Nội dung

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Bài 05: Các thuật toán sắp xếp nâng cao

Ngô Thế Quyền
ngoquyenbg@hus.edu.vn

Ngày 1 tháng 10 năm 2024

Nội dung

Sắp xếp trộn

Thuật toán

- Trường hợp cơ sở (*Base case*): Nếu dãy chỉ có một phần tử được coi là dãy đã được sắp xếp
- Chia (*Divide)*: Chia bài toán ban đầu kích thước n thành hai bài toán con có $n/2$ phần tử
- Trị (*Conquer*):
  - Sắp xếp mỗi dãy con một cách đệ quy sử dụng sắp xếp trộn
  - Khi dãy chỉ còn một phần tử thì trả lại phần tử này
- Tổ hợp (*Combine*): Trộn hai dãy con được sắp để thu được dãy được sắp gồm tất cả các phần tử của cả hai dãy con

Sắp xếp trộn

```
void merge_sort (int a[], int first, int last){
    if (first < last) {
        int mid = (first + last)/2; // Chia (Divide)
        merge_sort(a, first, mid); // Tri (Conquer)
        merge_sort(a, mid + 1, last); // Tri (Conquer)
        merge(a, first, mid, last); // To hop (Combine)
    }
}
```

Sắp xếp trộn

```
void merge(int a[], int first, int mid, int last){
    int n1 = mid - first + 1, n2 = last - mid;
    int aL[] = new int[n1];
    int aR[] = new int[n2];
    for (int i = 0; i < n1; ++i)
        aL[i] = a[first + i];
    for (int j = 0; j < n2; ++j)
        aR[j] = a[mid + 1 + j];
    int i = 0, j = 0;
    int k = first;
    // xem trang tiep
```

Sắp xếp trộn

```
    while (i < n1 && j < n2) {
        if (aL[i] <= aR[j]) {
            a[k] = aL[i]; i++;
        }
        else {
            a[k] = aR[j]; j++;
        }
        k++;
    }
    while (i < n1) {
        a[k] =aL[i];
        i++; k++;
    }
    while (j < n2) {
        a[k] = aR[j];
        j++; k++;
    }
}
```

Bài tập

Hãy chứng minh thuật toán sắp xếp trộn có độ phức tạp là *nlogn*

Độ phức tạp

Thời gian tính của phép trộn - merge

- Khởi tạo hai mảng tạm thời: $\Theta(n1 + n2) = \Theta(n)$
- Đưa các phần tử đã trộn đúng thứ tự vào mảng kết quả: có n lần lặp, mỗi lần đòi hỏi thời gian hằng số, do đó thời gian cần thiết để thực hiện là $\Theta(n)$
- Tổng cộng thời gian là $\Theta(n)$

Độ phức tạp (tiếp)

Thời gian tính của sắp xếp trộn - mergeSort

- Chia: Tính mid mất $\Theta(1)$
- Trị: Giải đệ quy hai bài toán con, mỗi bài kích thước $n/2 \Rightarrow 2T(n/2)$
- Tổ hợp: $\Theta(n)$

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

Định nghĩa

Định lý thợ rút gọn: Giả sử $a \geq 1, b > 1, c > 0$ là các hằng số. Xét $T(n)$ là công thức đệ quy:

$$T(n) = aT(\tfrac{n}{b}) + cn^k$$

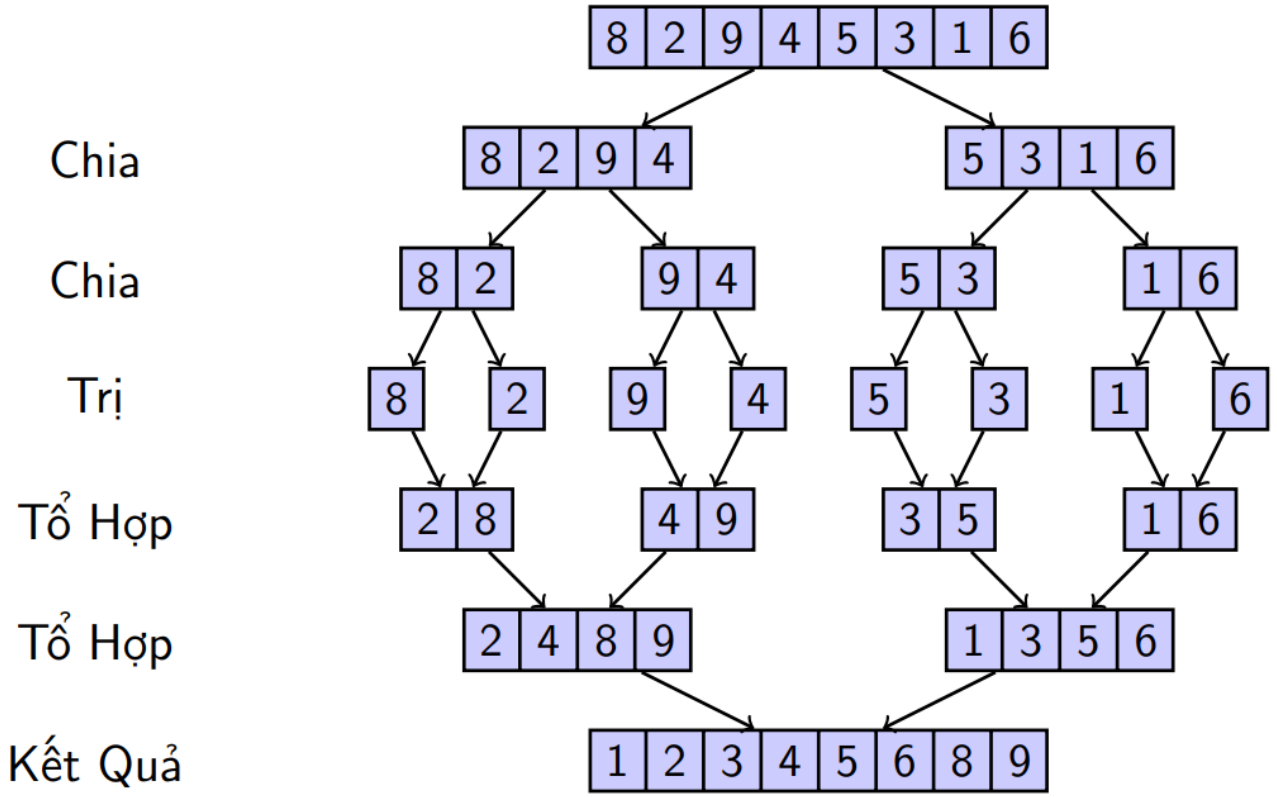
Xác định với $n \geq 0$

1. *nếu $a > b^k$ thì $T(n) = \Theta(n^{log_b^a})$*
2. *nếu $a = b^k$ thì $T(n) = \Theta(n^k log(n))$*
3. *nếu $a < b^k$ thì $T(n) = \Theta(n^k)$*

Sắp xếp trộn

Minh hoạ

Minh hoạ sắp xếp trộn của dãy $\{8, 2, 9, 4, 5, 3, 1, 6\}$



Nội dung

Sắp xếp nhanh

Thuật toán

Thuật toán sắp xếp nhanh được phát triển bởi C.A.R.Hoare vào năm 1960. Theo thống kê tính toán, đây là thuật toán sắp xếp tính nhanh nhất hiện nay. Thuật toán cũng được phát triển dựa theo phương pháp chia để trị

- Trường hợp cơ sở (Base Case): Nếu dãy có một phần tử thì nó là phần tử đã được sắp xếp.
- Chia (Divide):
  - Chọn một phần tử trong dãy làm chốt p (pivot)
  - Chia dãy thành hai dãy con: Dãy con trái L gồm các phần tử nhỏ hơn chốt, ngược lại, dãy con phải R gồm các phần tử lớn hơn chốt. Thao tác này gọi là phân hoạch - partition.
  - Trị (Conquer): Lặp lại đệ quy thuật toán với hai dãy con L và R
  - Tổ hợp (Combine): Dãy được sắp xếp là LpR

Sắp xếp nhanh

```
void quick_sort(a, left, right)
{
    if (left < right){
        p = partition (a, left, right);
        quick_sort (a, left, p-1);
        quick_sort (q, p+1, right);
    }
}
```

Sắp xếp nhanh

Một cải tiến mà D.Knuth đề nghị là nên dùng giải thuật sắp xếp khác khi số phần tử không quá lớn $n0 = 9$, ví dụ áp dụng giải thuật sắp xếp chèn

```
void quick_sort(int a[], int left, int right)
{
    if (right - left < n0)
        insertionSort (a, left, right);
    else{
        if (left < right) {
        int p = partition(a, left, right);
        quick_sort(a, left, p - 1);
        quick_sort(a, p + 1, right);
    }
}
```

Sắp xếp nhanh

Thao tác phân đoạn

- Chọn phần tử chốt p
- Chia dãy đã cho thành hai dãy con: Dãy con trái L gồm những phần tử nhỏ hơn chốt và dãy con phải R gồm các phần tử lớn hơn chốt

Thao tác phân đoạn có thể cài đặt tại chỗ, hiệu quả phụ thuộc vào việc chọn chốt p. Có mốt số cách để chọn chốt

- Chọn phần tử trái nhất
- Chọn phần tử phải nhất
- Chọn phần tử ở giữa
- Chọn phần tử trung vị (median) trong ba phần tử đầu, cuối hoặc giữa
- Chọn ngẫu nhiên một phần tử

Sắp xếp nhanh

Thao tác phân hoạch

Hàm `partition (a, left, right)` được xây dựng như sau:

- Đầu vào: mảng a[left...right]
- Đầu ra: Phân bố lại các phần tử của mảng ban đầu dựa vào phần tử chốt p và trả lại chỉ số jp sao cho:
  - $a[jp]$ chứa giá trị chốt p
  - $a[i] \leq a[jp]$ với mọi $left \leq i < jp$
  - $a[j] > a[jp]$ với mọi $jp < j \leq right$

Sắp xếp nhanh

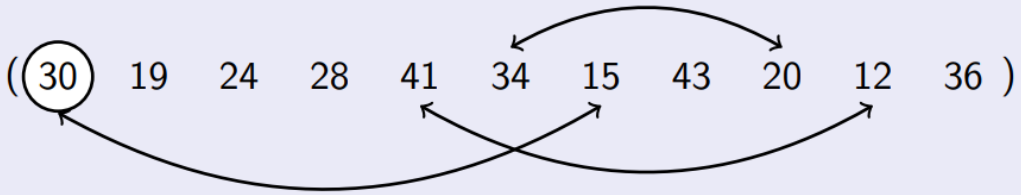
```
int partition(int a[], int left, int right) {
    int i = left, pivot = a[left], j = right;
    while (i < j) {
        while (i <= right && a[i] <= pivot)
            i++;
        while (j >= left && a[j] > pivot)
            j--;
        if (i > j)
            break;
        swap(a, i, j);
    }
    swap(a, left, j);
    return j;
}
```

Sắp xếp nhanh

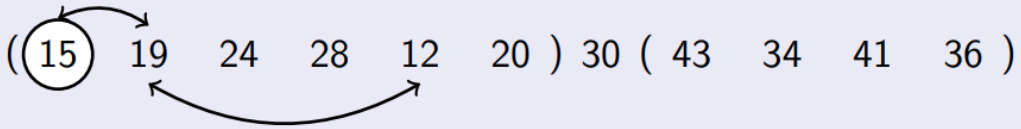
Ví dụ

Thao tác phân đoạn : Phần tử chốt là đứng đầu (tiếp)

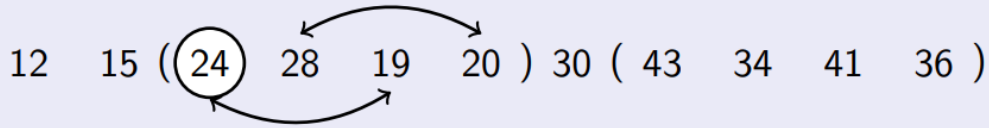
- Bước 1 :



- Bước 2 :



- Bước 3 :

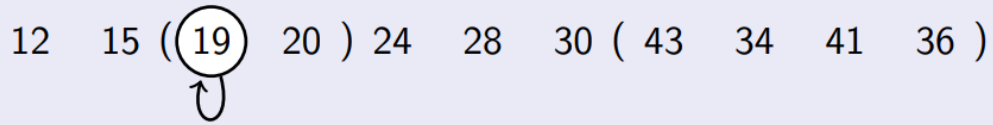


Sắp xếp nhanh

Ví dụ

Thao tác phân đoạn : Phần tử chốt là đứng đầu (tiếp)

- Bước 4 :



- Bước 5 :

- Bước 6 :

- Bước kết thúc :

12 15 19 20 24 28 30 34 36 41 43

Độ phức tạp

Độ phức tạp của sắp xếp nhanh

Thời gian tính của thuật toán sắp xếp nhanh phụ thuộc vào việc phân chia cân bằng (*balanced*) hay không cân bằng (*unbalanced*) và điều đó phụ thuộc vào việc phần tử nào được chọn làm chốt.

- Phân đoạn không cân bằng: một bài toán con có kích thước $n-1$, bài toán còn lại có kích thước 0
- Phân đoạn hoàn hảo: phân đoạn luôn thực hiện dưới dạng phân đôi, như vậy mỗi bài toán con có kích thước khoảng $n/2$
- Phân đoạn cân bằng: việc phân đoạn được thực hiện ở đâu đó quanh điểm giữa, nghĩa là một bài toán con có kích thước $n-k$, bài toán còn lại có kích thước k

Độ phức tạp

Phân đoạn không cân bằng - unbalanced partition

Công thức đệ quy cho thời gian tính trong tình huống này là:

- $T(n) = T(n-1) + T(0) + \Theta(n)$
- $T(0) = T(1) = 1$

Vậy công thức đệ quy cho thời gian tính trong tình huống này là $T(n) = \Theta(n^2)$

Độ phức tạp

Phân đoạn hoàn hảo - perfect partition

Công thức đệ quy cho thời gian tính trong tình huống này là:

- $T(n) = T(n/2) + T(n/2) + \Theta(n) = 2T(n/2) + \Theta(n)$

Vậy công thức đệ quy cho thời gian tính trong tình huống này là $T(n) = \Theta(nlogn)$

Độ phức tạp

Phân đoạn cân bằng - balanced partition

Giả sử phần tử chốt được chọn ngẫu nhiên trong số các phần tử của dãy đầu vào.

- chốt là phần tử nhỏ nhất trong dãy
- chốt là phần tử nhỏ thứ nhì trong dãy
- ....
- chốt là phần tử lớn nhất trong dãy

Điều đó cũng đúng khi chốt luôn được chọn là phần tử đầu tiên, với giả thiết dãy đầu vào hoàn toàn ngẫu nhiên

Độ phức tạp

Phân đoạn cân bằng - balanced partition

Khi đó thời gian tính trung bình sẽ có công thức
$\Sigma(\textit{thời gian phân đoạn kích thước } i) \times (\textit{xác suất có phân đoạn kích thước } i)$ Khi dãy dầu vào ngẫu nhiên, tất cả các kích thước đồng khả năng xảy ra, xác suất đều $1/n$. Do thời gian phân đoạn kích thước i là:
$T(n) = T(i) + T(n-i-1) + cn$

Độ phức tạp

Phân đoạn cân bằng - balanced partition

$$T(n) = \frac{1}{n}\Sigma_{i=0}^{n-1}(T(i) + T(n-i-1)) + cn \quad (1)$$
$$= \frac{2}{n}\Sigma_{i=0}^{n-1} T(i) + cn$$

Nhân cả hai vế với n được

$$nT(n) = 2\Sigma_{i=0}^{n-1} T(i) + cn^2 (2) \quad (2)$$

Tương tự với $(n-1)$

$$(n-1)T(n-1) = 2\Sigma_{i=0}^{n-2} T(i) + c(n-1)^2 (3) \quad (3)$$

Lấy (2) - (3)

Độ phức tạp

Phân đoạn cân bằng - balanced partition

$$\begin{aligned} nT(n) - (n-1)T(n-1) &= 2\Sigma_{i=0}^{n-1} T(i) + cn^2 - \Sigma_{i=0}^{n-2} T(i) - c(n-1)^2 \\ &= 2T(n-1) + cn \end{aligned} \tag{4}$$

Vậy $nT(n) = (n+1)T(n-1) + cn$
Chia 2 vế cho $n \times (n+1)$ được

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{c}{n+1}(4) \tag{5}$$

Độ phức tạp

Phân đoạn cân bằng - balanced partition

Nếu đặt $n = n - 1$ để tính $T(n-1)$ theo công thức $T(n)$

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{c}{n+1} \tag{6}$$
$$\frac{T(n-1)}{n} = \frac{T(n-2)}{n-1} + \frac{c}{n}$$

Tương tự, ta có:

$$\frac{T(n-2)}{n-1} = \frac{T(n-3)}{n-2} + \frac{c}{n-1} \tag{7}$$
$$\frac{T(n-3)}{n-2} = \frac{T(n-4)}{n-3} + \frac{c}{n-2}$$
$$\dots$$

Độ phức tạp

Phân đoạn cân bằng - balanced partition

Thay vào (4) ta được

$$
\begin{aligned}
\frac{T(n)}{n+1} &= \frac{T(n-1)}{n} + \frac{c}{n+1} \\
&= \frac{T(n-2)}{n-1} + \frac{c}{n} + \frac{c}{n+1} \\
&= \frac{T(n-4)}{n-3} + c \times (\frac{1}{n-2} + \frac{1}{n-1} + \frac{1}{n} + \frac{1}{n+1}) \\
&= ... \\
&= \frac{T(n-(n-1))}{n-(n-2)} + c \times (\frac{1}{3} + \frac{1}{4} + ... + \frac{1}{n} + \frac{1}{n+1}) \\
&= \frac{T(1)}{2} + c\Sigma_{i=3}^{n} \frac{1}{i}
\end{aligned}
\tag{8}
$$

Độ phức tạp

Phân đoạn cân bằng - balanced partition

Tổng chuỗi $\sum_{i=1}^{n} \frac{1}{i} \approx ln(n) + \gamma$
với γ là hằng số Euler–Mascheroni[a] $\gamma \approx 0.577$

$$\begin{aligned} \frac{T(n)}{n+1} &= \frac{T(n-1)}{n} + \frac{c}{n+1} \\ &= \frac{T(1)}{2} + c\Sigma_{i=3}^{n} \frac{1}{i} \\ &= O(1) + c\Sigma_{i=3}^{n} \frac{1}{i} \\ &= O(1) + O(clogn) \end{aligned} \tag{9}$$

Vậy $T(n) = O(nlogn)$

[a] https://en.wikipedia.org/wiki/Euler%27s_constant