42. Trapping Rain Water

Условие задачи

Дан массив $height$ из n неотрицательных целых чисел, задающий карту высот. Ширина каждого столбца равна 1. После дождя между столбцами может скапливаться вода.

Требуется вычислить суммарный объём воды, который может быть удержан.

Ограничения:

- $n == height.length$
- $1 \le n \le 2 \cdot 10^4$
- $0 \le height[i] \le 10^5$

Пример 1

```
Вход:  height = [0,1,0,2,1,0,1,3,2,1,2,1]
Выход: 6
```

Пример 2

```
Вход:  height = [4,2,0,3,2,5]
Выход: 9
```

Алгоритм

Идея: количество воды над каждой позицией ограничено максимальной высотой слева и максимальной высотой справа. Мы реализуем это в два прохода, но немного необычным способом: сначала считаем *потенциальную* воду, ограниченную только слева, а затем "обрезаем" её справа.

Пусть h_i — высота столбца на позиции i (нумерация с 0).

Шаг 1. Проход слева направо (массив `need`)

Поддерживаем l_max — максимальную высоту среди $h_0, h_1, \ldots, h_i$. Тогда

$$need[i] = l\_max - h_i.$$

Это значение показывает, сколько воды **могло бы** стоять над i, если бы справа была бесконечно высокая стенка (то есть ограничение только слева).

Шаг 2. Проход справа налево (обрезка потенциала)

Поддерживаем r_max — максимальную высоту среди $h_i, h_{i+1}, \ldots, h_{n-1}$. Тогда правая граница позволяет накопить над i не больше, чем

$$r\_cont(i) = r\_max - h_i.$$

Реальное количество воды над позицией i равно

$$add(i) = \max\big(0,\ \min(need[i],\ r\_cont(i))\big).$$

Суммируя $add(i)$ по всем i, получаем ответ.

Корректность

Докажем, что алгоритм возвращает правильный объём воды.

Лемма 1. Корректность массива `need`

Для каждого i значение $need[i]$ равно разности между максимальной высотой слева и высотой текущего столбца:

$$need[i] = \max_{0 \le k \le i} h_k - h_i.$$

Доказательство. Переменная l_max на шаге i хранит максимум среди $h_0, \dots, h_i$ по определению, так как на каждом шаге обновляется присваиванием $l\_max = \max(l\_max, h_i)$. Затем записывается $need[i] = l\_max - h_i$. □

Лемма 2. Ограничение водой справа

Для каждого i значение $r\_cont(i) = r\_max - h_i$ равно максимально возможному объёму воды над i, если учитывать только правую границу (то есть считать, что слева стенка бесконечно высокая).

Доказательство. Аналогично лемме 1: на проходе справа налево переменная r_max хранит максимум среди $h_i, \dots, h_{n-1}$, потому что обновляется как $r\_max = \max(r\_max, h_i)$. Тогда уровень воды над i не может быть выше r_max, а значит объём воды над столбцом i не превышает $r\_max - h_i$. □

Лемма 3. Формула для воды над одной позицией

Пусть $L_i = \max_{0 \le k \le i} h_k$ и $R_i = \max_{i \le k \le n-1} h_k$. Тогда количество воды над позицией i равно

$$w_i = \max\bigl(0,\ \min(L_i, R_i) - h_i\bigr).$$

Доказательство. Если уровень воды выше $\min(L_i, R_i)$, то она “перельётся” через более низкую стенку (либо левую, либо правую), что невозможно в устойчивом состоянии. Если же уровень не выше $\min(L_i, R_i)$, то обе стороны удерживают воду. Значит максимально возможный уровень воды — $\min(L_i, R_i)$, откуда и формула. □

Следствие. Корректность вычисления `add`

По лемме 1: $need[i] = L_i - h_i$. По лемме 2: $r\_cont(i) = R_i - h_i$.

Тогда

$$\min(need[i], r\_cont(i)) = \min(L_i - h_i,\ R_i - h_i) = \min(L_i, R_i) - h_i.$$

С учётом отсечения отрицательных значений получаем ровно w_i из леммы 3. Следовательно, сумма всех добавлений равна суммарному объёму воды. □

Код (базовая реализация)

```
from typing import List

class Solution:
    def trap(self, height: List[int]) -> int:
        n = len(height)
        if n < 3:
            return 0

        need = [0] * n

        l_max = 0
        i = 0
        while i < n:
            cur_h = height[i]

            if cur_h > l_max:
                l_max = cur_h

            need[i] = l_max - cur_h
            i += 1

        total_water = 0
        r_max = 0

        j = n - 1
        while j >= 0:
            cur_h = height[j]

            if cur_h > r_max:
                r_max = cur_h

            r_cont = r_max - cur_h

            if need[j] < r_cont:
                add_water = need[j]
            else:
                add_water = r_cont

            if add_water > 0:
                total_water += add_water

            j -= 1

        return total_water
```

Временная сложность — асимптотика

Алгоритм выполняет два линейных прохода по массиву длины n:

- построение массива `need`: $O(n)$,
- проход справа налево и суммирование ответа: $O(n)$.

Итого: $O(n)$.

Затраты памяти — асимптотика

Используется массив `need` длины n и несколько переменных. Дополнительная память: $O(n)$.