ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phân tích và thiết kế thuật toán

Sửa bài tập nhóm 9

Sinh viên:

Hồ Ngọc Luật - 23520900

Nguyễn Trần Quang Minh - 23520943

Giáo viên hướng dẫn: Nguyễn Thanh Sơn

Ngày 04 tháng 12 năm 2024

Mục lục

1 Lý thuyết **3**
1.1 Có phải mọi bài toán đều có thể giải quyết bằng quy hoạch động không? Tại sao? 3
1.2 Trong thực tế, bạn đã gặp bài toán nào có thể áp dụng quy hoạch động? Hãy chia sẻ cách tiếp cận. 3
1.3 Hãy phân tích và làm rõ ưu, nhược điểm của 2 phương pháp Top-down và Bottom-up. Bạn sẽ ưu tiên phương pháp nào? Vì sao? 3

2 Thực hành **5**
2.1 Tóm tắt ý tưởng 5
2.2 Độ phức tạp 5
2.3 Mã giả C++ 5

1 Lý thuyết

1.1 Có phải mọi bài toán đều có thể giải quyết bằng quy hoạch động không? Tại sao?

Trả lời:

Không phải mọi bài toán đều có thể giải quyết bằng quy hoạch động. Điều kiện áp dụng quy hoạch động bao gồm:

- **Tính tối ưu con** (Optimal Substructure): Bài toán lớn có thể chia thành các bài toán con, và lời giải của bài toán lớn được xây dựng từ lời giải của các bài toán con.
- **Tính trùng lặp của bài toán con** (Overlapping Subproblems): Các bài toán con xuất hiện lặp đi lặp lại, cho phép lưu trữ kết quả để tránh tính toán lại.

Nếu một bài toán không thỏa mãn hai điều kiện trên, thì không thể áp dụng quy hoạch động.

1.2 Trong thực tế, bạn đã gặp bài toán nào có thể áp dụng quy hoạch động? Hãy chia sẻ cách tiếp cận.

Trả lời:

Ví dụ về các bài toán thực tế sử dụng quy hoạch động:

- **Bài toán ba lô** (Knapsack Problem): Lựa chọn các món đồ sao cho tổng giá trị tối đa nhưng không vượt quá trọng lượng cho phép.
  - **Cách tiếp cận**: Sử dụng mảng 2D để lưu trữ giá trị tối ưu cho từng trọng lượng và số lượng đồ.
- **Bài toán chuỗi con chung dài nhất** (Longest Common Subsequence - LCS): Tìm độ dài chuỗi con chung dài nhất giữa hai chuỗi.
  - **Cách tiếp cận**: Dùng mảng 2D để lưu kết quả các bước nhỏ và xây dựng kết quả lớn hơn dựa trên các kết quả đã tính.

1.3 Hãy phân tích và làm rõ ưu, nhược điểm của 2 phương pháp Top-down và Bottom-up. Bạn sẽ ưu tiên phương pháp nào? Vì sao?

Trả lời:

Phương pháp Top-down (Sử dụng đệ quy + ghi nhớ - Memoization):

- **Ưu điểm:**
  - Trực quan và dễ viết mã khi suy nghĩ từ bài toán lớn xuống bài toán con.
  - Chỉ tính toán những bài toán con cần thiết.

- **Nhược điểm:**
  - Tốn không gian ngăn xếp do đệ quy (có nguy cơ tràn ngăn xếp).
  - Có thể chậm nếu số lần gọi đệ quy lớn.

Phương pháp Bottom-up (Dùng vòng lặp + bảng - Tabulation):

- **Ưu điểm:**
  - Không tốn không gian ngăn xếp vì không sử dụng đệ quy.
  - Hiệu quả về mặt tốc độ khi tính toán từ bài toán nhỏ lên bài toán lớn.
- **Nhược điểm:**
  - Phải tính toàn bộ các bài toán con, kể cả những bài toán không cần thiết.

2 Thực hành

2.1 Tóm tắt ý tưởng

Bài toán được giải bằng phương pháp quy hoạch động:

- Gọi $dp[i]$ là chi phí tối thiểu để chú ếch đến được hòn đá thứ i.
- Chú ếch có thể nhảy từ hòn đá $i-j$ (với $1 \leq j \leq k$) đến hòn đá i. Chi phí là $|h[i]-h[i-j]|$.
- Công thức truy hồi:

$$dp[i] = \min_{j=1}^{k} \left(dp[i-j] + |h[i]-h[i-j]| \right), \quad \text{với } i-j \geq 1.$$

- Khởi tạo:

$$dp[1] = 0 \quad \text{(chú ếch bắt đầu ở hòn đá đầu tiên, chi phí là 0).}$$

- Kết quả cuối cùng:

$$dp[n] \quad \text{(chi phí tối thiểu để đến hòn đá cuối cùng).}$$

2.2 Độ phức tạp

- **Thời gian**: $O(n \times k)$, với $n \leq 10^5$ và $k \leq 100$.
- **Không gian**: $O(n)$, sử dụng mảng dp để lưu kết quả.

2.3 Mã giả C++

Dưới đây là mã C++ để giải bài toán:

```cpp
int frog_jump(int n, int k, vector<int>& heights) {
    vector<int> dp(n, INT_MAX); // Khởi tạo mảng dp với giá trị lớn ban đầu
    dp[0] = 0; // Chi phí đến hòn đá đầu tiên là 0

    // Duyệt qua từng hòn đá
    for (int i = 1; i < n; i++) {
        for (int j = 1; j <= k && i - j >= 0; j++) {
            dp[i] = min(dp[i], dp[i - j] + abs(heights[i] - heights[i - j]));
        }
    }

    return dp[n - 1]; // Chi phí tối thiểu để đến hòn đá cuối cùng
}
```

Ví dụ minh họa

Input:

```
5 3
10 30 40 50 20
```

Output:

```
30
```

Giải thích:

- $dp[1] = 0$
- $dp[2] = \min(dp[1] + |30 - 10|) = 20$
- $dp[3] = \min(dp[2] + |40 - 30|, dp[1] + |40 - 10|) = 30$
- $dp[4] = \min(dp[3] + |50 - 40|, dp[2] + |50 - 30|, dp[1] + |50 - 10|) = 40$
- $dp[5] = \min(dp[4] + |20 - 50|, dp[3] + |20 - 40|, dp[2] + |20 - 30|) = 30$

Kết quả cuối cùng là 30.