PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN (Design and Analysis of Algorithms)

L/O/G/O

GV: HUỲNH THỊ THANH THƯƠNG
Email: hh.thanhthuong@gmail.com
thuonghtt@uit.edu.vn

CHƯƠNG 3

THIẾT KẾ THUẬT TOÁN

Algorithm Design

Nội dung

- Phương pháp chia để trị (Divide and Conquer)
- Phương pháp giảm để trị
- Phương pháp tham lam
- Phương pháp quay lui, nhánh cận, cắt tỉa
- Phương pháp quy hoạch động
- Các phương pháp khác

Chia để trị - Divide and Conquer

❖ Ý tưởng:

- Để giải 1 bài toán có kích thước n, chia bài toán đã cho thành 1 số bài toán con có kích thước nhỏ hơn.
- Giải các bài toán con rồi tổng hợp kết quả lại để được lời giải của bài toán ban đầu
  - Giải các bài toán con : lại chia kích thước nhỏ hơn nữa
  - Quá trình dẫn đến những bài toán mà lời giải hiển nhiên, dễ dàng thực hiện (bài toán cơ sở)

The Divide and Conquer Paradigm

1. DIVIDE into smaller subproblems

2. CONQUER subproblems recursively.

3. COMBINE solutions of subproblems into one for the original problem.

5/31/2022

Chia để trị - Divide and Conquer

❖ Mô hình

```
void D&C(N)  // N là kích thước dữ liệu của bài toán
{
        if( N đủ nhỏ)
                Giải bài toán; (đôi khi không làm gì cả)
        else
        {
                Chia bài toán N thành các bài toán con
        kích thước N1, N2, …, Nm

                for (i = 1; i <= m; i++)
                        D&C(Ni);

                Tổng hợp kết quả ( có hoặc không tường minh )
        }
}
```

Chia để trị - Divide and Conquer

❖ Ví dụ 1: Bài toán tìm MaxMin

Tìm giá trị Max, Min trong đoạn [l, r] của mảng A có n phần tử.

- Tìm thuật toán có độ phức tạp O(n) ?
- Giải bằng nhiều cách:
  - Cách 1: so sánh từng phần tử a[i] với min/max và cập nhật min/max
  - Cách 2: sort + trả về min = a[0], max=a[n-1]
  - Cách 3: chia để trị

Chia để trị - Divide and Conquer

❖ Ví dụ 1: Bài toán tìm MaxMin

- Chia

  - Nếu l = r → giải trực tiếp

  - Ngược lại, chia bài toán thành 2 bài toán con rời nhau

- Trị

  - Tìm kiếm Max1, Min1 trên bài toán con 1

  - Tìm kiếm Max2, Min2 trên bài toán con 2

- Tổng hợp: - Tổng hợp kết quả

Chia để trị - Divide and Conquer

❖ Ví dụ 1: Bài toán tìm MaxMin

```
void MaxMin(A, l, r, &Max, &Min)
{
            if (l == r) {Min = Max = a[l]; }
            else
            {       mid = (l+r)/2;
                    // gọi đệ quy để giải các bài toán con
                    MaxMin (A, l, mid, Max1 , Min1);
                    MaxMin (A, mid+1, r, Max2 , Min2);
                    // tổng hợp kết quả từ các bài toán con
                    if (Min1 < Min2)  Min = Min1;
                    else Min = Min2;
                    if (Max1 > Max2)  Max = Max1;
                    else Max = Max2;
            }
}
```

Chia để trị - Divide and Conquer

❖ Bài toán tìm MaxMin

Phân tích:

- Thành lập phương trình đệ quy:

$$T(n) = \begin{cases} C_1 & \text{nếu} \quad n = 1\ ||n = 2 \\ 2T\left(\frac{n}{2}\right) + C_2 & \text{nếu}\ n > 2 \end{cases}$$

- Giải tìm nghiệm: SV tự giải để suy ra Độ phức tạp

Chia để trị - Divide and Conquer

❖ Ví dụ 2: Thuật toán Merge Sort

- Chia

  - Nếu mảng A rỗng hoặc chỉ có một phần tử thì trả về chính A (đã có thứ tự).

  - Ngược lại, chia A thành 2 mảng con

- Đệ quy

  - Sắp xếp 2 mảng con

- Tổng hợp

  - Xếp xen kẽ hai mảng con đã có thứ tự

Chia để trị - Divide and Conquer

❖ Ví dụ 2: Thuật toán MergeSort

```
Merge-Sort(A,l,r)
{
        if (l < r) then
        {
                mid = (l+r)/2;
                Merge-Sort(A,l,mid);
                Merge-Sort(A,mid+1,r);
                Merge(A,l,mid,r);
        }
}
```

Chia để trị - Divide and Conquer

❖Ví dụ 2: Thuật toán Merge Sort

Phân tích:

- Chia: chỉ tốn thời gian là hằng số để tính phần tử giữa dãy, chi phí là O(1)

- Đệ quy: giải quyết đệ quy 2 dãy con, mỗi dãy có kích thước n/2, chi phí là 2T(n/2)

- Tổng hợp: giải thuật trộn n phần tử có chi phí O(n)

- Thành lập phương trình đệ quy:
- Giải tìm nghiệm: SV tự giải để suy ra Độ phức tạp

Chia để trị - Divide and Conquer

❖ Ví dụ 3: Thuật toán Quick Sort

- Chia
  - Chọn 1 phần tử bất kỳ trong mảng làm nút trục, xác định vị trí hợp lệ của nút này trong mảng (vị trí pivot).
  - Phân hoạch các phần tử còn lại sao cho từ vị trí 0 đến pivot-1 đều có giá trị nhỏ hơn hoặc bằng nút trục, từ vị trí pivot+1 đến n-1 lớn hơn nút trục.
- Đệ quy
  - Sắp xếp 2 mảng con.
- Tổng hợp
  - Không tổng hợp kết quả 1 cách tường minh (đã thực hiện trong quá trình phần hoạch)

Chia để trị - Divide and Conquer

❖Ví dụ 3: Thuật toán Quick Sort

```
void QuickSort (int a[], int left, int right )
{

    int pivot;
    if (left >= right)    return;        //điều kiện dừng

    if (left < right)                    //bước đệ qui

            partition (a, left, right, &pivot);

                    dãy con 1: a[i] <= a[pivot], i < pivot

                    dãy con 2: a[i]  >  a[pivot], i > pivot

            QuickSort (a, left, pivot - 1);

            QuickSort (a, pivot+1, right);

    }
```

Chia để trị - Divide and Conquer

❖ Ví dụ 4: Bài toán Vạch thước

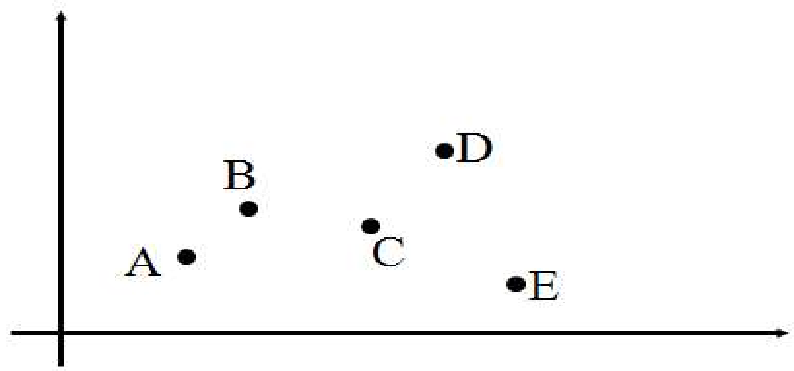
- Cho một cây thước có độ dài L và một chiều cao h nguyên cho trước.
- Tại vị trí chính giữa của cây thước, vạch một vạch có chiều cao h.
- Tại vị trí 1/4 và 3/4 của cây thước, vạch một vạch có chiều cao h-1.
- Tại vị trí 1/8, 3/8, 5/8, và 7/8 của cây thước, vạch một vạch có chiều cao h-2.
- ...
- Cho đến khi không thể vạch được nữa (chiều của vạch bằng 0)

(Đã giải trên lớp)

Chia để trị - Divide and Conquer

❖ Ví dụ 5: Bài toán sắp hạng trong không gian 2D

- Cho điểm $A(a_1, a_2)$ và $B(b_1, b_2)$. A được gọi là "trội hơn" B nếu $a_1 > b_1$ và $a_2 > b_2$.



rank(A)= 0 rank(B) = 1 rank(C) = 1

rank(D) = 3 rank(E) = 0

Chia để trị - Divide and Conquer

❖ Ví dụ 5: Bài toán tìm hạng trong không gian 2D

- Cho tập S có n điểm trong 2D, hạng của điểm X là số lượng các điểm mà X trội hơn
- Thiết kế thuật toán để sắp hạng các điểm trong tập S?

(Đã giải trên lớp)

Chia để trị - Divide and Conquer

❖ Ví dụ 5: Bài toán tìm hạng trong không gian 2D

- Ý tưởng 1: So sánh trực tiếp từng cặp điểm

  - Độ phức tạp $O(n^2)$

- Ý tưởng 2: Áp dụng pp chia để trị

  - Độ phức tạp O(?)

(Đã giải trên lớp)

Chia để trị - Divide and Conquer

❖ Ví dụ 6: Bài toán Nhân 2 số nguyên lớn

- Trong các ngôn ngữ lập trình, kiểu dữ liệu số nguyên đều có miền giá trị hạn chế
- Ứng dụng cần số nguyên lớn hơn (hàng chục hay hàng trăm chữ số)→ xây dựng cấu trúc dữ liệu số nguyên lớn + Các thao tác cộng, trừ, nhân,…
- Xây dựng giải thuật "Nhân 2 số nguyên lớn có n chữ số" sao cho hiệu quả.

(Đã giải trên lớp)

Chia để trị - Divide and Conquer

❖ Ví dụ 6: Bài toán Nhân 2 số nguyên lớn

- Cách 1: cách nhân thông thường = từng chữ số nhân với nhau rồi cộng lại→chi phí là $O(n^2)$.
- Cách 2: áp dụng kỹ thuật chia để trị

Ta chia 2 số nguyên X, Y (có n chữ số) thành các số nguyên lớn có n/2 chữ số:

$X = A\ 10^{n/2} + B$ và $Y = C\ 10^{n/2} + D$

Ví dụ: X = 1288 thì $X = 12 \times 10^{n/2} + 88$

Khi đó, $X.Y = (A\ 10^{n/2} + B). (C\ 10^{n/2} + D) = AC10^{n}+(AD+BC)10^{n/2}+BD$

Giống như trên ta lại chia tiếp tục để có bài toán cơ sở dễ dàng thực hiện

Phương pháp chia để trị

❖ Ví dụ 6: Bài toán Nhân 2 số nguyên lớn

Phân tích

X.Y = thực hiện { 4 phép nhân các số nguyên lớn n/2 chữ số (AC, AD, BC, BD),

3 phép cộng các số nguyên lớn n chữ số,

2 phép nhân với 10^n và $10^{n/2}$ để tổng hợp}

Phép cộng số nguyên lớn → O(n), Phép nhân 10^n (thêm n chữ số 0) → O(n).

$$T(n) = \begin{cases} C_1 & \text{nếu} \quad n = 1 \\ 4T\left(\frac{n}{2}\right) + C_2\, n & \text{nếu } n > 1 \end{cases}$$

Giải phương trình: T(n) = O(n^2) → Không cải thiện

Phương pháp chia để trị

❖ Ví dụ 6: Bài toán Nhân 2 số nguyên lớn

- Cách 3:

$$X.Y = AC\ 10^n + [\ (A-B)(D-C) + AC + BD]\ 10^{n/2} + BD$$

X.Y = thực hiện { 3 phép nhân các số nguyên lớn n/2 chữ số (AC, BD), và (A-B)(D-C)

6 phép cộng trừ các số nguyên lớn,

2 phép nhân với 10^n và $10^{n/2}$ để tổng hợp}

SV tự thành lập phương trình đệ quy và giải để xác định độ phức tạp → cải thiện hơn

Nội dung

- Phương pháp chia để trị

  (Divide and Conquer)
- Phương pháp giảm để trị

  (Decrease and Conquer)
- Phương pháp tham lam
- Phương pháp quay lui, nhánh cận, cắt tỉa
- Phương pháp quy hoạch động
- Các phương pháp khác

Giảm để trị - Decrease and Conquer

❖ Ý tưởng:

- Để giải 1 bài toán P(n) có kích thước n, ta giảm kích thước của nó, giải 1 bài toán con có kích thước nhỏ hơn.
- Giải bài toán con rồi tìm cách suy ra lời giải của bài toán ban đầu

Giảm để trị - Decrease and Conquer

❖ Mô hình:

- Kiểu top-down (dẫn đến 1 giải thuật đệ quy)
- Kiểu bottom-up (giải thuật lặp)

```
void D&C(n)  // n là kích thước dữ liệu của bài toán
{
        if( n đủ nhỏ)
                Giải bài toán; (đôi khi không làm gì cả)
        else
        {
                Giảm kích thước bài toán n ta được 1 bài toán con
                kích thước m (m<n)

                D&C(m);

                Suy ra kết quả từ lời giải của bài toán con
        }
}
```

Giảm để trị - Decrease and Conquer

❖ 3 trường hợp giảm kích thước:

1. Decrease by a constant: hằng số -1, -2, -3,..
2. Decrease by a factor: hệ số 1/2, 1/3,…
3. Variable size decrease

Giảm để trị - Decrease and Conquer

❖ Ví dụ 1: Tính lũy thừa x^n ($x \neq 0$)

- Công thức đệ quy: $x^n = x^{n-1}.x$
- Giảm kích thước bởi cùng 1 hằng số (không đổi) là 1

```
Pow(x,n)
{
    if(n=0) return 1;
    return Pow(x,n-1)*x;
}
```

Giảm để trị - Decrease and Conquer

❖ Ví dụ 2: Binary Search

- Giảm kích thước bởi cùng 1 hệ số là ½
- binarySearch(a,n) chuyển thành tìm kiếm trên mảng con có kích thước giảm đi 1 nửa

```
int binarySearch(int a[], int n, int l, int r, int x)
{
    if (l>r) return -1;

    int mid = (l+r)/2;

    if (x == a[mid]) return mid;
    if (x < a[mid]) return binarySearch(a,n,l,mid-1,x);
    return binarySearch(a,n,mid+1,r,x);

}
```

Giảm để trị - Decrease and Conquer

❖ Ví dụ 3: Tìm USCLN (a,b)

- **Brute-force** (dựa theo định nghĩa của USCLN): phân tích 2 số ra thừa số nguyên tố → tìm thừa số chung lớn nhất
- 2 cách giải theo kiểu Giảm để trị: dãy liên tiếp các phép toán %, -
- Giảm kích thước của biến/tham số

```
Mã giả: CHƯA XÉT SỐ ÂM
Topdown(Đệ quy)
ucln(a,b)
{
    if(b==0) return a
    return ucln(b,a%b)
}
```

```
Bottom-up(Quá trình lặp)
ucln(a,b)
{
    while(b khác 0)
    {
        r=a%b;
        a=b;
        b=r;
    }
    return a;
}
```

Giảm để trị - Decrease and Conquer

❖ Ví dụ 4: Tìm n! hoán vị của 1 tập gồm n phần tử a = (a_1, a_2, …, a_n)

- Cách 1: Quay lui - Backtracking
- Cách 2: Giảm để trị - Decrease and Conquer

```
*** Cách hình thành ý tưởng là đệ quy top-down
Bước phân tích:
Giải A4     = (7 2 4 1)
Giải A3     = (7 2 4)   -->LG={724, 742,274,247,427,472}
Giải A2     = (7 2)     -->LG={72,27}
Giải A1     = (7)       --> LG={7}
Bước thế ngược:
LG(A1)={7} + chèn 2     --> LG(A2)={72,27}
LG(A2)={72,27} + chèn 4 --> LG(A3)={724, 742,274,247,427,472}
Hỏi: chèn bằng cách nào?

***Cài đặt (mã giả): tiếp cận bottom-up (giải thuật lặp)
Đã giải trên lớp
```