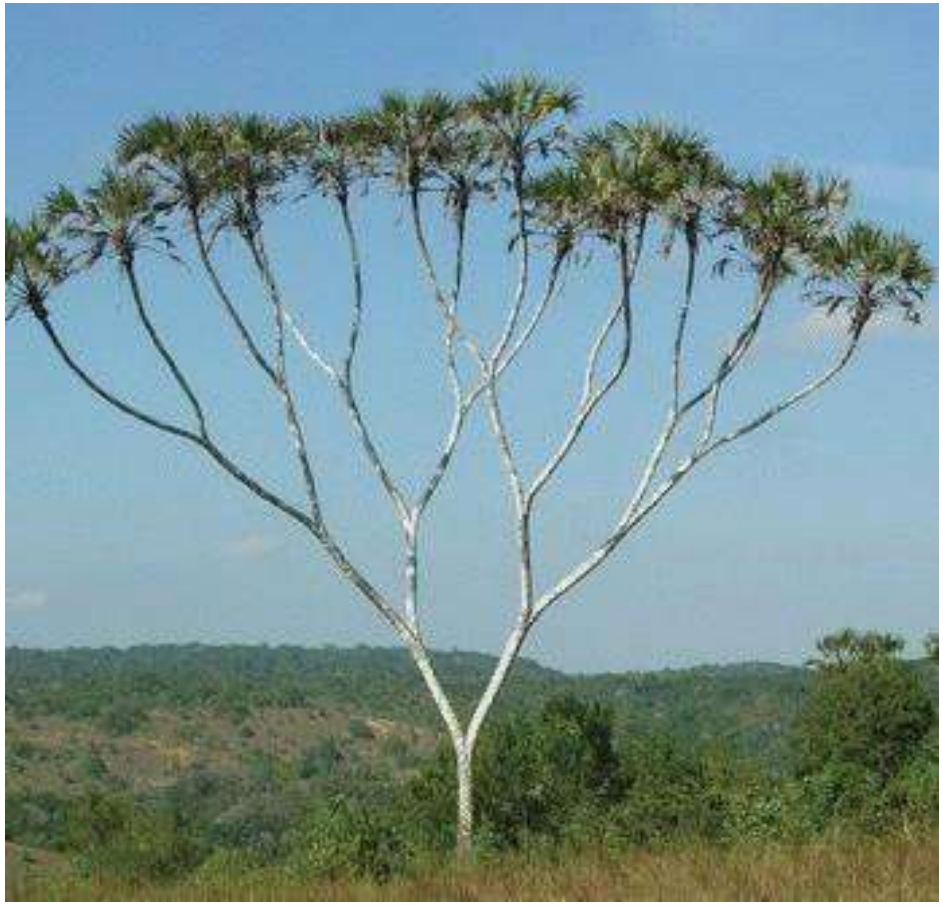
Heap và hàng đợi ưu tiên

Nội dung

- Heap tree
- Hàng đợi ưu tiên
- Heap sort



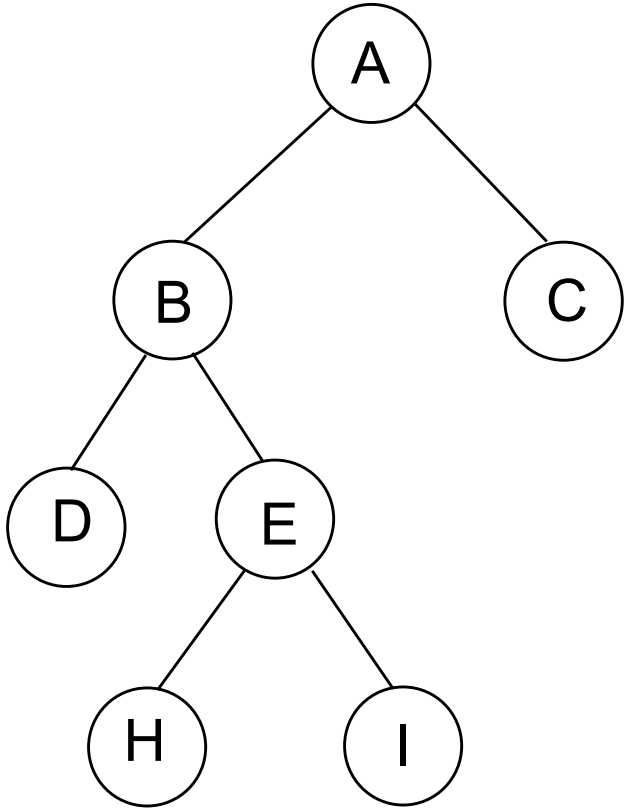
Array, Linked list, và BST

OPERATION	SORTED ARRAY	UNSORTED LINKED LIST	BST (WORST CASE)	BST (BALANCED)
SEARCH	O(log(n))	O(n)	O(n)	O(log(n))
DELETE	O(n)	O(n)	O(n)	O(log(n))
INSERT	O(n)	O(1)	O(n)	O(log(n))

- Tìm kiếm nhanh giá trị lớn nhất?
- Thêm và bớt nút hiệu quả?
- Lưu trữ hiệu quả?

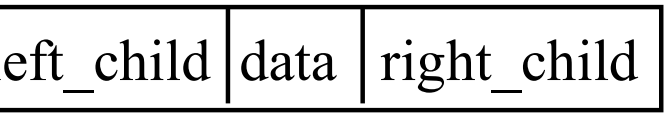
Cấu trúc cây và cây nhị phân

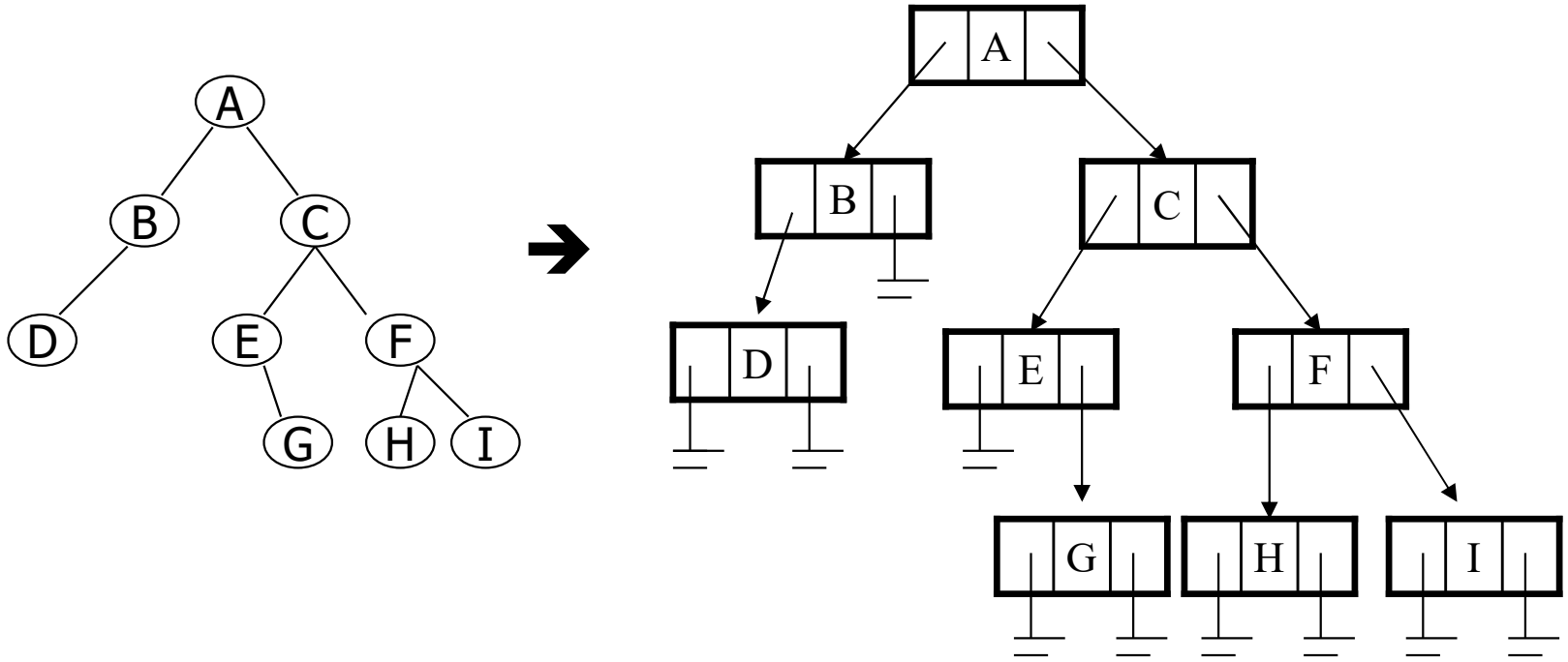
- Là cấu trúc dữ liệu dựa trên liên kết các node
- Mỗi node có thể có liên kết với các nút con
  - Không có vòng lặp
  - Nút gốc: không có nút cha; Nút lá: không có nút con
  - Cây con: bộ phân của cây có đỉnh không phải là nút gốc
- Cây nhị phân
  - Chỉ có tối đa 2 nút con
  - Là cấu trúc cây phổ biến



Cây nhị phân: dùng linked list

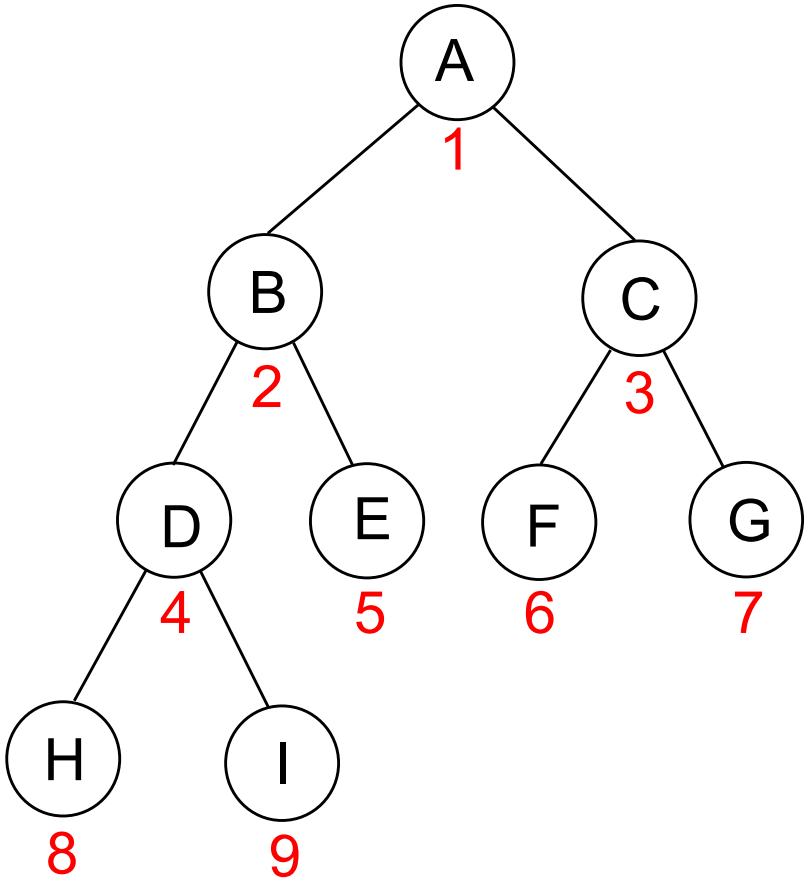
```
struct node {
    long data;
    node* left_child,* right_child;
};
```





Cây nhị phân: dùng mảng

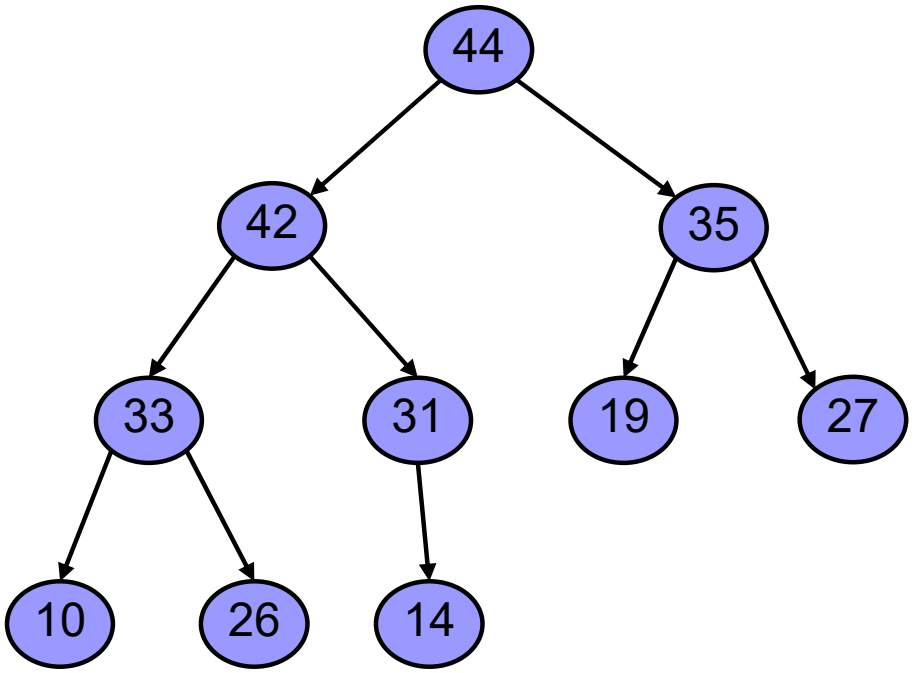
- Cây dạng hoàn chỉnh (complete tree)
  - Cây cân bằng
  - Cần lưu số lượng nút thực
- Quan hệ cha con
  - Nút cha của nút thứ n là ???



[1]	A
[2]	B
[3]	C
[4]	D
[5]	E
[6]	F
[7]	G
[8]	H
[9]	I

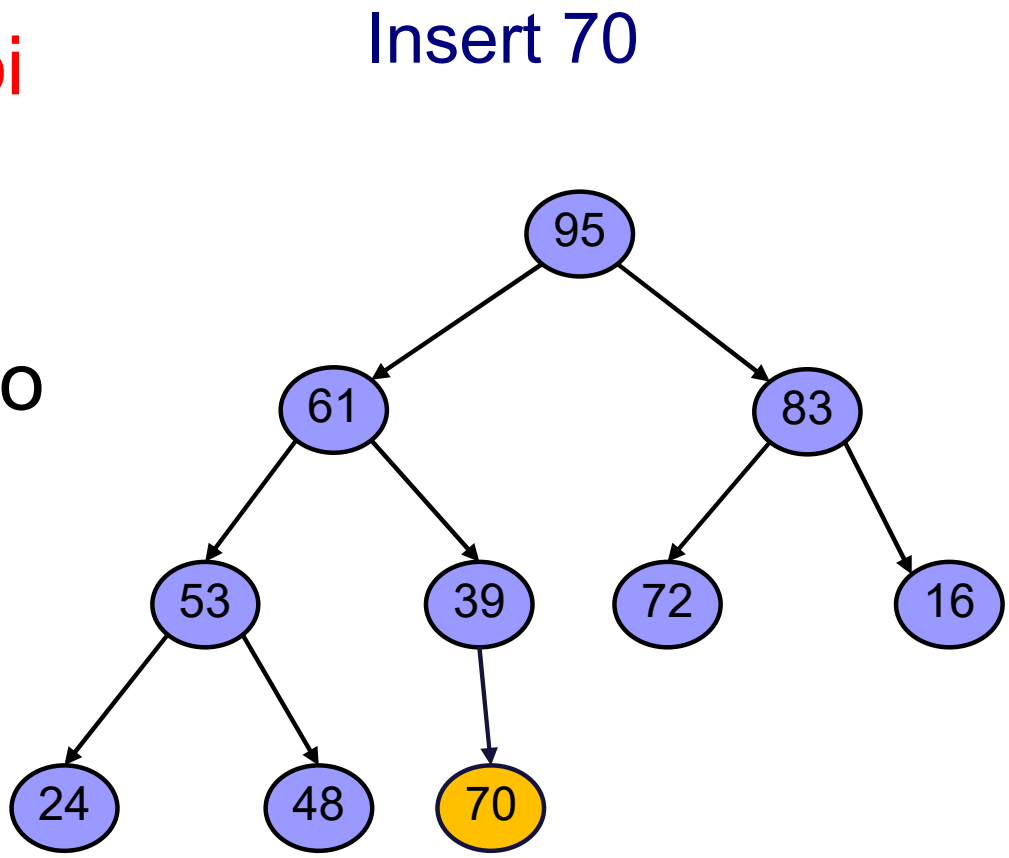
Binary Heap

- Là cây nhị phân dạng hoàn chỉnh
- Nút đỉnh có giá trị lớn hơn (hoặc nhỏ hơn) nút con
  - Max heap/ Min heap
- Được lưu trữ thực tế dưới dạng mảng

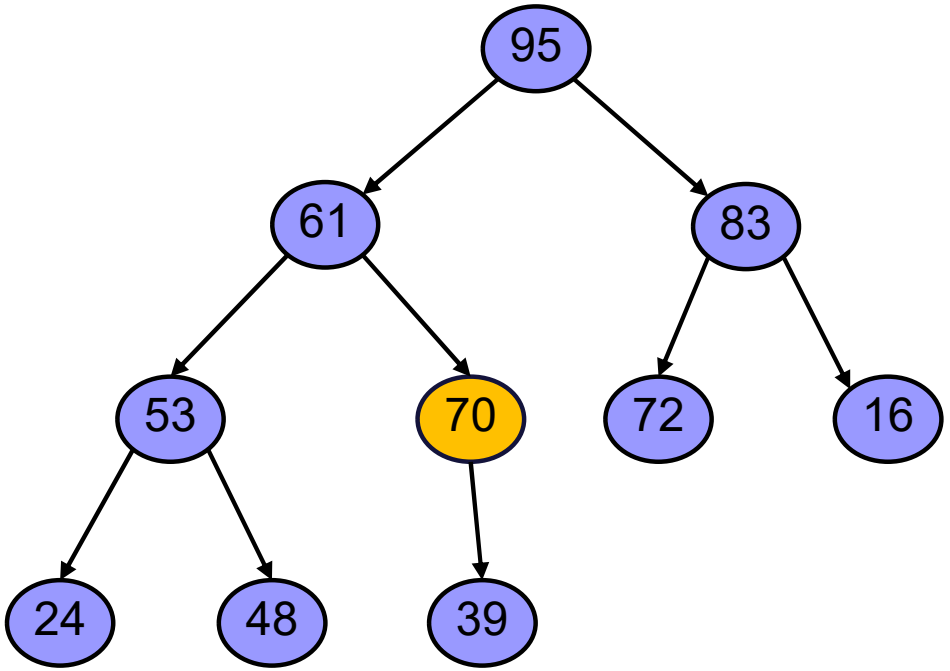


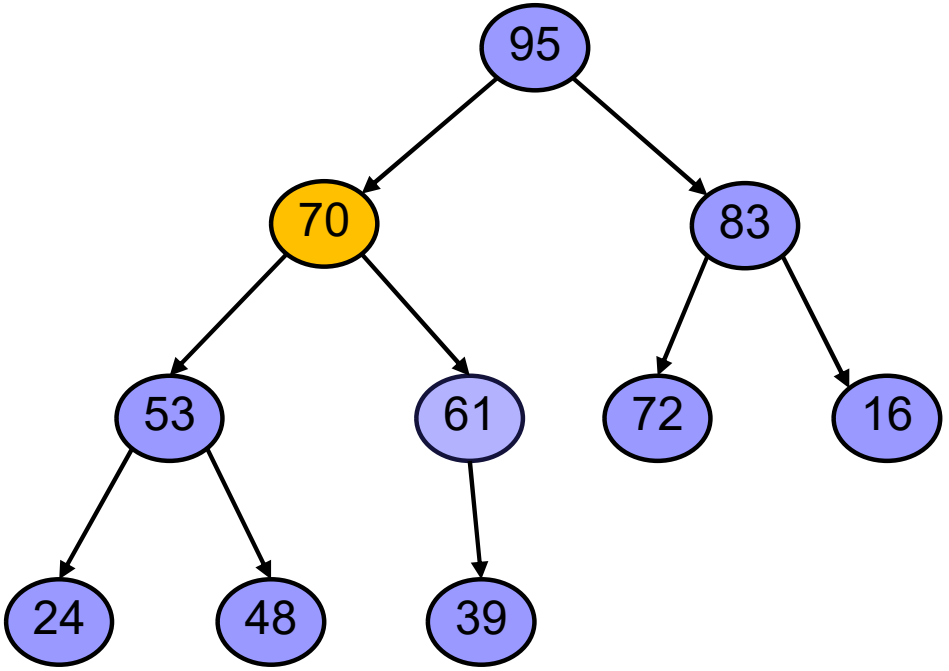
Heap: thêm nút

- Thêm nút mới vào cuối mảng
- So sánh nút mới với nút cha, nếu thứ tự đảo ngược thì đổi chỗ
  - Lặp lại đổi chỗ cho đến heap đúng thứ tự



Heap: swim up





Ví dụ cài đặt

```
void insert(int a[], int size, int n) {
    a[size] = n;
    swim(size++);
}

void swim(int a[], int k) {



        // ???
}
```

Ví dụ cài đặt

```
void insert(int a[], int size, int n) {
    a[size] = n;
    swim(a, size++);
}

void swim(int a[], int k) {
    while (k > 0 && a[k/2] < a[k]) {
        swap(a, k/2, k);
        k = k/2;
    }
}
```

Bài tập

Hãy tạo cây heap cho dữ liệu được thêm vào tuần tự như sau

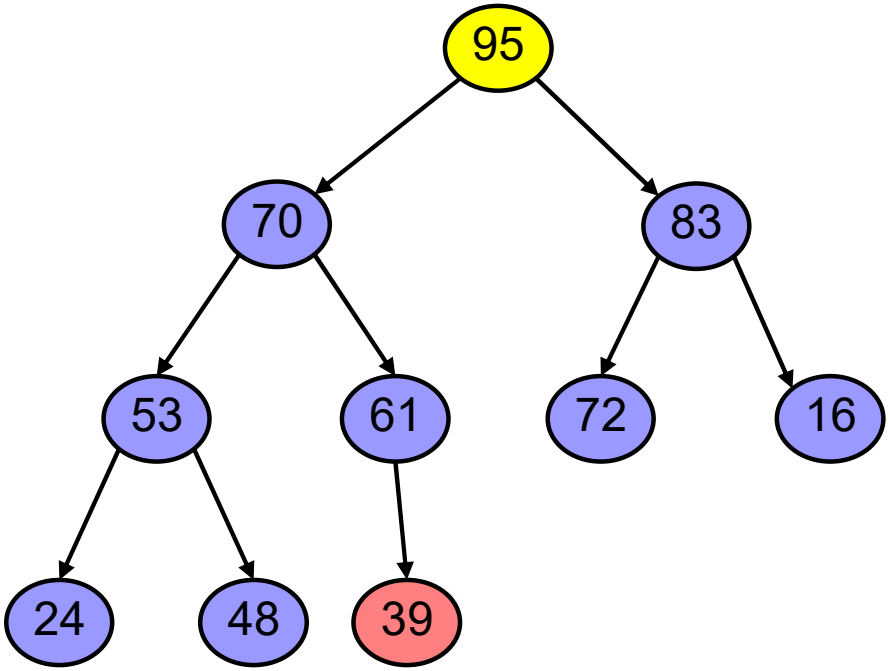
H P E X A M P

Heap: xóa nút đỉnh

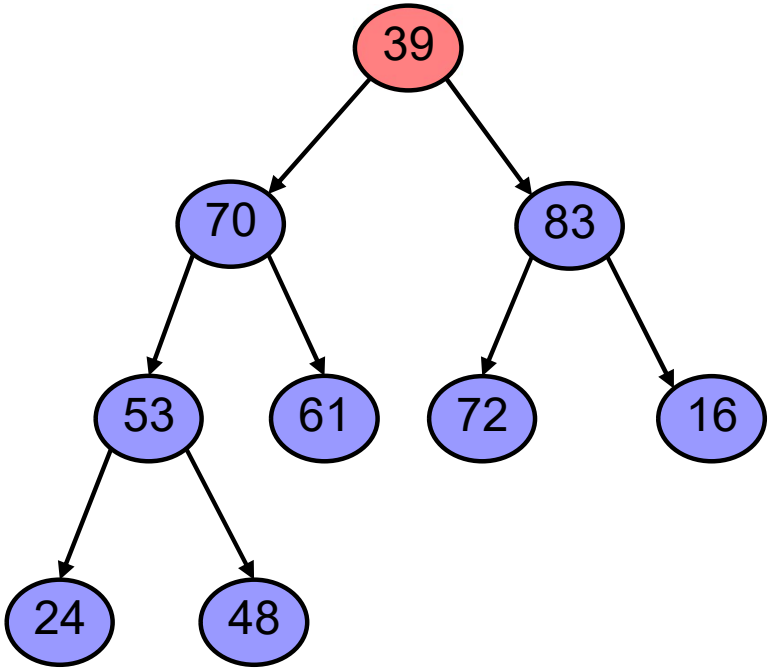
- Loại bỏ nút đỉnh (lấy ra nút có giá trị max hoặc min)
- Chuyển nút cuối cùng đến đỉnh, giảm kích thước heap
- So sánh nút đỉnh với các nút con, nếu thứ tự đảo ngược thì đổi chỗ
  - Lặp lại việc đổi chỗ cho đến khi heap đúng thứ tự

Xóa nút đỉnh

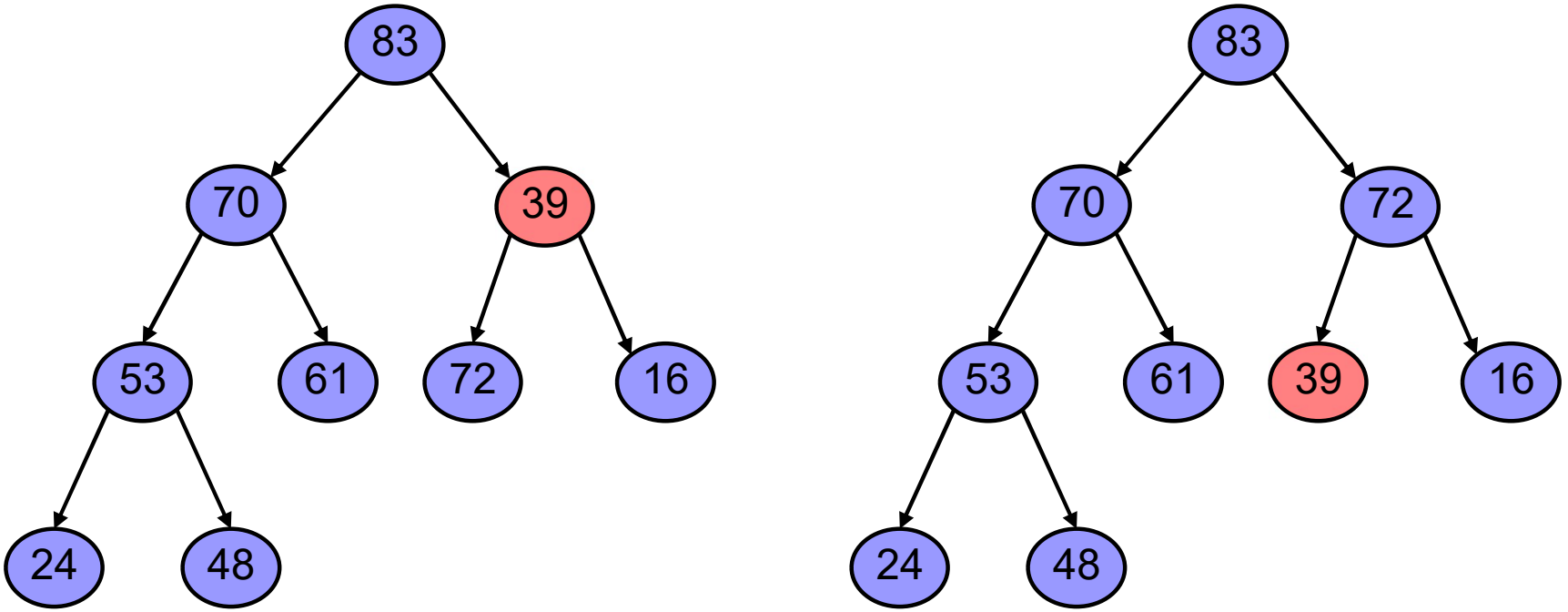
Trả về 95



Đổi chỗ



Xóa nút đỉnh: sink down



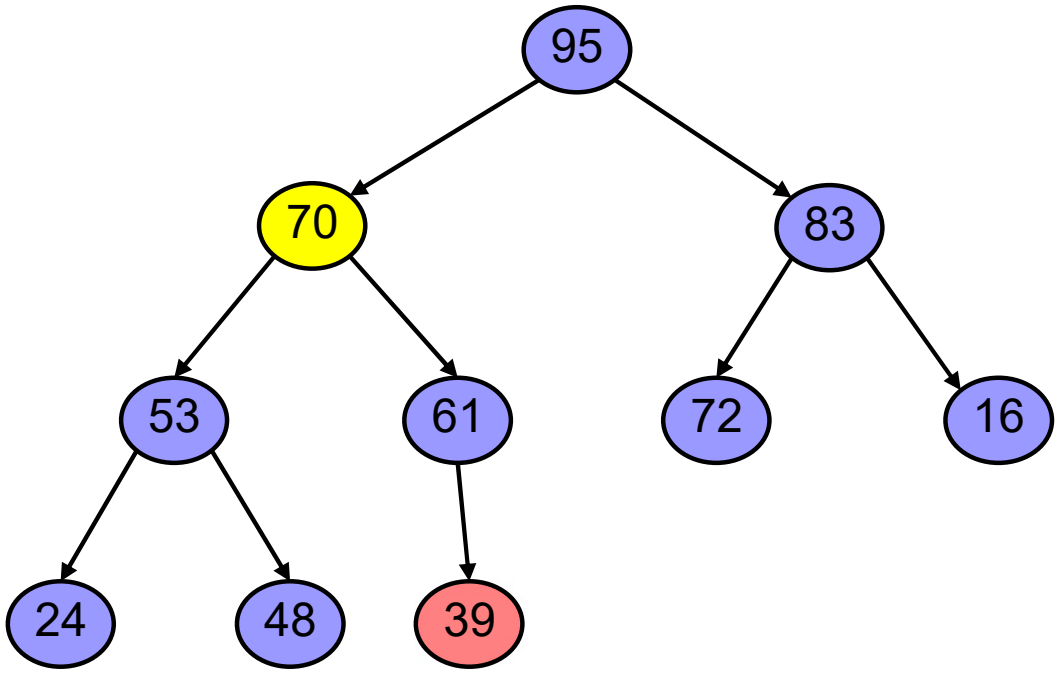
Ví dụ cài đặt

```
int remove_max(int a[], int size) {
    int max = a[0];
    a[0] = a[--size];
    sink(a, 0, size);
    return max;
}

void sink(int a[], int k, int size) {
   ???
}
```

Xóa nút bất kỳ

- Xóa nút *n* bất kỳ
- Thay thế *n* bằng nút cuối heap
- Thực hiện sink down với đỉnh *n*



Hàng đợi ưu tiên
Priority queue

- Cấu trúc dữ liệu hàng đợi có thêm thông tin độ ưu tiên của các phần tử
- Phần tử có độ ưu tiên cao nhất được lấy ra trước
- Áp dụng trong các bài toán lập lịch, các giải thuật đồ thị (tìm đường đi ngắn nhất…),…

Hàng đợi ưu tiên: Ví dụ

operation	argument	return value	size	contents (unordered)	contents (ordered)
insert	P		1	P	P
insert	Q		2	P Q	P Q
insert	E		3	P Q E	E P Q
remove max		Q	2	P E	E P
insert	X		3	P E X	E P X
insert	A		4	P E X A	A E P X
insert	M		5	P E X A M	A E M P X
remove max		X	4	P E M A	A E M P
insert	P		5	P E M A P	A E M P P
insert	L		6	P E M A P L	A E L M P
insert	E		7	P E M A P L E	A E E L M
remove max		P	6	E E M A P L	A E E L M

A sequence of operations on a priority queue

Hàng đợi ưu tiên: cài đặt

- Cài đặt bằng mảng
  - Sắp xếp mảng để đảm bảo thứ tự ưu tiên
- Cài đặt bằng danh sách liên kết
  - Dễ dàng thêm nút
- Cài đặt bằng heap
  - Thêm vào / Lấy ra?
  - Kết hợp lợi điểm của cả mảng có thứ tự và danh sách liên kết

Hàng đợi ưu tiên: so sánh

implementation	insert	remove max	max
Unordered array (linked list)	*O(1)*		
Ordered array			
Binary heap			

Hàng đợi ưu tiên: so sánh

implementation	insert	remove max	max
Unordered array (linked list)	*O(1)*	*O(n)*	*O(n)*
Ordered array	*O(n)*	*O(1)*	*O(1)*
Binary heap	*O(log n)*	*O(log n)*	*O(1)*

Bài tập

- Vẽ “cây heap” cho hàng đợi ưu tiên khi thực hiện chuỗi thao tác sau:

Insert P
Insert Q
Insert E
Remove max
Insert X
Insert A
Insert M
Remove max
Insert P
Insert L
Insert E

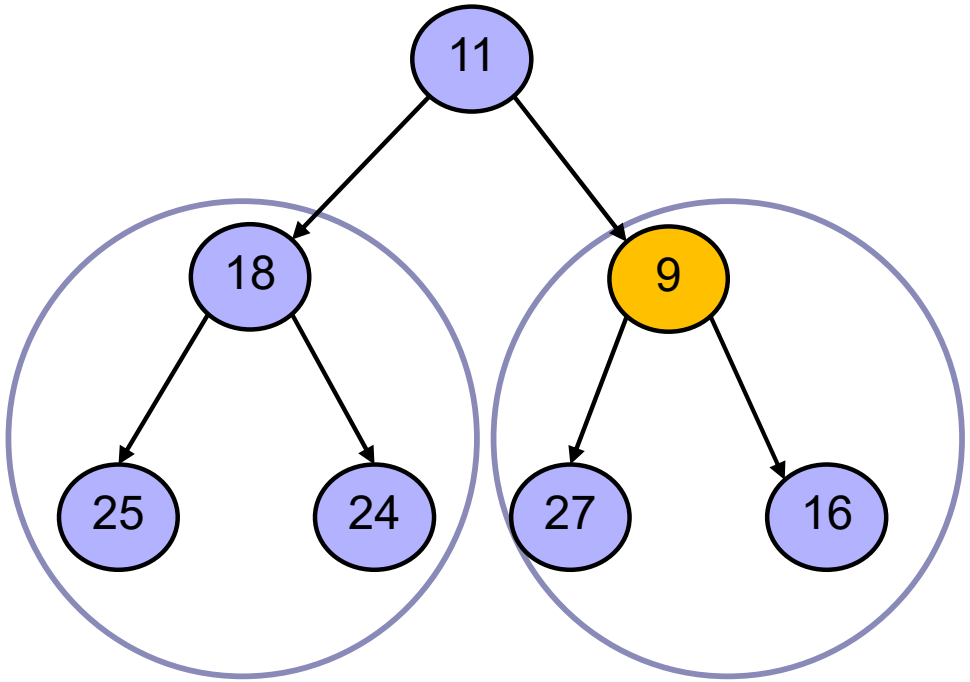
Heapsort - sắp xếp vun đống

- Là giải thuật sắp xếp ứng dụng tính chất của heap
- Xem mảng như một cây nhị phân hoàn chỉnh
- Sắp xếp mảng gồm 2 pha
  1. Tạo heap từ mảng bằng cách vun đống từ dưới lên
  2. Tạo mảng được sắp xếp bằng cách lần lượt lấy phần tử lớn nhất ra khỏi heap và đổi chỗ với phần tử cuối mảng

Tạo heap: vun đống

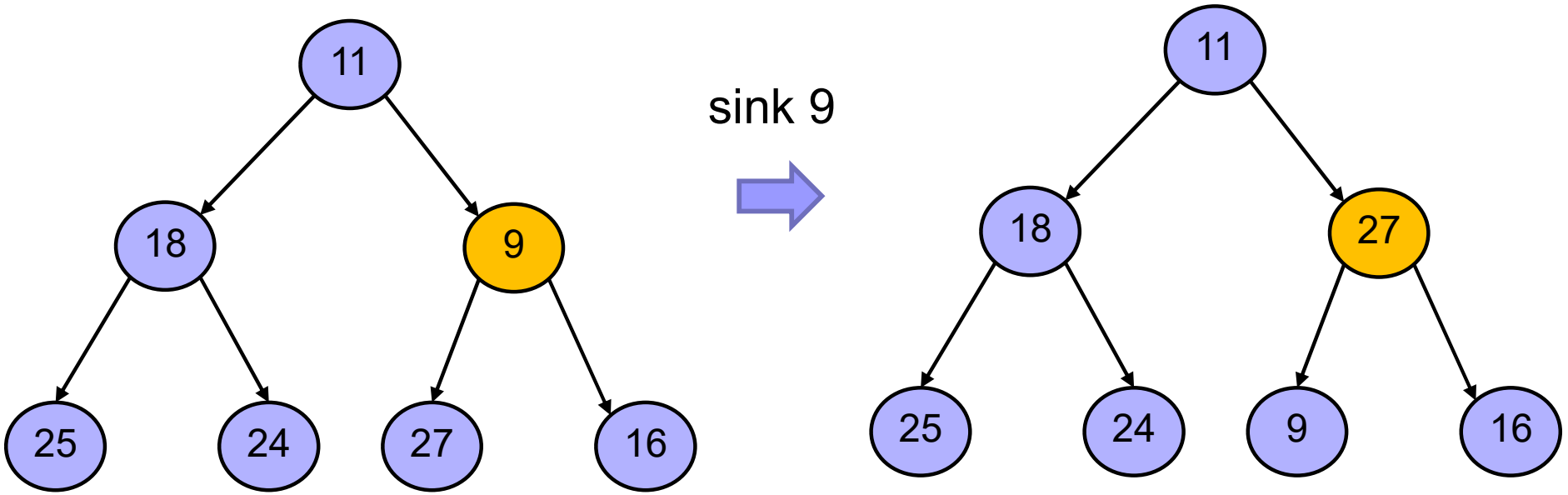
- Vun đống cây con từ dưới lên, từ phải qua trái
- Tăng độ dần độ cao của cây
  - Các cây con của nó đã được vun đống

11 18 9 25 24 27 16

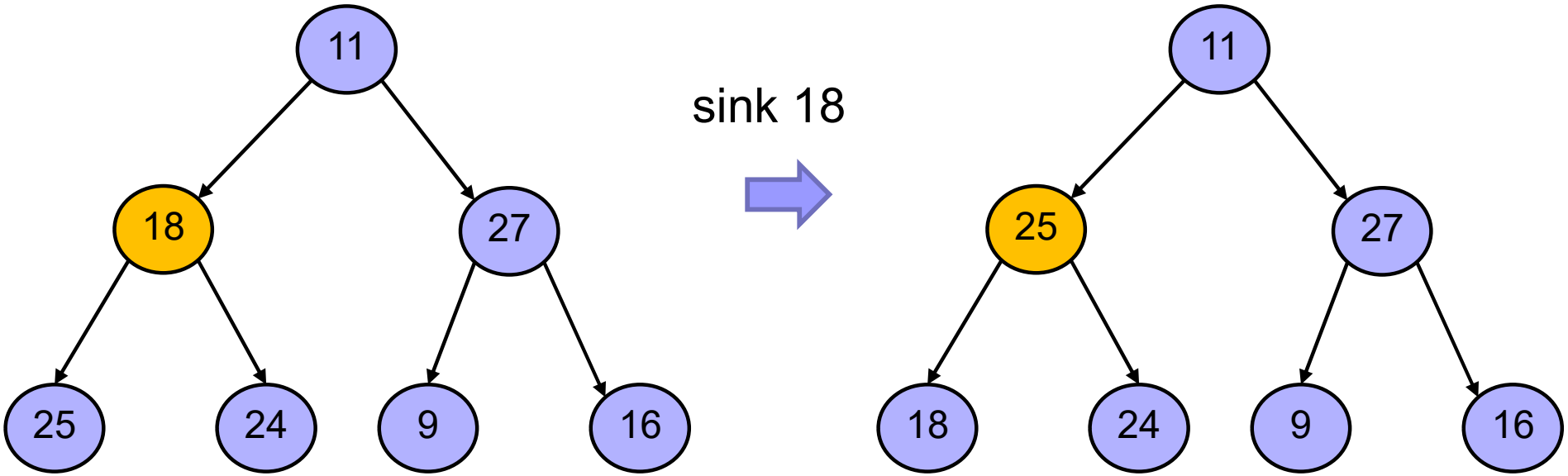


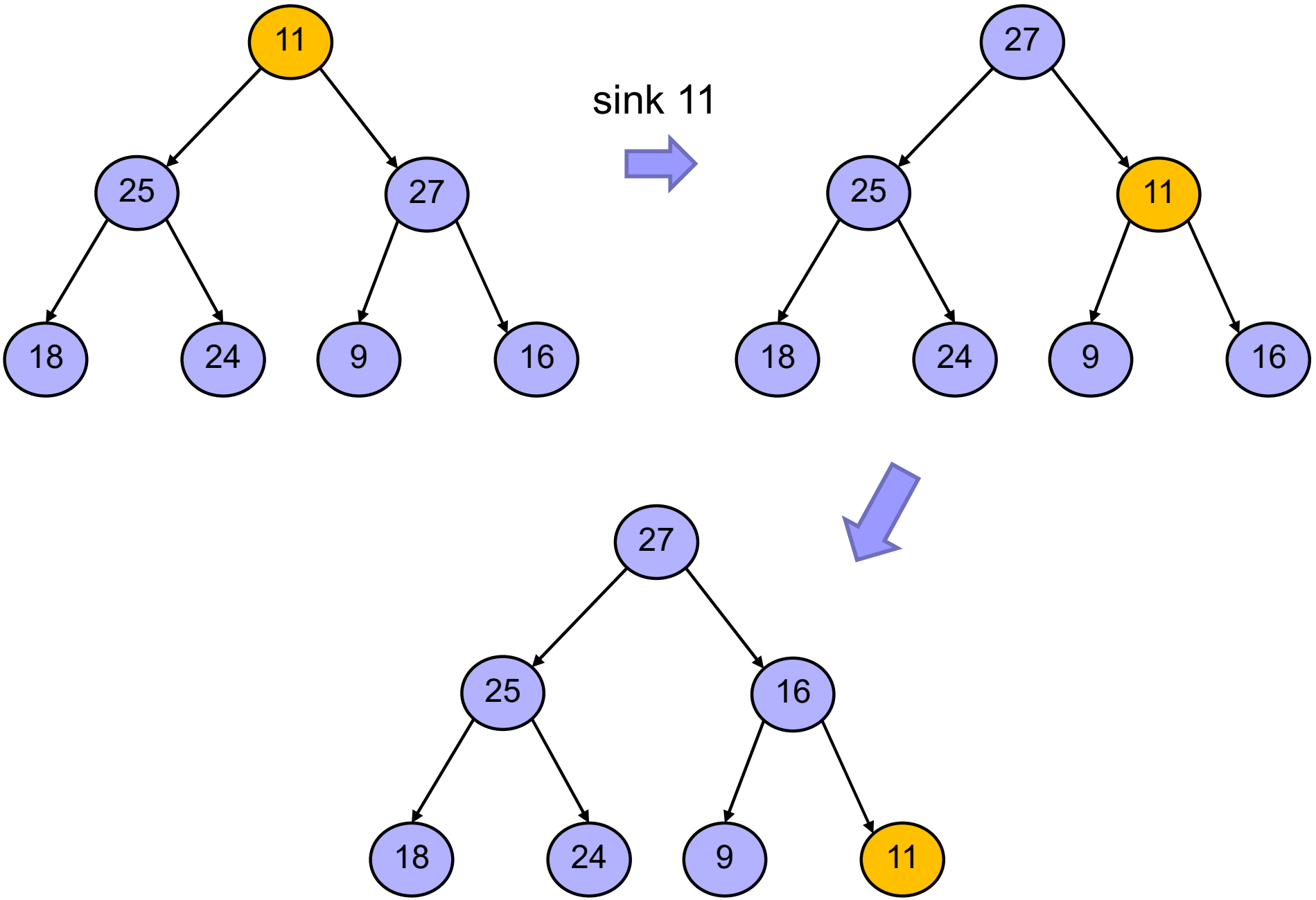
Phase 1: built heap

11 18 9 25 24 27 16



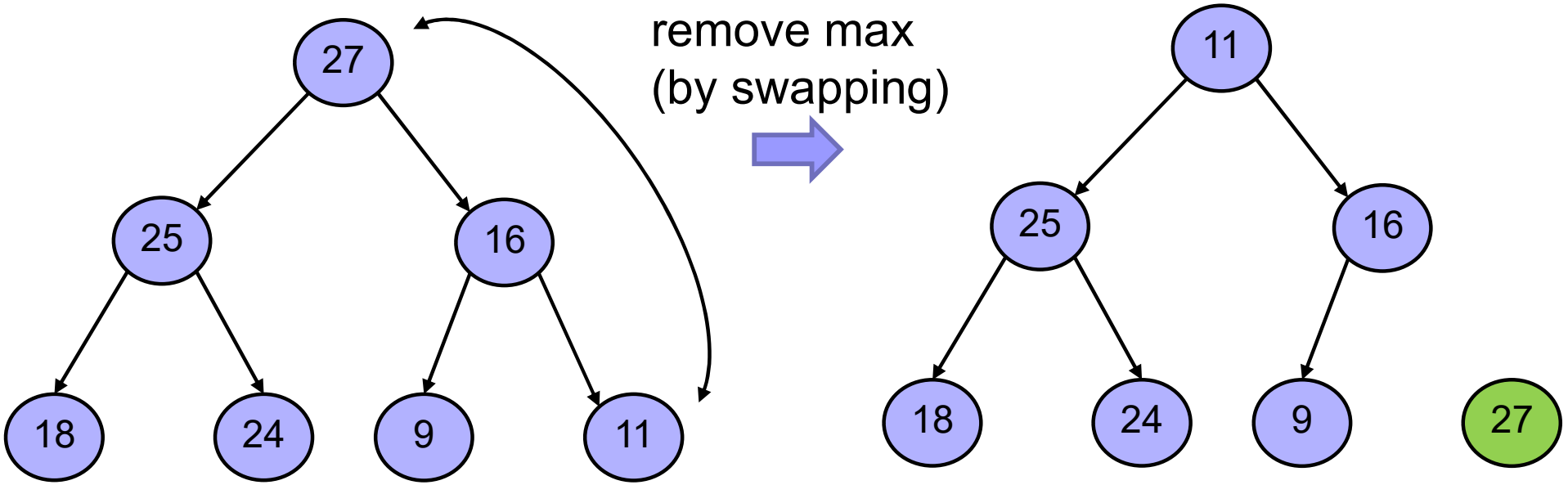
Phase 1: built heap

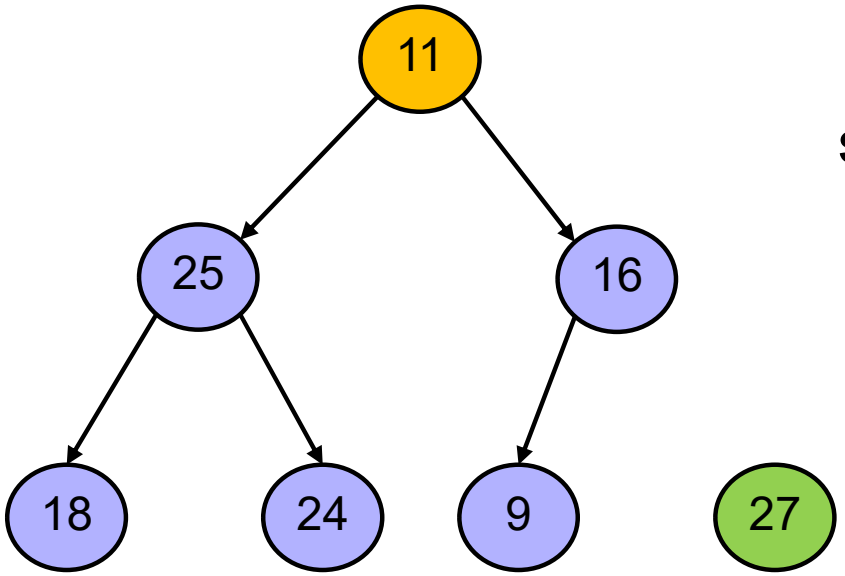



11
25
27
18
24
9
16
sink 11
27
25
11
18
24
9
16
27
25
16
18
24
9
11

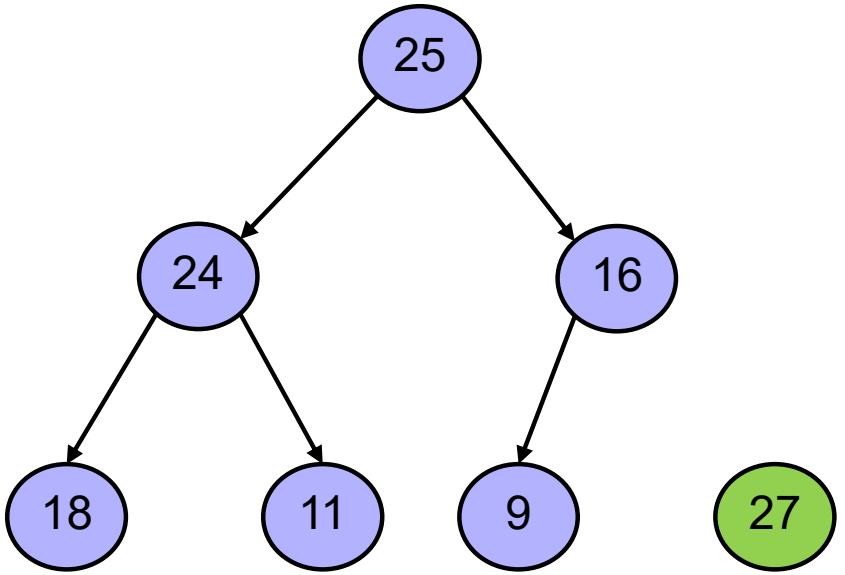
Phase 2: remove max

11 25 16 18 24 9 27

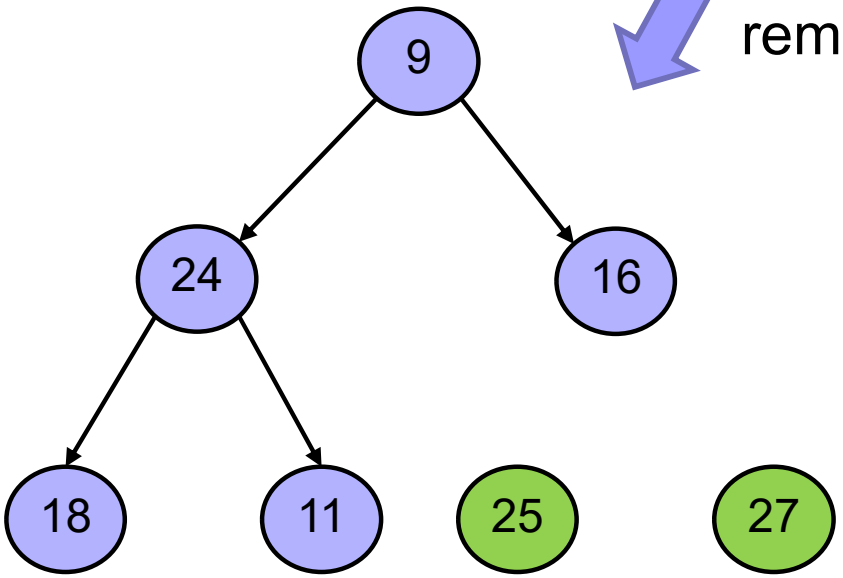




sink 11



remove max



9 24 16 18 11 25 27

Heapsort: Cài đặt

- Phase 1:

```
for (int i = n/2-1; i>=0; i--)
    sink(a, i, n);
```

- Phase 2:

```
    //???
```

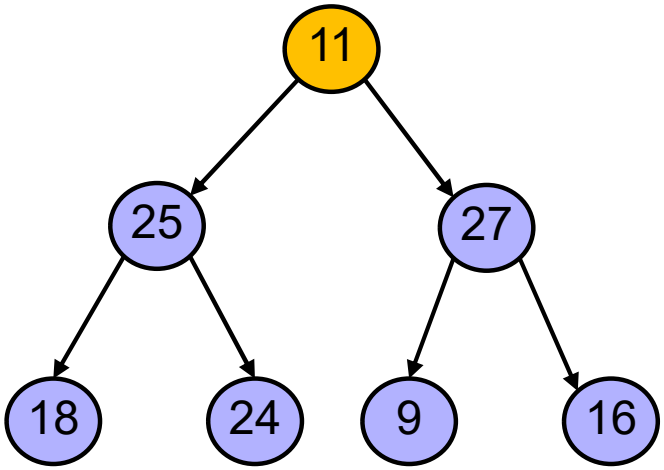
Heapsort: Cài đặt

- Phase 1:

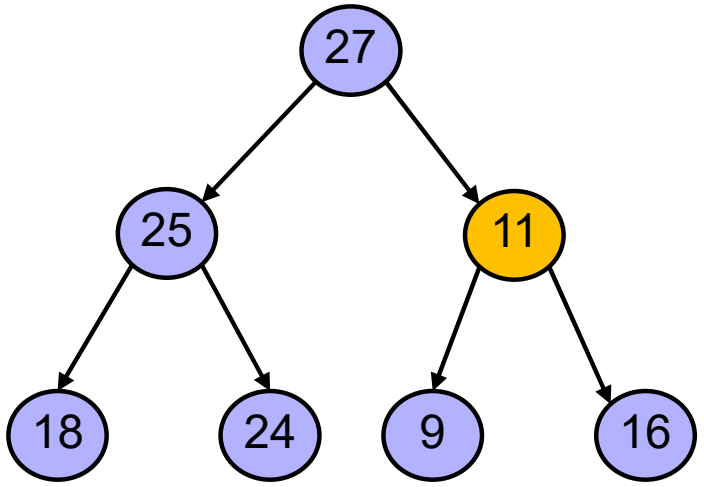
```
for (int i = n/2-1; i>=0; i--)
    sink(a, i, n);
```

- Phase 2:

```
for (int i=n-1; i>0; i--) {
    swap(a, 0, i);
    sink(a, 0, i);
}
```

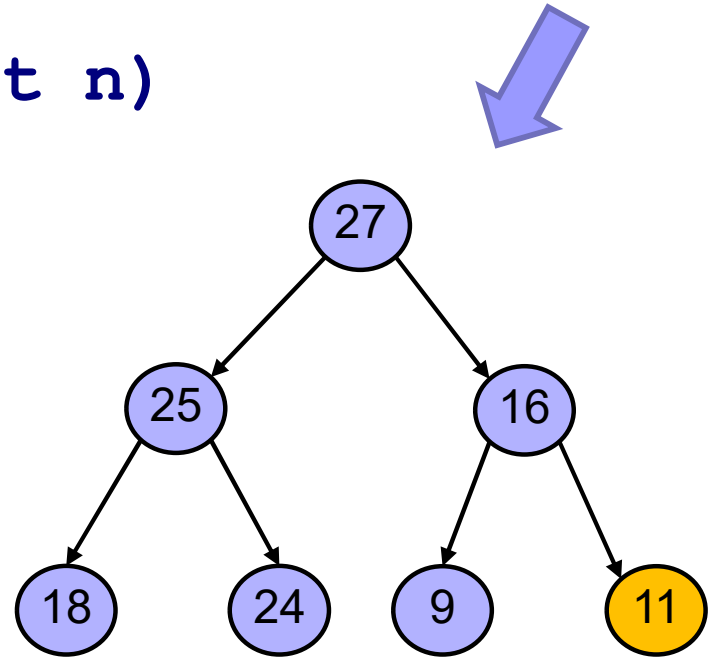


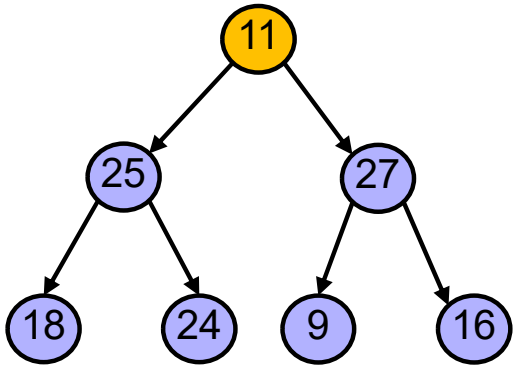
sink 11



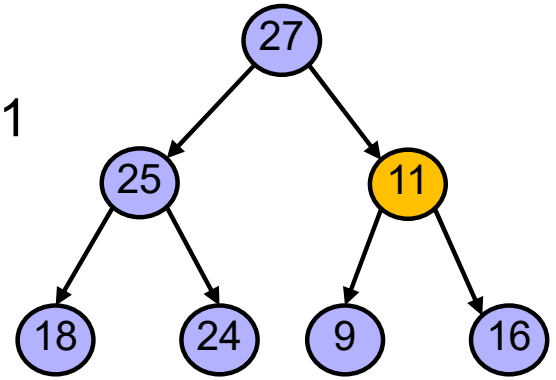

```
void sink(int a[], int k, int n)
{


//???
}
```

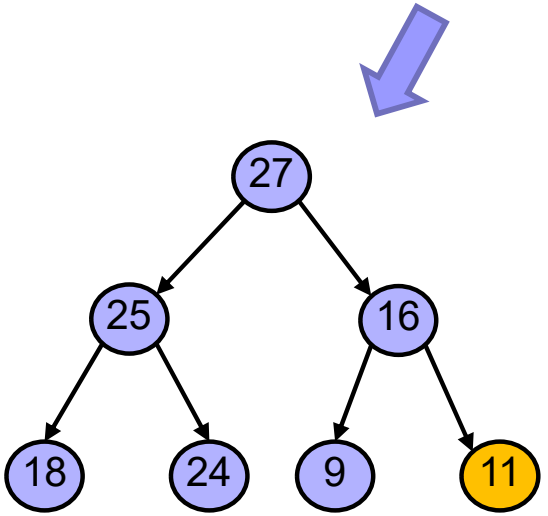




sink 11




```
void sink(int a[], int k, int n) {
  while (k < n/2) {
     int j = 2*k + 1;
     if (j < n-1 && a[j] < a[j+1]) j++;
     if (a[k] >= a[j]) break;
     swap(a, k, j);
     k = j;
  }
}
```



So sánh các thuật toán sắp xếp

	best	average	worst
merge	*?*	*n log n*	*?*
quick	*?*	*?*	*?*
heap	*?*	*?*	*?*

- Phép toán / bộ nhớ?
- Xấu nhất/tốt nhất khi nào?

So sánh các thuật toán sắp xếp

	best	average	worst
merge	*n log n*	*n log n*	*n log n*
quick	*n log n*	*n log n*	n^2
heap	*n*	*n log n*	*n log n*

Mergesort cần dùng mảng phụ
Heapsort đảm bảo trường hợp xấu nhất *O(n log n)*
Quicksort thực tế là thuật toán hiệu quả nhất

Bài tập/thực hành

- Cài đặt hàng đợi ưu tiên bằng binary heap
  - Tạo lớp Heap với các hàm insert, remove_max, …
- Cài đặt thuật toán sắp xếp heapsort
  - Thực nghiệm so sánh với mergesort và quicksort

Chuẩn bị

- Bảng băm (hash table)