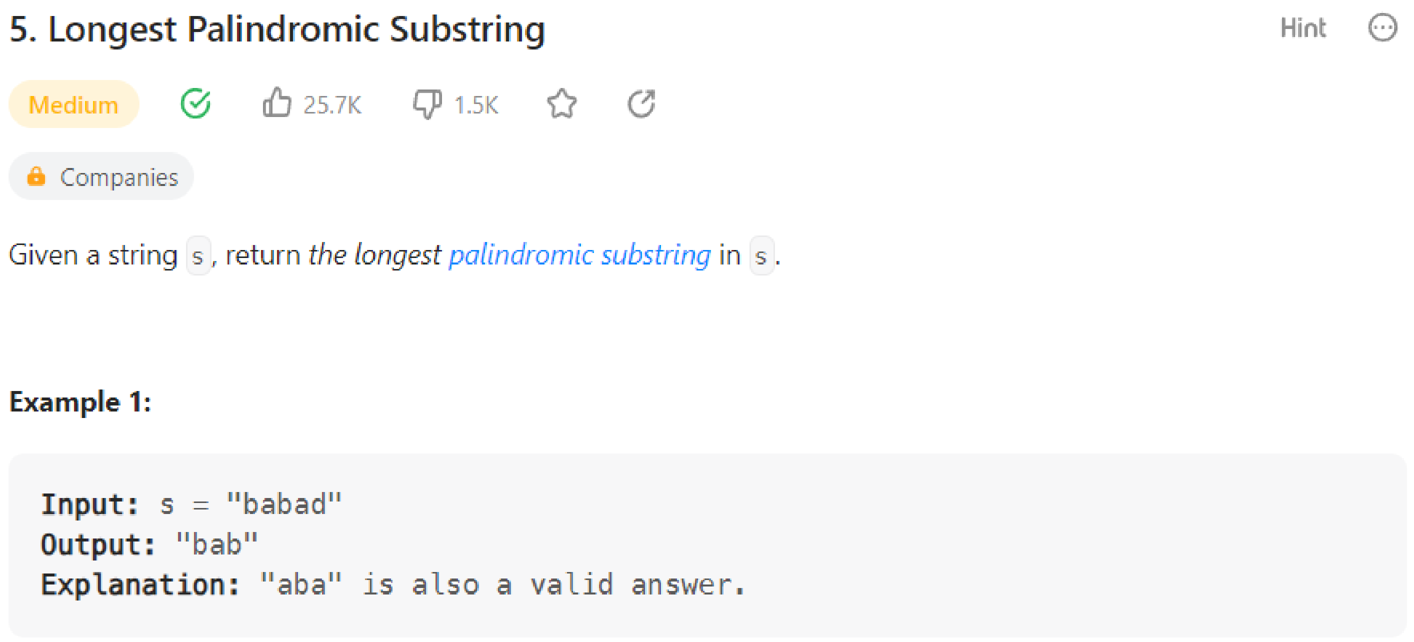
C++

Runtime 7 ms Beats 94.72%

Memory 18.3 MB Beats 63.31%

Click the distribution chart to view more details

5. Longest Palindromic Substring



5. Longest Palindromic Substring

Given a string `s`, return *the longest* palindromic substring in `s`.

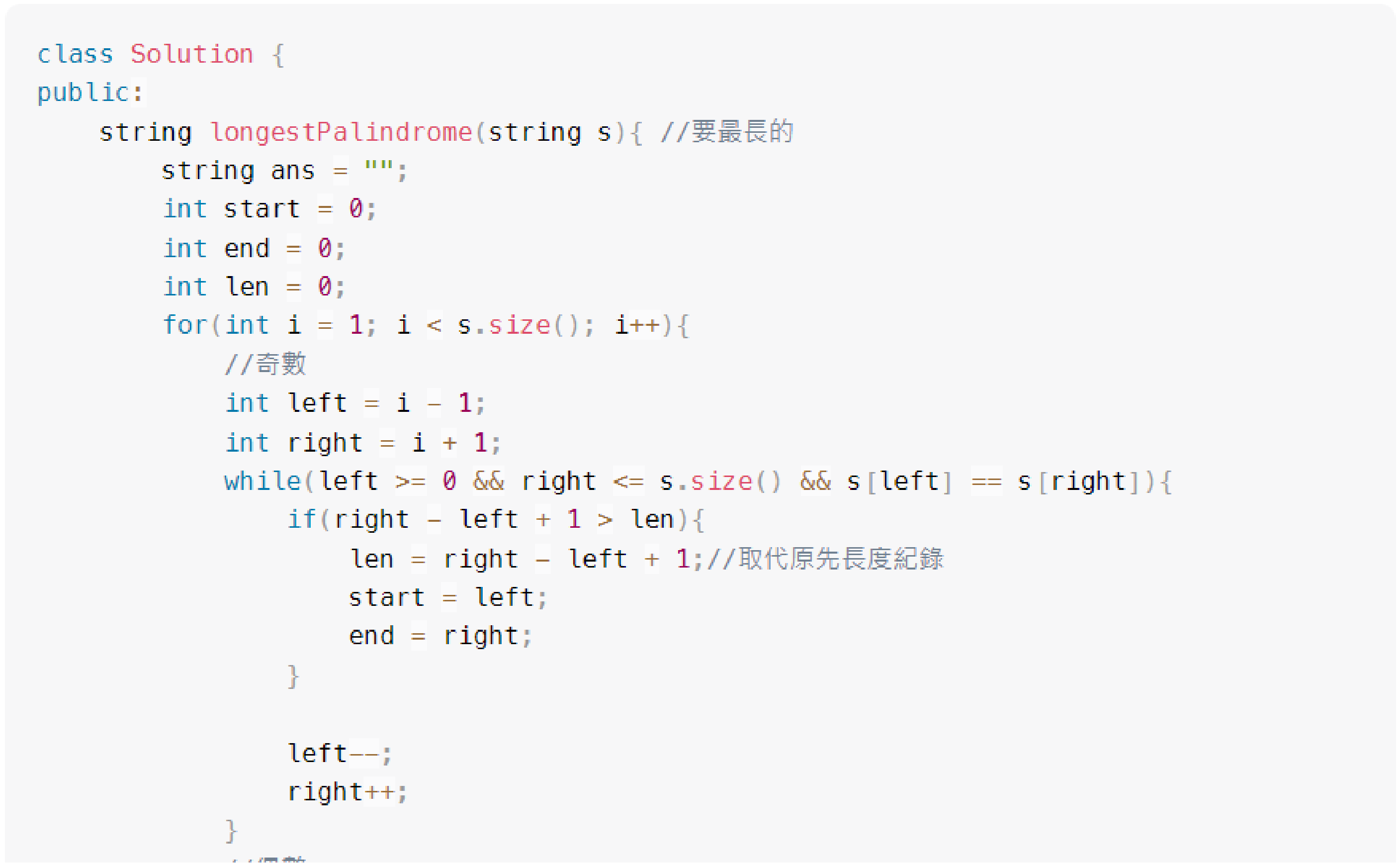
Example 1:

```
Input: s = "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
```

Think:

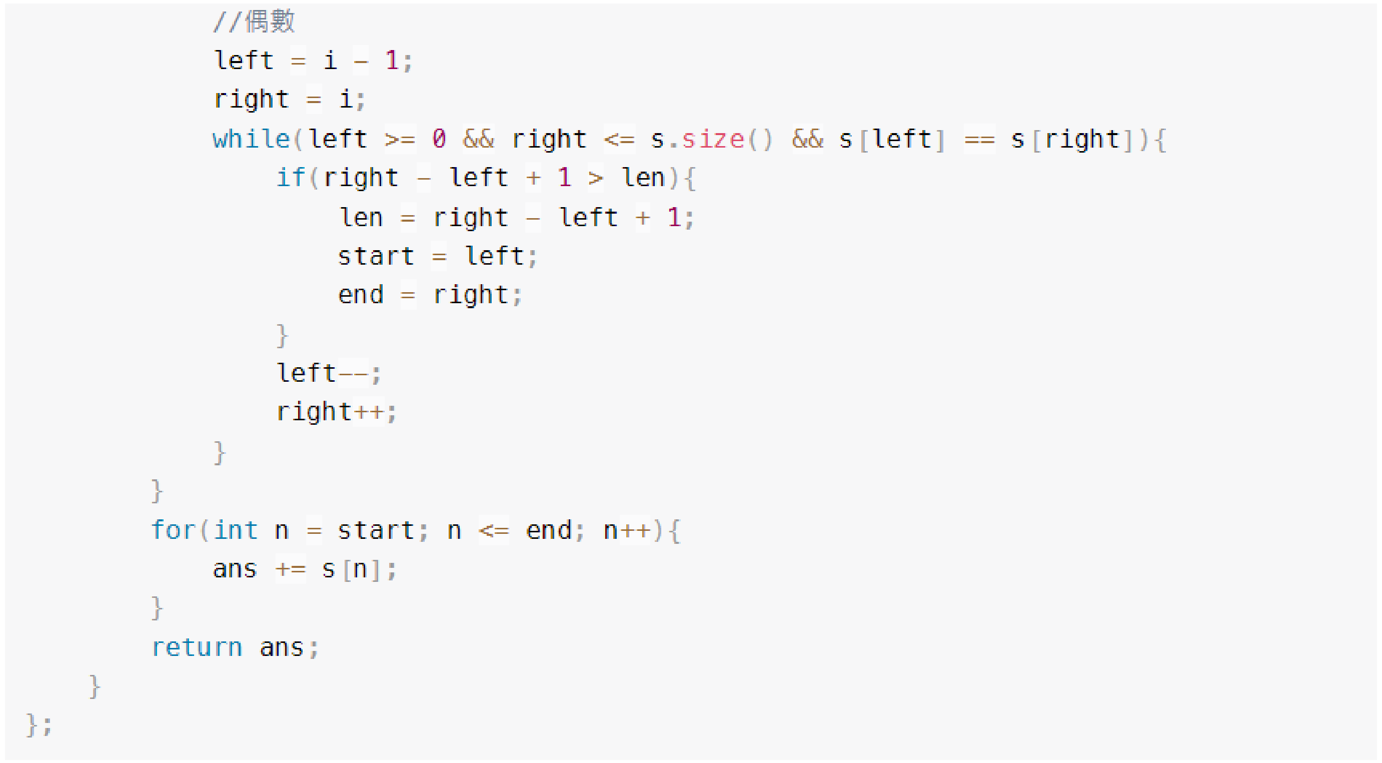
- 一組字串有多種可能的回文字串，但最長的只有一組。
- 針對奇數與偶數數量的字母要有不同的處理方式。
- 雙向檢查。

Solution: (1)



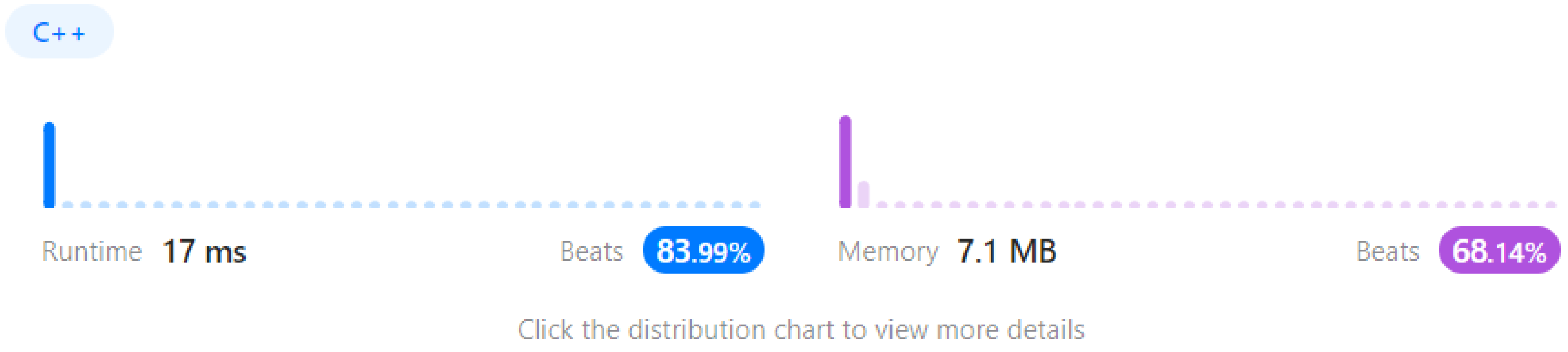
```
class Solution {
public:
    string longestPalindrome(string s){ //要最長的
        string ans = "";
        int start = 0;
        int end = 0;
        int len = 0;
        for(int i = 1; i < s.size(); i++){
            //奇數
            int left = i - 1;
            int right = i + 1;
            while(left >= 0 && right <= s.size() && s[left] == s[right]){
                if(right - left + 1 > len){
                    len = right - left + 1;//取代原先長度紀錄
                    start = left;
                    end = right;
                }

                left--;
                right++;
            }
```

Solution: (2)



```
            //偶數
            left = i - 1;
            right = i;
            while(left >= 0 && right <= s.size() && s[left] == s[right]){
                if(right - left + 1 > len){
                    len = right - left + 1;
                    start = left;
                    end = right;
                }
                left--;
                right++;
            }
        }
        for(int n = start; n <= end; n++){
            ans += s[n];
        }
        return ans;
    }
};
```

Output



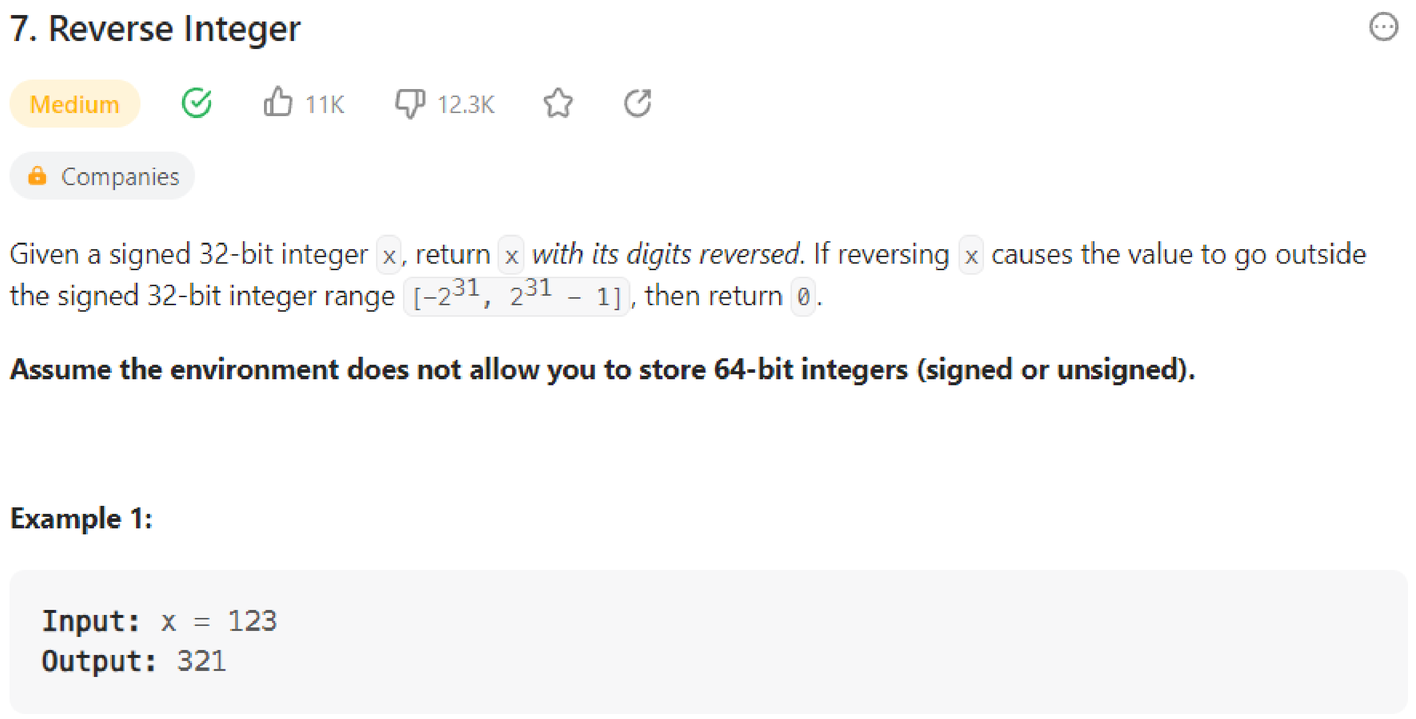
C++

Runtime 17 ms Beats 83.99%

Memory 7.1 MB Beats 68.14%

Click the distribution chart to view more details

7. Reverse Integer



7. Reverse Integer

Medium ✓ 11K 12.3K

Companies

Given a signed 32-bit integer `x`, return `x` *with its digits reversed*. If reversing `x` causes the value to go outside the signed 32-bit integer range $[-2^{31}, 2^{31} - 1]$, then return `0`.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1:

```
Input: x = 123
Output: 321
```

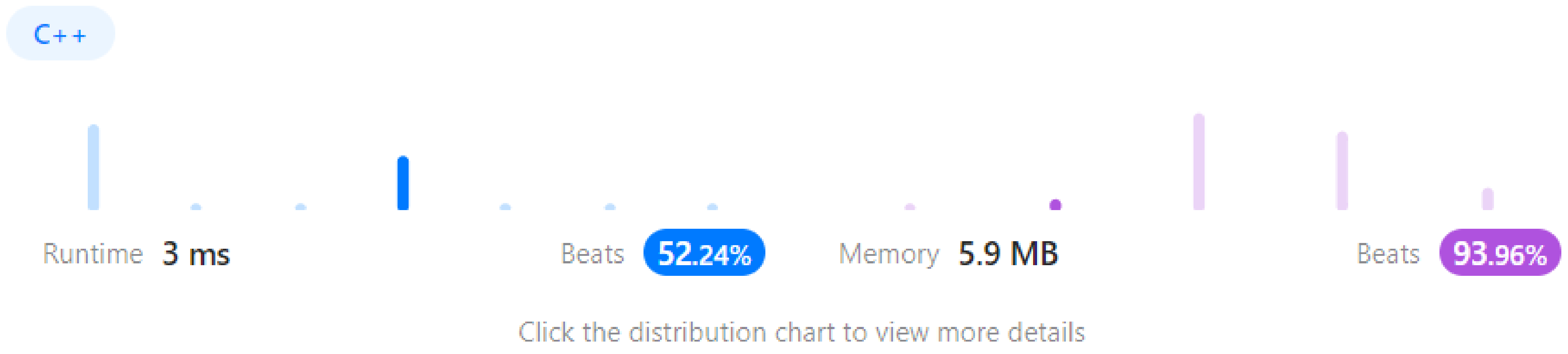
Think:

- 可以使用在回文數題目中的方法
- 但要注意，在運算的過程中會不會在生成回文數的過程中超出整數的極限
- Ex：INT_MAX - 1 = 2147483646，若直接將前後顛倒，會得到6463847412，這個數字在在32位元中的系統是無法表達的。
- 解決方法：提早觀察前一位的數字，若當前顛倒數再加一位數字必然會超出整數範圍，則回傳0

Solution: 雜湊表

```
class Solution {
public:
    int reverse(int x) {
        int n = 0;
        while(x){
            if(x > INT_MAX || x < INT_MIN) return 0;
            if(n > INT_MAX / 10 || n < INT_MIN / 10) return 0;
            n = n * 10 + x % 10;
            x /= 10;
        }
        return n;
    }
};
```

Output

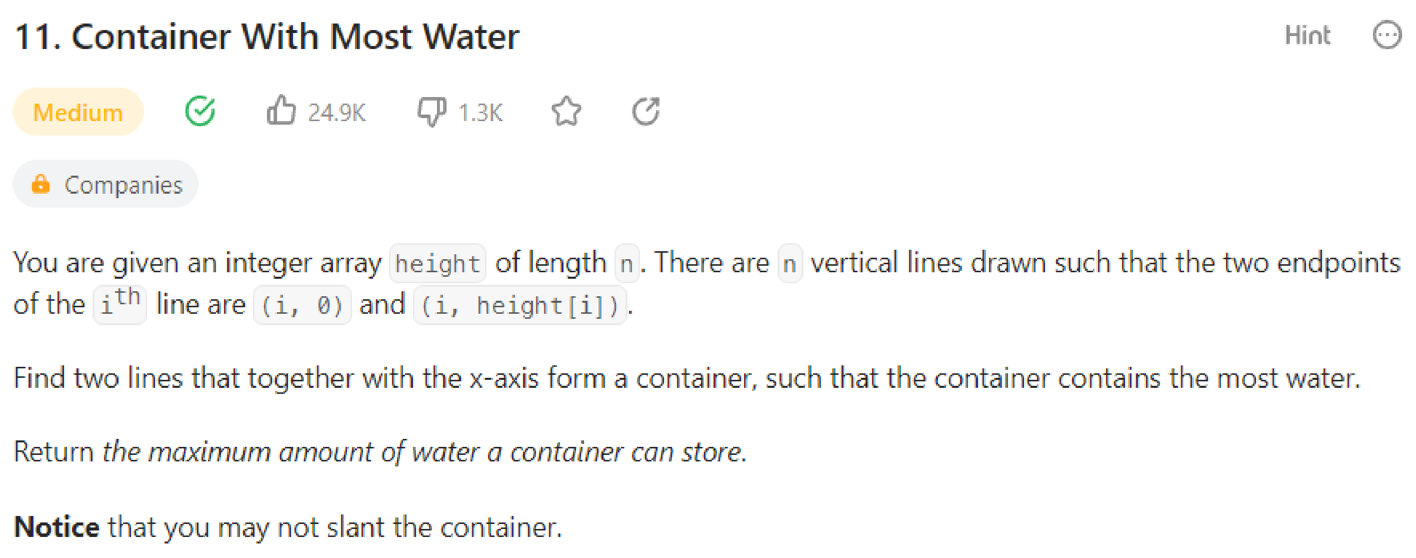


C++

Runtime 3 ms Beats 52.24% Memory 5.9 MB Beats 93.96%

Click the distribution chart to view more details

11. Container With Most Water



11. Container With Most Water

Hint

Medium 24.9K 1.3K

Companies

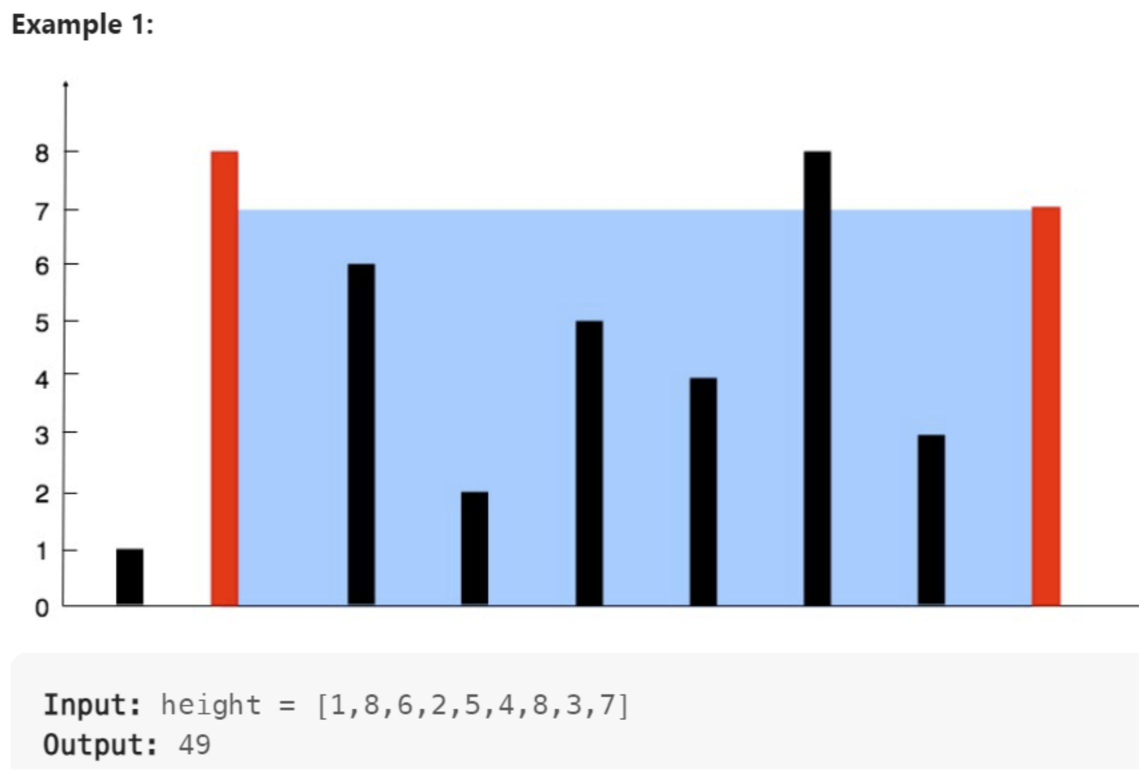
You are given an integer array `height` of length `n`. There are `n` vertical lines drawn such that the two endpoints of the `i`th line are `(i, 0)` and `(i, height[i])`.

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return *the maximum amount of water a container can store.*

Notice that you may not slant the container.

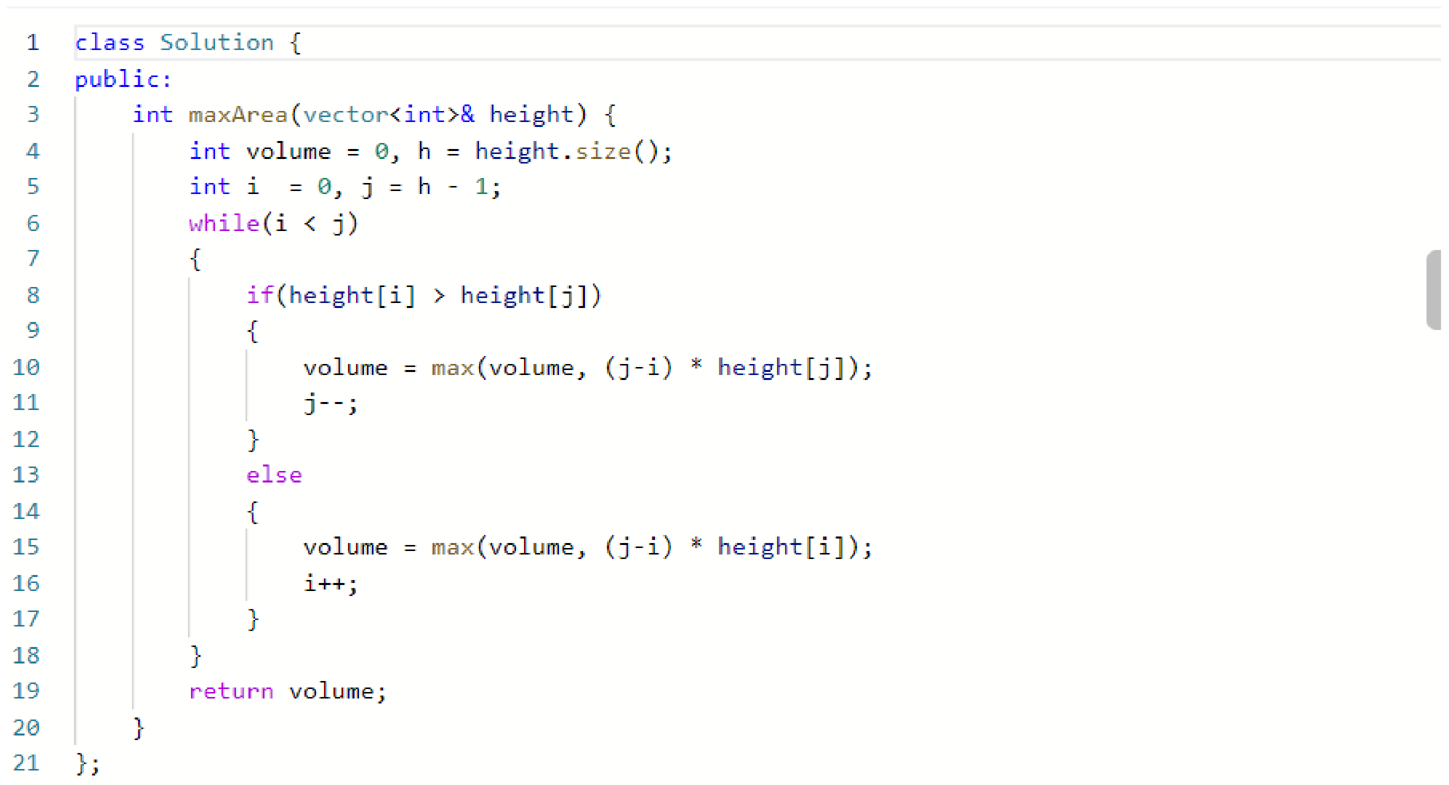
Think:


Example 1:

```
Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49
```

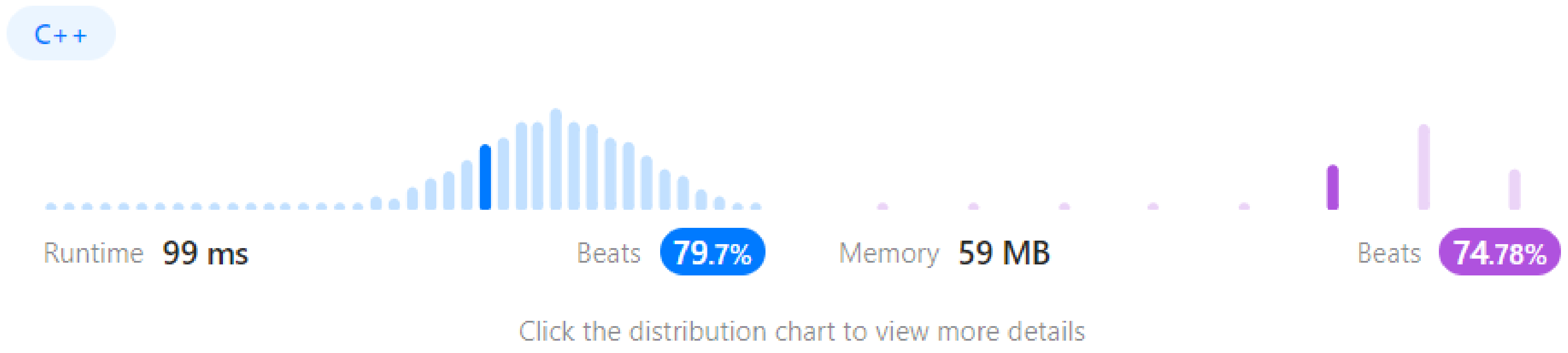
- 最後只會有一條水平線，不會有不同的高度
- 從左右兩端各取一根柱子，相互比較，並計算容量。
- 備註：當時間複雜度為O(n**2)時過不了

Solution:



```cpp
class Solution {
public:
    int maxArea(vector<int>& height) {
        int volume = 0, h = height.size();
        int i  = 0, j = h - 1;
        while(i < j)
        {
            if(height[i] > height[j])
            {
                volume = max(volume, (j-i) * height[j]);
                j--;
            }
            else
            {
                volume = max(volume, (j-i) * height[i]);
                i++;
            }
        }
        return volume;
    }
};
```

Output



C++

Runtime 99 ms Beats 79.7%

Memory 59 MB Beats 74.78%

Click the distribution chart to view more details

92. Reverse Linked List II

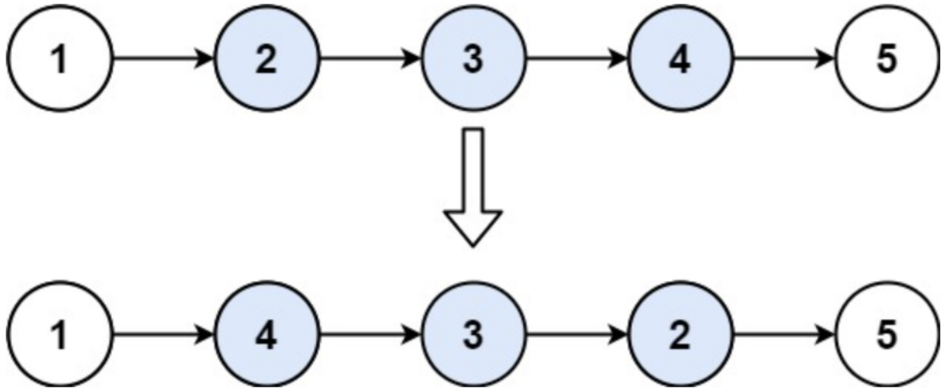
92. Reverse Linked List II

Medium 9.4K 427

Companies

Given the `head` of a singly linked list and two integers `left` and `right` where `left <= right`, reverse the nodes of the list from position `left` to position `right`, and return *the reversed list*.

Example 1:

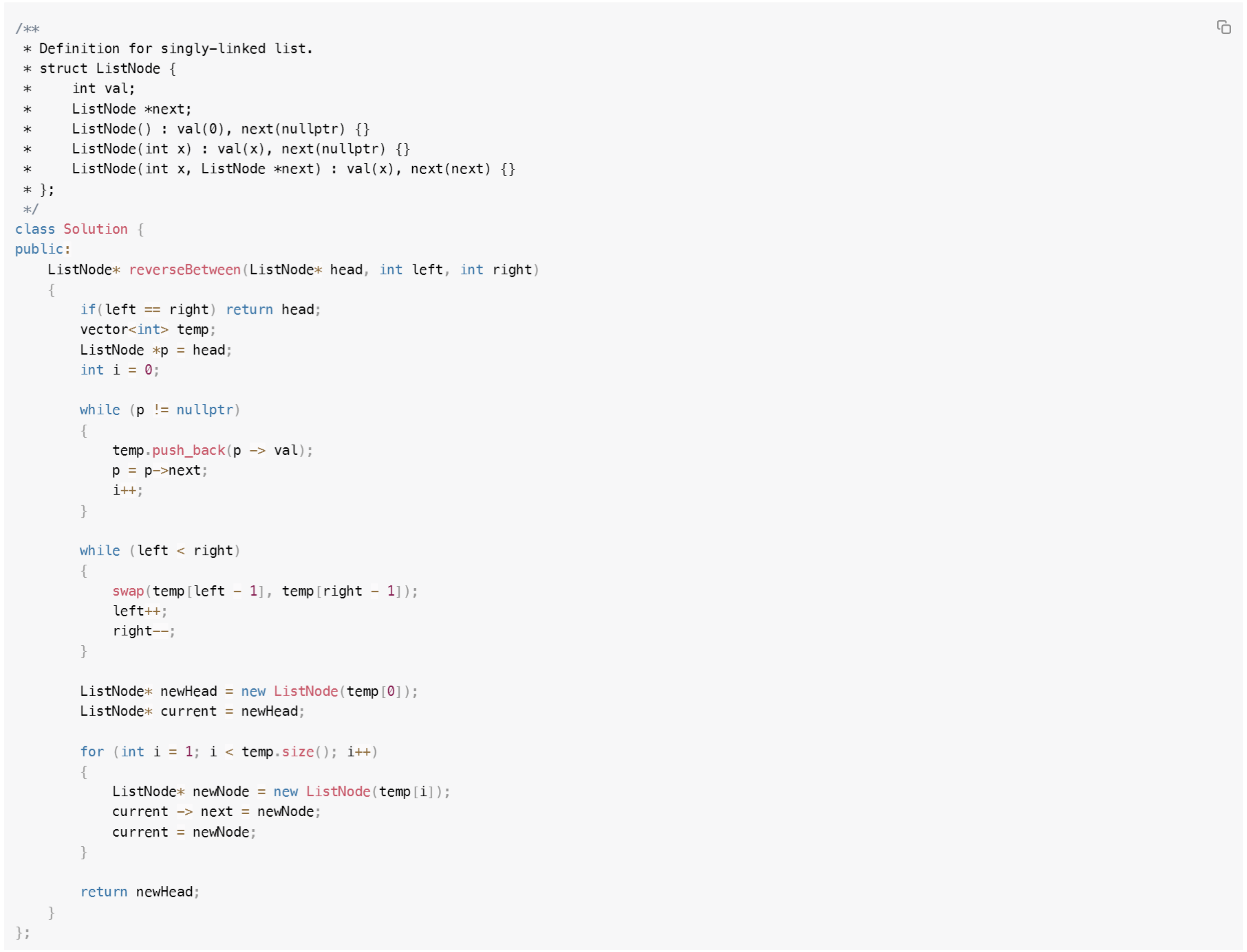


```
Input: head = [1,2,3,4,5], left = 2, right = 4
Output: [1,4,3,2,5]
```

Think:

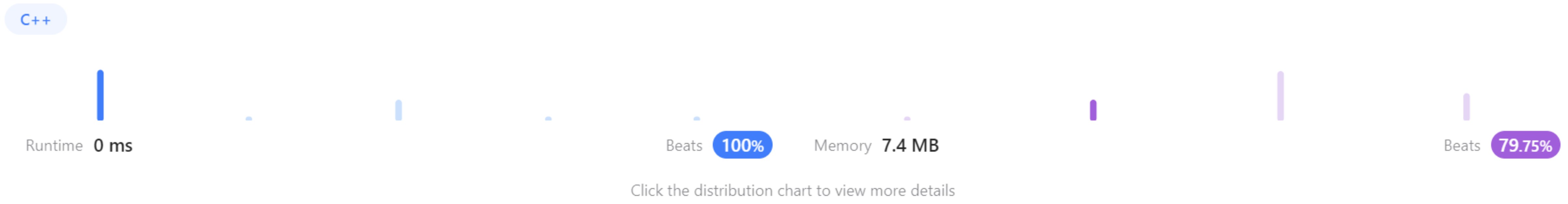
- Vector 的LIFO特性剛好契合這題的指示，因為執行完後順序會剛好顛倒。
- 接下來只需要把linkedlist和vector結合即可。
- 輸入的數字一定不重複。
- 能省則省：數字相同就直接回傳原Node

Solution:



```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* reverseBetween(ListNode* head, int left, int right)
    {
        if(left == right) return head;
        vector<int> temp;
        ListNode *p = head;
        int i = 0;

        while (p != nullptr)
        {
            temp.push_back(p -> val);
            p = p->next;
            i++;
        }

        while (left < right)
        {
            swap(temp[left - 1], temp[right - 1]);
            left++;
            right--;
        }

        ListNode* newHead = new ListNode(temp[0]);
        ListNode* current = newHead;

        for (int i = 1; i < temp.size(); i++)
        {
            ListNode* newNode = new ListNode(temp[i]);
            current -> next = newNode;
            current = newNode;
        }

        return newHead;
    }
};
```

Output



C++

Runtime 0 ms Beats 100% Memory 7.4 MB Beats 79.75%

Click the distribution chart to view more details

42. Trapping Rain Water

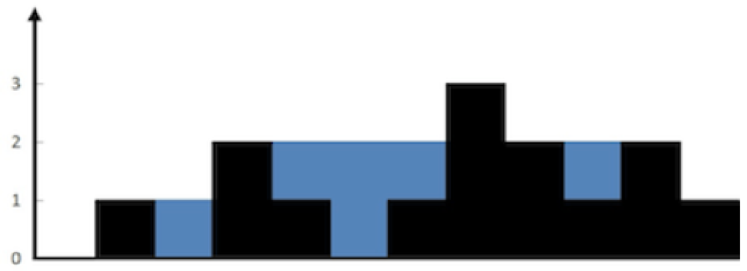
42. Trapping Rain Water

Hard 27.3K 375

Companies

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

Example 1:



```
Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
Explanation: The above elevation map (black section) is represented by array
[0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section)
are being trapped.
```

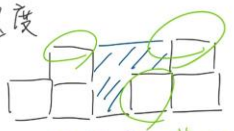
Think:

- 不再有單一高度限制，只要有凹槽就能夠有雨水。
- 如何判斷凹槽，及計算凹槽能裝多少水？
- 凹槽的容積取決於凹槽兩端的高度與水面下已存在的階梯，每單位的凹槽區域所能容納的水為「水平面（凹槽兩端較低者）減去階梯高」

42

2022年9月29日 星期四　下午10:50

✡ 寬度為 1 就不用乘寬度

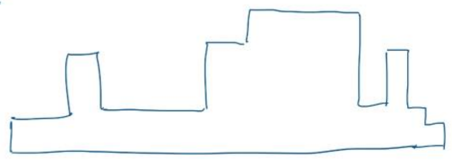
ex [1, 2, 0, 1, 2] ⇒ 

☆ 左右端裝不了　　用高度減去占有的高度

height[i]

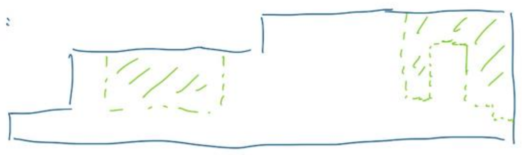
✡

ex:

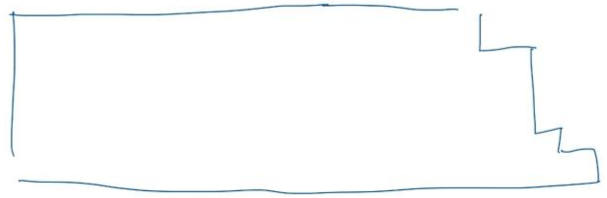


左右比較

從左看:

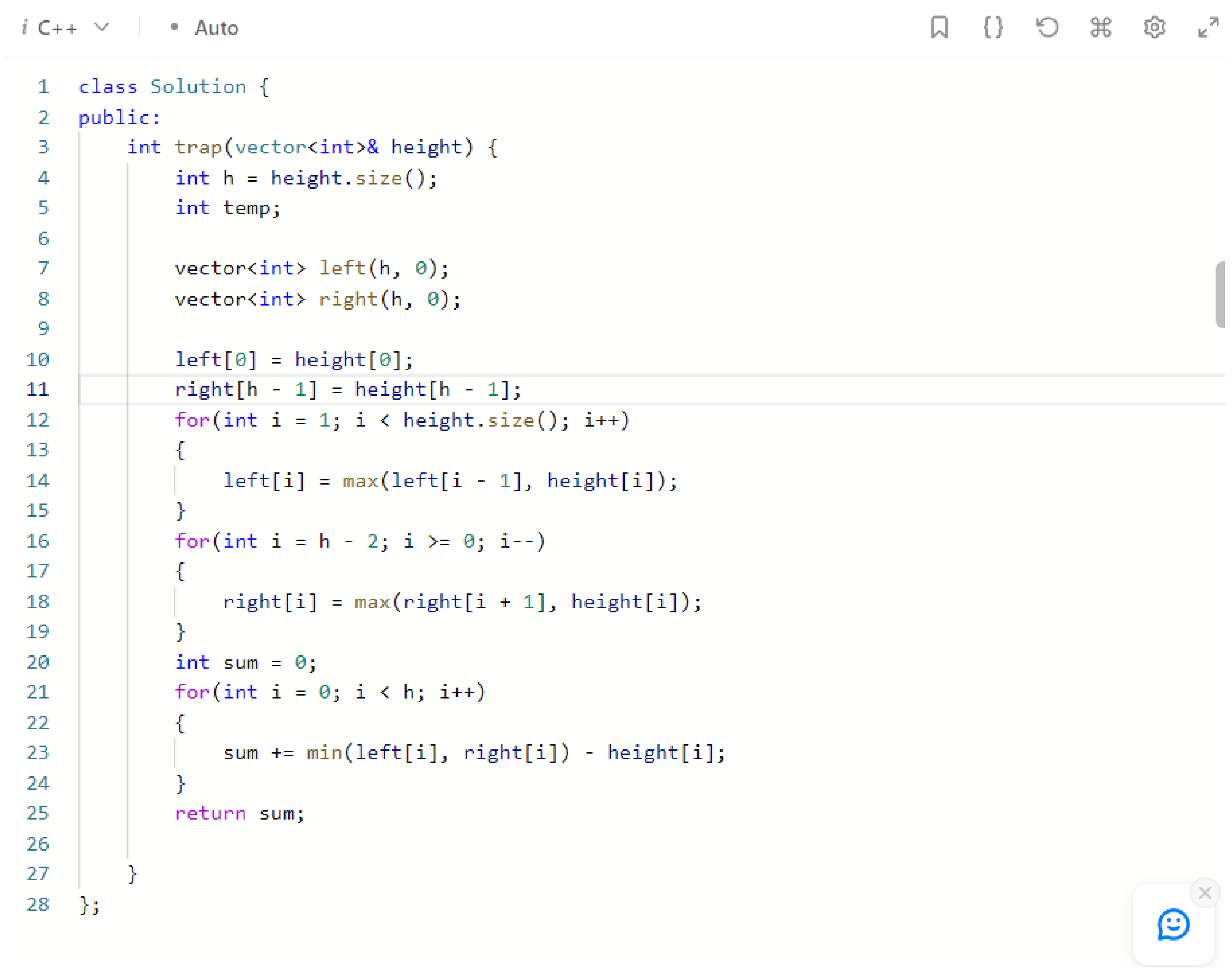


從右看:



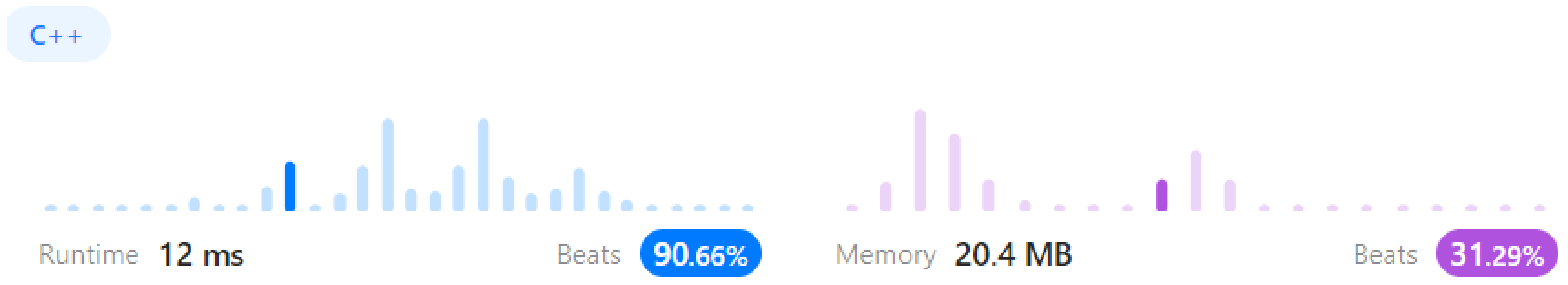
⇒ 交互比對再減去原高度可得 V

Solution:



```cpp
class Solution {
public:
    int trap(vector<int>& height) {
        int h = height.size();
        int temp;

        vector<int> left(h, 0);
        vector<int> right(h, 0);

        left[0] = height[0];
        right[h - 1] = height[h - 1];
        for(int i = 1; i < height.size(); i++)
        {
            left[i] = max(left[i - 1], height[i]);
        }
        for(int i = h - 2; i >= 0; i--)
        {
            right[i] = max(right[i + 1], height[i]);
        }
        int sum = 0;
        for(int i = 0; i < h; i++)
        {
            sum += min(left[i], right[i]) - height[i];
        }
        return sum;

    }
};
```

Output



C++

Runtime 12 ms Beats 90.66%

Memory 20.4 MB Beats 31.29%