BÀI 3. GIẢI THUẬT CHIA VÀ TRỊ

CHIA VÀ TRỊ

1. Khái niệm – Phân tích độ phức tạp
2. Các bài toán áp dụng

GIẢI THUẬT CHIA VÀ TRỊ

Thuật toán chia để trị (Divide and Conquer)

- Mục tiêu: Giảm thời gian chạy
- Ý tưởng: Áp dụng đệ quy theo 3 bước

1. **Divide (Chia).** Chia bài toán lớn thành những bài toán con có cùng kiểu với bài toán lớn.
2. **Conquer (Trị).** Giải các bài toán con. Thông thường các bài toán con chỉ khác nhau về dữ liệu vào nên ta có thể thực hiện bằng một thủ tục đệ quy.
3. **Combine (Tổng hợp).** Tổng hợp lại kết quả của các bài toán con để nhận được kết quả của bài toán lớn.

GIẢI THUẬT CHIA VÀ TRỊ

Áp dụng khi nào?

- Bài toán có thể được chia thành các bài toán con giống bài toán gốc nhưng với kích thước nhỏ hơn
- Ứng dụng trong xử lý song song ...

Một số thuật ngữ liên quan

- Chia và trị (Divide and Conquer)
- Giảm và trị (Decrease and Conquer)
- Chặt nhị phân
- Chặt tam phân

GIẢI THUẬT CHIA VÀ TRỊ

```
void solve(int n) {
  if (n == 0)
     return;

  solve(n/2);
  solve(n/2);

  for (int i = 0; i < n; i++) {
     // some constant time operations
  }
}
```

PHÂN TÍCH ĐỘ PHỨC TẠP

Gọi T(n) là độ phức tạp tính toán, n_0 là trường hợp dừng.

Theo thuật toán tổng quát

- T(n) = O(1) nếu $n = n_0$
- T(n) = aT(n/b) + D(n) + C(n) nếu $n > n_0$

Trong đó

- a: Số bài toán con
- n/b: Kích thước bài toán con
- D(n): Độ phức tạp thời gian của phần Devide
- C(n): Độ phức tạp thời gian của phần Combine

Ví dụ với b=2.

- Theo định lý Master: $T(n) = 2T(n/2) + n$ thì độ phức tạp sẽ là $O(n \log n)$

CÁC BÀI TOÁN ÁP DỤNG

1. Tìm kiếm nhị phân
2. Nhân số nguyên lớn
3. Tính tổng dãy con liên tục lớn nhất
4. Sắp xếp trộn
5. Tính lũy thừa
6. Dãy xâu Fibonacci
7. Tính số Fibonacci thứ N với N lớn

1. TÌM KIẾM NHỊ PHÂN

Bài toán: Cho số x và mảng A[] các số nguyên được sắp xếp theo thứ tự không giảm. Tìm i sao cho A[i] = x.

Phân tích bài toán:

Số x cho trước:

- Hoặc là bằng phần tử nằm ở vị trí giữa mảng A
- Hoặc là nằm ở nửa bên trái (x < phần tử ở giữa mảng A)
- Hoặc là nằm ở nửa bên phải (x > phần tử ở giữa mảng A)

1. TÌM KIẾM NHỊ PHÂN

Giải thuật

```
index  location(index  low, index  high){
 index  mid;
 If (low > high)   return 0;
 Else {
          mid =  (low + high)/2                    ;
          If  (x == A[mid])  return mid
         Else
                  If  (x < A[mid])  return  location(low,  mid - 1)
                  Else
                          Return    location (mid + 1, high);
      }
 }
```

Decrease and Conquer

1. TÌM KIẾM NHỊ PHÂN

Độ phức tạp của thuật toán:

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\frac{n}{2}) + 1 & n > 1 \end{cases}$$

Kết luận:

$$T(n) = O(\log_2 n)$$

2. BÀI TOÁN NHÂN SỐ NGUYÊN LỚN

Thuật toán cổ điển

- Nhân hai số nguyên có n chữ số có độ phức tạp $O(n^2)$.

Cải tiến 1:

- Rút gọn phép nhân hai số nguyên n chữ số còn bốn phép nhân hai số nguyên n/2

Cải tiến 2 (thuật toán *Karatsuba)*

- Giảm xuống còn 3 phép nhân

2. BÀI TOÁN NHÂN SỐ NGUYÊN LỚN

Input: $x = x_{n-1}x_{n-2}...x_1x_0$ và $y = y_{n-1}y_{n-2}\cdots y_1y_0$

Output: $z = x * y = z_{2n-1}z_{2n-2}\cdots z_1z_0$

Phân tích:

$$x = \sum_{i=0}^{n-1} x_i * 10^i = x_{n-1} * 10^{n-1} + x_{n-2} * 10^{n-2} + ... + x_1 * 10^1 + x_0 * 10^0$$

$$y = \sum_{i=0}^{n-1} y_i * 10^i = y_{n-1} * 10^{n-1} + y_{n-2} * 10^{n-2} + ... + y_1 * 10^1 + y_0 * 10^0$$

$$\Rightarrow z = x * y = \sum_{i=0}^{2n-1} z_i * 10^i = (\sum_{i=0}^{n-1} x_i * 10^i) * (\sum_{i=0}^{n-1} y_i * 10^i)$$

Cải tiến 1

Đặt các biến phụ a,b,c,d:

$$a = x_{n-1}x_{n-2}\ldots x_{n/2} \qquad b = x_{(n/2)-1}x_{(n/2)-2}\ldots x_0$$

$$c = y_{n-1}y_{n-2}\cdots y_{n/2} \qquad d = y_{(n/2)-1}y_{(n/2)-2}\cdots y_0$$

Khi đó: $x = a*10^{n/2} + b$ $y = c*10^{n/2} + d$

$$\Rightarrow z = x*y = (a*10^{n/2}+b)(c*10^{n/2}+d) = (a*c)*10^{n} + (a*d+b*c)*10^{n/2} + (b*d)$$

Cải tiến 2: Thuật toán Karatsuba

Cải tiến để còn lại 3 phép nhân :

$$\text{Đặt} \quad U = a \times c; V = b \times d; W = (a+b) \times (c+d)$$

$$\Rightarrow a \times d + b \times c = W - U - V$$

$$\Rightarrow Z = U \times 10^{n} + (W - U - V) \times 10^{n/2} + V$$

Thuật toán Karatsuba

```
Large_integer Karatsuba(x, y, n){
    If  n == 1  Return x*y
   Else  {
       a = x[n-1]. . . x[n/2]; b = x[n/2-1] . . . x[0];
       c = y[n-1]. . . y[n/2]; d = y[n/2-1] . . . y[0];
       U = Karatsuba(a, c, n/2);
       V = Karasuba(b, d, n/2);
       W = Karatsuba(a+b, c+d, n/2);
       Return U*10^n + (W-U-V)*10^(n/2) + V
   }
}
```

Thuật toán Karatsuba

Độ phức tạp thuật toán:

- *Gọi T(n) là thời gian cần thiết để thực hiện thuật toán. Khi đó:*

$$T(n) = \begin{cases} 1 & n = 1 \\ 3T(\frac{n}{2}) + cn & n > 1 \end{cases}$$

- *T(n) = O(nlog3) ≈ O($n^{1.58}$)*

3. Tìm tổng dãy con liên tục lớn nhất

Bài toán: Cho dãy số nguyên bao gồm cả số âm lẫn số dương. Nhiệm vụ của ta là tìm dãy con liên tục có tổng lớn nhất.

Ví dụ. Với dãy số A = {-2, -5, **6, -2, -3, 1, 5**, -6}

Tổng lớn nhất của dãy con liên tục ta nhận được là 7 tương ứng với dãy con {6, -2, -3, 1, 5}.

3. Tìm tổng dãy con liên tục lớn nhất

Thuật toán 1. Max-SubArray(Arr[], n): *//Độ phức tạp* O(n^2).

Bước 1 (Khởi tạo):

```
Max = Arr[0];
```

Bước 2 (lặp):

```
for (i=1; i<n; i++) {
        S = 0;
        for ( j =i+1; j<=n; j++) {
                S = S + Arr[j];
                if (Max < S )
                        Max = S;
        }
}
```

Bước 3 (Trả lại kết quả):

```
Return(Max).
```

3. Tìm tổng dãy con liên tục lớn nhất

Ý tưởng chia và trị

- Với một vị trí mốc:
  - Dãy con tổng lớn nhất có thể nằm hoàn toàn bên trái mốc
  - Dãy con tổng lớn nhất có thể nằm hoàn toàn bên phải mốc
  - Dãy con tổng lớn nhất có thể có một phần ở bên trái và một phần ở bên phải mốc.

- Thực hiện đệ quy để tính tổng bên trái và bên phải với mốc là vị trí ở giữa.

- Kết quả là max của ba giá trị: đệ quy bên trái, đệ quy bên phải, hoặc ở cả hai bên.

3. Tìm tổng dãy con liên tục lớn nhất

Thuật toán 2. Chia và trị (Arr[], n): *//Độ phức tạp* O(nlog(n).

```
int maxCrossingSum(int arr[], int l, int m, int h) {
        int sum = 0, left_sum = INT_MIN, right_sum = INT_MIN;
        for (int i = m; i >= l; i--)  {
                sum = sum + arr[i];
                if (sum > left_sum)   left_sum = sum;
        }
        sum = 0;
        for (int i = m+1; i <= h; i++){
                sum = sum + arr[i];
                if (sum > right_sum)  right_sum = sum;
        }
        return left_sum + right_sum;
}
int maxSubArraySum(int arr[], int l, int h) {
         if (l == h)    return arr[l];
         int m = (l + h)/2;
         return max(maxSubArraySum(arr, l, m),
                        maxSubArraySum(arr, m+1, h),
                                maxCrossingSum(arr, l, m, h));
}
```

4. BÀI TOÁN SẮP XẾP

Cho A[] là một mảng n phần tử. Hãy sắp xếp các phần tử của A theo thứ tự không giảm.

Giải thuật theo mô hình chia và trị (đã trình bày trong bài Sắp xếp):

1. *MergeSort*
2. *QuickSort*

Độ phức tạp chung: O(nlog(n))

5. Tính lũy thừa

Bài toán:

- Tính a^n
- Điều kiện: a, n là các số nguyên và n không âm.

Thuật toán lặp:

```
int expose(a,n) {
   int result = 1;
   for (int i = 1; i <= n; ++i)
        result *= a;
 }
```

Độ phức tạp O(n)

5. Tính lũy thừa

Cải tiến hàm *expose*

- a^n tính theo $(a^{n/2})^2$ khi n chẵn.

$$a^n = \begin{cases} 1 & , n = 0 \\ (a^2)^{\lfloor n/2 \rfloor} & \text{, n chẵn} \\ a(a^2)^{\lfloor n/2 \rfloor} & \text{n lẻ} \end{cases}$$

5. Tính lũy thừa

```
int power(int a, int n) {
   if(n == 0)  return 1;
   int x = power(a,n/2)
   if(n %2 == 0)
         return x*x;
   return a*x*x;
}
```

T(n) = O(logn).

Vấn đề:

Kiểu dữ liệu

Chia dư để tránh tràn số

Tính chất đồng dư

Với một giá trị m, các phép toán khi chia modulo cho m sẽ có tính chất sau

- **Phép cộng:** (A+B) mod m = A mod m + B mod m
- **Phép trừ:** (A-B) mod m = (A mod m – B mod m + m) mod m
- **Phép nhân:** (A*B) mod m = (A mod m * B mod m) mod m

Vì sao thường chọn cơ số m = 10^9+7

- Là một số nguyên tố
- Tích hai số vẫn chưa vượt quá 2^64

6. Bài toán Fibonacci Word

Dãy Fibonacci

- *F[1] = 1*
- *F[2] = 1*
- *F[n] = F[n-1] + F[n-2]*

Các phần tử đầu tiên của dãy

1; 1; 2; 3; 5; 8; 13; 21; 34; 55; …

Bài toán Fibonacci Word

- Cho trước hai xâu g1 và g2. Quy tắc tạo xâu tiếp theo như sau (với dấu + là nối xâu)
  - *g*(1) = *A*
  - *g*(2) = *B*
  - *g*(*n*) = *g*(*n*-2) + *g*(*n*-1)

6. Bài toán Fibonacci Word

Các xâu đầu tiên trong dãy G

A
B
AB
BAB
ABBAB
BABABBAB
ABBABBABABBAB
BABABBABABBABBABABBAB

Nhận xét: Độ dài của xâu thứ i chính là F[i]

BÀI TOÁN: Cho số tự nhiên N không quá 92 và vị trí K. Xác định ký tự thứ K trong xâu G(n)

6. Bài toán Fibonacci Word

Ý tưởng Chia và Trị

- Sinh dãy Fibonacci
- So sánh vị trí K với F[n-2]
  - Nhỏ hơn hoặc bằng: gọi đệ quy để tìm bên trái
  - Lớn hơn: gọi để quy để tìm bên phải.
- Thuật toán dừng khi gặp vị trí 1 hoặc 2 (đã biết giá trị)

7. Tính số Fibonacci thứ N

Sử dụng vòng lặp tính dãy Fibonacci

- Độ phức tạp O(N)

Làm thế nào để tính F[N] với độ phức tạp O(logN)?

Ý tưởng

- Nhận xét:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}.$$

- Tính lũy thừa ma trận theo cách tính lũy thừa O(logN)