Lập trình C++: Quy hoạch động (Dynamic Programming)

Chiến thuật "Chia để trị và Ghi nhớ"

Slide Learning CPP

Ngày 15 tháng 1 năm 2026

Lộ trình khám phá

4 Trạm dừng chân quan trọng

- **Chương 1:** Khái niệm "Ghi nhớ để không lãng phí".
- **Chương 2:** Chiếc túi chứa kiến thức (Vector 1D).
- **Chương 3:** Bản đồ tọa độ (Vector 2D).
- **Chương 4:** Thực hành giải bài toán kinh điển.

Tư duy cốt lõi

Thay vì mỗi ngày đều đo đạc lại từ đầu, bạn chỉ cần nhìn vào bản vẽ đã lưu lại để tiếp tục xây cao hơn.

Chương 1: Ghi nhớ để không lãng phí

Phép toán đơn giản

Tính: $1+1+1+1+1$. Kết quả $= 5$.
Thêm một số "+1"vào cuối hàng. Kết quả $= 6$.

- **Tại sao nhanh?**: Bạn không đếm lại từ đầu mà sử dụng kết quả 5 đã nhớ.
- **Ẩn dụ "Hố cát"**: Mỗi khi giải xong bài toán nhỏ, hãy đào một cái hố (ô nhớ) và đặt kết quả vào đó.

Chương 1: Công cụ "Đào hố"trong C++

Để lưu trữ trong C++, chúng ta sử dụng `vector`.

- **Vector 1D**: Dãy các hộp xếp hàng ngang.
- **Vector 2D**: Tủ nhiều ngăn kéo (hàng và cột).

```
1 // Khai bao vector 1 chieu co 10 phan tu, ban dau deu bang 0
2 vector<int> f(10, 0);
3
4 // Khai bao vector 2 chieu (10 dong, 10 cot), ban dau bang 0
5 vector<vector<int>> dp(10, vector<int>(10, 0));
```

Chương 2: Chiếc túi kiến thức (Vector 1D)

Ẩn dụ: Chiếc thắt lưng thợ sửa chữa

`vector<int> f(n)` là thắt lưng có n chiếc túi đánh số từ 0 đến $n-1$.

2 Bước vận hành DP

1. **Khởi tạo (Base case)**: Đặt những giá trị cơ bản đầu tiên vào túi.
2. **Công thức truy hồi (State transition)**: Cách dùng các túi cũ để tính túi mới.

Chương 2: Ví dụ dãy Fibonacci

Công thức: $f[i] = f[i-1] + f[i-2]$

```cpp
int n = 10;
vector<int> f(n + 1);

// Buoc 1: Khoi tao
f[0] = 0;
f[1] = 1;

// Buoc 2: Truy hoi
for (int i = 2; i <= n; i++) {
    f[i] = f[i-1] + f[i-2];
}
```

Chương 3: Bản đồ tọa độ (Vector 2D)

Ẩn dụ: Tòa nhà nhiều tầng

`vector<vector<int>> dp(hang, vector<int>(cot))`

- `dp[i]`: Tầng thứ i của tòa nhà.
- `dp[i][j]`: Căn phòng số j tại tầng thứ i.

Tư duy lưới

Để tính giá trị `dp[i][j]`, ta có thể nhìn vào phòng bên cạnh `dp[i][j-1]` hoặc phòng ở tầng trên `dp[i-1][j]`.

Thử thách tư duy

Câu hỏi

Nếu bạn chỉ được phép **đi sang phải** hoặc **đi xuống dưới** trên một lưới ô vuông. Để biết số cách đi đến ô `dp[i][j]`, bạn cần cộng số cách từ những ô nào?

Thử thách tư duy

Câu hỏi

Nếu bạn chỉ được phép **đi sang phải** hoặc **đi xuống dưới** trên một lưới ô vuông. Để biết số cách đi đến ô `dp[i][j]`, bạn cần cộng số cách từ những ô nào?

- **Đáp án**: Ô bên trái `dp[i][j-1]` và ô phía trên `dp[i-1][j]`.

Đề bài

Đếm số cách đi từ (0,0) đến (M,N) nếu chỉ được đi sang phải hoặc xuống dưới.

```
// Khoi tao o dau tien
dp[0][0] = 1;

for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
        if (i == 0 && j == 0) continue;

        int tu_phia_tren = (i > 0) ? dp[i-1][j] : 0;
        int tu_ben_trai = (j > 0) ? dp[i][j-1] : 0;

        dp[i][j] = tu_phia_tren + tu_ben_trai;
    }
}
```

Tổng kết

Ghi nhớ

Quy hoạch động giống như việc **xây cầu**. Bạn không thể xây nhịp thứ 10 nếu chưa xây nhịp thứ 9.

- Lưu trữ kết quả bài toán nhỏ.
- Tái sử dụng để giải bài toán lớn hơn.
- Tiết kiệm thời gian tính toán.

Chúc các bạn chinh phục thành công DP!