CHƯƠNG 3. CÁC THUẬT TOÁN TÌM KIẾM & SẮP XẾP

SẮP XẾP NHANH (QUICKSORT)

ThS. Nguyễn Chí Hiếu

2021

NỘI DUNG

1. Giới thiệu kỹ thuật Chia để trị
2. Giải thuật sắp xếp nhanh (Quick Sort)

NỘI DUNG

Giới thiệu kỹ thuật Chia để trị

Các bước thực hiện

- Chia bài toán: chia bài toán thành những bài toán con nhỏ hơn.
- Trị bài toán nhỏ: giải những bài toán con này.
- Tổng hợp: kết hợp lời giải của những bài toán con thành lời giải cho bài toán ban đầu.

Một số bài toán áp dụng kỹ thuật chia để trị: tìm kiếm nhị phân *(BinarySearch)*, sắp xếp nhanh *(QuickSort)*, sắp xếp trộn *(MergeSort)*, $\cdots$

NỘI DUNG

Thuật toán sắp xếp nhanh *(Quick Sort)*

Ý tưởng

Sắp xếp nhanh *(Quick sort)* dựa vào ý tưởng chọn một phần tử chốt *(pivot)* $x = a_i$ trong dãy a để phân hoạch/chia dãy thành 2 dãy con

- Dãy bên trái: các phần tử nhỏ hơn x

$$a_0, a_1, \cdots, a_{i-1}$$

- Dãy bên phải: các phần tử lớn hơn x

$$a_{i+1}, a_{i+2}, \cdots, a_{n-1}$$

Tiếp tục thực hiện giải thuật đối với hai dãy con cho đến khi dãy con chỉ còn một phần tử *(xem như có thứ tự)*.

Thuật toán sắp xếp nhanh *(Quick Sort)*

Phương pháp đệ quy

Thuật toán 1: QuickSort(a[], left, right)
- Đầu vào: mảng a, vị trí left và right của mảng con đang xét.
- Đầu ra: mảng a có thứ tự tăng dần.

```
1    x ← a[(left + right) / 2]
2    i ← left
3    j ← right
4
5    Partition(a, i, j, x)
6    if left < j
7      QuickSort(a, left, j)
8    if i < right
9      QuickSort(a, i, right)
```


Thuật toán sắp xếp nhanh *(Quick Sort)*

Giải thuật phân hoạch *(Partition)*

- Nếu dãy số còn phần tử chưa xét
  - Mỗi bước thứ i tìm phần tử lớn hơn x ở dãy con bên trái *(thực hiện từ đầu đến cuối dãy con)*.
  - Mỗi bước thứ j tìm phần tử nhỏ hơn x ở dãy con bên phải *(thực hiện từ cuối về đầu dãy con)*.
  - Hoán vị hai phần tử này.
- Sau khi phân hoạch, dãy a gồm các dãy con sau:
  - Dãy con 1: $a_{left}, \ldots, a_i < x$.
  - Dãy con 2: $a_{i+1}, \ldots, a_{j-1} = x$.
  - Dãy con 3: $a_j, \ldots, a_{right} > x$.

Thuật toán sắp xếp nhanh *(Quick Sort)*

Thuật toán 2: Partition(a[], i, j, x)

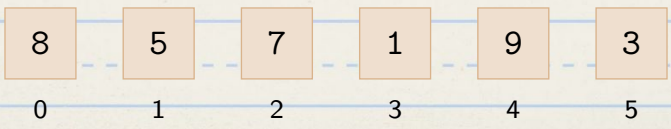
- Đầu vào: mảng a gồm n phần tử.
- Đầu ra: mảng a sau khi phân hoạch.

```
1   do
2     while a[i] < x
3       i ← i + 1
4     while a[j] > x
5       j ← j - 1
6     if i ≤ j
7       Swap(a[i], a[j])
8       i ← i + 1
9       j ← j - 1
10  while i < j
```

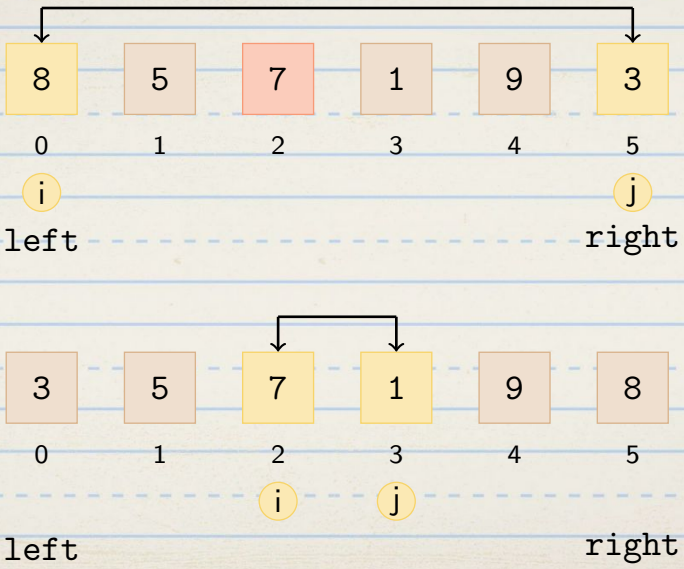

Thuật toán sắp xếp nhanh *(Quick Sort)*

Ví dụ 1

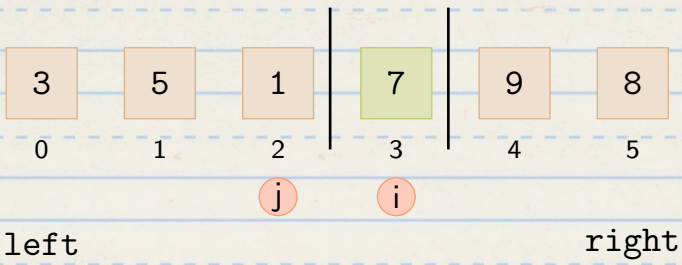
Cho dãy số a gồm 7 phần tử: $8, 5, 7, 1, 9, 10$. Áp dụng giải thuật sắp xếp nhanh (Quick Sort) sắp dãy a theo thứ tự tăng dần.



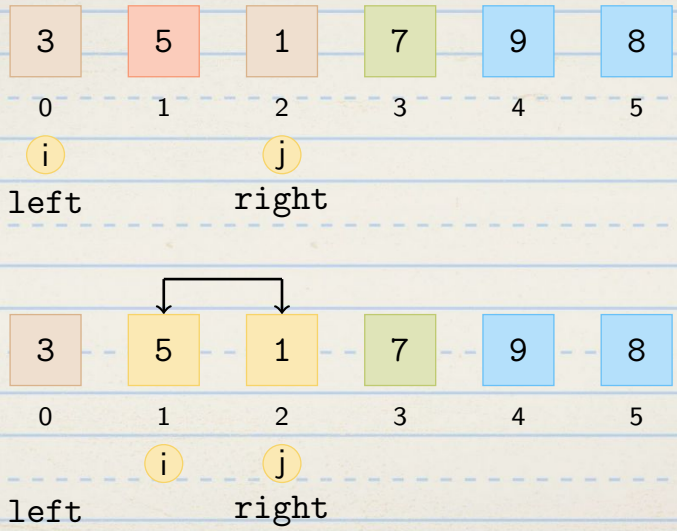
Thuật toán sắp xếp nhanh *(Quick Sort)*



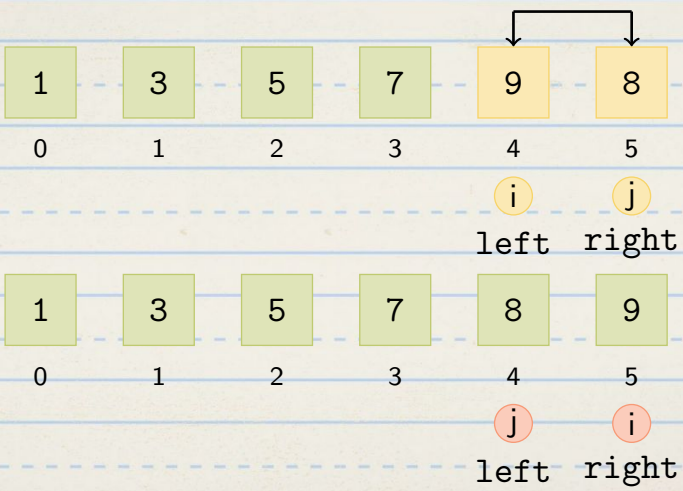
Thuật toán sắp xếp nhanh *(Quick Sort)*



Thuật toán sắp xếp nhanh *(Quick Sort)*



Thuật toán sắp xếp nhanh *(Quick Sort)*



Giải thuật sắp xếp nhanh *(Quick sort)*

Chọn phần tử chốt

- Chọn phần tử trung vị của 3 phần tử trái, giữa, phải *(median of three)*
  - So sánh 3 phần tử: trái, giữa, phải của dãy.
  - Hoán vị các phần tử sao cho:
    - a_{left} = smallest
    - a_{middle} = median of three
    - a_{right} = largest
  - Chọn $median = a_{middle}$ là phần tử chốt.

Giải thuật sắp xếp nhanh *(Quick sort)*

Chọn phần tử trung vị của 3 phần tử

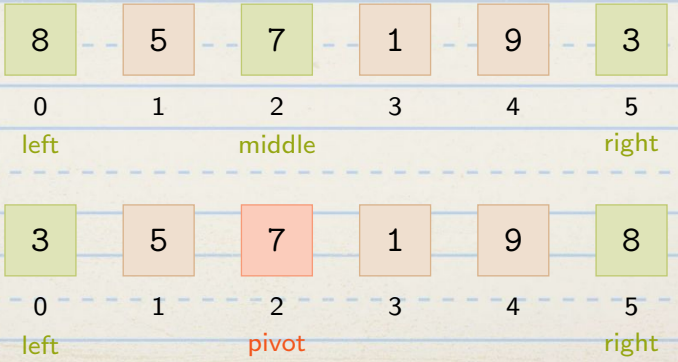
Thuật toán 3: `Median3(a[], left, right)`
- Đầu vào: mảng a gồm n phần tử.
- Đầu ra: trả về phần tử trung vị của mảng a.

```
1   middle = (left + right) / 2
2
3   if a[left] > a[middle]
4     Swap(a[left], a[middle])
5   if a[left] > a[right]
6     Swap(a[left], a[right])
7   if a[middle] > a[right]
8     Swap(a[middle], a[right])
9
10  return a[middle]
```

Giải thuật sắp xếp nhanh *(Quick sort)*

Ví dụ 2

Cho dãy số a gồm 7 phần tử: $8, 5, 7, 1, 9, 10$. Chọn phần tử pivot là trung vị của 3 phần tử: trái, giữa và phải.



Đánh giá giải thuật

Trường hợp tốt nhất

- Xảy ra khi mỗi lần phân hoạch chia dãy thành 2 phần bằng nhau.

$$T(n) = \begin{cases} 0 & , n = 1 \\ 2T\left(\frac{n}{2}\right) + (n-1) & , n > 1 \end{cases}$$

trong đó,

- $2T\left(\frac{n}{2}\right)$: thời gian sắp thứ tự *2 dãy con*.
- $n-1$: số phép so sánh giữa x và $n-1$ phần tử khác.

Độ phức tạp thời gian

$$T(n) = O(n) + O(n \log n) = O(n \log n)$$

Đánh giá giải thuật

Trường hợp xấu nhất

- Xảy ra khi dãy đã có thứ tự. Mỗi lần phân hoạch chia dãy đang xét thành 2 dãy con
  - Một dãy gồm 1 phần tử.
  - Một dãy gồm $n-1$ phần tử.

$$T(n) = \begin{cases} 0 & ,n = 1 \\ T(n-1) + (n-1) & ,n > 1 \end{cases}$$

trong đó,

- $T(n-1)$: thời gian sắp thứ tự *1 dãy con*.
- $n-1$: số phép so sánh giữa x và $n-1$ phần tử khác.

Độ phức tạp thời gian

$$T(n) = O\left(n^2\right).$$

Đánh giá giải thuật

Trường hợp trung bình

- Xảy ra khi dãy đã có thứ tự. Mỗi lần phân hoạch chia dãy đang xét thành 2 dãy con khác rỗng.

$$T(n) = \begin{cases} 0 & ,n \leq 1 \\ \frac{1}{n}\sum_{i=0}^{n-1}\left[T(i) + T(n-1-i)\right] + (n-1) & ,n > 1 \end{cases}$$

trong đó,

- $T(i)$: thời gian sắp thứ tự dãy con thứ nhất.
- $T(n-1-i)$: thời gian sắp thứ tự dãy con thứ hai.

- Do đó,

$$T(n) = O(n \log n).$$

Bài tập

1. Áp dụng giải thuật QuickSort sắp xếp dãy sau theo thứ tự tăng dần.

$$6, 5, 4, 3, 2, 1$$

2. Giả sử, một dãy gồm n phần tử có giá trị bằng nhau, khi đó độ phức tạp của thuật toán QuickSort như thế nào?
   - Chọn phần tử chốt tại vị trí đầu tiên
   - Chọn phần tử chốt tại vị trí giữa như giải thuật đã học.
3. Áp dụng giải thuật sắp xếp nhanh xây dựng giải thuật chọn một phần tử nhỏ thứ k trong dãy không thứ tự.

Tài liệu tham khảo


Donald E. Knuth.
The Art of Computer Programming, Volume 3.
Addison-Wesley, 1998.


Dương Anh Đức, Trần Hạnh Nhi.
Nhập môn Cấu trúc dữ liệu và Thuật toán.
Đại học Khoa học tự nhiên TP Hồ Chí Minh, 2003.


Niklaus Wirth.
Algorithms + Data Structures = Programs.
Prentice-Hall, 1976.


Robert Sedgewick.
Algorithms in C.
Addison-Wesley, 1990.