INTRODUCTION TO DATA STRUCTURES AND ALGORITHM COURSE

DATA STRUCTURES AND ALGORITHMS

ThS Nguyễn Thị Ngọc Diễm
diemntn@uit.edu.vn

Nội dung

- Các giải thuật sắp xếp nội:

1. Chọn trực tiếp - Selection Sort
2. Đổi chỗ trực tiếp - Interchange Sort
3. Chèn trực tiếp - Insertion Sort
4. Chèn nhị phân - Binary Insertion Sort
5. Quick Sort
6. **Heap Sort**
7. Merge Sort

6. Heap Sort

Từ khóa: Heap Sort

Phân tích: Trong phương pháp chọn trực tiếp (Selection Sort), mỗi lần chọn phần tử cực tiểu theo quan hệ $\Re$ đều không tận dụng được các kết quả so sánh trước đó → độ phức tạp theo phép so sánh là $\mathbf{O(n^2)}$.

→ Tận dụng kết quả so sánh bằng cấu trúc **Heap**

Lịch sử: Sắp xếp Heapsort dựa trên một cấu trúc dữ liệu được gọi là Binary Heap, được giới thiệu bởi J. W. J. Williams vào năm 1964.

6. Heap Sort

Ý tưởng:

- Xây dựng cấu trúc Heap:
  - Là một cây nhị phân hoàn chỉnh
  - Nếu giá trị khóa của nút cha và hai nút con lần lượt là K, K_1, K_2, thì:

$$K_1 \ \mathfrak{R} \ K$$

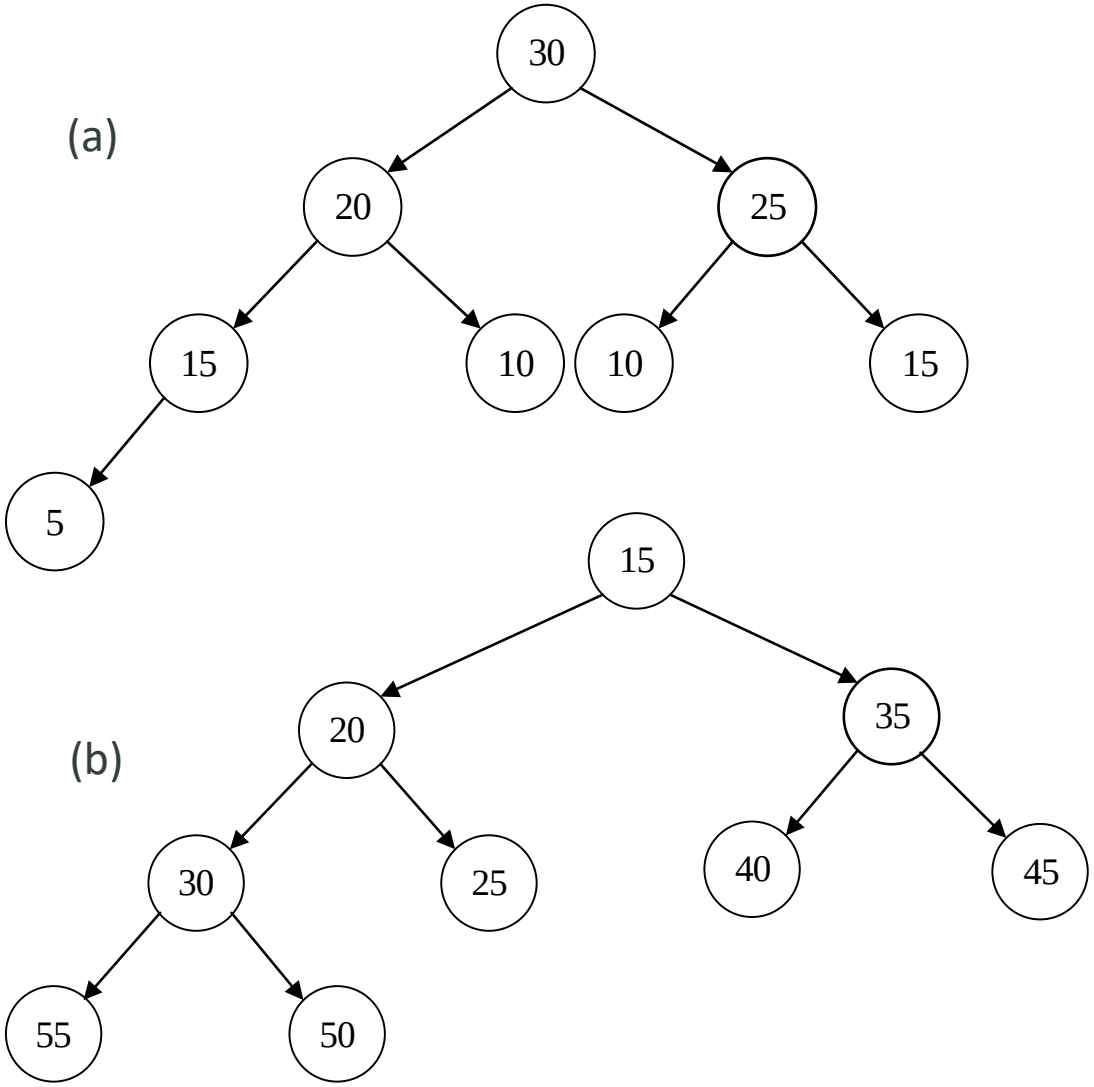
$$K_2 \ \mathfrak{R} \ K$$

Định nghĩa cây Heap

- Cây **Binary Max-Heap** có tính chất: Là cây nhị phân hoàn chỉnh (complete binary tree) và giá trị của một node bất kỳ trên cây sẽ **luôn lớn hơn hoặc bằng** giá trị của các node con của nó (nếu có).
- Cây **Binary Min-Heap** có tính chất: Là cây nhị phân hoàn chỉnh (complete binary tree) và giá trị của một node bất kỳ trên cây sẽ **luôn nhỏ hơn hoặc bằng** giá trị của các node con của nó (nếu có).

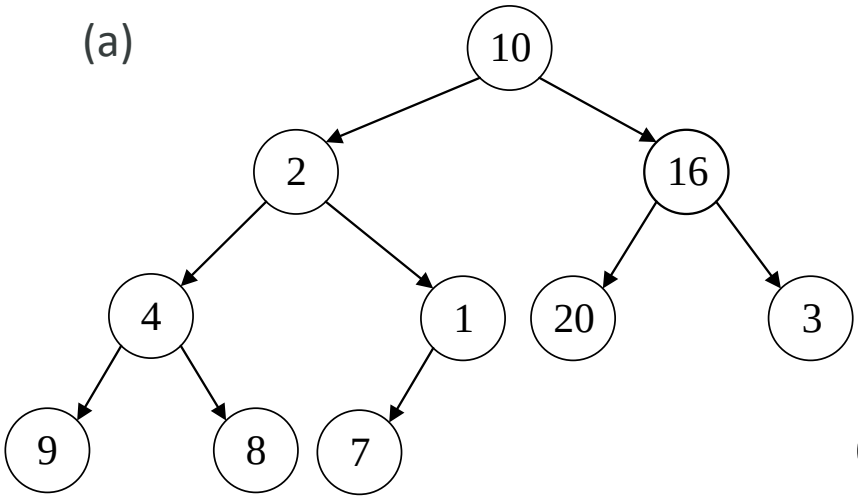
Bài tập:

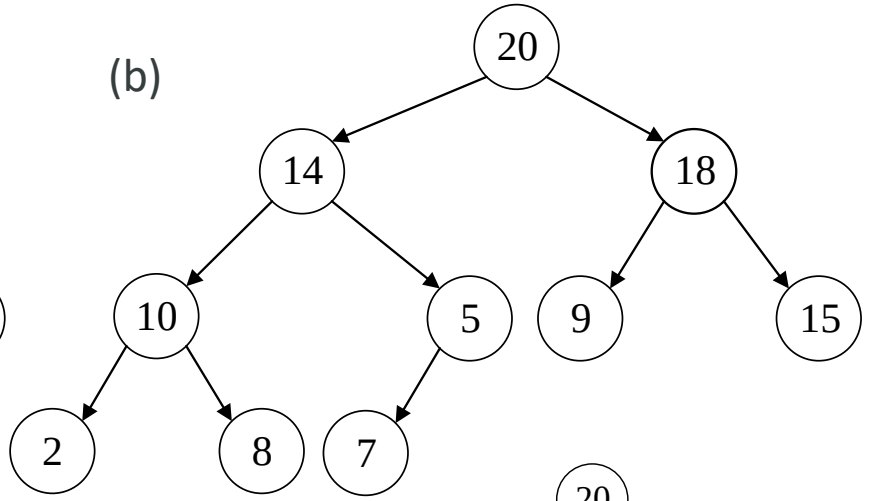
Câu 1. Cho biết cây Min-Heap, Max-Heap tương ứng trong 2 hình bên ?

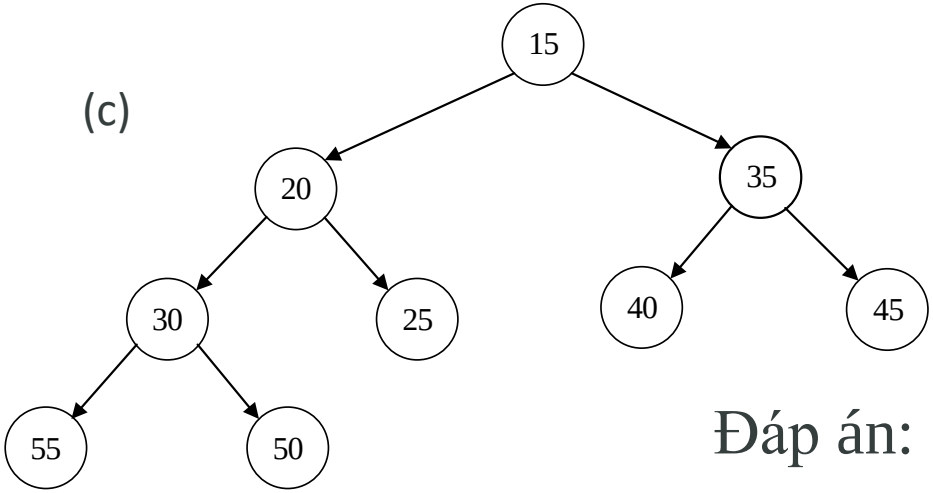


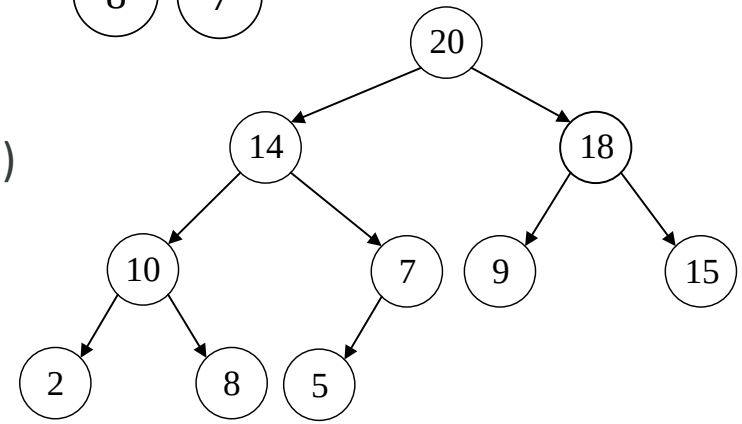
Bài tập: (tt)

Câu 2. Cho biết cây nào trong những hình bên dưới không thỏa tính chất Heap.







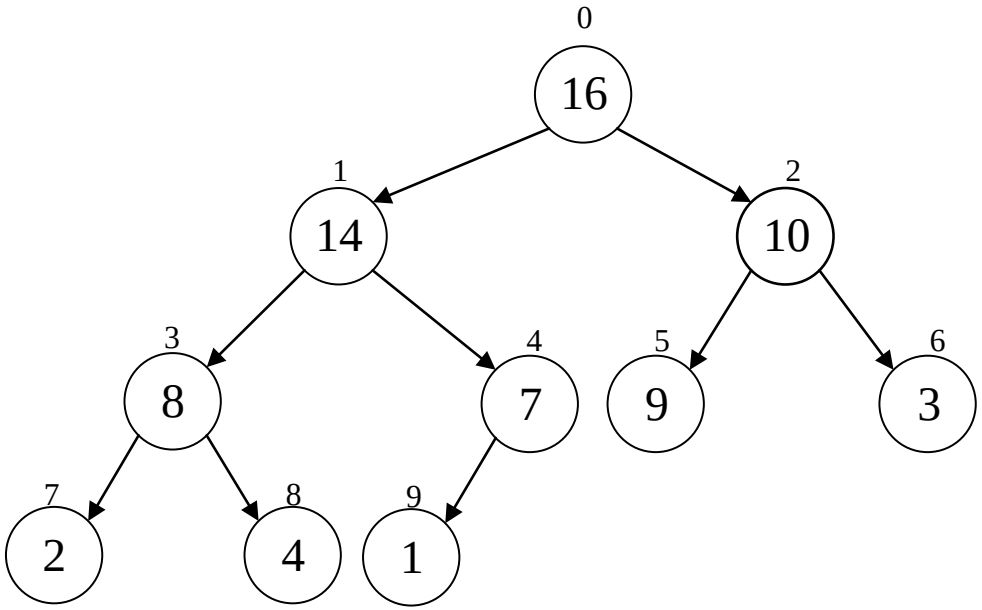


Đáp án: (a), (b)

Array-based Binary Heap

Binary Heap thường được biểu diễn bởi một mảng (array-based binary heap) thay vì một cây nhị phân (pointer-based binary heap).

0	1	2	3	4	5	6	7	8	9
16	14	10	8	7	9	3	2	4	1



Array-based Binary Heap

Gốc của cây A là A[0].

Gọi i là chỉ số của một node bất kỳ trong cây, ta ký hiệu sau:

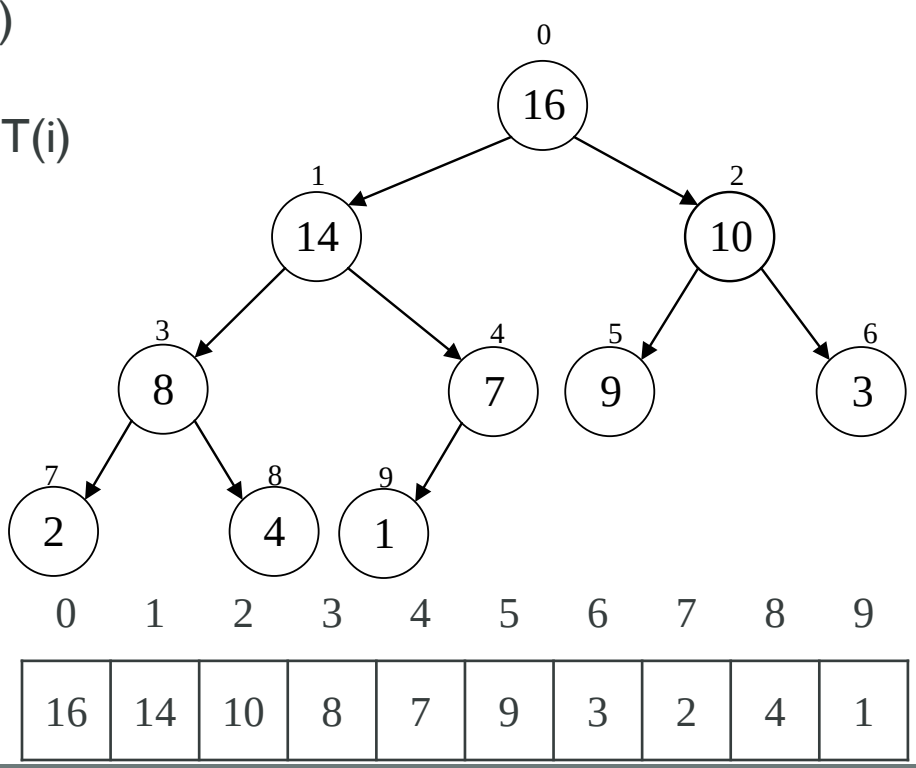
+ Chỉ số của node cha của i gọi là: PARENT(i),

+ Chỉ số của node con trái của i gọi là: LEFT(i)

+ Chỉ số của node con phải của i gọi là: RIGHT(i)

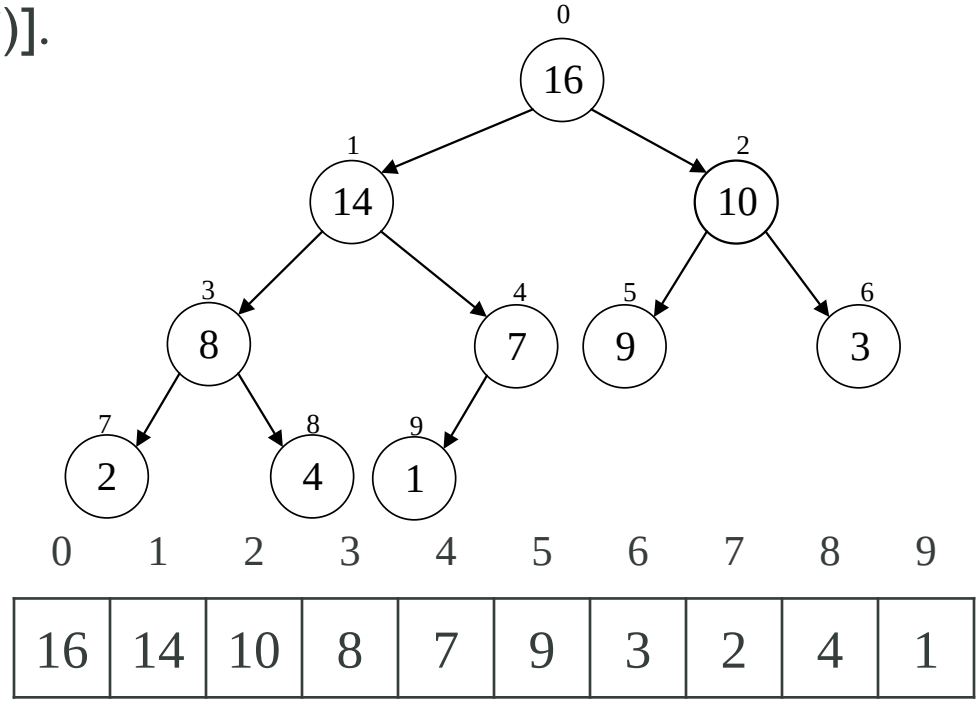
3 chỉ số này được tính qua công thức sau:

```
PARENT(i)    return [(i-1)/2]

LEFT(i)      return 2i+1

RIGHT(i)     return 2i+2
```



Array-based binary heap

- Array-based binary heap phải thỏa mãn tính chất heap (*heap property):*
  - Với Binary Max-Heap: Tất cả các node i (không kể node gốc) thỏa A[i] ≤ A[PARENT(i)].
  - Với Binary Min-Heap: Tất cả các node i (không kể node gốc) thỏa A[i] ≥ A[PARENT(i)].



Bài tập: (tt)

Cho biết dãy nào sau đây thỏa tính chất Heap.

a) 20, 18, 15, 17, 5, 10

b) 20, 18, 15, 17, 15, 17, 14, 5

c) 20, 18, 15, 17, 15, 10, 14, 5

d) 20, 40, 30, 38, 45, 35

e) 20, 40, 30, 48, 45, 35

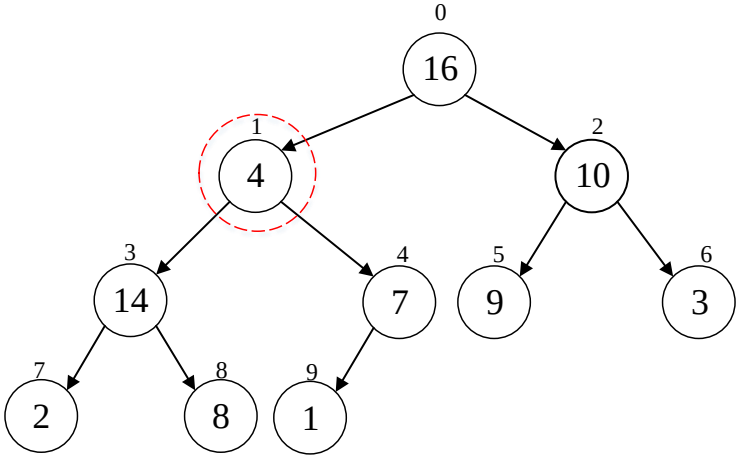
Duy trì tính chất Heap: HEAPIFY hay Re-Heap

- HEAPIFY is to let the value at A[i] "float down" in the heap so that the subtree rooted at i becomes a heap:
- Thuật toán:

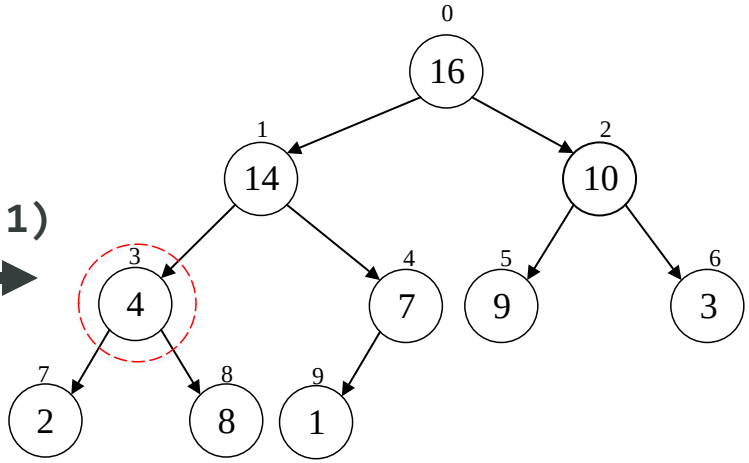
HEAPIFY(A, heap-size, i)

```
1.  l <- LEFT(i)
2.  r <- RIGHT(i)
3.  largest <- i
4.  if l < heap-size[A] and A[l] > A[i]
5.      then largest <- l
6.  if r < heap-size[A] and A[r] > A[largest]
7.      then largest <- r
8.  if largest ≠ i
9.      then exchange A[i] <-> A[largest]
10.     HEAPIFY(A, heap-size, largest )
```

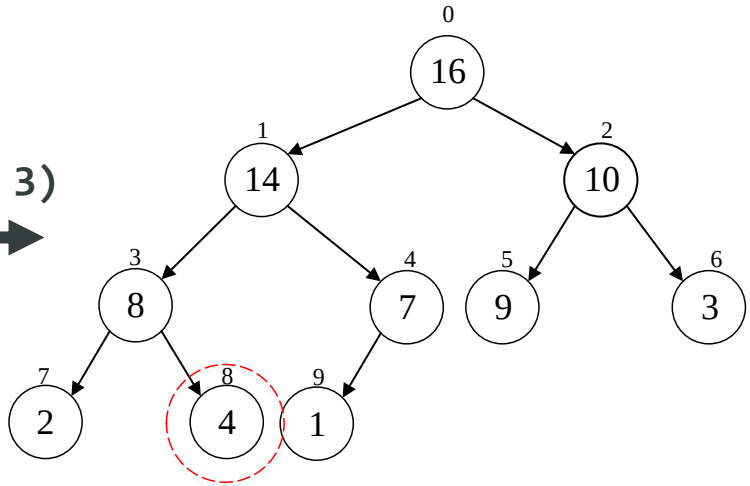
Duy trì tính chất Heap: Ví dụ trên cây Max-Heap



HEAPIFY(A, 10, 1)



HEAPIFY(A, 10, 3)



The action of

HEAPIFY(A, 10, 1)
HEAPIFY(A, 10, 3)
HEAPIFY(A, 10, 8)

where **heap-size[A]=10**

Xây dựng cây Heap: BUILD-HEAP

- **BUILD-HEAP** thực hiện chuyển đổi dãy A[0 .. n-1] thành dãy có tính chất heap (array-based binary heap). Với N là số lượng phần tử của A, ký hiệu length[A]=n. Quá trình đó sẽ sử dụng **HEAPIFY** trên các phần tử của mảng. Trong đó:
  - Các phần tử từ **A[n/2]** → **A[n-1]** là các node lá của cây nên không cần xét.
  - **BUILD-HEAP** sẽ thực hiện **HEAPIFY** tất cả các phần tử còn lại bắt đầu từ dưới lên trên (bottom-up manner). Nghĩa là từ **A[n/2-1] ngược lên A[0]**.

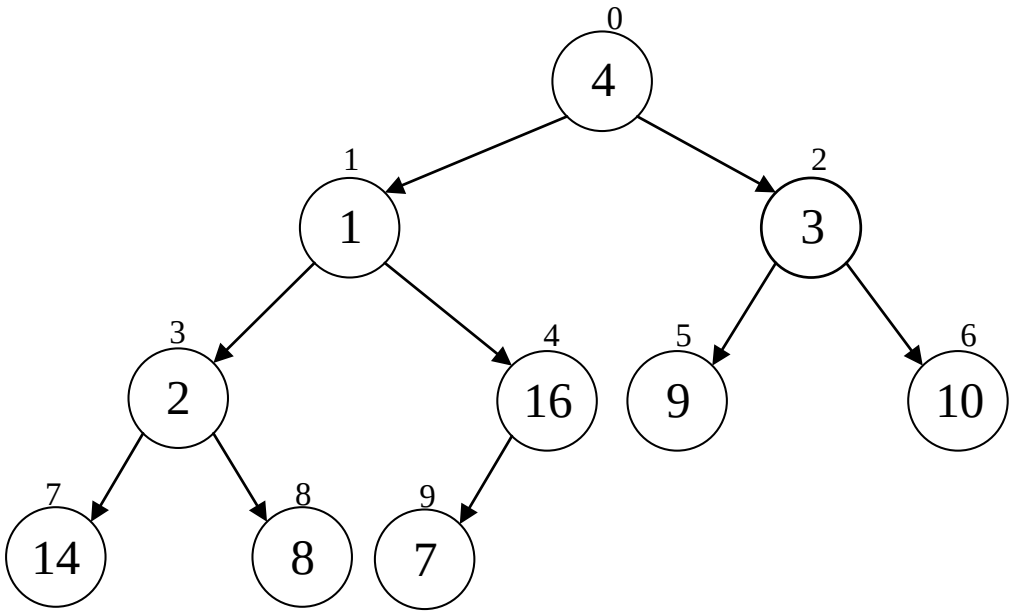
BUILD-HEAP(A)	
1.	heap-size[A] <- length[A]
2.	for i <- length[A]/2-1 downto 0 do
3.	HEAPIFY(A, heap-size, i)

BUILD-HEAP: Ví dụ

Xây dựng cây Heap cho dãy số sau:

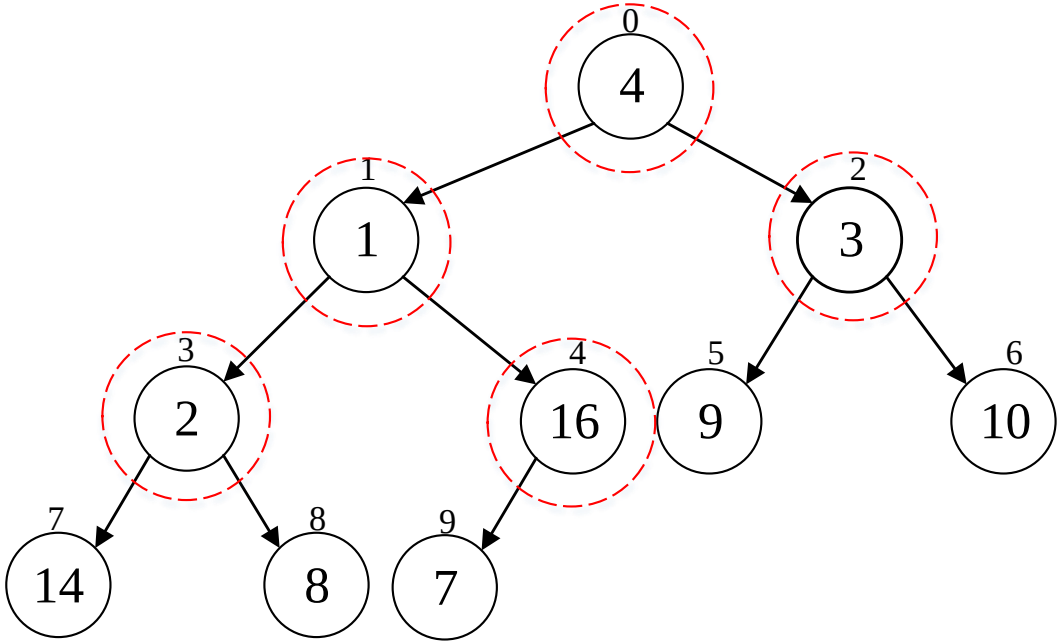
0	1	2	3	4	5	6	7	8	9
4	1	3	2	16	9	10	14	8	7

Ghi lại dãy số dưới dạng cây:

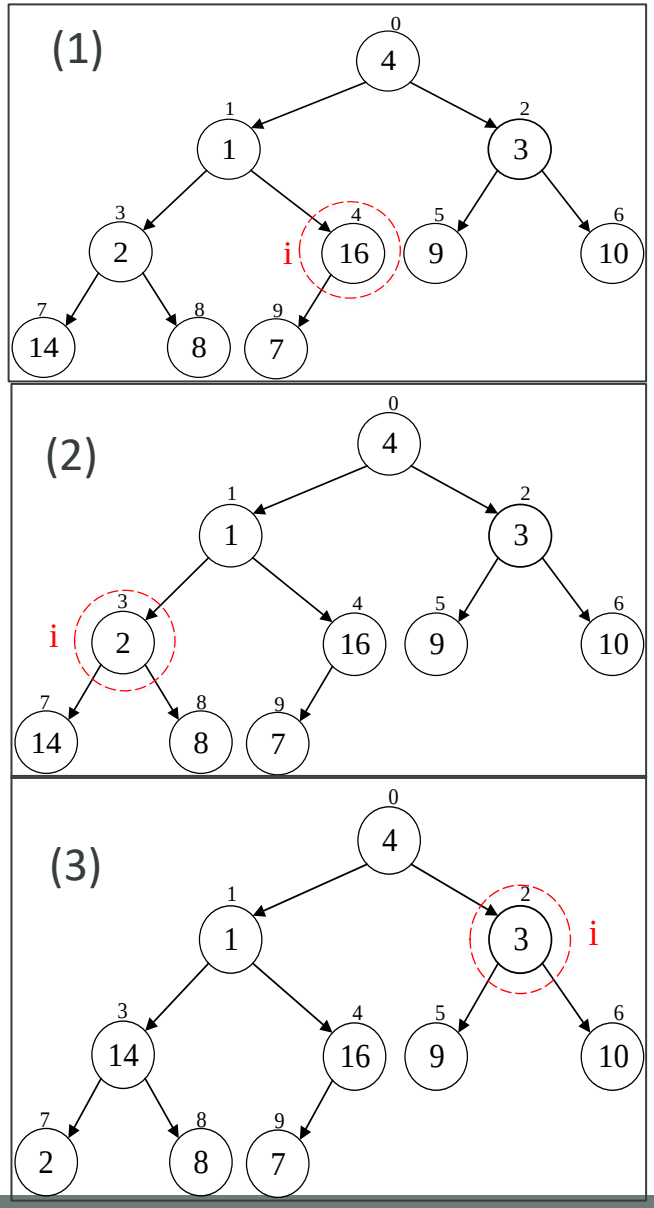


BUILD-HEAP: Ví dụ (tt)

- Các giá trị trên dãy cần xem xét trong quá trình BUILD-HEAP lần lượt là 16, 2, 3, 1, 4:



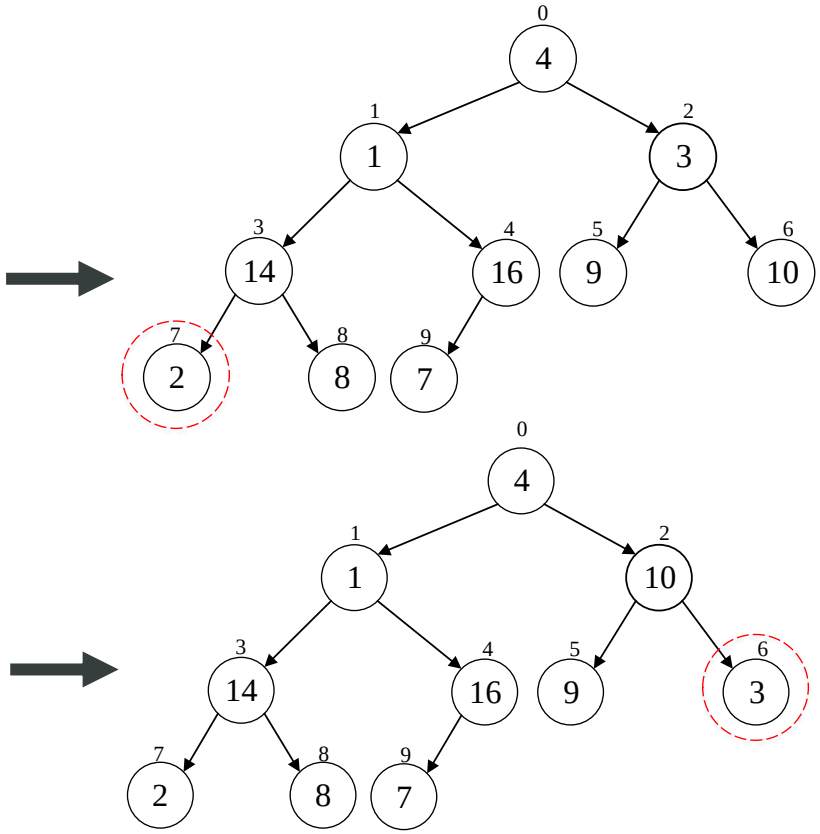
BUILD-HEAP: Ví dụ (tt)



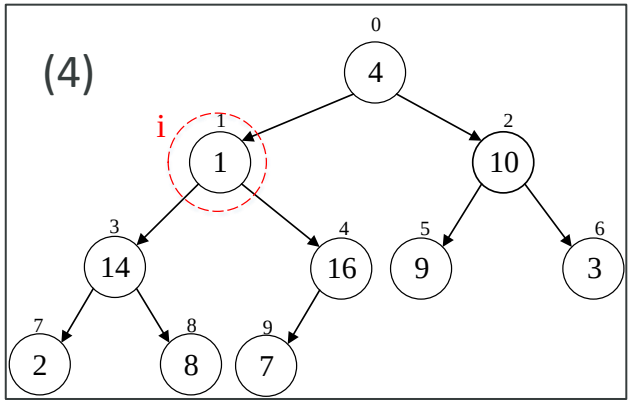
The operation of **`BUILD-HEAP`**, before the call **`HEAPIFY`**:

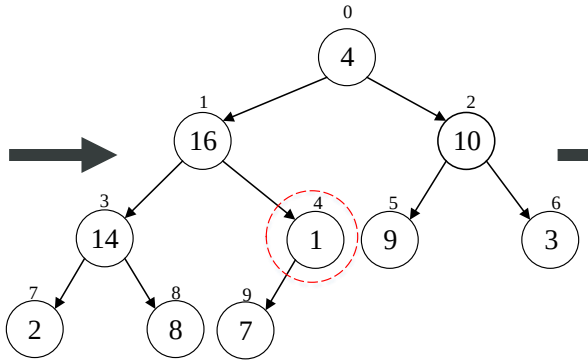
(1): A 10-element input array **`A`** and the binary tree it represents. The figure shows that the loop index **`i`** points to node 4 before the call **`HEAPIFY(A, i)`**.

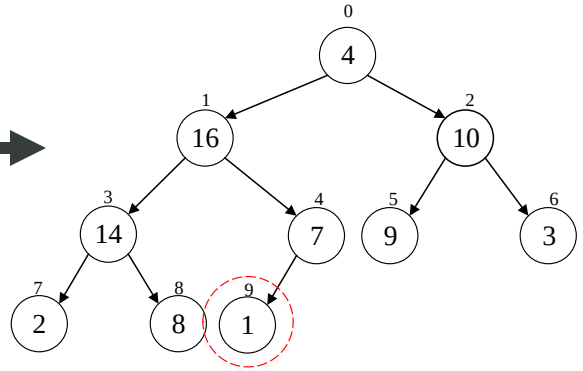
(2): The data structure that results. The loop index **`i`** for the next iteration points to node 3.

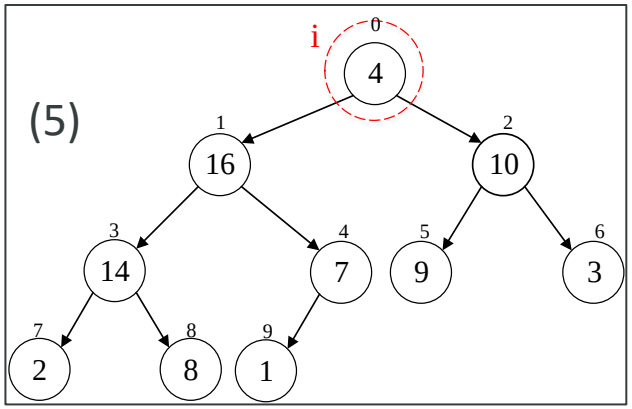


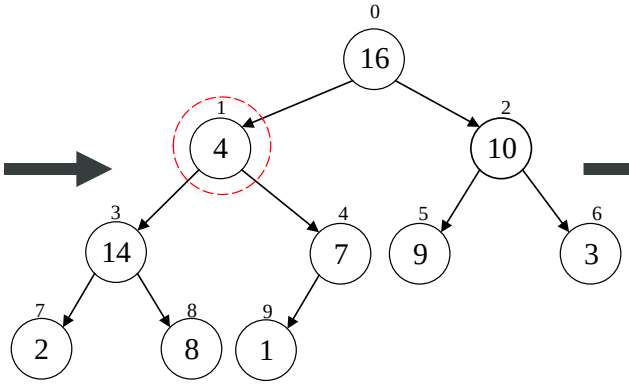
BUILD-HEAP: Ví dụ (tt)

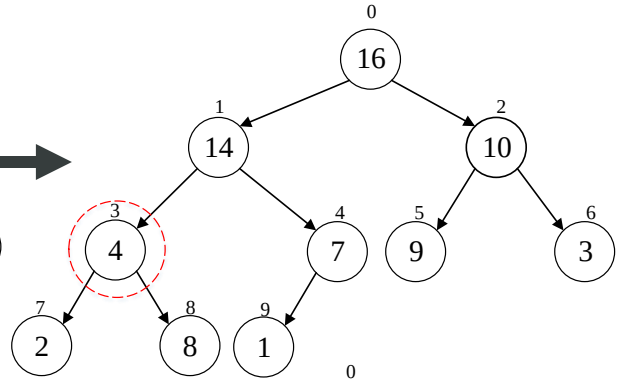


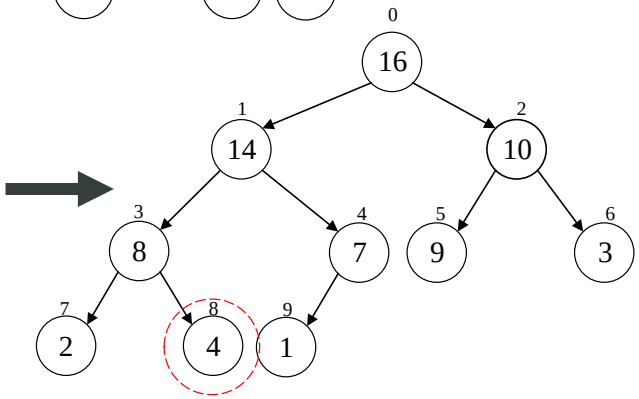










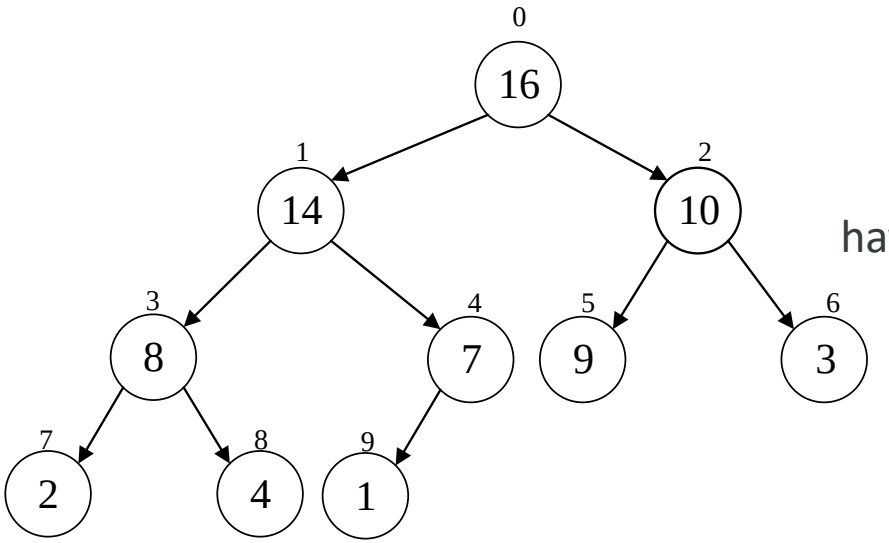


(3)-(5): Subsequent iterations of the for loop in **BUILD-HEAP**. Whenever **HEAPIFY** is called on a node, the two subtrees of that node are both heaps.

BUILD-HEAP: Ví dụ (tt)

	0	1	2	3	4	5	6	7	8	9
Dãy ban đầu:	4	1	3	2	16	9	10	14	8	7

The heap after **`BUILD-HEAP`** finishes :



	0	1	2	3	4	5	6	7	8	9
hay:	16	14	10	8	7	9	3	2	4	1

Heap Sort: Ý tưởng

- Bước 1: Sử dụng BUILD-HEAP để tạo dãy Heap cho dãy A[0..n-1] với n là chiều dài của dãy.
- Bước 2: Sau đó thực hiện đưa A[0] về cuối dãy bằng cách hoán vị với phần tử cuối cùng của dãy heap đang xét. Xét dãy heap mà đã loại bỏ phần tử cuối cùng. Tuy nhiên phần tử gốc A[0] của dãy heap hiện tại sẽ không còn giữ tính chất heap. Vì vậy ta cần HEAPIFY lại phần tử A[0] để đưa dãy hiện tại về dãy có tính chất heap. Thực hiện lặp lại các thao tác ở bước 2 trên cho tới khi dãy heap còn 1 phần tử.

Heap Sort: Thuật toán

- The heapsort algorithm then repeats this process for the heap of size **n-1** down to a heap of size 2.

HEAPSORT(A)	
Đầu vào: Mảng A chưa có thứ tự Đầu ra : Mảng A đã có thứ tự	
1.	BUILD-HEAP(A)
2.	for i <- length[A]-1 downto 1 do
3.	exchange A[0] <-> A[i]
4.	heap-size[A] <- heap-size[A]-1
5.	HEAPIFY(A, heap-size, 0)

- The **HEAPSORT** procedure takes time O(nlogn), since the call to **BUILDHEAP** takes time O(n) and each of the **n-1** calls to **HEAPIFY** takes time O(logn).

Heap Sort: Độ phức tạp

Trường hợp	Độ phức tạp
Tốt nhất	$nlog_2n$
Trung bình	$nlog_2n$
Xấu nhất	$nlog_2n$

HeapSort: Code C/C++

```
void Heapify(int a[], int heapSize, int i) {
    int childLeft  = i*2+1;
    int childRight = i*2+2;
    int max = i;
    if(childLeft<heapSize && a[max]<a[childLeft])
            max=childLeft;
    if(childRight<heapSize && a[max]<a[childRight])
            max= childRight;
    if(max != i) {
        swap(a[max], a[i]);
        Heapify(a, heapSize, max);
    }
}
```

HeapSort: Code C/C++ (tt)

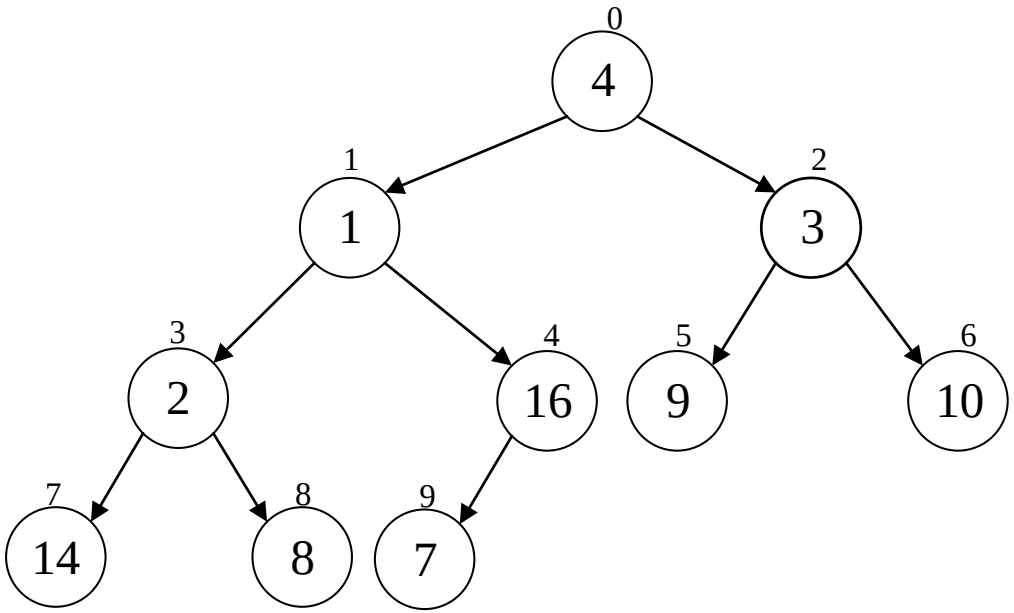
```
void buildHeap(int a[], int n) {
    int heapSize = n;
    for(int i = n/2-1; i>=0; i--)
            Heapify(a, heapSize, i);
}

void HeapSort(int a[], int n) {
    int heapSize;
    heapSize = n;
    buildHeap(a, n);
    for(int i = n-1; i>=1 ; i--) {
        swap(a[0], a[i]);
        heapSize -= 1;
        Heapify(a, heapSize, 0);
    }
}
```

Heap Sort: Minh họa

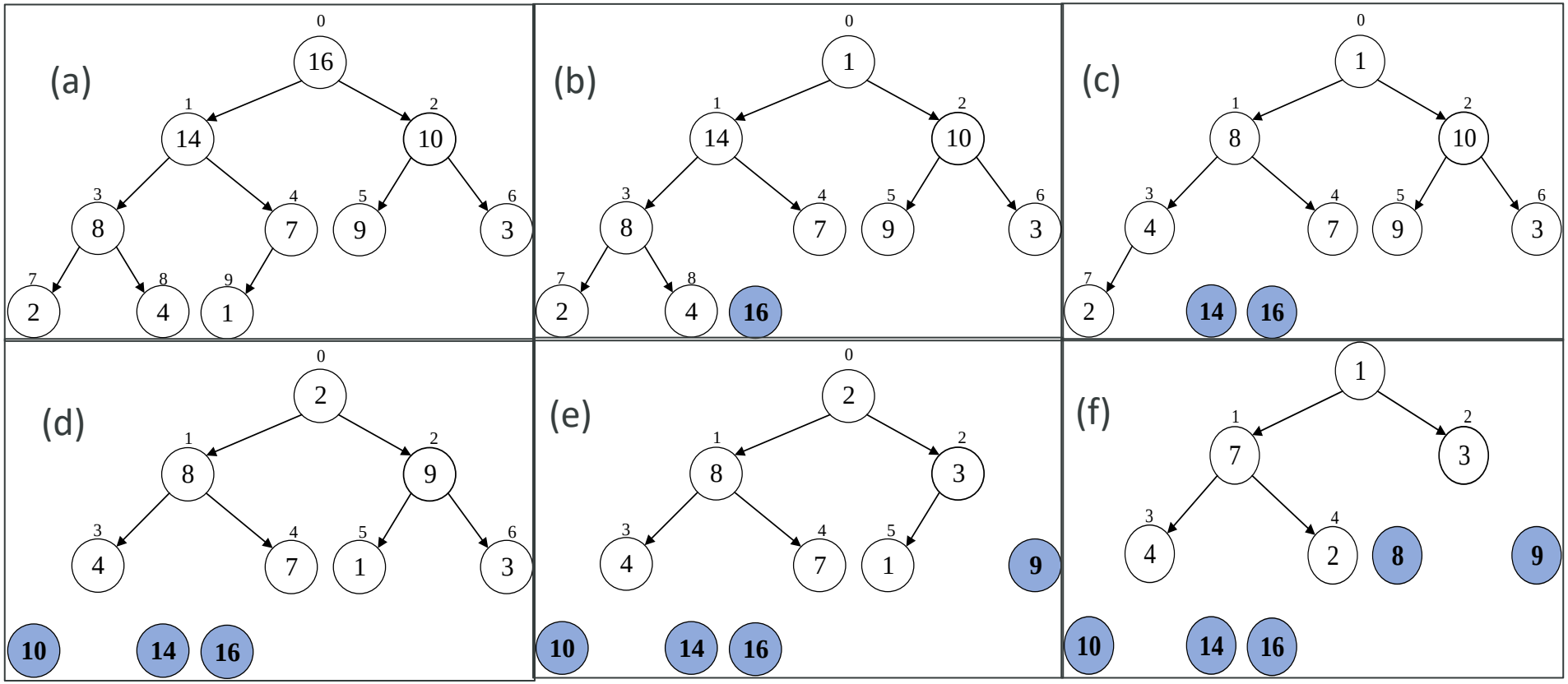
Sắp xếp dãy số sau bằng Heap Sort:

0	1	2	3	4	5	6	7	8	9
4	1	3	2	16	9	10	14	8	7

Ghi lại dãy số dưới dạng cây:



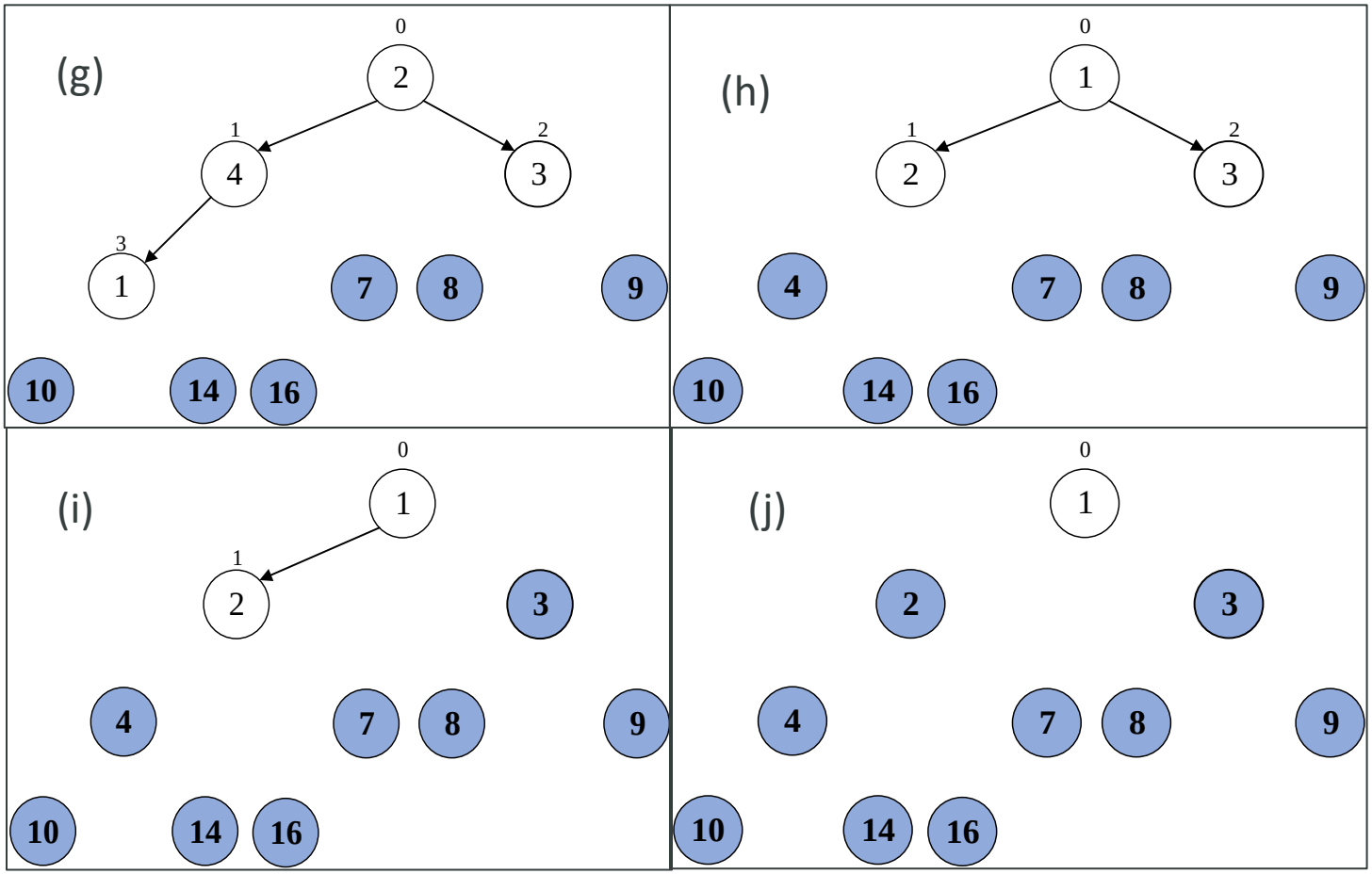
Heap Sort: Minh họa (tt)



Hình (a): The heap just after it has been built by **`BUILD-HEAP.`**

Hình (b)-(j): The heap just after calling of **`HEAPIFY`** in line 5.

Heap Sort: Minh họa (tt)



1	2	3	4	7	8	9	10	14	16

The resulting sorted array A.

Bài tập về nhà

1. Is the sequence (23, 17, 14, 6, 13, 10, 1, 5, 7, 12) a heap?
2. Illustrate the operation of HEAPIFY(A,3,) on the array A= (27, 17,3,16, 13, 10, 1,5,7, 12,4,8,9,0).
3. The code for HEAPIFY is quite efficient in terms of constant factors, except possibly for the recursive call in line 10, which might cause some compilers to produce inefficient code. Write an efficient HEAPIFY that uses an iterative control construct (a loop) instead of recursion.
4. Illustrate the operation of BUILD-HEAP on the array A = (5, 3, 17, 10, 84, 19, 6, 22, 9)
5. Show that the running time of heapsort is Ω(nlgn).
6. Illustrate the operation of HEAPSORT on the array A = (5, 13, 2, 25, 7, 17, 20, 8, 4)

Chúc các em học tốt!