CHƯƠNG 4
CHIẾN LƯỢC CHIA ĐỂ TRỊ

Nội dung chính chương 4

4.1 Cơ sở của chiến lược chia để trị (Divide and Conquer)

4.2 Sắp xếp trộn (Merge sort)

4.3. Sắp xếp nhanh (Quick sort)

4.4 Tìm kiếm nhị phân

4.1 Cơ sở của chiến lược chia để trị

- **Ý tưởng:**
  - Để giải quyết vấn đề phức tạp, ta chia vấn đề thành nhiều vấn đề con nhỏ hơn đơn giản hơn.
  - Mỗi vấn đề nhỏ được giải quyết độc lập.
  - Kết hợp kết quả của chúng lại (nếu cần) để được kết quả vấn đề lớn.

4.1 Cơ sở của chiến lược chia để trị

Các bước thực hiện:

- **Divide**: chia bài toán ban đầu thành các bài toán con.
- **Conquer**: giải quyết bài toán con một cách đệ quy. Nếu kích thước bài toán con đủ nhỏ thì có thể giải trực tiếp.
- **Combine**: kết hợp lời giải của các bài toán con thành lời giải của bài toán ban đầu (có thể không cần bước này).

4.1 Cơ sở của chiến lược chia để trị

Chia để trị (A,x) {tìm nghiệm x của bài toán A}.

- Nếu A đủ nhỏ thì GiảiBàiToán(A) ngược lại thì thực hiện:
- Phân tích A thành các bài toán con A_1, A_2, …, A_m.
- Giải các bài toán A_1, A_2, …, A_m để được các nghiệm x_1, x_2, …,x_m tương ứng.
- Kết hợp các nghiệm x_i (i=1,2…m) của các bài toán con A_i (i =1,2..m) để được nghiệm của bài toán A ban đầu.

4.1 Cơ sở của chiến lược chia để trị

Divide And Conquer algorithm :

```
DAC(a, i, j)
{
   if(small(a, i, j))
     return(Solution(a, i, j))
   else
     mid = divide(a, i, j)          // f1(n)
     b = DAC(a, i, mid)             // T(n/2)
     c = DAC(a, mid+1, j)           // T(n/2)
     d = combine(b, c)              // f2(n)
  return(d)
}
```

$$T(n) = \begin{cases} O(1) \text{ if n is small} \\ 2T(n/2) + f_1(n) + f_2(n) \end{cases}$$

4.1 Cơ sở của chiến lược chia để trị

- **Khó khăn**
  - Vấn đề lớn nhất là làm sao để chia bài toán lớn một cách hợp lý thành các bài toán nhỏ và các bài toán nhỏ phải có cùng cách giải.
  - Nếu các bài toán con có thuật toán khác nhau sẽ phức tạp.
  - Kết hợp như thế nào?

4.1 Cơ sở của chiến lược chia để trị

Ví dụ: tính a^N

Để tính a^N ta để ý công thức sau:

$$a^N = \begin{cases} 1 \text{ if } N = 0 \\ (a^{N/2})^2 \text{ if } N \% 2 = 0 \\ a*(a^{N/2})^2 \text{ if } N \% 2 = 1 \end{cases}$$

4.1 Cơ sở của chiến lược chia để trị

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

4.2 Sắp xếp trộn (merge sort)

–Ý tưởng

- Divide: Chia dãy n phần tử cần sắp xếp thành 2 dãy con, mỗi dãy có n/2 phần tử.
- Conquer: Sắp xếp các dãy con bằng cách gọi đệ quy Merge Sort. Dãy con chỉ có 1 phần tử thì mặc nhiên có thứ tự, không cần sắp xếp.
- Combine: Trộn 2 dãy con đã sắp xếp để tạo thành dãy ban đầu có thứ tự.

4.2 Sắp xếp trộn (merge sort)

– Giải thuật

```
MERGE-SORT (A, p, r)
    if p < r then
        q = (p + r) / 2
        MERGE-SORT(A, p, q)
        MERGE-SORT(A, q+1, r)
        MERGE(A, p, q, r)
```

4.2 Sắp xếp trộn (merge sort)

```
MERGE (A, p, q, r)
    n1 = q – p +1
    n2 = r – q
    for i = 0 to n1-1
        L[i] = A[p+i]
    for j = 0 to n2-1
        R[j] = A[q+j+1]
    L[n1] = infinity
    R[n2] = infinity
    i = 0
    j = 0
    for k = p to r
        if L[i] <= R[j] then
            A[k] = L[i]
            i = i + 1
        else
            A[k] = R[j]
            j = j + 1
```

4.2 Sắp xếp trộn (merge sort)

– Phân tích

- Divide: chỉ tốn thời gian là hằng số để tính phần tử giữa dãy, chi phí là O(1).
- Conquer: giải quyết đệ quy 2 dãy con, mỗi dãy có kích thước n/2, chi phí là 2T(n/2).
- Combine: giải thuật trộn n phần tử có chi phí O(n).

4.2 Sắp xếp trộn (merge sort)

– Phân tích

- Hệ thức truy hồi

$$\begin{cases} O(1) \text{ nếu } n=1 \\ 2T\left(\frac{n}{2}\right)+O(n) \text{ nếu } n>1 \end{cases}$$

- Giải hệ thức truy hồi, độ phức tạp của giải thuật là O(nlogn).

4.3 Sắp xếp nhanh (Quick sort)

Ý tưởng

- Divide: phân hoạch dãy A[p..r] thành 2 dãy con A[p..q-1] chứa các phần tử nhỏ hơn hoặc bằng A[q] và A[q+1..r] chứa các phần tử lớn hơn A[q].
- Conquer: sắp xếp 2 dãy con A[p..q-1] và A[q+1..r] bằng cách gọi đệ quy Quicksort.
- Combine: Vì 2 dãy con được sắp xếp đã đặt đúng vị trí nên không cần làm gì trong bước này.

4.3 Sắp xếp nhanh (Quick sort)

– Giải thuật

```
QUICK-SORT (A, p, r)
    if p < r then
        q = PARTITION(A,p,r)
        QUICK-SORT(A, p, q-1)
        QUICK-SORT(A, q+1, r)
```

4.3 Sắp xếp nhanh (Quick sort)

– Giải thuật

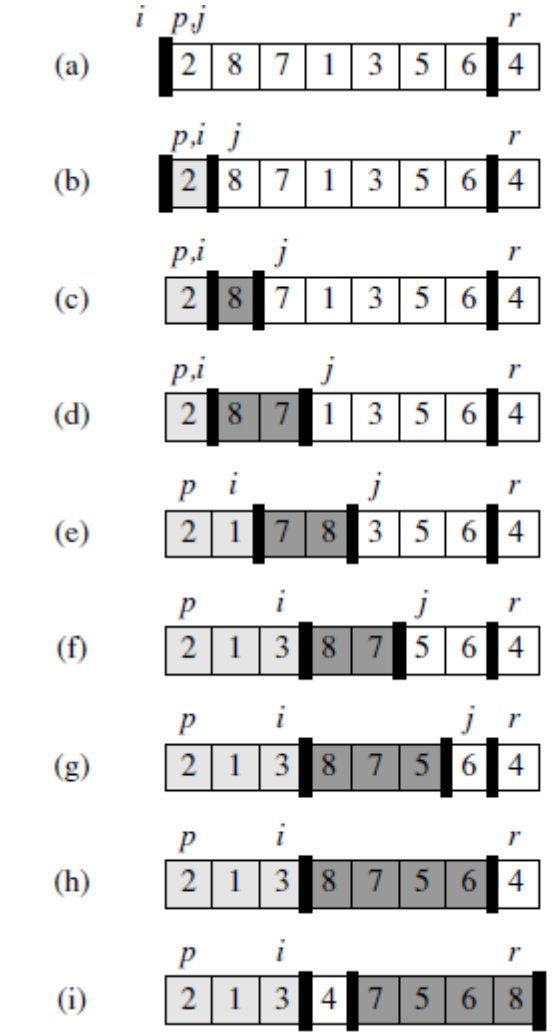
```
PARTITION (A, p, r)
    x = A[r]
    i = p – 1
    for j = p to r-1
        if A[j] <= x then
            i = i + 1
            swap(A[i], A[j])
    swap(A[i+1],A[r])
    return i+1
```

4.3 Sắp xếp nhanh (Quick sort)

– Phân tích

```
PARTITION (A, p, r)
    x = A[r]
    i = p – 1
    for j = p to r-1
        if A[j] <= x then
            i = i + 1
            swap(A[i], A[j])
    swap(A[i+1],A[r])
    return i+1
```

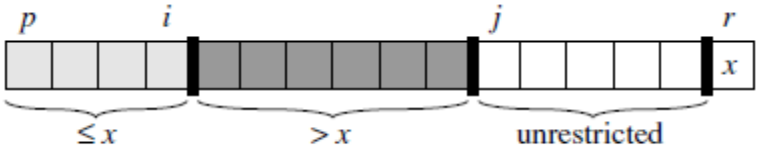
1. If $p \le k \le i$, then $A[k] \le x$.
2. If $i + 1 \le k \le j - 1$, then $A[k] > x$.
3. If $k = r$, then $A[k] = x$.



4.3 Sắp xếp nhanh (Quick sort)

– Phân tích

```
PARTITION (A, p, r)
    x = A[r]
    i = p – 1
    for j = p to r-1
        if A[j] <= x then
            i = i + 1
            swap(A[i], A[j])
    swap(A[i+1],A[r])
    return i+1
```



4.3 Sắp xếp nhanh (Quick sort)

– Phân tích:

Độ phức tạp trong trường hợp xấu nhất: $O(n^2)$

Độ phức tạp trong trường hợp tốt nhất: O(nlogn)

Độ phức tạp trong trường hợp trung bình: O(nlogn)

4.4 Tìm kiếm nhị phân

–Tìm một phần tử cho trước trong một dãy đã có thứ tự

–**Ý tưởng**

- Divide: xác định phần tử giữa dãy. Nếu phần tử cần tìm bằng phần tử này thì trả về vị trí tìm được và kết thúc. Nếu phần tử cần tìm nhỏ hơn phần tử này thì ta chỉ cần tìm trong dãy con bên trái. Nếu phần tử cần tìm lớn hơn phần tử này thì ta chỉ cần tìm trong dãy con bên phải.

4.4 Tìm kiếm nhị phân

– Ý tưởng

- Conquer: Tiếp tục tìm kiếm trong các dãy con đến khi tìm thấy hoặc đến khi hết dãy.
- Combine: Không cần trong trường hợp này.

4.4 Tìm kiếm nhị phân

```
BINARY-SEARCH (A, key)
    left = 0
    right = n-1
    while left <= right
        mid = (left + right) / 2
        if key = A[mid] then
                return mid
        else if key < A[mid]
                right = mid – 1
        else
                left = mid + 1
    return -1
```

4.4 Tìm kiếm nhị phân

– Phân tích: Độ phức tạp của giải thuật

- Trong trường hợp tốt nhất là: O(1).
- Trong trường hợp xấu nhất là: O(logn).
- Trong trường hợp trung bình là: O(logn).

Bài tập: dùng chia để trị

1. Cho đa thức $P_n(x) = a_0x^n + a_1x^{n-1} +\ldots+ a_{n-1}x + a_n$
Nhập các giá trị $a_0, a_1,\ldots,a_n$ và x. Tính $P_n(x)$
2. Các chuỗi Fibonacci được định nghĩa đệ quy như sau:

$g_1=A$, $g_2=B$, $g_n= g_{n-2}+g_{n-1}$ (ghép 2 chuỗi)
Tìm từ thứ M của chuỗi thứ N
3. Cho 3 số nguyên dương A, B, C. Hãy tính giá trị biểu thức A^B %C
4. Cho dãy $A=(a_1, a_2, .., a_n)$. Tìm diff(A)= chênh lệch nhỏ nhất giữa hai phần tử trong A

Bài tập: dùng chia để trị

5. Cho dãy A=(a_1, a_2, .., a_n). Một phần tử gọi là phần tử phổ biến nếu nó xuất hiện trong ít nhất một nửa các giá trị của A. Biết dãy A có phần tử phổ biến, hãy tìm giá trị của phần tử này.

6. Bạn được yêu cầu cắt dãy A=(a_1, a_2, .., a_n) thành K đoạn con. Trong một đoạn con, nếu có một cặp số đứng sau nhỏ hơn số đứng trước (không nhất thiết phải đứng liền trước) sẽ bị phạt 1 điểm. Hãy tìm phương án cắt có tổng số điểm phạt thấp nhất.

A=(20, 50, 30, 60, 40, 100) K=3

Cách chia: 20, 50 | 30, 60 | 40, 100 (số điểm phạt 0)

A=(20, 50, 30, 60, 40, 100, 5, 1) K=3

Cách chia: 20, 50, 30, 60 | 40, 100 | 5, 1 (số điểm phạt 2)


Thank You!