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

BÀI TẬP NHÓM 12
PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

Giáo viên hướng dẫn: ***Nguyễn Thanh Sơn***

Nhóm 12:

- ***Hoàng Minh Thái*** **- 23521414**
- ***Nguyễn Trọng Tất Thành*** **- 23521414**

Mục lục

1. Câu hỏi 1.2

[label=**Câu 0.**]**Có phải mọi bài toán đều có thể giải quyết bằng quy hoạch động không? Tại sao?**
Không. Quy hoạch động (Dynamic Programming - DP) chỉ phù hợp với các bài toán có cấu trúc con tối ưu và không có phụ thuộc vòng lặp. Những bài toán không thoả mãn các điều kiện này sẽ không thể áp dụng DP hiệu quả.
Trong thực tế, bạn đã gặp bài toán nào có thể áp dụng quy hoạch động? Hãy chia sẻ cách tiếp cận.
Ví dụ: *Bài toán ba lô (Knapsack Problem).*
Cách tiếp cận:

2.
   - Xây dựng mảng `dp[i][w]` lưu giá trị tối ưu cho i vật và trọng lượng tối đa w.
   - Quy hoạch:

$$dp[i][w] = \max(dp[i-1][w], dp[i-1][w - \text{weight}_i] + \text{value}_i)$$

   - Kết quả là `dp[n][W]`, với n là số vật và W là trọng lượng tối đa.

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?**
   - **Top-down (Ghi nhớ - Memoization):**
     - Ưu điểm: Dễ triển khai, sử dụng đệ quy tự nhiên.
     - Nhược điểm: Tốn nhiều bộ nhớ do lưu trữ trạng thái đệ quy và sử dụng stack.
   - **Bottom-up (Lập bảng - Tabulation):**
     - Ưu điểm: Tiết kiệm bộ nhớ, tránh lỗi stack overflow.
     - Nhược điểm: Có thể khó hiểu và phức tạp hơn trong triển khai.
   - **Ưu tiên:** Chọn phương pháp **Bottom-up** nếu bộ nhớ hạn chế hoặc bài toán yêu cầu tối ưu hiệu suất.

2. Câu 2.1:

Ý tưởng:

Sử dụng quy hoạch động (Dynamic Programming) để tìm chi phí nhảy tối thiểu. Gọi `dp[i]` là chi phí tối thiểu để nhảy đến hòn đá thứ i. Công thức truy hồi:

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

Với `dp[1]` $= 0$ (bắt đầu từ hòn đá đầu tiên, không mất phí).

Mã giả:

```
Input: n, k, h[1..n]
Initialize dp[1] = 0, dp[2..n] = INF
For i = 2 to n:
    For j = 1 to min(k, i-1):
        dp[i] = min(dp[i], dp[i-j] + abs(h[i] - h[i-j]))
Output dp[n]
```

Độ phức tạp:

- **Thời gian:** $\mathcal{O}(n \cdot k)$, do duyệt qua n hòn đá và với mỗi hòn đá xét tối đa k bước nhảy. - **Không gian:** $\mathcal{O}(n)$, do chỉ lưu mảng `dp`.