盛最多水的容器

三种时间复杂度的对比题解

Problem ID: 3123 / ZJUT

题目简述

给定数组 height$[0 \ldots n-1]$，第 i 条竖线的高度为 height$[i]$。在所有两条竖线组成的容器中，求最大面积：

$$\text{area}(i, j) = (j - i) \cdot \min(\text{height}[i], \text{height}[j]).$$

本题有三个典型解法：暴力 $O(n^2)$、排序贪心 $O(n \log n)$、双指针 $O(n)$。

一、暴力枚举 $O(n^2)$

算法思路

- 用两重循环枚举所有下标对 (i, j)，其中 $0 \le i < j < n$。
- 对每一对 (i, j) 计算面积 $(j - i) \cdot \min(h_i, h_j)$，维护最大值。
  这里 $h_i = \text{height}[i]$。
- 时间复杂度为 $O(n^2)$，当 n 稍大时代码会偏慢，但逻辑最直观，适合作为入门思路。

```cpp
#include <bits/stdc++.h>
using namespace std;

int n, a[105], ans;

int main() {
    cin >> n;
    for (int i = 0; i < n; i++) cin >> a[i];

    // 枚举每一种 (i, j) 组合
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            ans = max(ans, (j - i) * min(a[i], a[j]));
        }
```

```
    }

    cout << ans;
    return 0;
}
```

二、排序贪心 $O(n \log n)$

算法思路

从另一个角度看问题：

- 将所有下标按高度从大到小排序，依次处理第 k 高的竖线。处理到它时，之前出现的竖线高度都不低于它。
  因此，这根竖线一定是当前容器的“短板”。
- 容器高度固定为当前高度，只需尽量拉大宽度。
  维护目前为止出现过的最小下标 mn 与最大下标 mx，对当前下标 idx 计算

$$\text{area} = h_{idx} \cdot \max\big(|idx - mn|, |idx - mx|\big).$$

- 每个下标仅参与一次排序和一次更新，整体复杂度为 $O(n \log n)$。

```
#include <bits/stdc++.h>
using namespace std;

int n, a[105], p[105], ans;

int main() {
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> a[i];
        p[i] = i; // 记录原始下标
    }

    // 按高度从大到小排序下标
    sort(p, p + n, [&](int x, int y) {
        return a[x] > a[y];
    });
```

```cpp

    int mn = (int)1e9, mx = -1;
    for (int i = 0; i < n; i++) {
        int idx = p[i]; // 当前柱子的原始位置
        if (mx != -1) {
            int width = max(abs(idx - mn), abs(idx - mx));
            ans = max(ans, a[idx] * width);
        }
        mn = min(mn, idx);
        mx = max(mx, idx);
    }

    cout << ans;
    return 0;
}
```

三、双指针法（最优解）$O(n)$

算法核心与证明直观

设左右指针初始为 $L = 0, R = n - 1$，当前面积为

$$S = (R - L) \cdot \min(h_L, h_R).$$

- 若 $h_L < h_R$，则高度由左边决定。如果此时只移动右指针向左，宽度变小，高度最多仍为 h_L，面积不会变大，所以这些状态可以全部剪枝；因此我们只需要右移短的一侧：$L \leftarrow L + 1$。
- 同理，当 $h_L \geq h_R$ 时，左侧再怎么不动，右侧往左移动都不可能得到比当前更大的面积，应当移动右指针：$R \leftarrow R - 1$。
- 每一步都缩小区间，指针一共移动 $O(n)$ 次，即可穷尽所有“有希望”的状态。

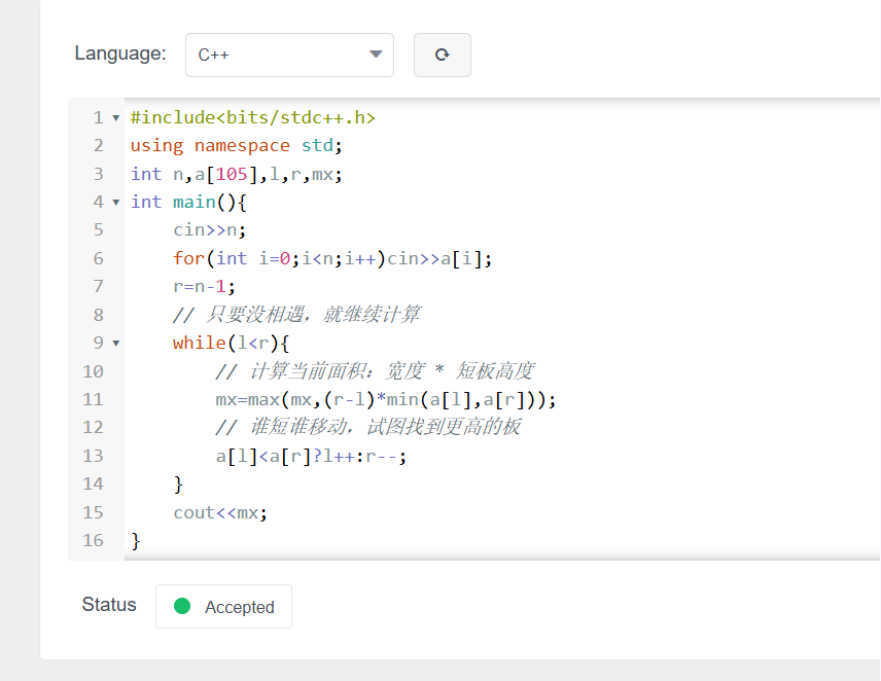
```cpp
#include <bits/stdc++.h>
using namespace std;

int n, a[105], l, r, mx;

int main() {
```

```
    cin >> n;
    for (int i = 0; i < n; i++) cin >> a[i];

    l = 0;
    r = n - 1;

    // 只要没有相遇，就持续计算并尝试移动短板
    while (l < r) {
        mx = max(mx, (r - l) * min(a[l], a[r])); // 当前面积

        // 谁更短就移动谁，寻找更高的一侧
        if (a[l] < a[r]) {
            l++;
        } else {
            r--;
        }
    }

    cout << mx;
    return 0;
}
```

四、通过测试截图

说明

上述三份 C++ 代码均在评测系统上编译运行通过，状态为 `Accepted`。下图给出其中双指针解法的提交截图作为证明。



图：双指针解法提交记录，状态为 Accepted