Huấn luyện viên tư duy thuật toán

Bài toán: Base Conversion (Chuyển đổi cơ số)

Slide Learning C++

Ngày 16 tháng 1 năm 2026

Lời chào dẫn nhập

Mục tiêu bài học

Chào mừng bạn! Chúng ta sẽ cùng nhau "mổ xẻ"bài toán chuyển đổi cơ số theo phong cách **Learning How to Learn**, tập trung vào bản chất logic và hình ảnh hóa thay vì chỉ nhìn vào code.

1. Phẫu thuật đề bài (Briefing)

Cốt lõi vấn đề

- **Nhiệm vụ:** Nhập số thập phân (cơ số 10), in ra dạng Nhị phân (cơ số 2).
- **Điều kiện bắt buộc:** Sử dụng **Đệ quy (Recursion)**.
- **Dữ liệu:** Nhiều bộ thử nghiệm, N có thể lên tới 10^{18}.

Lộ trình tư duy

1. Hiểu bản chất "chia để trị".
2. Xây dựng cấu trúc đệ quy (Điểm dừng & Hành động).
3. Xử lý thứ tự in ấn (Tránh bẫy in ngược).

2. Chunk 1: Cơ chế "Rút tỉa"

- Hệ nhị phân chỉ hiểu hai trạng thái: **Dư lẻ (1)** hoặc **Chẵn đôi (0)**.
- **Quy tắc:**
  1. Lấy số đó chia cho 2.
  2. Ghi lại số dư (0 hoặc 1).
  3. Lấy phần nguyên tiếp tục chia cho đến khi bằng 0.

Bẫy logic (The Trap)

Nếu vừa chia vừa in ngay, kết quả sẽ bị **ngược** (Ví dụ 6 ra 011 thay vì 110).

Thử thách tư duy (Mental Check)

Câu hỏi

Nếu $N = 13$, hãy thực hiện các bước chia cho 2. Liệt kê các số dư thu được theo thứ tự từ bước đầu đến cuối?

Thử thách tư duy (Mental Check)

Câu hỏi

Nếu $N = 13$, hãy thực hiện các bước chia cho 2. Liệt kê các số dư thu được theo thứ tự từ bước đầu đến cuối?

Đáp án & Giải thích

$13 \div 2 = 6$ dư $\mathbf{1}$ (đơn vị 2^0)
$6 \div 2 = 3$ dư $\mathbf{0}$ (hàng 2^1)
$3 \div 2 = 1$ dư $\mathbf{1}$ (hàng 2^2)
$1 \div 2 = 0$ dư $\mathbf{1}$ (hàng 2^3)
$\rightarrow$ Kết quả: $\mathbf{1101}$. Chữ số cuối cùng tìm được lại có giá trị lớn nhất.

Mảnh ghép 2: Cơ chế Stack trong Đệ quy

- Đệ quy sử dụng cơ chế **Stack (Ngăn xếp)**: Vào sau - Ra trước.
- **Bước gọi:** Chia cho 2 đến khi chạm đáy ($N = 0$).
- **Bước in:** Chỉ thực hiện **sau khi** lệnh gọi đệ quy quay trở về.

Mô phỏng đệ quy

1. `convert(13)` gọi `convert(6)`
2. `convert(6)` gọi `convert(3)`
3. `convert(3)` gọi `convert(1)`
4. `convert(1)` gọi `convert(0)` $\rightarrow$ Chạm đáy!

Mảnh ghép 3: Thiết kế hàm đệ quy

Thành phần quan trọng

1. **Điểm dừng (Base Case):** Khi nào không chia nữa?
2. **Bước đệ quy (Recursive Step):** Gọi lại chính nó với $N/2$.

Thử thách lựa chọn

Nên đặt lệnh in ở đâu để đúng thứ tự?

- **Phương án A:** In số dư $\rightarrow$ Gọi `convert(N/2)`
- **Phương án B:** Gọi `convert(N/2)` $\rightarrow$ In số dư

Mảnh ghép 3: Thiết kế hàm đệ quy

Thành phần quan trọng

1. **Điểm dừng (Base Case):** Khi nào không chia nữa?
2. **Bước đệ quy (Recursive Step):** Gọi lại chính nó với $N/2$.

Thử thách lựa chọn

Nên đặt lệnh in ở đâu để đúng thứ tự?

- **Phương án A:** In số dư $\rightarrow$ Gọi `convert(N/2)`
- **Phương án B:** Gọi `convert(N/2)` $\rightarrow$ In số dư

Đáp án: Phương án B. Lệnh in sau lời gọi đệ quy sẽ chờ "bung"ngược từ đáy lên.

Mô phỏng "Vụ nổ ngược"

Tầng	Trạng thái	Hành động tiếp theo
Tầng 1	`convert(13)`	Gọi `convert(6)`, chờ in $13\%2 = 1$
Tầng 2	`convert(6)`	Gọi `convert(3)`, chờ in $6\%2 = 0$
Tầng 3	`convert(3)`	Gọi `convert(1)`, chờ in $3\%2 = 1$
Tầng 4	`convert(1)`	Gọi `convert(0)`, chờ in $1\%2 = 1$
Tầng 5	`convert(0)`	**Chạm đáy!** Kết thúc gọi.

Mô phỏng "Vụ nổ ngược"

Tầng	Trạng thái	Hành động tiếp theo
Tầng 1	`convert(13)`	Gọi `convert(6)`, chờ in $13\%2 = 1$
Tầng 2	`convert(6)`	Gọi `convert(3)`, chờ in $6\%2 = 0$
Tầng 3	`convert(3)`	Gọi `convert(1)`, chờ in $3\%2 = 1$
Tầng 4	`convert(1)`	Gọi `convert(0)`, chờ in $1\%2 = 1$
Tầng 5	`convert(0)`	**Chạm đáy!** Kết thúc gọi.

Thứ tự in thực tế: $1 \to 1 \to 0 \to 1$.

Mảnh ghép cuối cùng: Điểm dừng (Base Case)

Điều kiện dừng

Khi nào chúng ta ngừng chia?

- A. Khi $N = 1$
- B. Khi $N = 0$
- C. Khi N âm

Mảnh ghép cuối cùng: Điểm dừng (Base Case)

Điều kiện dừng

Khi nào chúng ta ngừng chia?

- A. Khi $N = 1$
- B. Khi $N = 0$
- C. Khi N âm

Đáp án: B. Khi $N = 0$, không còn gì để chia.

Lưu ý: Nếu đề cho $N = 0$ ngay từ đầu, cần xử lý riêng để in ra số 0.

Tổng kết Logic & Mã giả

```
void convert(long long n) {
    // 1. Diem dung (Base Case)
    if (n == 0) return;

    // 2. Goi de quy (Di sau vao stack)
    convert(n / 2);

    // 3. In ket qua (Thuc hien khi quay lui)
    cout << n % 2;
}

```

Listing 1: Cấu trúc hàm đệ quy

Kết quả với N = 3

1. convert(1) chạy trước → in 1.
2. convert(3) chạy sau → in 1.

Kết quả: 11.

Bước cuối cùng: Thực hành

Bạn đã nắm vững "trái tim"của đệ quy. Hãy thử hiện thực hóa nó bằng ngôn ngữ lập trình của bạn!

- Bạn muốn tôi hỗ trợ viết mã hoàn chỉnh cho bài này?
- Hay bạn muốn chuyển sang một thử thách đệ quy khó hơn?