ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BỘ MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Báo cáo

Đề tài: Giải bài toán Set Cover và TSP bằng thuật toán gần đúng

---

Môn học: Phân tích và thiết kế thuật toán

Sinh viên thực hiện:

Nguyen Minh Huy(23520634)

Do Quang Luc(23520902)

Giáo viên hướng dẫn:

Nguyen Thanh Son

Ngày 5 tháng 12 năm 2024

Trả lời câu hỏi

1. **Có phải mọi bài toán đều có thể giải quyết bằng quy hoạch động không? Tại sao?**
   Không, không phải mọi bài toán đều có thể giải quyết bằng quy hoạch động. Quy hoạch động chỉ áp dụng được cho các bài toán thỏa mãn hai tính chất cơ bản:

   - **Tính chất con tối ưu:** Bài toán lớn có thể được giải quyết bằng cách giải các bài toán con nhỏ hơn và kết hợp kết quả.
   - **Tính chất lặp lại:** Các bài toán con trùng lặp và có thể tái sử dụng kết quả thay vì tính lại.

   Nếu bài toán không thỏa mãn hai tính chất trên, thì không thể sử dụng quy hoạch động để giải quyết.

2. **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 **tối ưu hóa ba-lô (Knapsack Problem)**.

   - **Mô tả bài toán:** Cho một tập các vật phẩm với trọng lượng và giá trị, cùng một giới hạn trọng lượng của ba-lô. Tìm cách chọn các vật phẩm sao cho tổng giá trị lớn nhất mà không vượt quá giới hạn trọng lượng.
   - **Cách tiếp cận:**
     - Sử dụng mảng `dp[i][w]`, trong đó `dp[i][w]` đại diện cho giá trị tối đa có thể đạt được với `i` vật phẩm đầu tiên và giới hạn trọng lượng là `w`.
     - Truy hồi:

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

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 (Memoization):**
     - **Ưu điểm:**

    - Dễ triển khai, tự nhiên hơn vì đi từ bài toán lớn đến các bài toán con.
    - Chỉ tính toán các trạng thái cần thiết, tiết kiệm bộ nhớ và thời gian.
  - **Nhược điểm:**
    - Có thể tốn nhiều bộ nhớ do việc lưu kết quả tạm thời (memo table).
    - Gọi đệ quy sâu có thể gây lỗi tràn stack với các bài toán lớn.
- **Bottom-Up (Tabulation):**
  - **Ưu điểm:**
    - Không sử dụng đệ quy nên không lo lỗi tràn stack.
    - Tính toán tuần tự, dễ tối ưu hơn về mặt thời gian và không gian.
  - **Nhược điểm:**
    - Phải tính toán tất cả trạng thái, kể cả những trạng thái không cần thiết.
    - Cách triển khai có thể phức tạp hơn khi bài toán lớn.
- **Ưu tiên:**
  - Nếu bài toán đơn giản hoặc kích thước lớn dễ gây tràn stack, tôi sẽ ưu tiên **Bottom-Up** vì ổn định và tiết kiệm bộ nhớ hơn.
  - Nếu bài toán phức tạp, khó triển khai theo kiểu lặp, tôi sẽ ưu tiên **Top-Down** để dễ kiểm soát logic.

Giải pháp bài toán "Chú ếch"

Phân tích bài toán

Bài toán yêu cầu tìm chi phí tối thiểu để chú ếch nhảy từ hòn đá thứ nhất đến hòn đá thứ n, với điều kiện mỗi bước nhảy tốn chi phí $|h_i - h_j|$ khi nhảy từ hòn đá i đến hòn đá j $(i < j)$ và chú ếch chỉ được nhảy tối đa k hòn đá trong một lần.

Ý tưởng giải quyết

- Sử dụng kỹ thuật **quy hoạch động (Dynamic Programming)**. - Định nghĩa $dp[i]$ là chi phí tối thiểu để nhảy đến hòn đá i. - Công thức truy hồi:

$$dp[i] = \min_{1 \le j \le k} \{dp[i-j] + |h[i] - h[i-j]|\}, \quad \text{với } i - j \ge 0.$$

- Kết quả cuối cùng là $dp[n-1]$, chi phí tối thiểu để đến hòn đá thứ n.

Cài đặt bằng C++

```
#include <bits/stdc++.h>
#define double long double
#define endl "\n"
#define int long long
using namespace std;

const int INF = 1e18;

int32_t main() {
    int n, k;
    cin >> n >> k;
    vector<int> h(n);
    for (int i = 0; i < n; ++i) {
        cin >> h[i];
    }

    vector<int> dp(n, INF);
    dp[0] = 0;

    for (int i = 0; i < n; ++i) {
        for (int j = 1; j <= k && i + j < n; ++j) {
```

```
            dp[i + j] = min(dp[i + j], dp[i] + abs(h[i] - h[i + j]));
        }
    }

    cout << dp[n - 1] << endl;
    return 0;
}
```

Độ phức tạp

- **Thời gian:** $\mathcal{O}(n \times k)$, với n là số hòn đá và k là giới hạn bước nhảy. - **Không gian:** $\mathcal{O}(n)$, do sử dụng mảng dp để lưu kết quả.

Kết quả ví dụ

- **Input:**

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

- **Output:**

```
30
```

Kết luận

Giải pháp sử dụng quy hoạch động đảm bảo hiệu quả trong cả thời gian lẫn không gian, đáp ứng được ràng buộc $n \leq 10^5$ và $k \leq 100$.