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

BÀI TẬP MÔN HỌC
PHÂN TÍCH THIẾT KẾ THUẬT TOÁN

Sinh viên: Đỗ Phương Duy - 23520362

Sinh viên: Nguyễn Nguyên Khang - 22520623

Ngày 8 tháng 11 năm 2024

Mục lục

1 Bài 1: 3 bài toán và kỹ thuật áp dụng:

1.1 Kỹ thuật Divide and Conquer: Bài toán Strassen's Matrix Multiplication

Đề bài: Cho 2 ma trận vuông X và Y có kích thước $n \times n$ với $n = 2^k$ (k là số nguyên ≥ 1) Khi đó mỗi ma trận X và Y và ma trận kết quả Z có thể được chia thành 4 ma trận con có kích thước nhỏ hơn có dạng như sau:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix} \quad Z = X \times Y = \begin{bmatrix} I & J \\ K & L \end{bmatrix}$$

Hãy tính toán các giá trị của các ma trận con I, J, K, L thuộc ma trận kết quả Z.

1.1.1 Cách tiếp cận truyền thống

- Chúng ta tiến hành nhân ma trận theo cách thông thường:

$$\begin{aligned} Z[0][0] &= A \times E + B \times G \\ Z[0][1] &= A \times F + B \times H \\ Z[1][0] &= C \times E + D \times G \\ Z[1][1] &= C \times F + D \times H \end{aligned}$$

- Chúng ta phải thực hiện tổng cộng 8 phép nhân ma trận nếu xem mỗi phần tử là một ma trận con.
- Phương pháp này có độ phức tạp là $O(n^3)$.

1.1.2 Cách tiếp cận sử dụng phương pháp Strassen's Matrix Multiplication

- Chúng ta sử dụng cách tiếp cận mới: tính toán ra ma trận Z mà chỉ sử dụng 7 phép nhân ma trận nếu xem mỗi phần tử là một ma trận con

$$\begin{aligned} M_1 &= (A + D)(E + H) \\ M_2 &= (C + D)E \\ M_3 &= A(F - H) \\ M_4 &= D(G - E) \\ M_5 &= (A + B)H \\ M_6 &= (C - A)(E + F) \\ M_7 &= (B - D)(G + H) \end{aligned}$$

- Sau đó các ma trận con thuộc ma trận Z có thể được tính như sau:

$$\begin{aligned} I &= M_1 + M_4 - M_5 + M_7 \\ J &= M_3 + M_5 \\ K &= M_2 + M_4 \\ L &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

- Trong trường hợp các ma trận có kích thước $n \times n$ với $n \geq 2$: Chúng ta liên tục chia các ma trận này thành 4 ma trận có kích thước nhỏ hơn đến khi đạt kích thước 2×2, tiếp tục tính các giá trị M. Các ma trận con tính toán được sau đó được tổng hợp lại để tạo thành ma trận tích cuối cùng.
- Phương pháp này có độ phức tạp là $O(n^{\log_2 7}) \approx O(n^{2.81})$

1.2 Kỹ thuật Decrease and Conquer: Bài toán Russian Peasant Multiplication

Russian Peasant Multiplication: là thuật toán dùng để nhân hai số nguyên mà chỉ sử dụng các phép nhân chia với số 2. Hoạt động dựa trên hai nguyên lý cơ bản:

- Chia lấy phần nguyên của một trong 2 số cho 2 và nhân số còn lại với 2
- Cộng vào kết quả tích cuối cùng khi số bị chia là một số lẻ
- Ví dụ: thực hiện phép tính 18 x 25:

Cột 1	Cột 2	Giữ lại (số lẻ)
18	25	Không
9	50	Có (9)
4	100	Không
2	200	Không
1	400	Có (1)

Ta bỏ các hàng có cột 1 là số chẵn, nên ta lấy tổng $400 + 50 = 450$ là kết quả cuối cùng của phép tính.

Các yếu tố của kỹ thuật Decrease and Conquer:

- Decrease: Ở mỗi bước, một trong hai số sẽ bị chia đôi khiến giá trị giảm dần đến khi đạt đến 1, giúp giảm kích thước bài toán qua từng bước.
- Conquer: Khi đã đạt đến giá trị 1, ta tính tổng các giá trị được giữ lại để tạo thành kết quả cần tìm của bài toán.
- Kỹ thuật này có độ phức tạp là $O(\log_2 n)$.

1.3 Kỹ thuật Transform and Conquer: Bài toán Gaussian Elimination

Phương pháp Gaussian Elimination: Là một phương pháp để giải hệ phương trình tuyến tính có dạng tổng quát như sau:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

Ta có thể biểu diễn hệ này bằng ma trận mở rộng $[A|b]$:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & | & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & | & b_2 \\ \vdots & \vdots & \ddots & \vdots & | & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} & | & b_m \end{bmatrix}$$

Chúng ta sẽ sử dụng các phép biến đổi sơ cấp để đưa các phần tử phía dưới đường chéo chính của ma trận về 0. Cụ thể:

- Chúng ta chọn phần tử khác 0 đầu tiên của mỗi hàng gọi là pivot
- Dùng pivot để triệt tiêu các phần tử bên dưới nó bằng các phép biến đổi sơ cấp (cộng trừ nhân chia)
- Chuyển đến hàng tiếp theo và lặp lại cho đến khi toàn bộ ma trận có dạng bậc thang (ma trận tam giác trên)
- Sau đó chúng ta giải từng ẩn của phương trình từ dưới lên trên bằng phương pháp thế ngược, bắt đầu từ hàng cuối cùng và sử dụng giá trị của các ẩn đã biết để giải các ẩn còn lại

Giả sử ta có hệ phương trình tuyến tính:

$$\begin{cases} 2x + y - z = 8 \\ -3x - y + 2z = -11 \\ -2x + y + 2z = -3 \end{cases}$$

- Viết dưới dạng ma trận mở rộng:

$$\begin{bmatrix} 2 & 1 & -1 & | & 8 \\ -3 & -1 & 2 & | & -11 \\ -2 & 1 & 2 & | & -3 \end{bmatrix}$$

Chúng ta tiến hành giải như sau

$$R_3 = R_3 + R_1$$

$$R_2 = R_2 + 1.5R_1$$

$$\begin{bmatrix} 2 & 1 & -1 & | & 8 \\ 0 & 0.5 & 0.5 & | & 1 \\ 0 & 2 & 1 & | & 5 \end{bmatrix}$$

$$R_3 = R_3 - 4R_2$$

$$\begin{bmatrix} 2 & 1 & -1 & | & 8 \\ 0 & 0.5 & 0.5 & | & 1 \\ 0 & 0 & -1 & | & 1 \end{bmatrix}$$

Sau khi tiến hành thế ngược từng ẩn lên để giải, ta sẽ thu được kết quả cuối cùng:

$$\begin{cases} x = 2 \\ y = 3 \\ z = -1 \end{cases}$$

- Gaussian Elimination là thuật toán thuộc chiến lược Transform and Conquer, ta biến đổi bài toán ban đầu, thay đổi cấu trúc của nó để biến nó thành bài toán đơn giản và dễ giải hơn
- Bài toán ban đầu được biến đổi thành cấu trúc ma trận bậc thang, có thể giải một cách dễ dàng bằng cách thế ngược
- Gaussian Elimination có độ phức tạp tính toán là $O(n^3)$, với n là số phương trình hoặc số ẩn trong hệ.

2 Bài 2:

Cho hai số nguyên x và n ($x \leq 10^{18}, n \leq 10^{18}$), tính tổng:

$$S = x^0 + x^1 + x^2 + x^3 + ... + x^n$$

Ví dụ: Với $x = 5$ và $n = 5$, ta có $S = 3906$.

2.1 Brute force

Kỹ thuật: Phương pháp cơ bản, tính trực tiếp từng lũy thừa x^i với $0 \leq i \leq n$ và cộng chúng lại.
Độ phức tạp thời gian: $O(n^2)$
Pseudocode:

```
function sum_of_powers(x, n)
    S = 0
    for i = 0 to n
        tmp = x
        if i == 0 then
            tmp = 1
        else
            for j = 0 to i
                tmp = tmp * x
        end if
        S = S + tmp
    return S
```

2.2 Lũy thừa nhanh:

Kỹ thuật: Phương pháp chia để trị (divide and conquer). Kỹ thuật này giảm số phép nhân không cần thiết để tính lũy thừa x một cách hiệu quả bằng cách dùng lũy thừa nhanh.
Độ phức tạp thời gian: $O(nlogn)$
Pseudocode:

```
function fast_power(x, i)
    if i == 0
        return 1
    t = fast_power(x, i // 2)
    t = t * t
    if i is even
        return t
```

```
    else return t * x
function sum_of_powers(x, n)
    S = 0
    for i = 0 to n
        S = S + fast_power(x, i)
    return S
```

2.3 Công thức cấp số nhân

Kỹ thuật: Phương pháp biến đổi và trị (transform and conquer). Biến đổi công thức ban đầu thành công thức thu gọn và dễ tính toán hơn.

$$S = x_0 + x_1 + x_2 + x_3 + \cdots + x_n$$

$$xS = x_1 + x_2 + x_3 + \cdots + x_{n+1}$$

$$xS - S = x_1 + x_2 + x_3 + \cdots + x_{n+1} - (x_0 + x_1 + x_2 + x_3 + \cdots + x_n)$$

$$S(x-1) = x_1 + x_2 + x_3 + \cdots + x_{n+1} - x_0 - x_1 - x_2 - x_3 - \cdots - x_n$$

$$S(x-1) = x_{n+1} - x_0$$

$$S(x-1) = x^{n+1} - 1$$

$$S = \frac{x^{n+1} - 1}{x - 1}$$

Độ phức tạp thời gian: $O(log(n))$ (tính x^{n+1} bằng lũy thừa nhanh)
Pseudocode:

```
function sum_of_power(x, n)
    if (x == 1)
        return n + 1
    else
        (return fast_power(x, n + 1) - 1) // (x - 1)
```

Tài liệu