ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

BÀI TẬP VỀ NHÀ NHÓM 9

Môn học: Phân tích và thiết kế thuật toán

Sinh viên thực hiện:
Đặng Quốc Cường
Nguyễn Đình Thiên Quang
(Nhóm 1)

Giảng viên môn học:
Nguyễn Thanh Sơn

Ngày 5 tháng 12 năm 2024

Mục lục

1 Bài tập lý thuyết

Câu 1: Có phải mọi bài toán đều có thể giải quyết bằng quy hoạch động không? Tại sao?

Không phải mọi bài toán đều có thể giải quyết bằng quy hoạch động. Quy hoạch động chỉ áp dụng hiệu quả đối với những bài toán có tính chất tối ưu con (optimal substructure) và tính trùng lặp con (overlapping subproblems). Điều này có nghĩa là bài toán cần được chia nhỏ thành các bài toán con độc lập, và những bài toán con này có thể được giải quyết và lưu trữ để tránh tính toán lại nhiều lần. Nếu bài toán không thỏa mãn các điều kiện này, việc sử dụng quy hoạch động có thể không mang lại hiệu quả.

Câu 2: Trong thực tế, bạn đã gặp bài toán nào có thể áp dụng quy hoạch động? Hãy chia sẻ cách tiếp cận.

Một bài toán phổ biến có thể áp dụng quy hoạch động là bài toán "Lợi nhuận tối đa từ việc mua và bán cổ phiếu". Cách tiếp cận là chia bài toán thành các bài toán con: tìm giá trị tối đa lợi nhuận có thể thu được khi mua và bán cổ phiếu ở các ngày khác nhau, sau đó lưu trữ kết quả của mỗi ngày để tránh tính toán lại. Quy hoạch động giúp giảm thiểu thời gian tính toán, thay vì tính lại lợi nhuận cho mỗi cặp ngày bán-mua.

Câu 3: Phân tích ưu, nhược điểm của phương pháp Top-down và Bottom-up. Bạn ưu tiên phương pháp nào? Vì sao?

Top-down (Từ trên xuống): Phương pháp này sử dụng đệ quy để giải quyết bài toán. Ưu điểm là dễ hiểu và dễ triển khai, nhưng có thể gây ra vấn đề về hiệu năng nếu không sử dụng kỹ thuật lưu trữ kết quả (memoization) do tính trùng lặp trong các bài toán con.
Bottom-up (Từ dưới lên): Phương pháp này giải quyết các bài toán con từ đơn giản đến phức tạp, bắt đầu từ các trường hợp cơ bản và xây dựng dần dần các bài toán phức tạp hơn. Ưu điểm là không có đệ quy, giúp tiết kiệm bộ nhớ và tránh việc tính toán lại. Tuy nhiên, phương pháp này có thể khó hiểu hơn và yêu cầu nhiều công sức trong việc xác định thứ tự xử lý.
Ưu tiên phương pháp Bottom-up do tính hiệu quả trong việc sử dụng bộ nhớ và tránh lặp lại tính toán.

2 Bài tập thực hành

2.1 Bài 1 : Chú ếch

1. Phân tích bài toán

- **Input**:
  - Số hòn đá n và giới hạn bước nhảy k.
  - Độ cao của từng hòn đá $h_1, h_2, \ldots, h_n$.
- **Output**:

  - Chi phí tối thiểu để nhảy từ hòn đá đầu tiên đến hòn đá cuối cùng.

- **Quy tắc nhảy**:
  - Tại mỗi hòn đá i, ếch có thể nhảy đến các hòn $i+1, i+2, \ldots, i+k$ (nếu tồn tại).
  - Chi phí nhảy từ i đến j là $|h[i] - h[j]|$.

2. Ý tưởng giải bài toán

Sử dụng mảng động dp:

- $dp[i]$: chi phí tối thiểu để nhảy đến hòn đá thứ i.
- **Công thức truy hồi**:

$$dp[i] = \min_{j=1}^{\min(k,i-1)} \{dp[i-j] + |h[i] - h[i-j]|\}$$

- **Khởi tạo**:

$$dp[1] = 0 \quad \text{(chi phí ban đầu là 0)}$$

- **Kết quả**:

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

3. Mã giả

```
def frog_jump(n, k, heights):
    # Khởi tạo mảng động
    dp = [float('inf')] * n
    dp[0] = 0

    for i in range(1, n):
        for j in range(1, min(k, i) + 1):
            dp[i] = min(dp[i], dp[i-j] + abs(heights[i] - heights[i-j]))

    return dp[n-1]
```

4. Độ phức tạp

- **Thời gian**: $O(n \cdot k)$, do duyệt n hòn đá và mỗi hòn đá xét tối đa k bước nhảy.
- **Không gian**: $O(n)$, do lưu trữ mảng dp.