Báo cáo Cấu trúc dữ liệu & Giải thuật

Ngô Phạm Minh Đức

Bài toán "Trapping Rain Water"

1 Trình bày mô tả giải thuật theo các bước phân tích sơ bộ

1.1 Mô tả bài toán

Bài toán yêu cầu tính tổng lượng nước mưa có thể được giữ lại giữa các cột có độ cao khác nhau trong một mảng 1 chiều.

1.2 Phương pháp giải quyết

Phương pháp 1: Sử dụng mảng phụ (Extra Space)

- Tạo hai mảng phụ `left_max` và `right_max` để lưu trữ giá trị lớn nhất từ trái qua phải và từ phải qua trái tại mỗi vị trí.
- Với mỗi cột, lượng nước giữ được là:

$$\text{water}[i] = \min(\text{left\_max}[i], \text{right\_max}[i]) - \text{height}[i]$$

- Tổng lượng nước là tổng các giá trị nước giữ được tại mỗi cột.

Phương pháp 2: Sử dụng hai con trỏ (Two Pointers)

- Dùng hai con trỏ `left` và `right` để di chuyển từ hai đầu mảng về giữa.
- Giữ hai giá trị `left_max` và `right_max` để theo dõi độ cao lớn nhất hiện tại từ hai phía.
- Tính lượng nước giữ được tại mỗi bước bằng cách so sánh giá trị tại con trỏ với giá trị lớn nhất tương ứng.

2 Mã giả

2.1 Sử dụng mảng phụ

```
Hàm trapRainWater(height, n):
    If n <= 2:
        return 0

    Tạo mảng left_max, right_max
    left_max[0] = height[0]
    For  i = 1 đến n-1:
        left_max[i] = max(left_max[i-1], height[i])

    right_max[n-1] = height[n-1]
    For  i = n-2 đến 0:
        right_max[i] = max(right_max[i+1], height[i])

    water = 0
    For  i = 0 đến n-1:
        water += min(left_max[i], right_max[i]) - height[i]
    return water
```

2.2 Sử dụng hai con trỏ

```
Hàm trapRainWater(height, n):
    If n <= 2:
        return 0
    left = 0, right = n-1
    left_max = 0, right_max = 0
    water = 0
    while left <= right:
        If height[left] <= height[right]:
            If height[left] >= left_max:
                left_max = height[left]
            Else:
                water += left_max - height[left]
            left++
        Else:
            If height[right] >= right_max:
                right_max = height[right]
            Else:
                water += right_max - height[right]
            right--
    return water
```

3 Cài đặt một ví dụ bao gồm tập dữ liệu đầu vào và đầu ra theo giải thuật tương ứng

3.1 Cách 1: Sử dụng mảng phụ

Code:

```
#include <iostream>
using namespace std;

int trapRainWater(int height[], int n) {
    if (n <= 2) return 0;

    int left_max[n], right_max[n];
    left_max[0] = height[0];
    for (int i = 1; i < n; i++) {
        left_max[i] = max(left_max[i - 1], height[i]);
    }

    right_max[n - 1] = height[n - 1];
    for (int i = n - 2; i >= 0; i--) {
        right_max[i] = max(right_max[i + 1], height[i
            ]);
    }

    int water = 0;
    for (int i = 0; i < n; i++) {
        water += min(left_max[i], right_max[i]) -
            height[i];
    }

    return water;
}

int main() {
    int height[] = {7, 0, 4, 2, 5, 0, 6, 4, 0, 5};
    int n = sizeof(height) / sizeof(height[0]);
    cout << "Tong luong nuoc: " << trapRainWater(
        height, n);
    return 0;
}
```

Output:

```
Tong luong nuoc: 25
```

3.2 Cách 2: Sử dụng hai con trỏ

Code:

```
#include <iostream>
using namespace std;

int trapRainWater(int height[], int n) {
    if (n <= 2) return 0;

    int left = 0, right = n - 1;
    int left_max = 0, right_max = 0;
    int water = 0;

    while (left <= right) {
        if (height[left] <= height[right]) {
            if (height[left] >= left_max) {
                left_max = height[left];
            } else {
                water += left_max - height[left];
            }
            left++;
        } else {
            if (height[right] >= right_max) {
                right_max = height[right];
            } else {
                water += right_max - height[right];
            }
            right--;
        }
    }

    return water;
}

int main() {
    int height[] = {7, 0, 4, 2, 5, 0, 6, 4, 0, 5};
    int n = sizeof(height) / sizeof(height[0]);
```

```
    cout << "Tong luong nuoc: " << trapRainWater(
        height, n);
    return 0;
}
```

Output:

```
Tong luong nuoc: 25
```

4 Phức tạp thuật toán

- **Phương pháp 1: Sử dụng mảng phụ**
  - **Độ phức tạp thời gian:** $O(n)$, với n là kích thước mảng đầu vào. Các vòng lặp tính `left_max`, `right_max`.
  - **Độ phức tạp không gian:** $O(n)$, do cần lưu trữ hai mảng `left_max` và `right_max`.
- **Phương pháp 2: Sử dụng hai con trỏ**
  - **Độ phức tạp thời gian:** $O(n)$, do chỉ sử dụng một vòng lặp để tính lượng nước.
  - **Độ phức tạp không gian:** $O(1)$, vì không cần sử dụng thêm không gian lưu trữ ngoài các biến `left_max`, `right_max`, `left`, `right`.