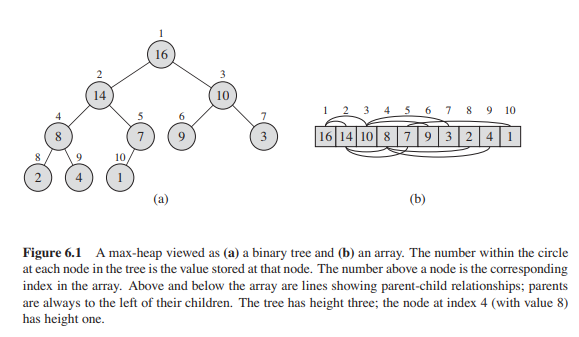
# Heap\_Sort

1. Cấu trúc dữ liệu ***heap*** là một đối tượng mảng, chúng ta có thể xem như là một cây nhị phân gần hoàn chỉnh. Mổi node của cây tương ứng với một phần tử của mảng. Heap có 2 tính chất như sau:

* Tính cấu trúc : Tất cả các mưc (level) đều được điền đầy ngoại trừ mức thấp nhất, tại mức thấp nhất thì chúng được điền từ trái qua phải.
* Tính có thứ tự hay còn gouj là tính chất đống: Parent(x) > x.

1. Các thành phần của cấu trúc Heap.

* A.length (Số phần tử trong mảng A ), A.heap-size (Số phần tử trong heap được chứa bên trong mảng A ).



* Tại vị trí i của mảng:

*Parent(i) = return i/2*

*Left children = return 2i*

*Right children = return 2i+1*

Trên phần lớn các máy tính, *Left children* có thể tính toán 2i trong một thực thi đơn giản bẳng việc dịch bit nhị phân của i sang trái 1 bit. Vs *Right children* thì dịch sang trái 1 bit và cộng thêm 1 ở bit thấp nhất . Còn tìm cha thì dịch sang phải 1 bit.

* Height : height của một node là số cạnh bắt đấu từ node và đi xuốt theo một đường đơn giản dài nhất xuống gặp lá. Và ta định nghĩa height của heap là height của root.

1. Phân loại Heap

Có 2 loại heap nhị phân: max-heaps và min-heaps

* Max-heaps : Với mọi node i trừ root thì giá trị của cha nó luôn lớn hơn. A[Parent(i)]>A[i]
* Min-heaps : Ngược lại vs max-heaps.

1. Các phép toán đối với Heap

* MAX-HEAPIFY : Khôi phục tính chất đống.
* BUILD-MAX-HEAP : Xây dựng một heap từ một mảng.
* HEAPSORT
* MAX-HEAP-INSERT, HEAP-EXTRACT-MAX, HEAP-INCREASE-KEY, HEAP-MAXIMUM

1. Maintaining the heap property

Để maintaining the max-heap property, chúng ta gọi tới MAX\_HEAPIFY. Nó thì input một mảng A và index i bên trong mảng này. Khi được gọi MAX-HEAPYIFY giả định rằng cây nhị phân đã có root, tại LEFT(i) và RIGHT(i) là max-heaps, nhưng A[i] có lẽ bị nhỏ hơn phần tử con của nó, do đó nó vi phạm max-heap property. MAX-HEAPIFY đưa giá trị A[i] trôi xuống dưới trong max-heap vì vậy mà cái subtree rooted này tại index I thì tuân theo max-heap property.

MAX-HEAPIFY(A,i)

1 l=LEFT(i)

2 r=RIGHT(i)

3 if(l<=A.heap-size && A[i]<A[l])

4 Largest=l

5 Else

6 Largest=i

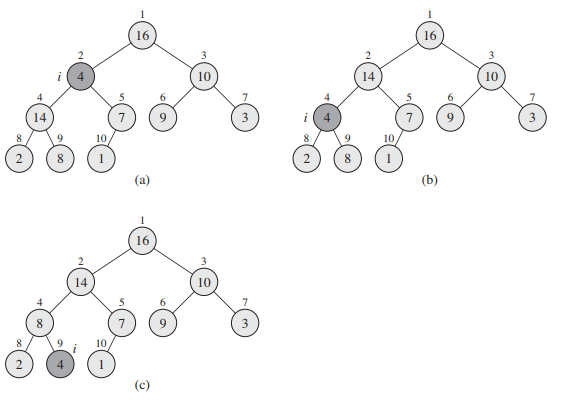
7 if(r<=A.heap-size && A[largest]<A[r])

8 Largest=r

9 if(i!=Largest)

10 Swap(A[i],A[Largest])

11 MAX-HEAPIFY(A,Largest)



1. Building a heap

Chúng ta có thể sử dụng thủ tục MAX-HEAPIFY để biến đổi một mảng A[1..n] vào bên trong một max-heap.

Điều kiện để sử dụng MAX-HEAPIFY là tại node i thì các root tại RIGHT(i) và LEFT(i) phải tuân thủ max-heap. Chính vì vậy thủ tục MAX-HEAPIFY sẽ áp dụng trên A.length/2 downto 1. Vì trong một mảng thì A[(n/2+1)..n] là các lá, mà nếu là lá thì nó mặc định là thỏa mãn max-heap và ta chỉ việc maintain bắt đầu với các node là Parent của chúng.

BUILD-MAX\_HEAP(A)

1. A.heap-size=A.length
2. For i=A.length/2 downto 1
3. MAX-HEAPIFY(A,i)

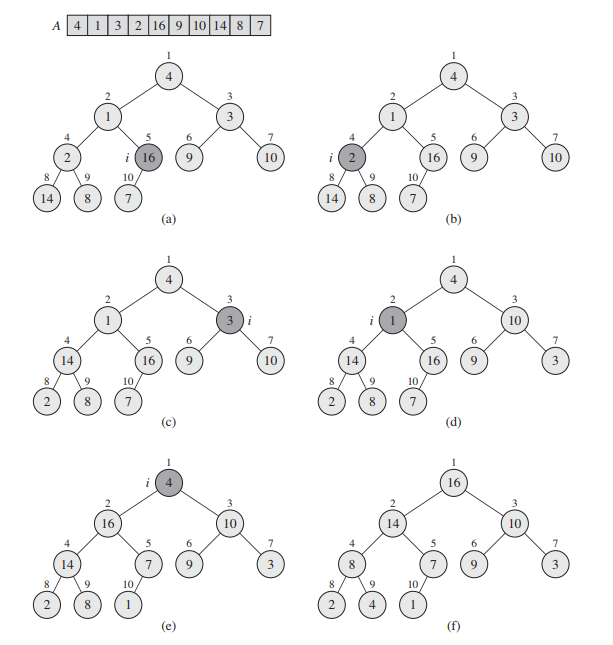
Ta sẽ giải thích tính chính xác của BUILD-MAX-HEAP

Tại thời điểm bắt đầu của mỗi vòng lặp trong dòng 2-3 của mã gỉ trên, mỗi node i+1, i+2,….,n là root của một max-heap.

**Initialization:** Ưu tiên tới lần lặp đầu tiền của vòng lặp, i=n/2. Mỗi node [n/2]+1, [n/2]+2,....,n là một lá và nó tuân thủ max-heap.

**Maintenance:** Nhìn vào mỗi lần lặp maintains vòng lặp bất biến, để ý thấy rằng các children của node *i* thì được đánh số cao hơn *i* , Bởi vòng lặp bất biến, vì vậy chúng là 2 root max-heap. Cái này thỏa mãn điều kiện cần để gọi MAX-HEAPIFY để làm node *i* là một max-heap root.

**Termination:** Tại thời điểm chấn dứt, i=0. Bởi vòng lặp bất biến, mỗi node 1,2,…,n là root của max-heap.



1. The heapsort algorithm

Để làm việc sort này thì ta sử dụng BUILD-MAX-HEAP để xây dựng một max-heap trên mảng đầu vào A[1..n]. Khi đó phần tử lớn nhất của mảng thì được chứa tại root A[1], chúng ta có thể nó vào bên trong vị trí cuối cùng chính xác bằng việc swap nó với A[n]. Nếu chúng ta ngắt bỏ node n từ heap, sau khi thay đổi chúng ta có một new root nhưng nó thì vi phạm quy tắc của max-heap nên việc còn lại ta chỉ cần gọi MAX-HEAPIFY(A,1). Thuật toán này lặp đi lặp lại xử lý này cho max-heap với size n-1 down to size 2.

HEAPSORT(A)

1. BUILD-MAX-HEAP(A)
2. For i=A.length downto 2
3. Swap(A[1],A[i])
4. A.heap-size—
5. MAX-HEAPIFY(A,1)
6. Priority queues

Một trong những ứng dụng phổ biến của heap là hàng đợi ưu tiên hiệu quả. Với heap hàng đợi ưu tiên có 2 form là max-piority queues và min-piority queues.

Một piority queues (hàng đợi ưu tiên) là một cấu trúc dữ liệu cho việc maintaining một bộ S của các elements, mỗi element có một giá trị ràng buộc gọi là ***key*** . Một max-pq hỗ trợ các hoạt động sau:

INSERT(S,x) thêm một phần tử x vào trong S

MAXIMUM(S) trả về element có key lớn nhất.

EXTRACT-MAX(S) xóa và trả về element có key lớn nhất.

INCREASE-KEY(S,x,k) tăng giá trị của element với key là x tới giá trị mới k

Trong các ứng dụng, chúng ta có thể sử dụng mã-pq để lên kế hoạch các công việc trên máy tính được chia sẻ. max-pq