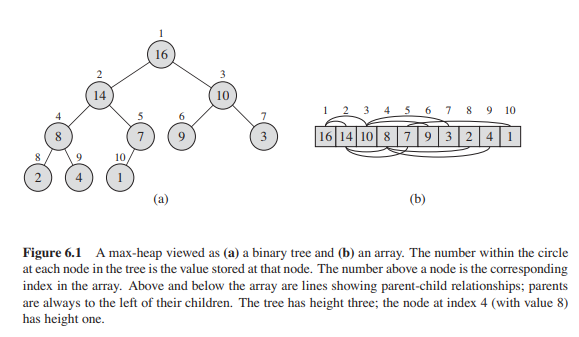
# Heap\_Sort

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)

