Chương 3:
Giải Thuật Chia Để Trị

Nội dung

- Ý tưởng
- Các bước thực hiện
- Một số bài toán minh hoạ

Ý tưởng

- Để giải quyết vấn đề phức tạp, ta chia vấn đề thành nhiều vấn đề con nhỏ hơn, đơn giản hơn. Giải quyết từng vấn đề nhỏ và kết hợp kết quả của chúng lại (nếu cần)
- Một số ví dụ
  - Phân nhỏ công việc cần làm
  - Phân nhỏ đồ vật
  - Sáng tạo: phân nhỏ không gian ruột xe, phân nhỏ quãng đường

Các bước thực hiện

- Divide: chia bài toán ban đầu thành một số bài toán con
- Conquer: giải quyết các bài toán con một cách đệ quy. Nếu kích thước của 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)

Một số bài toán minh hoạ

- 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.

Một số bài toán minh hoạ

- 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

Một số bài toán minh hoạ

Tìm kiếm nhị phân

- Giải thuật

```
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
```

Một số bài toán minh hoạ

- 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(lgn) và trong trường hợp trung bình là O(lgn).

Một số bài toán minh hoạ

- 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

Một số bài toán minh hoạ

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)
```

Một số bài toán minh hoạ

Merge Sort

- Giải thuật

```
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
```

Một số bài toán minh hoạ

- Merge Sort
  - Phân tích
    - Divide: chỉ tốn thời gian là hằng số để tính phần tử giữa dãy, chi phí là Θ(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 Θ(n)

Một số bài toán minh hoạ

- Merge Sort
  - Phân tích
    - Hệ thức truy hồi

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1\,, \\ 2T(n/2) + \Theta(n) & \text{if } n > 1\,. \end{cases}$$

    - Giải hệ thức truy hồi, độ phức tạp của giải thuật là Θ(nlgn)

Một số bài toán minh hoạ

- 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

Một số bài toán minh hoạ

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)
```

Một số bài toán minh hoạ

Quick Sort

- Giải thuật

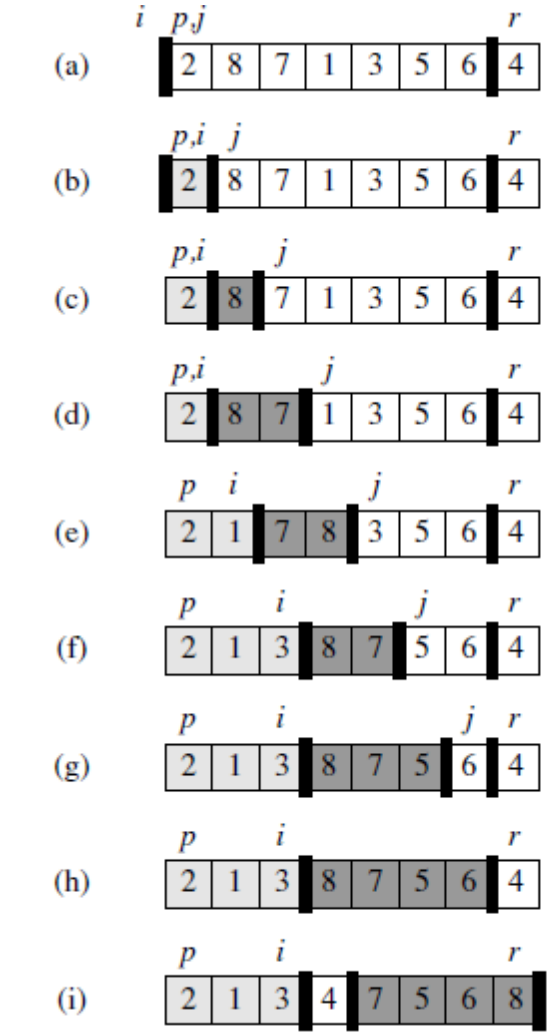
```
PARTITION (A, p, r)
    x = A[r]
    i = p – 1
    for j = p to r-1
        if A[j] <= x then
            i = i + 1
            swap(A[i], A[j])
    swap(A[i+1],A[r])
    return i+1
```

Một số bài toán minh hoạ

- 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 $A[k] = x$.

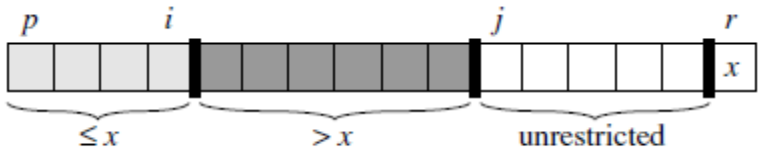


Một số bài toán minh hoạ

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
```



Một số bài toán minh hoạ

- Quick Sort
  - Phân tích: Độ phức tạp trong trường hợp xấu nhất: $\Theta(n^2)$, độ phức tạp trong trường hợp tốt nhất: O(nlgn), độ phức tạp trong trường hợp trung bình: O(nlgn)

Bài tập

- Cho một tập hợp S có n số nguyên và số nguyên x. Tìm xem có tồn tại 2 phần tử nào trong S có tổng bằng x hay không. Giải thuật đầu tiên bạn nghĩ tới là gì? Cài đặt và cho biết độ phức tạp của giải thuật này. Hãy cài đặt một giải thuật có độ phức tạp O(nlgn) giải quyết bài toán này (nếu chưa đạt)

Chiến lược giảm để trị

- Chiến lược giảm để trị tận dụng mối liên hệ giữa lời giải cho một thể hiện của một bài toán và lời giải cho một thể hiện nhỏ hơn của cùng một bài toán.
- Một số ví dụ điển hình
  - Insertion Sort
  - Giải thuật Euclide tìm ước số chung lớn nhất
  - Giải thuật DFS và BrFS

Insertion Sort

- Ý tưởng
  - Để sắp thứ tự mảng a[0..n-1], ta giả sử bài toán nhỏ hơn là sắp thứ tự mảng a[0..n-2] đã được thực hiện. Vấn đề là phải chèn phần tử a[n-1] vào mảng con đã có thứ tự a[0..n-2].

Insertion Sort

- Thực hiện: 2 cách
  - Duyệt mảng con đã có thứ tự từ trái sang phải đến khi tìm thấy phần tử đầu tiên lớn hơn hay bằng với phần tử a[n-1] và chèn phần tử a[n-1] vào bên trái phần tử này.
  - Duyệt mảng con đã có thứ tự từ phải sang trái đến khi tìm thấy phần tử đầu tiên nhỏ hơn hay bằng với phần tử a[n-1] và chèn phần tử a[n-1] vào bên phải phần tử này.

Insertion Sort

Giải thuật

```
INSERTION-SORT (A)
    for i = 1 to n-1
        temp = A[i]
        // Insert A[i] into the sorted sequence A[0..i-1]
        j = i – 1
        while j >= 0 and A[j] > temp
            A[j+1] = A[j]
            j = j – 1
        A[j+1] = temp
```

Giải thuật Euclide

- Ý tưởng
  - Giải thuật tìm ước số chung lớn nhất của 2 số theo công thức gcd(m,n) = gcd(n,m%n) là ví dụ của chiến lược giảm để trị theo hướng giảm kích thước của biến.
  - Ví dụ: gcd(12,8) = gcd(8,4) = gcd(4,0) = 4

Giải thuật Euclide

- Thực hiện: 2 cách
  - Đệ quy

```
EUCLIDE (m, n)
    if n==0 then
        return m
    else
        EUCLIDE(n,m%n)
```

  - Vòng lặp

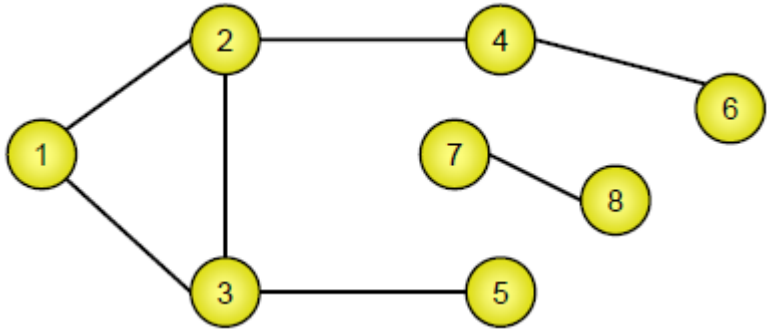
```
EUCLIDE (m, n)
    while n<>0
        r = m%n
        m = n
        n  = r
    return m
```

Giải thuật DFS và BrFS

- Tại mỗi bước của giải thuật duyệt đồ thị theo chiều sâu trước (DFS) hay duyệt theo chiều rộng trước (BrFS), giải thuật đánh dấu đỉnh đã được duyệt và tiếp tục xét các đỉnh kế cận của đỉnh đó.
- Hai giải thuật duyệt đồ thị này đã áp dụng kỹ thuật giảm bớt một trong chiến lược giảm để trị

Giải thuật DFS

- Bài toán: Duyệt tất cả các đỉnh có thể đến được từ một đỉnh xuất phát nào đó trong đồ thị



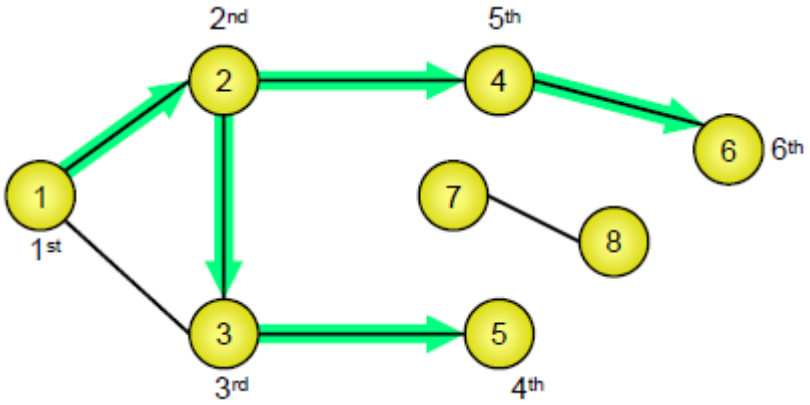
Giải thuật DFS

- Ý tưởng: Xét từ đỉnh xuất phát S, tất cả các đỉnh x kề với S sẽ đến được từ S. Với mỗi đỉnh x, đánh dấu đã xét (bớt một đỉnh) và tiếp tục lặp lại quá trình duyệt các đỉnh kề với đỉnh này…

Giải thuật DFS

Giải thuật

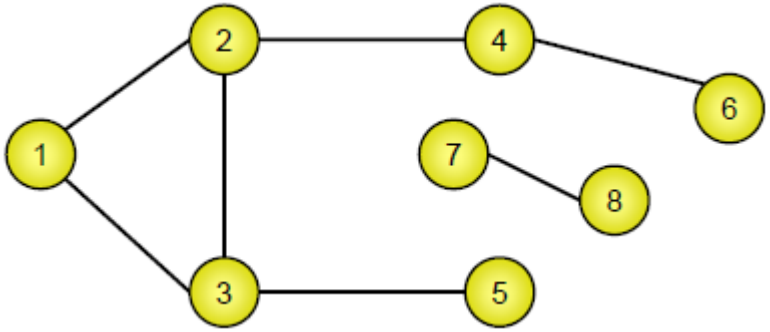
```
DFS (u)
    for v=0 to n-1
        free[u] = false
        if a[u,v] and free[v] then
            write(v)
            DFS(v)
```



Độ phức tạp: $O(n^2)$

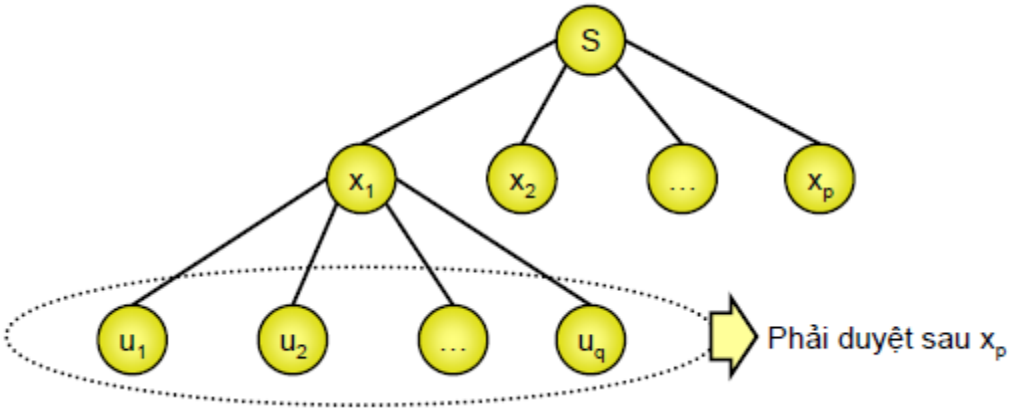
Giải thuật BrFS

- Bài toán: Duyệt tất cả các đỉnh có thể đến được từ một đỉnh xuất phát nào đó trong đồ thị



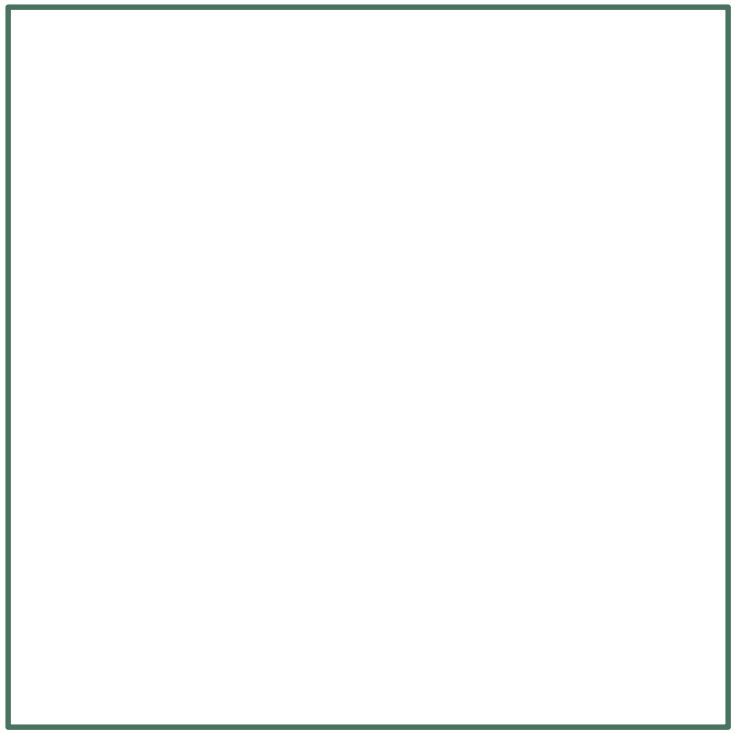
Giải thuật BrFS

- Ý tưởng: Xét từ đỉnh xuất phát S, tất cả các đỉnh x kề với S sẽ đến được từ S. Các đỉnh này sẽ được duyệt trước, sau đó là đến các đỉnh kề với các đỉnh này…



Giải thuật BrFS

- Giải thuật



- Độ phức tạp: $O(n^2)$

Bài tập

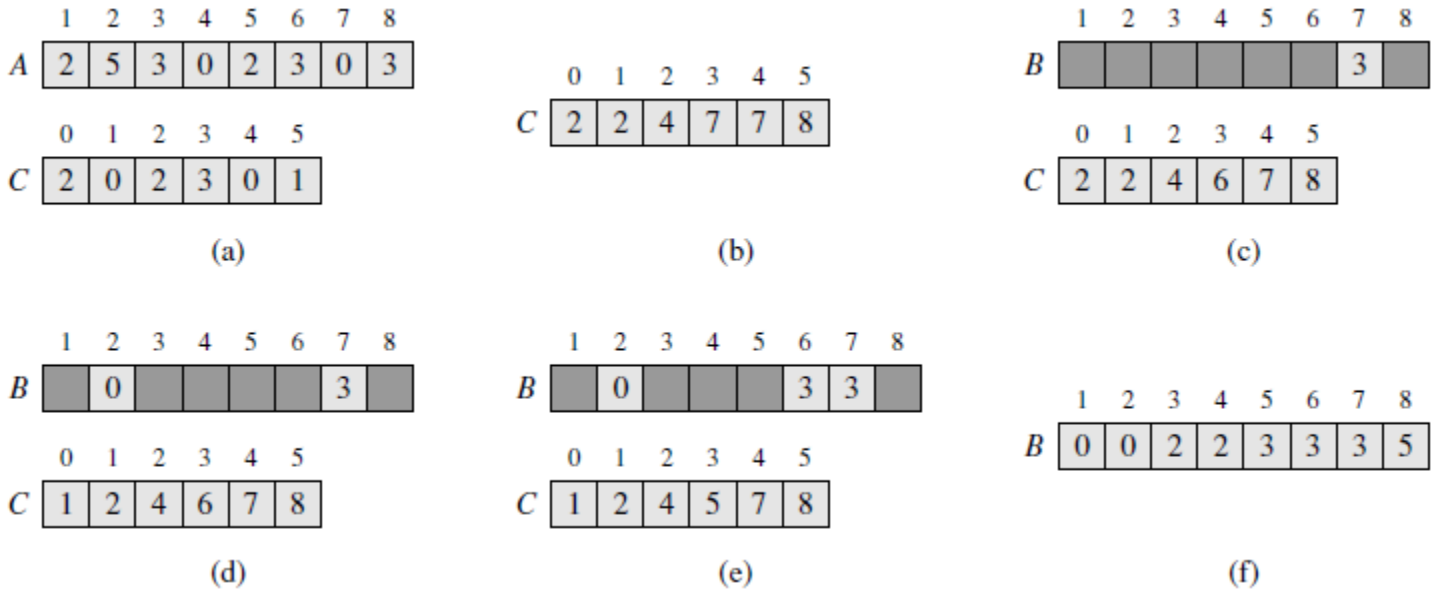
- Liệt kê tất cả các hoán vị của tập hợp S = {1, 2, …, n} bằng ý tưởng giảm để trị

Phụ lục - Counting Sort

- Ý tưởng
  - Giả sử mỗi phần tử trong mảng là một số nguyên trong khoảng 0 đến k.
  - Với mỗi phần tử x, tìm số phần tử nhỏ hơn x để xác định vị trí đúng của x trong mảng có thứ tự.
  - Có thể xảy ra trường hợp các phần tử trùng nhau.

Phụ lục - Counting Sort

- Ví dụ



Phụ lục - Counting Sort

- Giải thuật

```
COUNTING-SORT(A, B, k)
 1  for i ← 0 to k
 2       do C[i] ← 0
 3  for j ← 1 to length[A]
 4       do C[A[j]] ← C[A[j]] + 1
 5  ▷ C[i] now contains the number of elements equal to i.
 6  for i ← 1 to k
 7       do C[i] ← C[i] + C[i − 1]
 8  ▷ C[i] now contains the number of elements less than or equal to i.
 9  for j ← length[A] downto 1
10       do B[C[A[j]]] ← A[j]
11          C[A[j]] ← C[A[j]] − 1
```

Phụ lục - Counting Sort

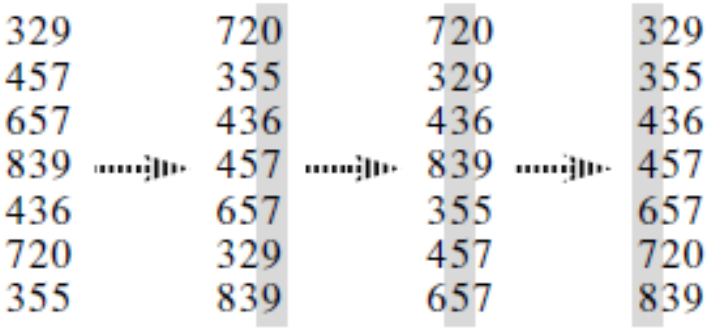
- Độ phức tạp của giải thuật: O(n+k). Nếu k = O(n) thì độ phức tạp giải thuật là O(n)

Phụ lục - Radix Sort

- Ý tưởng
  - Giả sử mỗi phần tử trong mảng A đều có d chữ số.
  - Sắp xếp mảng A lần lượt theo từng chữ số từ cuối về đầu

Phụ lục - Radix Sort

- Ví dụ



Phụ lục - Radix Sort

- Giải thuật

```
RADIX-SORT(A, d)
1  for i ← 1 to d
2        do use a stable sort to sort array A on digit i
```

- Độ phức tạp của giải thuật: O(d(n+k)). Nếu k = O(n) thì độ phức tạp giải thuật là O(n)