

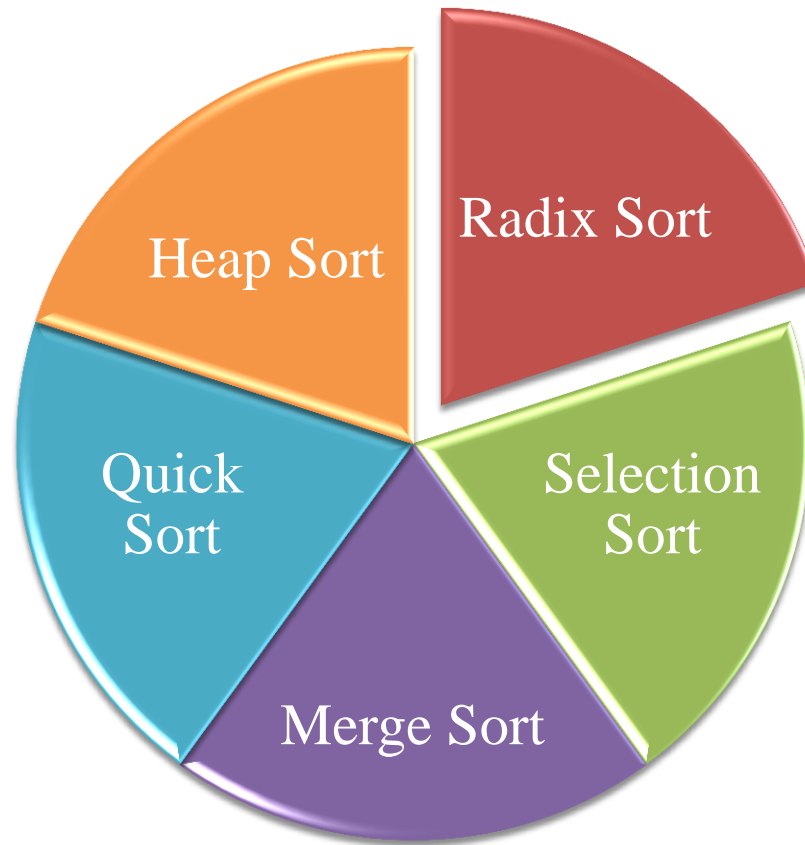
Cấu trúc dữ liệu và giải thuật

CÁC THUẬT TOÁN SẮP XẾP

Giảng viên:
Văn Chí Nam – Nguyễn Thị Hồng Nhung – Đặng Nguyễn Đức Tiến

Nội dung

2



Giới thiệu

Bài toán sắp xếp

Các thuật toán sắp xếp

Giới thiệu

4

- ◉ Bài toán sắp xếp: Sắp xếp là quá trình xử lý một danh sách các phần tử để đặt chúng theo một thứ tự thỏa yêu cầu cho trước
- ◉ Ví dụ: danh sách trước khi sắp xếp:
 $\{1, 25, 6, 5, 2, 37, 40\}$
Danh sách sau khi sắp xếp:
 $\{1, 2, 5, 6, 25, 37, 40\}$
- ◉ Thông thường, sắp xếp giúp cho việc tìm kiếm được nhanh hơn.

Giới thiệu

5

◉ Các phương pháp sắp xếp thông dụng:

- ▣ Bubble Sort
- ▣ Selection Sort
- ▣ Insertion Sort
- ▣ Quick Sort
- ▣ Merge Sort
- ▣ Heap Sort
- ▣ Radix Sort

💡 Cần tìm hiểu các phương pháp sắp xếp và lựa chọn phương pháp phù hợp khi sử dụng.

Sắp xếp chọn

Selection Sort

- ◉ Mô phỏng cách sắp xếp tự nhiên nhất trong thực tế
 - ▣ Chọn phần tử nhỏ nhất và đưa về vị trí đúng là đầu dãy hiện hành.
 - ▣ Sau đó xem dãy hiện hành chỉ còn $n-1$ phần tử.
 - ▣ Lặp lại cho đến khi dãy hiện hành chỉ còn 1 phần tử.

Thuật toán

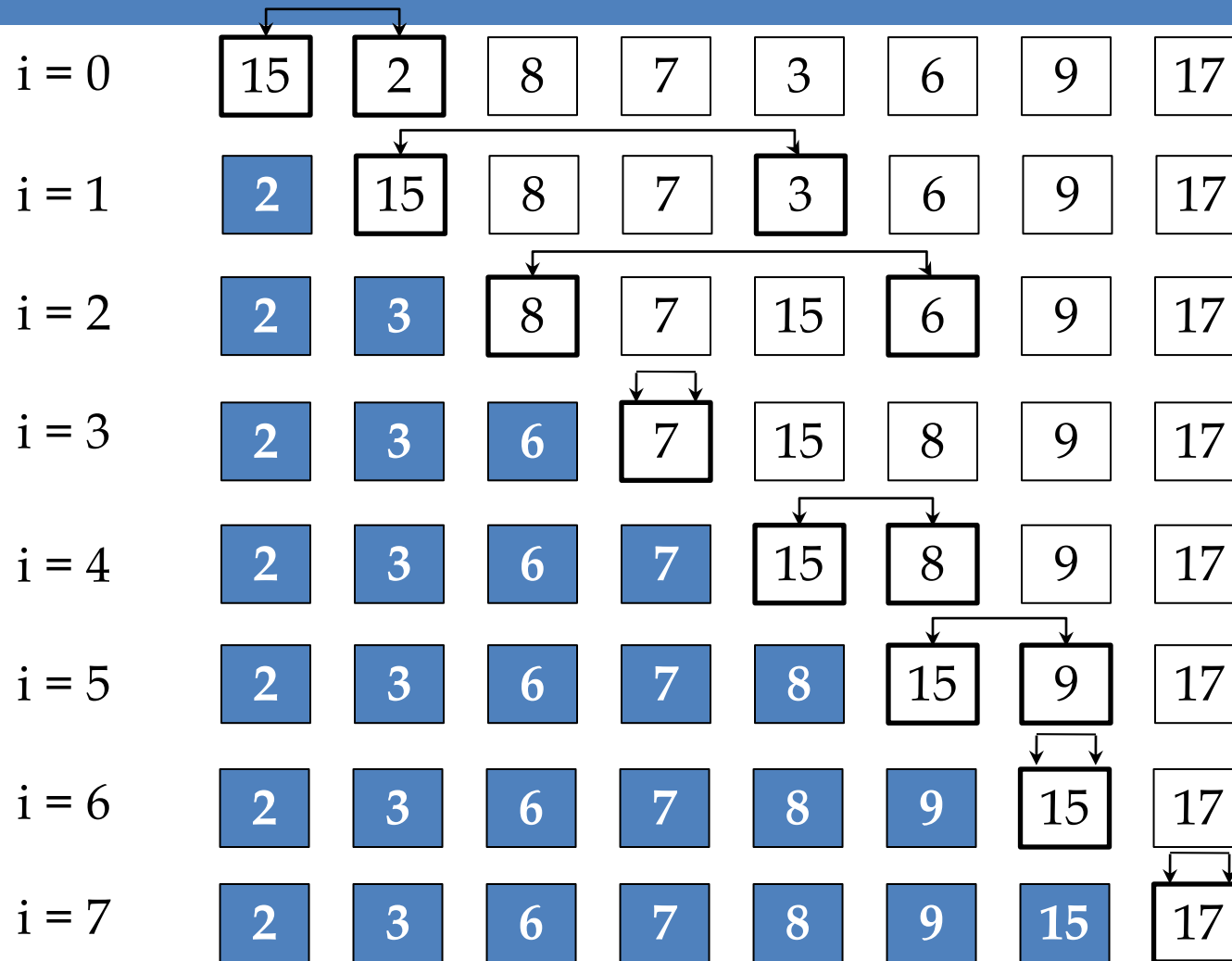
8

Các bước của thuật toán:

- ⊙ **Bước 1.** Khởi gán $i = 0$.
- ⊙ **Bước 2.** Bước lặp:
 - ▣ 2.1. *Tìm $a[\min]$ nhỏ nhất trong dãy từ $a[i]$ đến $a[n-1]$*
 - ▣ 2.2. *Hoán vị $a[\min]$ và $a[i]$*
- ⊙ **Bước 3.** So sánh i và n :
 - ▣ Nếu $i < n$ thì *tăng i thêm 1* và lặp lại bước 2.
 - ▣ Ngược lại: Dừng thuật toán.

Ví dụ

9



Đánh giá

10

◉ Đánh giá giải thuật:

▣ Số phép so sánh:

- Tại lượt i bao giờ cũng cần $(n-i-1)$ số lần so sánh
- Không phụ thuộc vào tình trạng dãy số ban đầu

Số phép so sánh =
$$\sum_{i=0}^{n-1} (n-i-1) = \frac{n(n-1)}{2}$$

Đánh giá

11

◉ Số phép gán:

▣ Tốt nhất: $\sum_{i=0}^{n-1} 4 = 4n$

▣ Xấu nhất:

$$\sum_{i=0}^{n-1} (4 + n - i - 1) = \frac{n(n+7)}{2}$$

Sắp xếp vun đống

Heap Sort

Ý tưởng

13

- ⊙ Ý tưởng: khi tìm phần tử nhỏ nhất ở bước i , phương pháp Selection sort không tận dụng được các thông tin đã có nhờ vào các phép so sánh ở bước $i-1 \rightarrow$ cần khắc phục nhược điểm này.
- ⊙ J. Williams đã đề xuất phương pháp sắp xếp Heapsort.

Heap

14

◉ Định nghĩa Heap:

- ▣ Giả sử xét trường hợp sắp xếp tăng dần, Heap được định nghĩa là một dãy các phần tử a_1, a_{1+1}, \dots, a_r thỏa: với mọi i thuộc $[1, r]$ (chỉ số bắt đầu từ 0)

$$a_i \geq a_{2i+1}$$

$$a_i \geq a_{2i+2} \{ (a_i, a_{2i+1}), (a_i, a_{2i+2}) \text{ là các cặp phần tử liên đới} \}$$

Các tính chất của Heap

15

- ⊙ Nếu a_l, a_{l+1}, \dots, a_r là một heap thì phần tử a_l (đầu heap) luôn là phần tử lớn nhất.
- ⊙ Mọi dãy a_i, a_{i+1}, \dots, a_r với $2i + 1 > r$ là heap.

Thuật toán

16

- ◉ Giai đoạn 1: Hiệu chỉnh dãy ban đầu thành heap (bắt đầu từ phần tử giữa của dãy)
- ◉ Giai đoạn 2: sắp xếp dựa trên heap.
 - ▣ Bước 1: đưa phần tử lớn nhất về vị trí đúng ở cuối dãy
 - ▣ Bước 2:
 - Loại bỏ phần tử lớn nhất ra khỏi heap: $r = r - 1$
 - Hiệu chỉnh lại phần còn lại của dãy.
 - ▣ Bước 3: So sánh r và l :
 - Nếu $r > l$ thì lặp lại bước 2.
 - Ngược lại, dừng thuật toán.

Heap Sort

17

⊙ Mã giả :

```
HeapSort (a: Array, n: int)
{
    TaoHeap (a, n-1) ;
    r = n-1;
    while (r > 0)
    {
        HoanVi (a[0], a[r]) ;
        r = r - 1;
        HieuChinh (a, 0, r) ;
    }
}
```

Heap Sort

18

⊙ Mã giả:

```
TaoHeap (a: Array, r: int)
{
    int l = r/2;
    while(l > 0)
    {
        HieuChinh(a, l, r);
        l = l - 1;
    }
}
```

Heap Sort

19

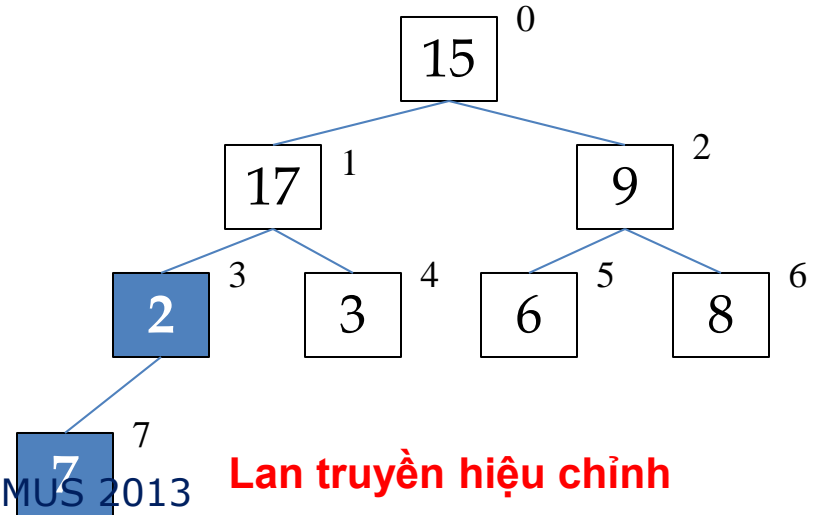
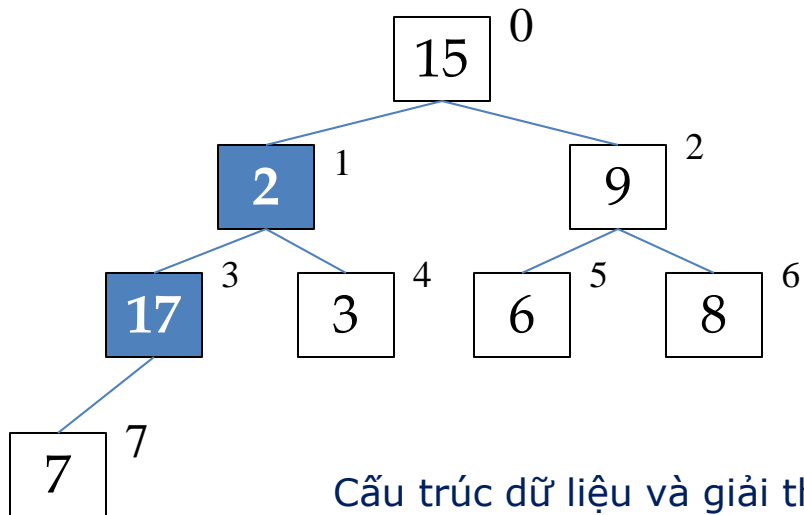
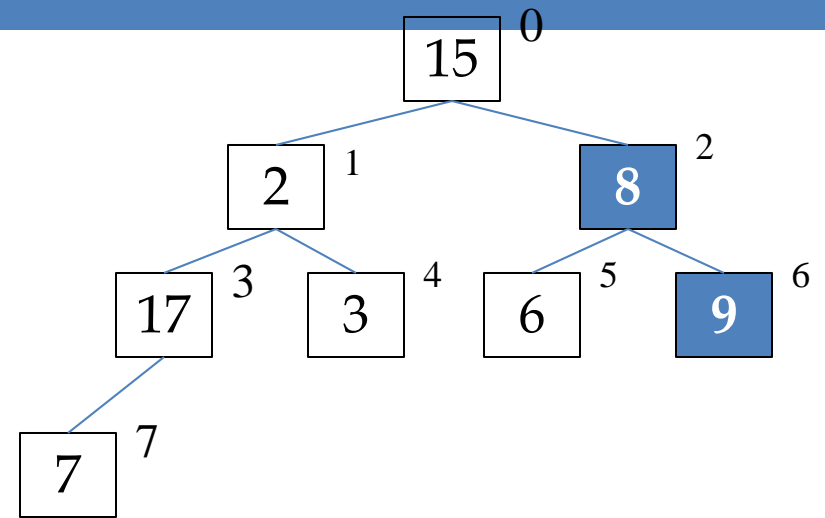
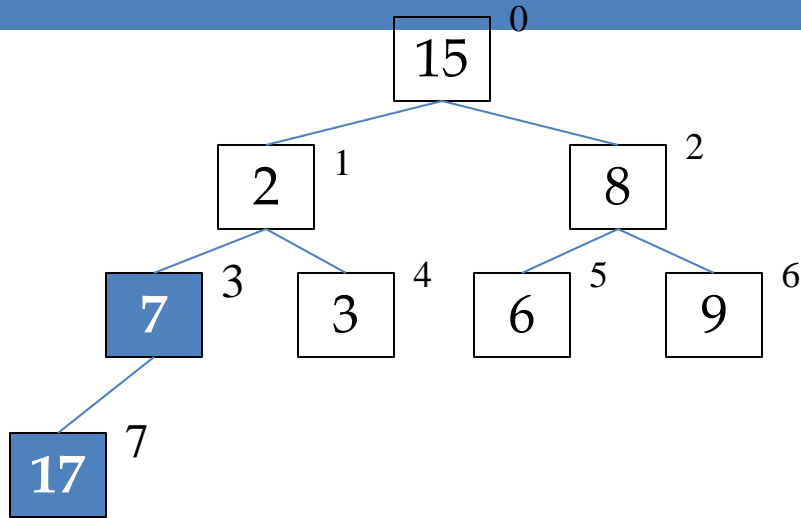
◉ Mã giả:

HieuChinh(a: Array, l: int, r: int)

```
{  
    i = l; j = 2*i+1; x = a[i];  
    while(j <= r)  
    {  
        if(có đủ 2 phần tử liên đới)  
            //xác định phần tử liên đới lớn nhất  
            if(a[j] < x) //thỏa quan hệ liên đới  
                //dùng  
            else  
                //hiệu chỉnh  
                //xét khả năng hiệu chỉnh lan truyền  
    }  
}
```

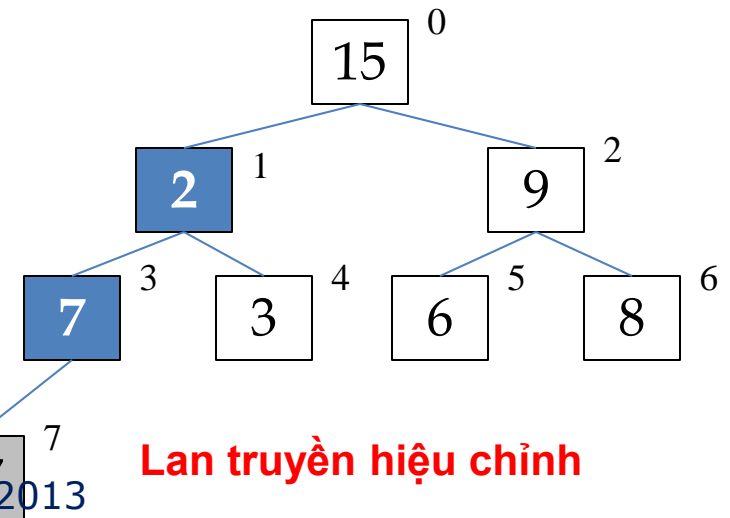
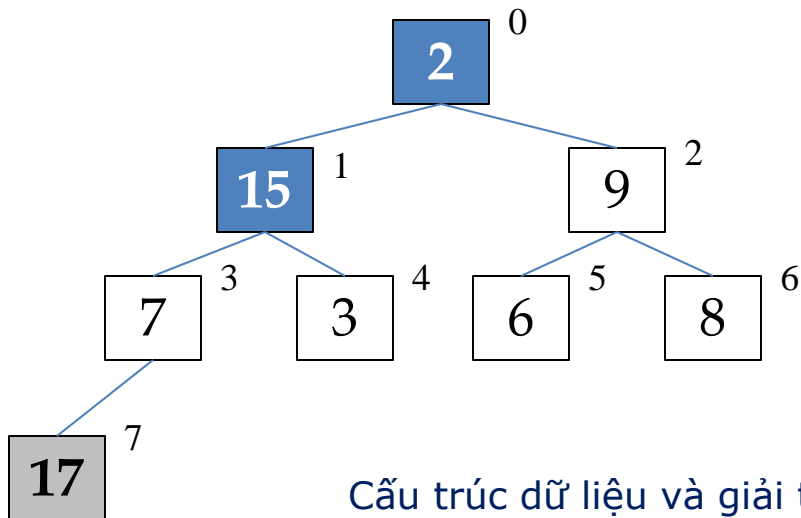
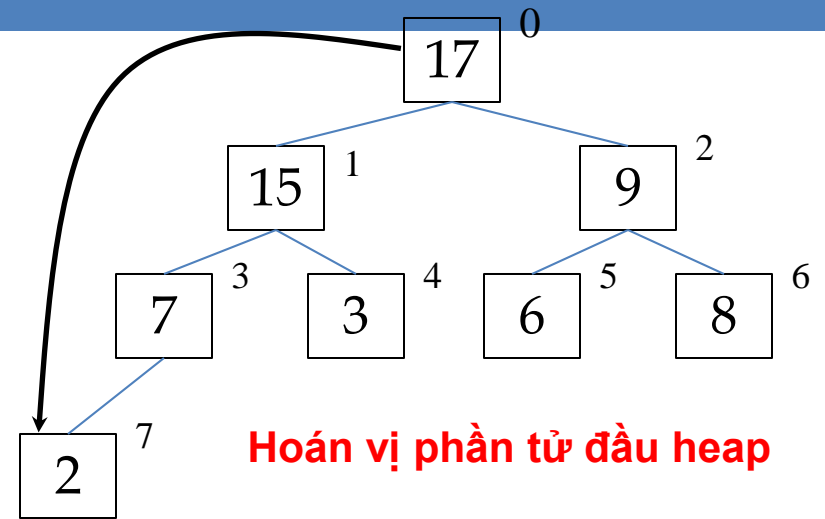
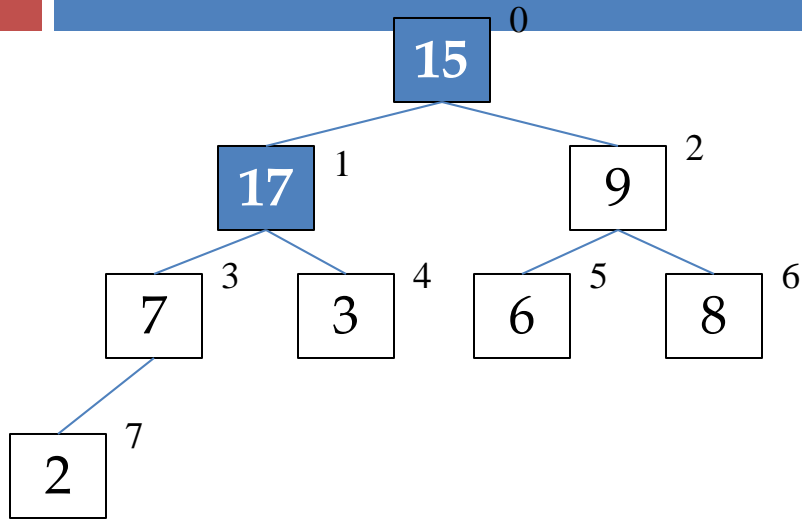
Ví dụ

20



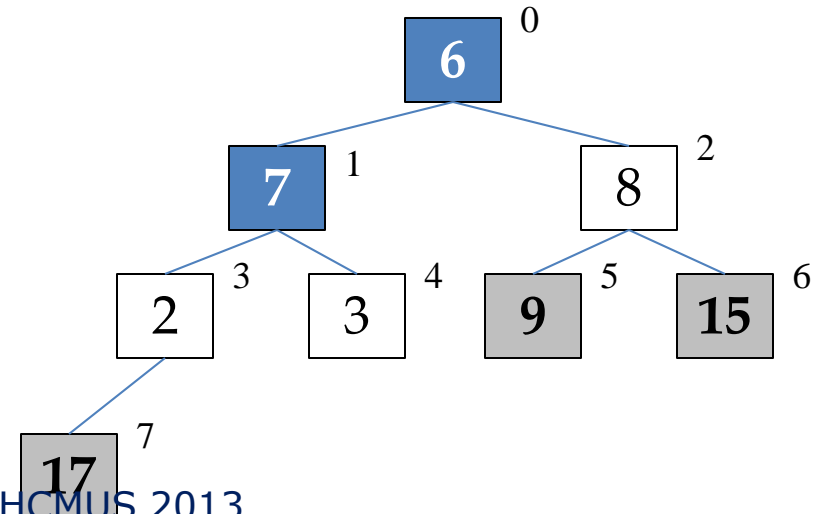
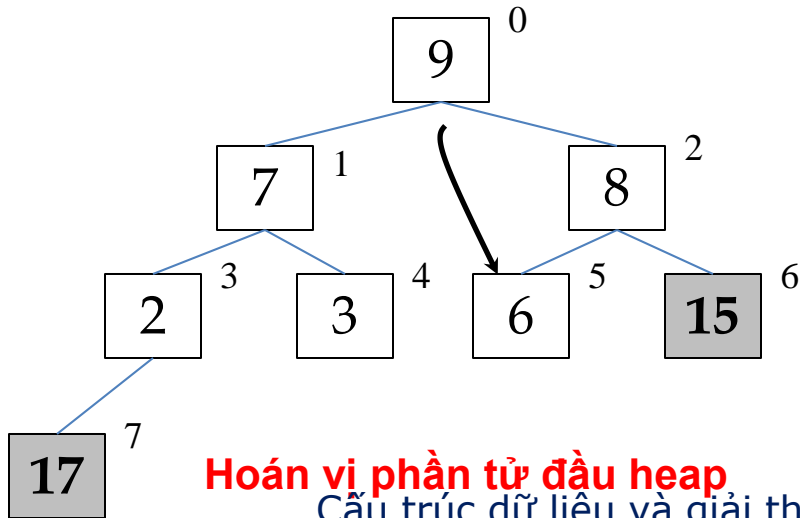
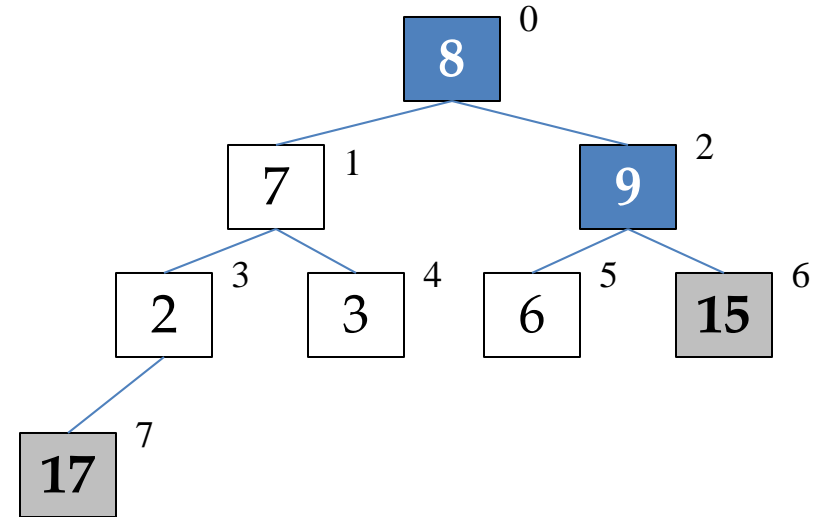
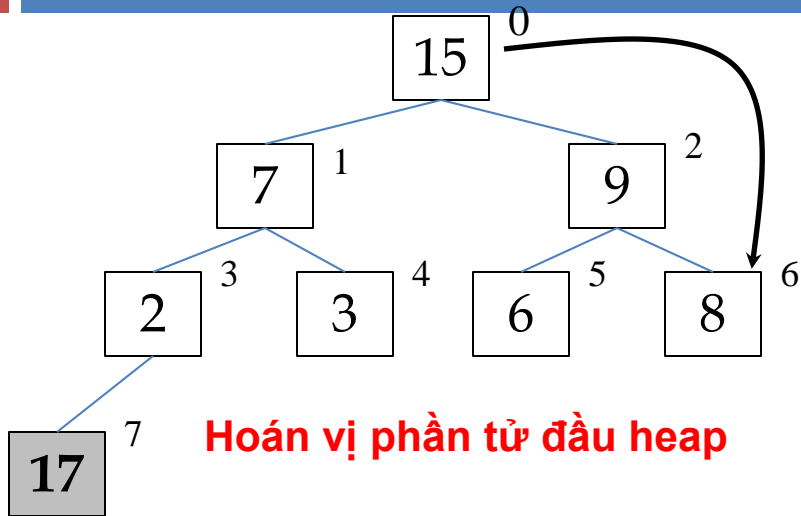
Ví dụ

21



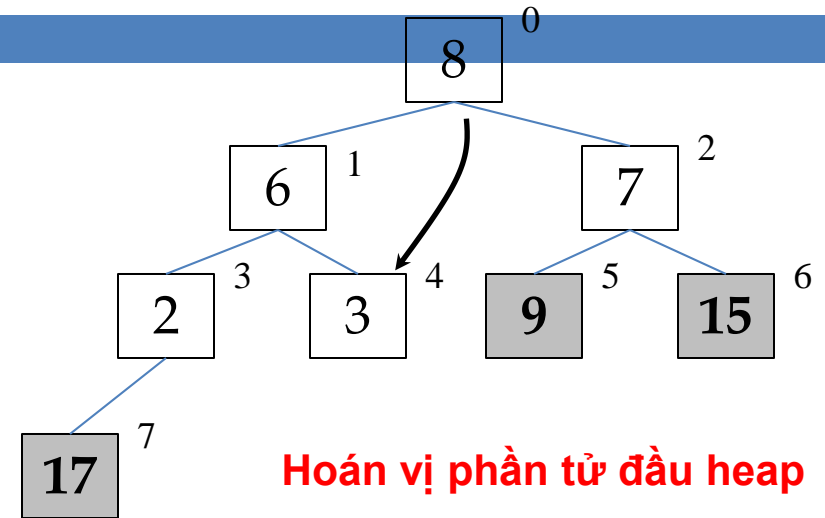
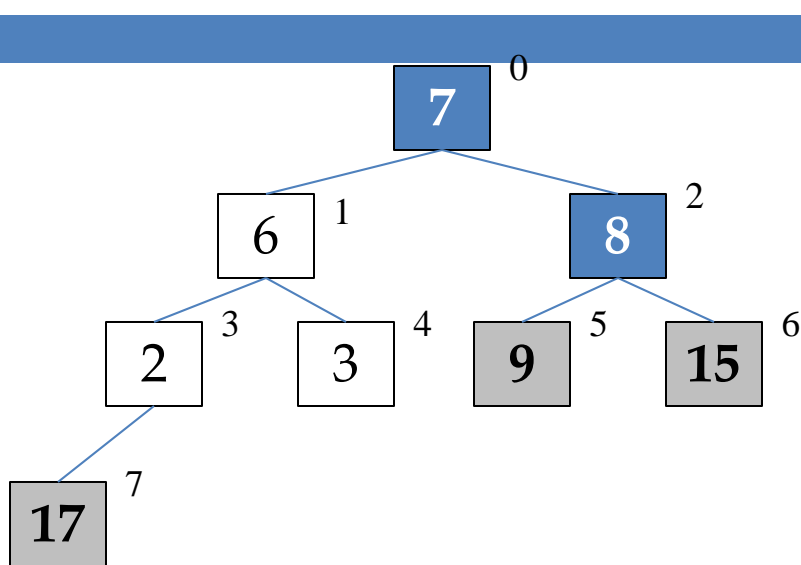
Ví dụ

22

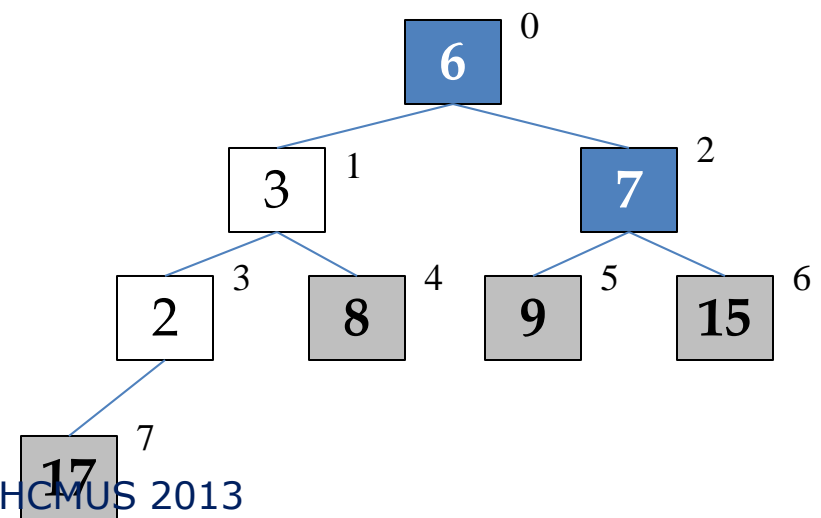
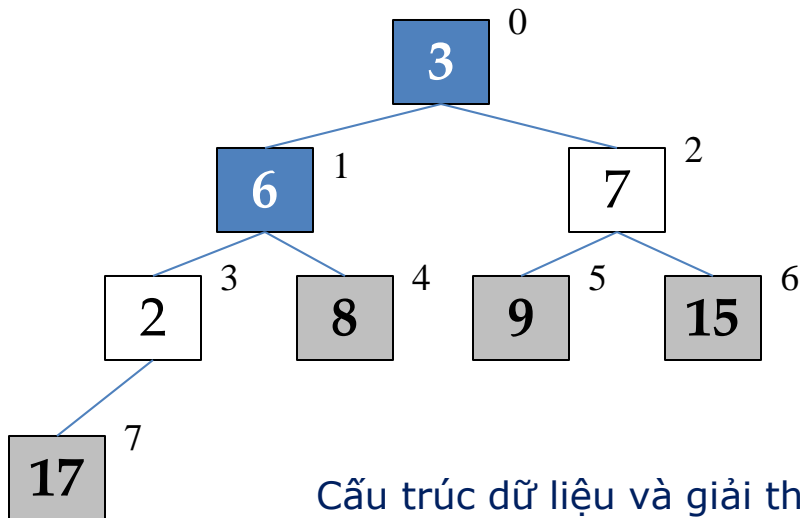


Ví dụ

23

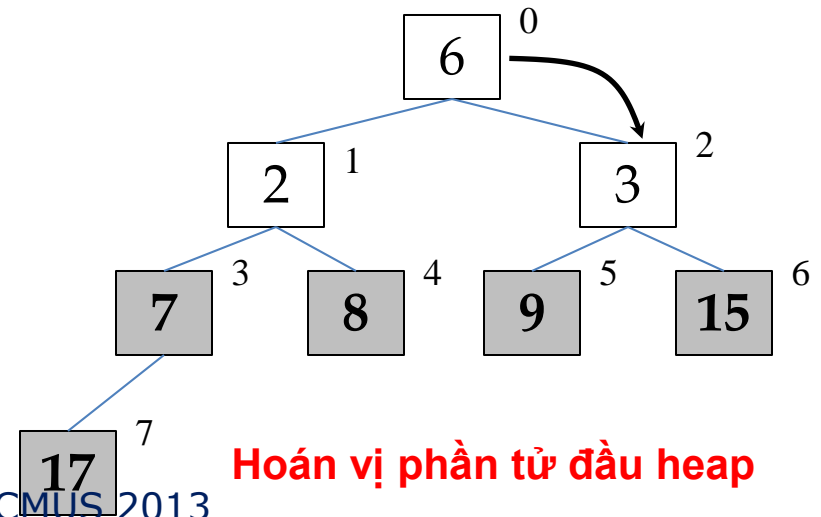
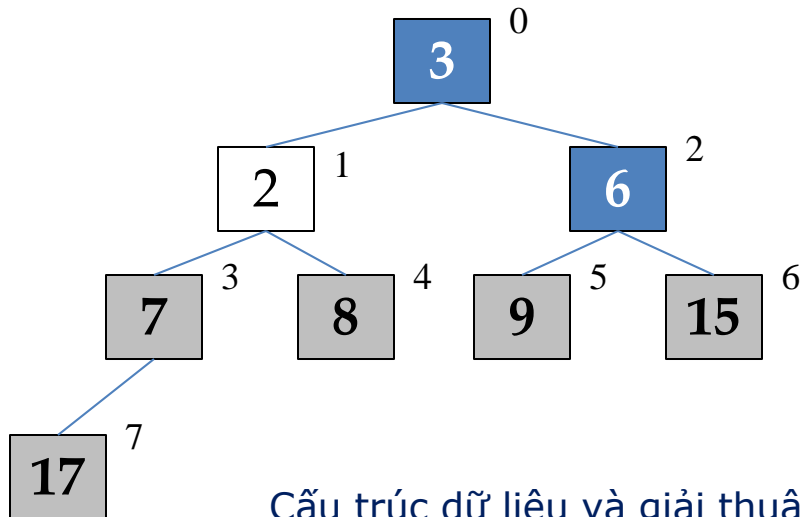
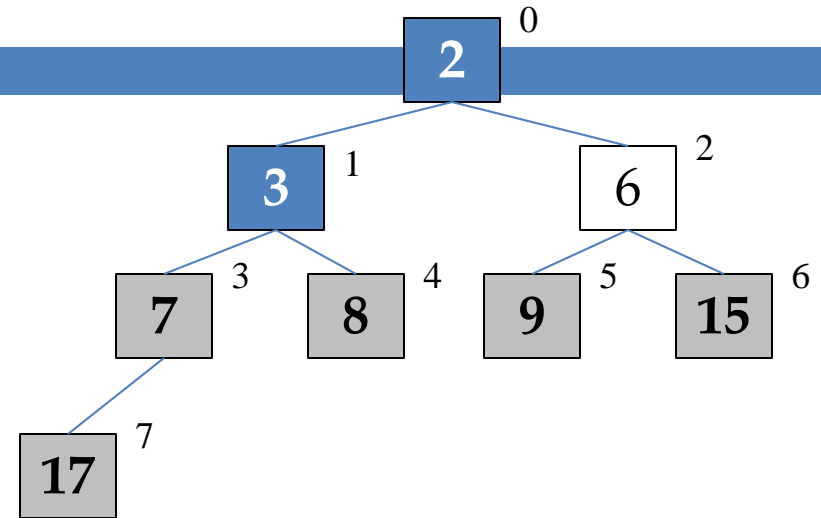
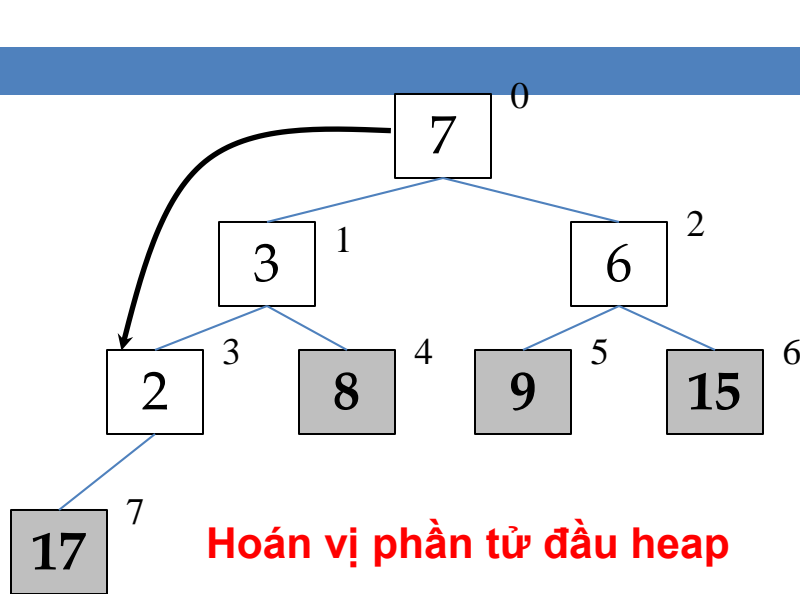


Hoán vị phần tử đầu heap



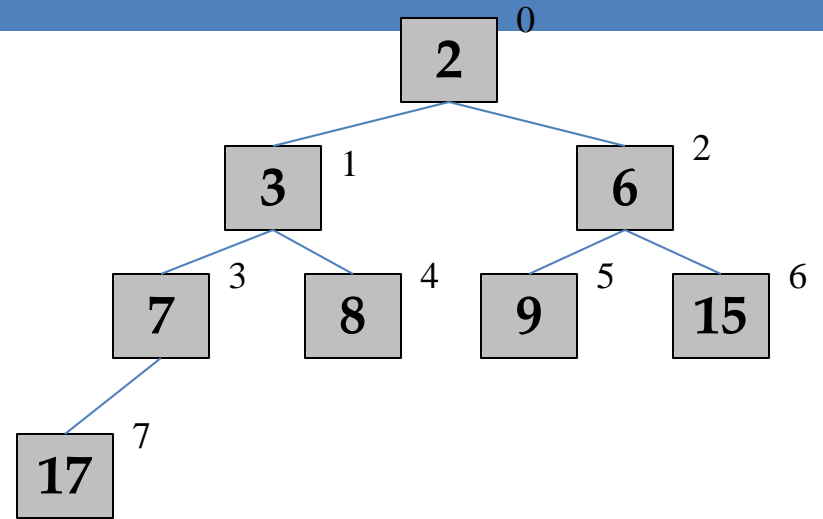
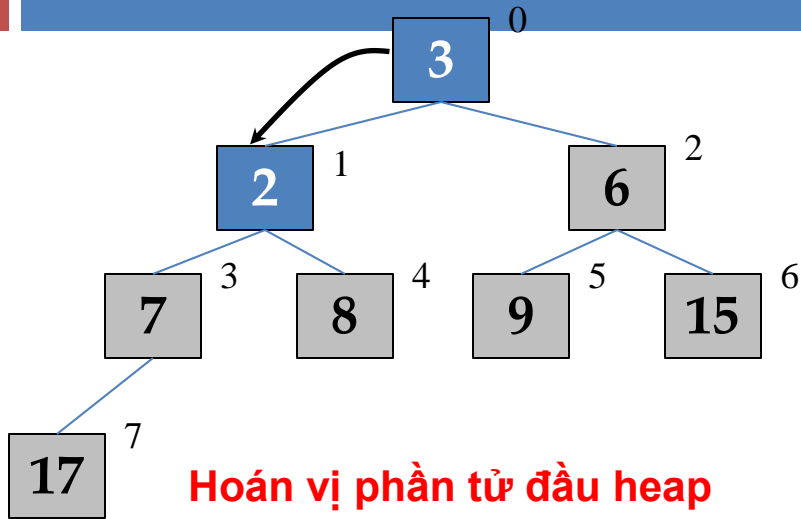
Ví dụ

24

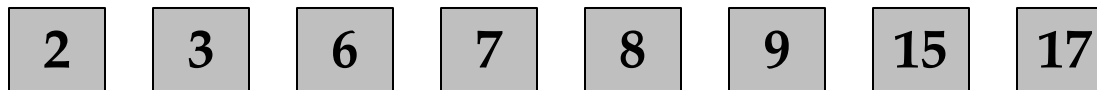


Ví dụ

25



Mảng sau khi sắp xếp:



Heap Sort

26

- ◉ Đánh giá giải thuật:
 - ▣ Độ phức tạp của giải thuật trong trường hợp xấu nhất là $O(n\log_2 n)$

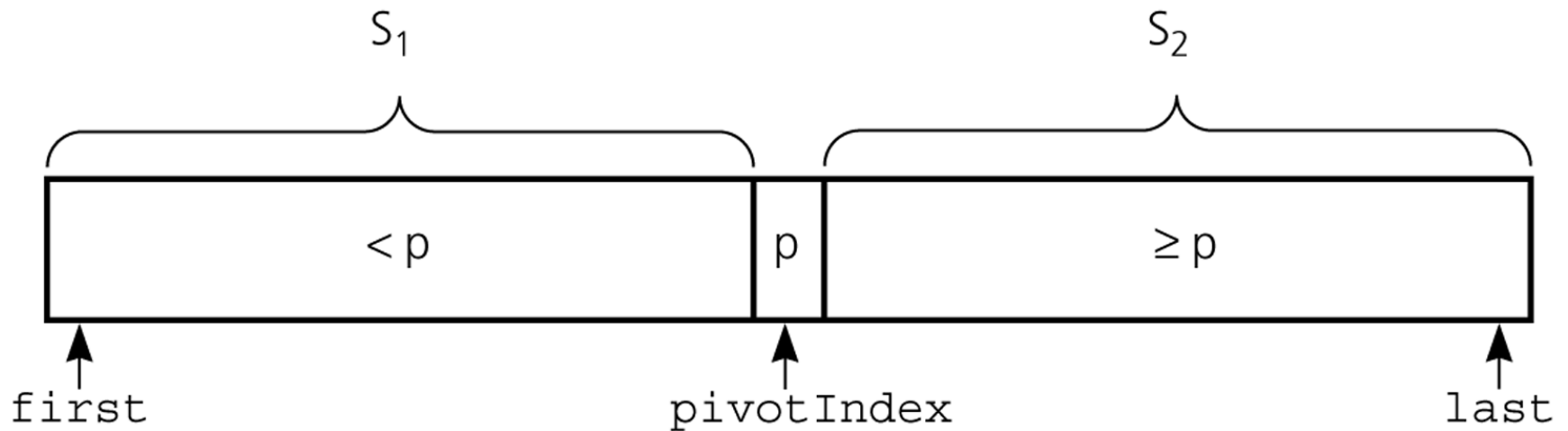
Sắp xếp nhanh

Quick Sort

Ý tưởng

28

- Phân chia dãy cần sắp xếp thành 2 phần **S1** và **S2** dựa vào phần tử mốc p :



◉ QuickSort(array[], first, last)

Nếu ($\text{first} < \text{last}$)

{

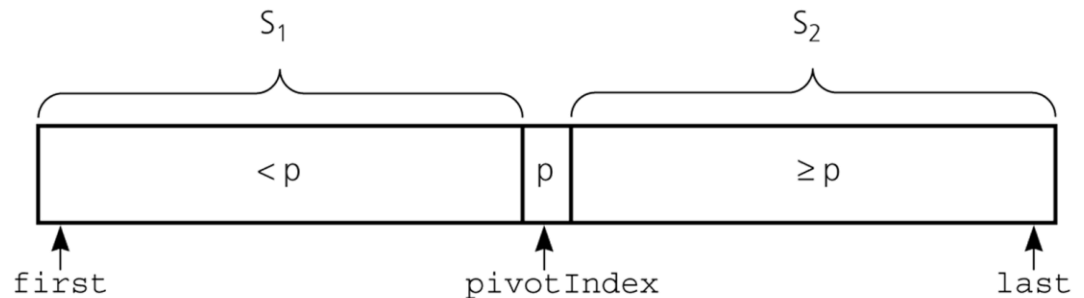
Chọn phần tử mốc ***pivot***.

Dựa vào giá trị pivot, phân hoạch dãy ***array*** thành 2 dãy mới ***S1*** ($\text{first} \dots \text{pivotIndex}-1$) và ***S2*** ($\text{pivotIndex}+1 \dots \text{last}$)

QuickSort (array, first, pivotIndex-1)

QuickSort (array, pivotIndex + 1, last)

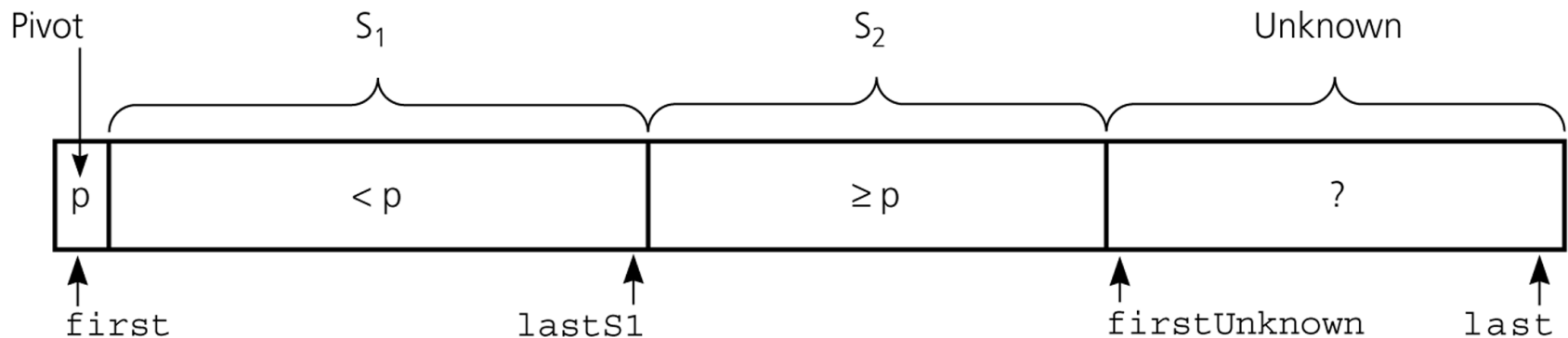
}



Phân hoạch

30

- ⊙ Sử dụng thêm 2 chỉ số **lastS1** và **firstUnknown** để phân hoạch.
- ⊙ Tiếp tục phân hoạch khi **firstUnknown \leq last**.



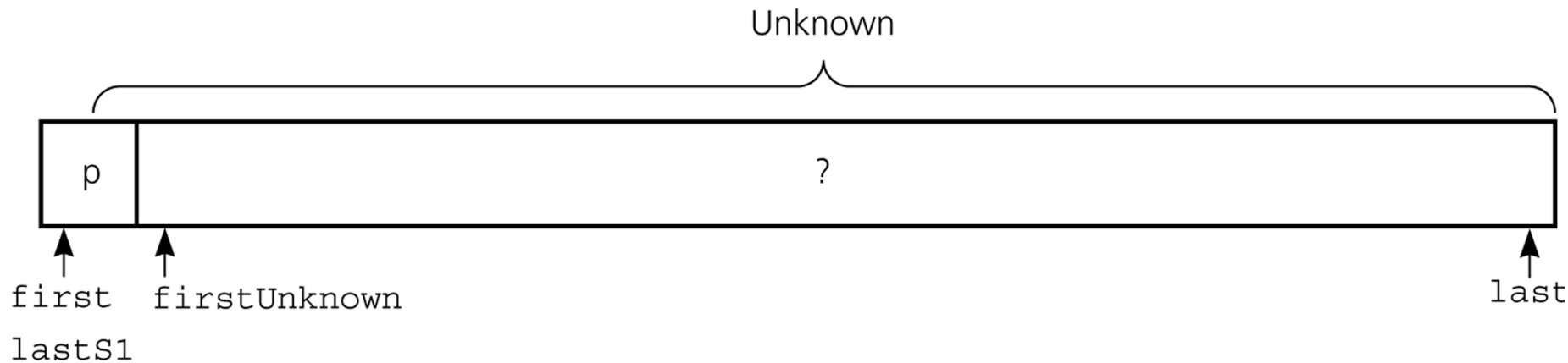
Phân hoạch

31

◉ Khởi tạo

▣ $\text{lastS1} = \text{first}$

▣ $\text{firstUnknown} = \text{first} + 1$



Phân hoạch

32

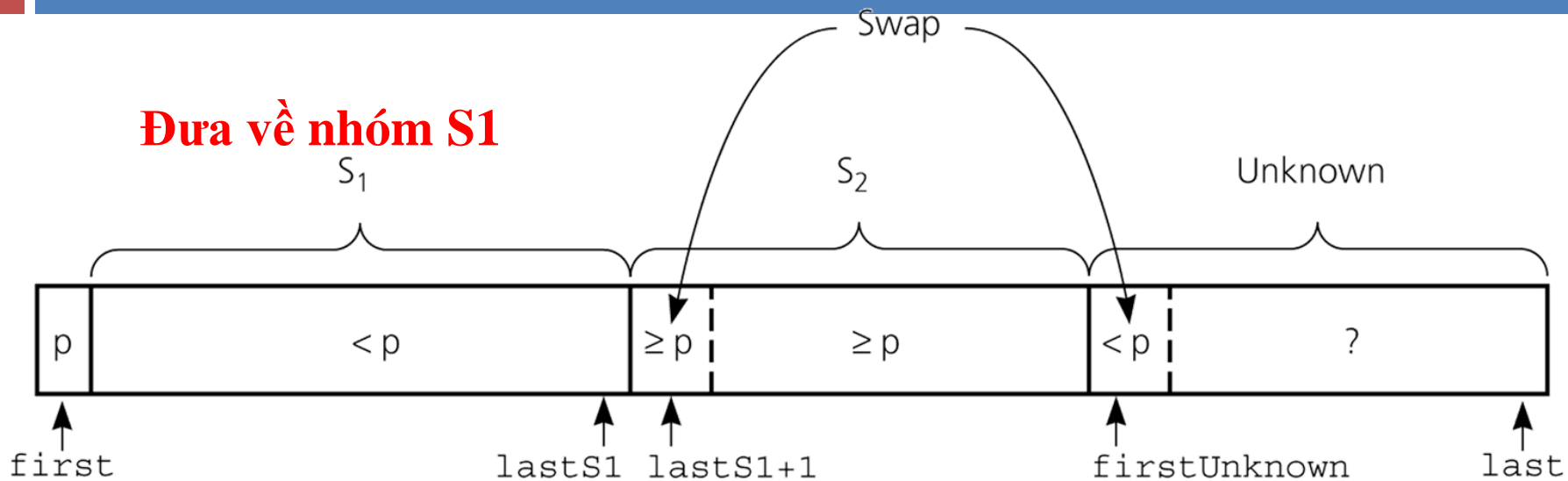
- ◉ Trong khi còn phân hoạch:
 - ▣ Nếu giá trị tại **firstUnknown** nhỏ hơn giá trị **pivot**
 - Chuyển sang nhóm S1
 - ▣ Ngược lại
 - Chuyển sang nhóm S2

- ◉ Kết thúc phân hoạch:
 - ▣ Đưa pivot về đúng vị trí (đổi chỗ giá trị **lastS1** và **first**).
 - ▣ $\text{pivotIndex} = \text{lastS1}$

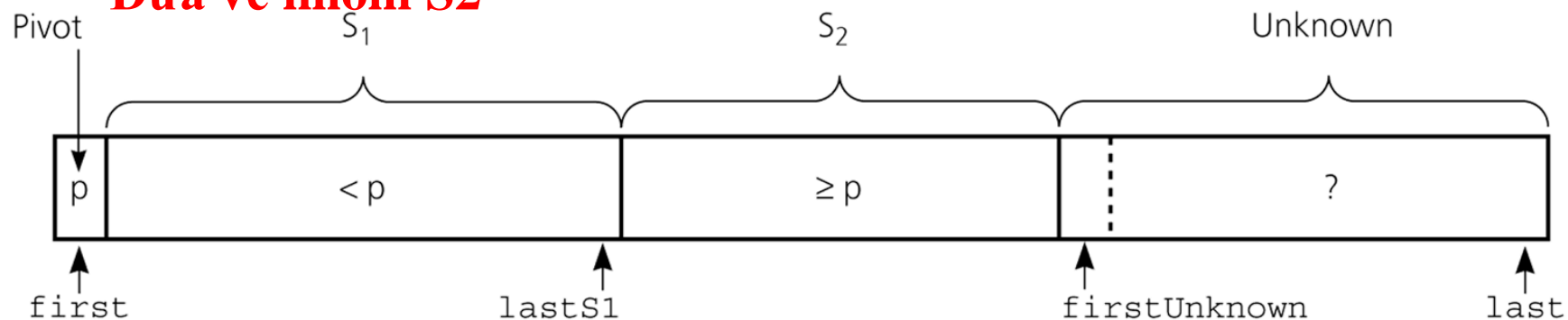
Phân hoạch

33

Đưa về nhóm S1



Đưa về nhóm S2




Ví dụ

34

- Phân hoạch dãy số: 27, 38, 12, 39, 27, 16

Pivot	Unknown				
27	38	12	39	27	16

Pivot	S2	Unknown			
27	38	12	39	27	16



Pivot	S1	S2	Unknown		
27	12	38	39	27	16


Ví dụ

35


⊙ Phân hoạch dãy số: 27, 38, 12, 39, 27, 16

Pivot	S1	S2	Unknown		
27	12	38	39	27	16

Pivot	S1	S2			U.K
27	12	38	39	27	16



Pivot	S1		S2		
27	12	16	39	27	38



S1		Pivot	S2		
16	12	27	39	27	38

Bài tập

36

- ◉ Chạy tay thuật toán Quick Sort để sắp xếp mảng A trong 2 trường hợp tăng dần và giảm dần.

$$A = \{2, 9, 5, 12, 20, 15, -8, 10\}$$

Quick Sort

37

◉ Đánh giá giải thuật:

- ▣ Hiệu quả phụ thuộc vào việc chọn giá trị mốc
 - Tốt nhất là phần tử median.
 - Nếu phần tử mốc là cực đại hay cực tiểu thì việc phân hoạch không đồng đều.
- ▣ Bảng tổng kết:

	Độ phức tạp
Tốt nhất	$O(n \log_2 n)$
Trung bình	$O(n \log_2 n)$
Xấu nhất	$O(n^2)$

Sắp xếp trộn

Merge Sort

Giới thiệu

39

- ⊙ Thực hiện theo hướng chia để trị.
- ⊙ Do John von Neumann đề xuất năm 1945.

Giải thuật

40

- ⊙ Nếu dãy có chiều dài là 0 hoặc 1: đã được sắp xếp.
- ⊙ Ngược lại:
 - ▣ Chia dãy thành 2 dãy con (chiều dài tương đương nhau).
 - ▣ Sắp xếp trên từng dãy con bằng thuật toán Merge Sort.
 - ▣ Trộn 2 dãy con (đã được sắp xếp) thành một dãy mới đã được sắp xếp.

Giải thuật

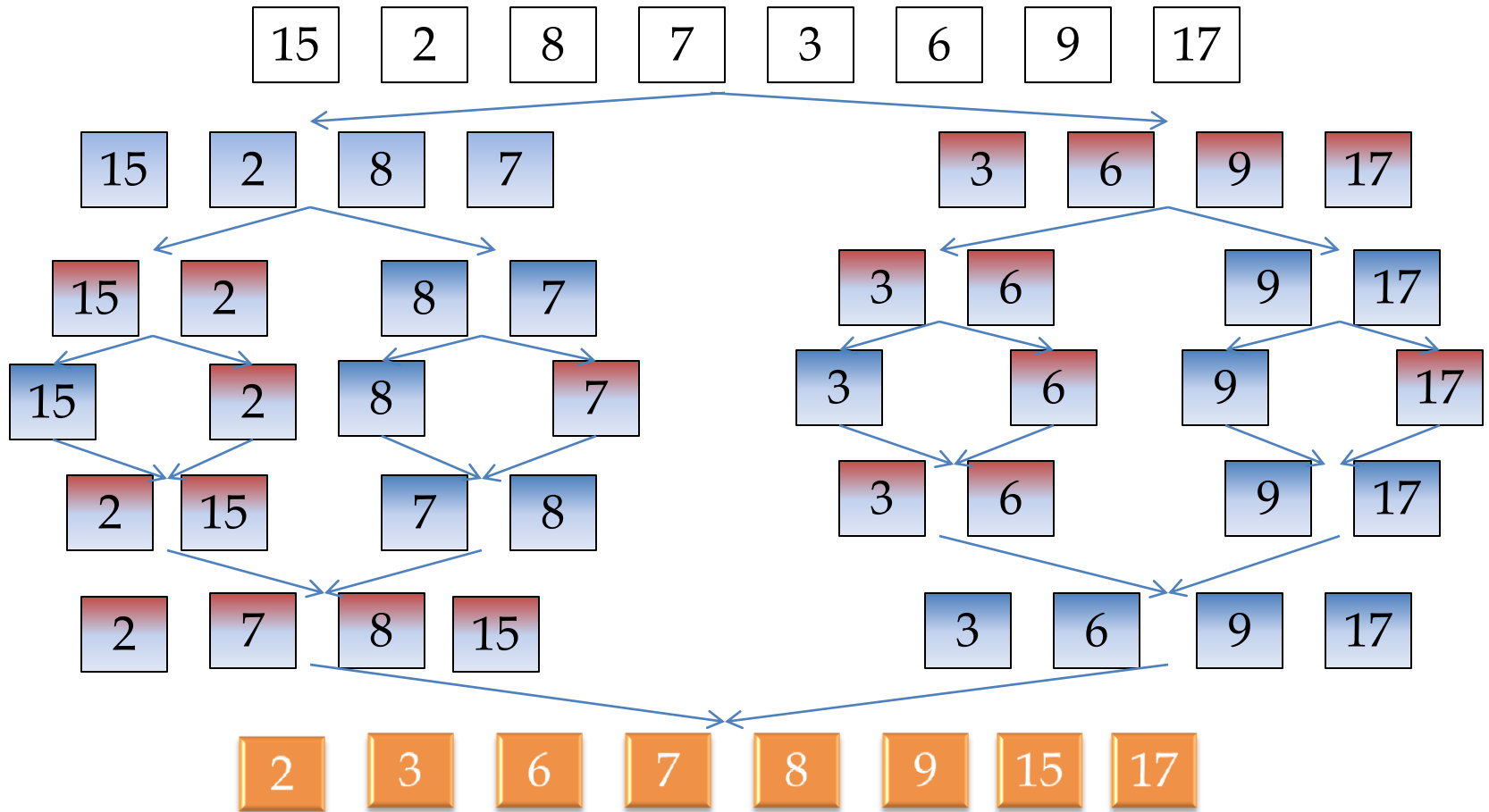
41

- ⊙ Input: Dãy A và các chỉ số left, right (sắp xếp dãy A gồm các phần tử có chỉ số từ *left* đến *right*).
- ⊙ Output: Dãy A đã được sắp xếp

```
MergeSort(A, left, right)
{
    if (left < right) {
        mid = (left + right)/2;
        MergeSort(A, left, mid);
        MergeSort(A, mid+1, right);
        Merge(A, left, mid, right);
    }
}
```

Ví dụ

42



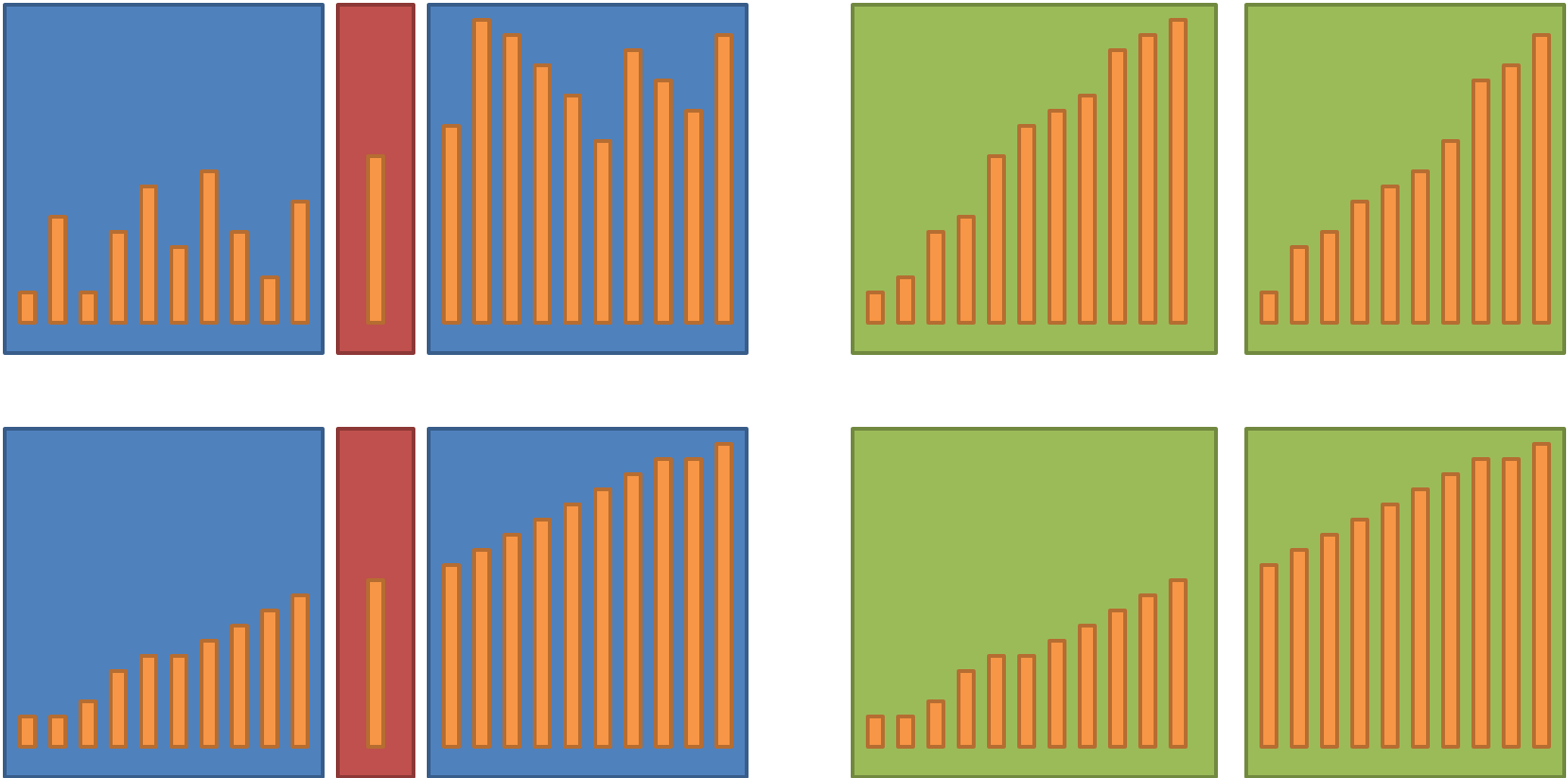
Đánh giá

43

- ◉ Số lần chia các dãy con: $\log_2 n$
- ◉ Chi phí thực hiện việc trộn hai dãy con đã sắp xếp tỷ lệ thuận với n .
- ◉ Chi phí của Merge Sort là $O(n \log_2 n)$
- ◉ Thuật toán không sử dụng thông tin nào về đặc tính của dãy cần sắp xếp \Rightarrow chi phí thuật toán là không đổi trong mọi trường hợp

So sánh tư tưởng sắp xếp giữa Quick sort và Merge sort

44



Sắp xếp theo cơ số

Radix Sort

- ⊙ Không dựa vào việc so sánh các phần tử
- ⊙ Sử dụng các ‘thùng’ để nhóm các giá trị theo cơ sở của vị trí đang xem xét.
- ⊙ Nối kết các giá trị trong ‘thùng’ để tạo thành dãy sắp xếp.

Ví dụ

48

- Cho dãy số sau: 27, 78, 52, 39, 17, 46
- Cơ số: 10, Số lượng ký số: 2
- Xét ký số thứ nhất

0	1	2	3	4	5	6	7	8	9
							17		
		52				46	27	78	39

Kết hợp lại: **52, 46, 27, 17, 78, 39**

Ví dụ

49

- ⊙ Xét ký số thứ 2 của: 52, 46, 27, 17, 78, 39

0	1	2	3	4	5	6	7	8	9
	17	27	39	46	52		78		

Kết hợp dãy có thứ tự: **17, 27, 39, 46, 52, 78**

Đánh giá

50

- ◉ Độ phức tạp của thuật toán: $O(n)$
(Chi tiết hơn: $O(k*n)$ với k là số lượng ký số)

Kết luận

Kết luận

52

- ⊙ Các thuật toán Bubble sort, Selection sort, Insertion sort
 - ▣ Cài đặt thuật toán đơn giản.
 - ▣ Chi phí của thuật toán cao: $O(n^2)$.
- ⊙ Heap sort được cải tiến từ Selection sort nhưng chi phí thuật toán thấp hơn hẳn ($O(n \log_2 n)$)

Kết luận

53

- ⊙ Các thuật toán Quick sort, Merge sort là những thuật toán theo chiến lược chia để trị.
 - ▣ Cài đặt thuật toán phức tạp
 - ▣ Chi phí thuật toán thấp: $O(n \log_2 n)$
 - ▣ Rất hiệu quả khi dùng danh sách liên kết.
 - ▣ Trong thực tế, Quick sort chạy nhanh hơn hẳn Merge sort và Heap sort.

Kết luận

54

- ◉ Người ta chứng minh $O(n \log_2 n)$ là ngưỡng chặn dưới của các thuật toán sắp xếp dựa trên việc so sánh giá trị của các phần tử.

Hỏi và Đáp