VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY

PHÂN TÍCH ĐỘ PHỨC TẠP CỦA ĐỆ QUY

CS112.P11.KHTN

Sinh Viên :
Nguyễn Văn Minh - 23520945
Đồng Quốc Thắng - 23521421

Giảng viên :
Nguyễn Thanh Sơn

Ngày 7 tháng 11 năm 2024

Mục lục

Chương 1

Phân tích độ phức tạp của các bài toán sau và giải thích cách giải

1.1 Bài 1

Cho một chuỗi đầu vào gồm các số, tìm tất cả các tổ hợp của các chữ số có thể được tạo ra bằng cách sử dụng các chữ số theo cùng thứ tự.
Giải pháp: Bài toán yêu cầu tìm tất cả các dãy con liền kề của chuỗi đầu vào. Đối với chuỗi có độ dài n, ta phải xét tất cả các tập hợp con của các chữ số có thể chọn theo thứ tự liên tiếp. Điều này dẫn đến $2^n - 1$ dãy con có thể có (vì mỗi chữ số có thể được chọn hoặc không chọn).
Độ phức tạp: Độ phức tạp thời gian là $O(2^n)$ vì ta phải tạo ra từng tập hợp con có thể của các chữ số, dẫn đến độ phức tạp theo cấp số nhân.
Phương pháp giải: Sử dụng phương pháp đệ quy hoặc quay lui để tạo ra tất cả các dãy con theo thứ tự giống như trong chuỗi ban đầu. Cũng có thể triển khai theo cách lặp bằng cách xem xét từng vị trí là bao gồm hoặc không bao gồm.
Phân tích chi tiết: Độ phức tạp thời gian tăng theo hàm số mũ với n, nên cần tối ưu hóa trong trường hợp n lớn. Vấn đề này ứng dụng hiệu quả trong xử lý chuỗi hoặc tổ hợp từ điển dữ liệu. Để cải thiện, có thể giảm chiều dài hoặc kích thước dữ liệu đầu vào.

1.2 Bài 2

Cho một tập hợp các ký tự và một số nguyên dương k, in ra tất cả các chuỗi có độ dài k có thể tạo ra từ tập hợp đã cho.
Giải pháp: Đây là bài toán tạo ra tất cả các tổ hợp của các ký tự với độ dài cố định là k, cho phép lặp lại ký tự.
Độ phức tạp: Đối với tập hợp có m ký tự, có m^k chuỗi có độ dài k có thể có. Do đó, độ phức tạp thời gian là $O(m^k)$, vì mỗi lần tạo chuỗi cần thời gian hằng số cho từng ký tự chọn.
Phương pháp giải: Sử dụng đệ quy hoặc vòng lặp lồng nhau để tạo ra tất cả các tổ hợp. Đệ quy thường được ưu tiên vì đơn giản hơn, trong đó mỗi cấp đệ quy đại diện cho việc thêm một ký tự từ tập hợp vào chuỗi.
Phân tích chi tiết: Độ phức tạp của thuật toán có xu hướng tăng nhanh với k, khi m hoặc k lớn sẽ dẫn đến số lượng tổ hợp rất lớn. Trong thực tiễn, thuật toán này thường dùng khi cần kiểm tra tất cả các tổ hợp ký tự trong các bài toán về mật khẩu.

1.3 Bài 3

Viết chương trình in ra tất cả các tổ hợp các ước của một số cho trước n.
Giải pháp: Bài toán này liên quan đến việc tìm tất cả các tổ hợp của các ước (không bao gồm 1 và n) mà khi nhân với nhau sẽ cho ra n. Ví dụ, với $n = 12$, các tổ hợp bao gồm $(2, 6), (3, 4), (2, 2, 3)$, và v.v.
Độ phức tạp: Số lượng ước có thể tăng xấp xỉ theo $O(2^{\sqrt{n}})$ trong trường hợp xấu nhất vì các tổ hợp ước phụ thuộc vào phân tích ra thừa số nguyên tố của n và liên quan đến việc khám phá nhiều tổ hợp.
Phương pháp giải: Sử dụng đệ quy và quay lui. Với mỗi ước từ 2 đến $\sqrt{n}$, tìm đệ quy các tổ hợp của ước, giữ một danh sách các ước hiện tại. Đảm bảo mỗi tổ hợp nhân lại cho ra n.
Phân tích chi tiết: Vì bài toán yêu cầu tìm tất cả tổ hợp của các ước nên độ phức tạp tăng nhanh với n. Cần có cách tối ưu hóa như bỏ qua các ước không cần thiết và lưu trữ các kết quả đã tính để tránh trùng lặp.

1.4 Bài 4

Cho hai số x và n, tìm số cách mà x có thể biểu diễn dưới dạng tổng của lũy thừa n của các số tự nhiên duy nhất.
Giải pháp: Đây là biến thể của bài toán tập con tổng (subset-sum), trong đó ta cần tìm các tổ hợp của các số mà lũy thừa bậc n của chúng bằng x. Ví dụ, nếu $x = 10$ và $n = 2$, ta cần tìm các số a và b sao cho $a^2 + b^2 = 10$.
Độ phức tạp: Bài toán này có độ phức tạp theo cấp số nhân, vì ta phải duyệt qua tất cả các tập hợp của các số được nâng lên lũy thừa n có thể cộng lại bằng x. Độ phức tạp thời gian thường là $O(2^x)$ trong trường hợp xấu nhất, nhưng memoization có thể giảm số lần tính toán thừa.
Phương pháp giải: Sử dụng đệ quy với memoization. Với mỗi số tự nhiên i (bắt đầu từ 1), kiểm tra xem việc thêm i^n vào tổng hiện tại có khả thi hay không. Nếu có, tiếp tục đệ quy. Memoization giúp tránh tính toán lại các kết quả cho cùng tham số.
Phân tích chi tiết: Đối với các bài toán liên quan đến tập con tổng, đệ quy kèm ghi nhớ (memoization) là rất hữu ích để tránh lặp lại tính toán. Đặc biệt khi x và n lớn, số lượng tổ hợp sẽ rất nhiều, memoization giúp tiết kiệm bộ nhớ và tăng tốc độ.

1.5 Bài 5

Tháp Hà Nội là một câu đố toán học với ba cọc và n đĩa. Mục tiêu là di chuyển toàn bộ chồng đĩa sang cọc khác theo các quy tắc sau:

- Chỉ một đĩa có thể di chuyển tại một thời điểm.
- Mỗi lần di chuyển bao gồm việc lấy đĩa trên cùng từ một cọc và đặt nó lên cọc khác.
- Không có đĩa nào có thể đặt trên đĩa nhỏ hơn.

Giải pháp: Bài toán này có giải pháp đệ quy đã biết:

1. Di chuyển $n-1$ đĩa từ cọc nguồn sang cọc trung gian.
2. Di chuyển đĩa thứ n trực tiếp sang cọc đích.
3. Di chuyển $n-1$ đĩa từ cọc trung gian sang cọc đích.

Độ phức tạp: Độ phức tạp thời gian là $O(2^n)$, vì mỗi bước liên quan đến việc di chuyển đệ quy $n-1$ đĩa, và số lượng di chuyển tối thiểu là $2^n - 1$.
Phương pháp giải: Triển khai hàm đệ quy gọi lại chính nó để di chuyển các đĩa giữa các cọc, tuân theo quy luật đệ quy như đã mô tả ở trên.
Phân tích chi tiết: Tháp Hà Nội là một bài toán điển hình cho việc ứng dụng đệ quy. Đối với mỗi lần tăng số đĩa, số bước giải cần tăng gấp đôi, tạo ra một độ phức tạp cấp số nhân. Đây là một bài toán minh họa rõ ràng cho độ phức tạp của thuật toán đệ quy.