TUGAS 1 : HEAP SORT

TUGAS PENGGANTI

PRAKTIKUM ANALISIS ALGORITMA

SEMESTER 4 TAHUN AJARAN 2018/2019

DISUSUN OLEH

TIVANI SHAKILLA ERVI

140810170014

KELAS B

DEPARTEMEN ILMU KOMPUTER

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

JATINANGOR

2019

- **Algoritma Heap Sort**

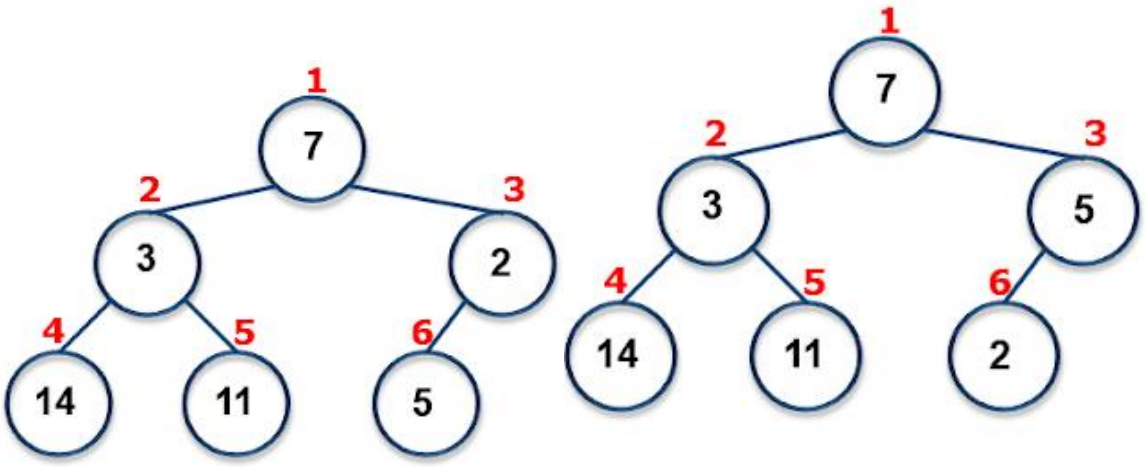
  Heap sort merupakan salah satu metode pengurutan data dari suatu array yang digambarkan menjadi sebuah pohon atau tree dan nilai pada masing-masing indeksnya akan diurutkan.

  Pada metode ini memiliki 3 bagian, yaitu Node, Root, Link, dan Leaf. Apa sih itu? Node adalah indeks yang berada pada array, root adalah node awal pada tree, link adalah sebuah garis yang menyambungkan antara node satu ke node lainnya, dan Leaf adalah node yang tidak memiliki anak atau tidak memiliki node turunan. Adapun proses dari Heap Sort :

  - Pembentukan Heap
  - Pengurutan Data pada Heap
  - Pada pembentukan Heap adapun cara-caranya yaitu dengan,

7	3	2	14	11	5
1	2	3	4	5	6

Dari sebuah array dibuat menjadi Complete Binary Tree, lalu lakukan proses pengurutan secara max heap dengan cara banyaknya simpul dibagi dua untuk mencari nilai tengah dari sebuah array, sebagai contoh N = 6, Tengah = 6/2 = 3. Lalu lakukan reorganisasi pada simpul atau node ke-3. Dengan cara jika angka yang sekarang dibandingkan dengan angka selanjutnya yang ada di node turunannya itu lebih kecil maka tukar posisi.



Barulah dapat kita lakukan pengurutan data heap dengan syarat :

1. Binary Tree dalam keadaan Max Heap, lalu
2. Hapus atau "Pecat" root dan tukarkan dengan simpul pada posisi terakhir.
3. Banyaknya simpul dikurangi 1
4. Jika n lebih dari 1, maka lakukan reorganisasi heap
5. Lakukan langkah ke-2 hingga ke-5 sampai n = 0

- **<u>Kompleksitas Waktu dan Big-O</u>**

  Algoritma pengurutan Heap Sort merupakan salah satu metode pengurutan tercepat setelah Merge Sort dan Quick Sort dengan kompleksitas **O(n log n)**

  Pseudo-code

```
BUILD-HEAP(A)
   heapsize := size(A);
   for i := floor(heapsize/2) downto 1
      do HEAPIFY(A, i);
   end for
END
```

  Kompleksitas Waktu

$$\mathrm{T(n)} = \sum_{h=0}^{log(n)} \left(\frac{n}{2^{h+1}}\right) * O(h)$$

$$= O\left(n * \sum_{h=0}^{\log(n)} \frac{h}{2^h}\right)$$

$$= O\left(n * \sum_{h=0}^{\infty} \frac{h}{2^h}\right)$$

Big-O notation

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$$

$$\sum_{n=0}^{\infty} nx^n = \frac{x}{(1-x)^2}$$

$$= O(n * \frac{\frac{1}{2}}{1-\frac{1}{2}})$$

$$=O(n*2)$$

$$=O(n)$$

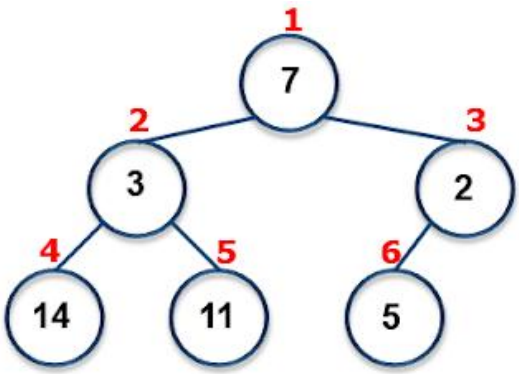
T(n) = O(n) + O(lg n) O(n)

=> T(n) = O(n lg n)

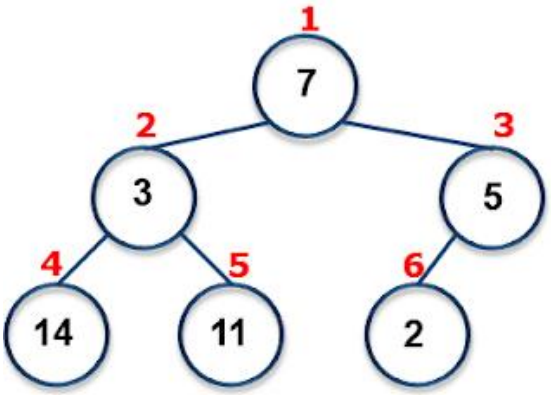
- **<u>Step-Step Heap Sort (6 Input)</u>**

Misal Input yang dimasukkan adalah :

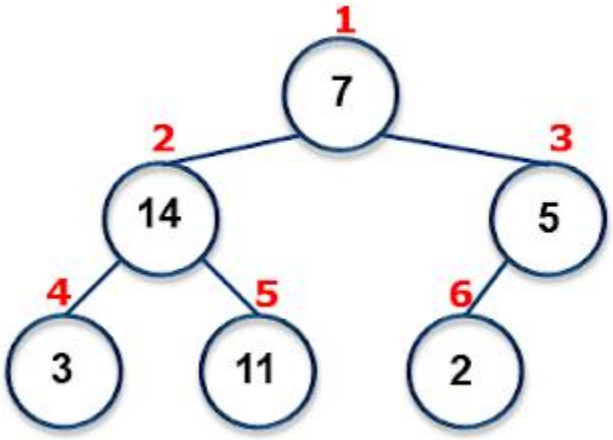
Konversi kedalam bentuk binary tree seperti ini :



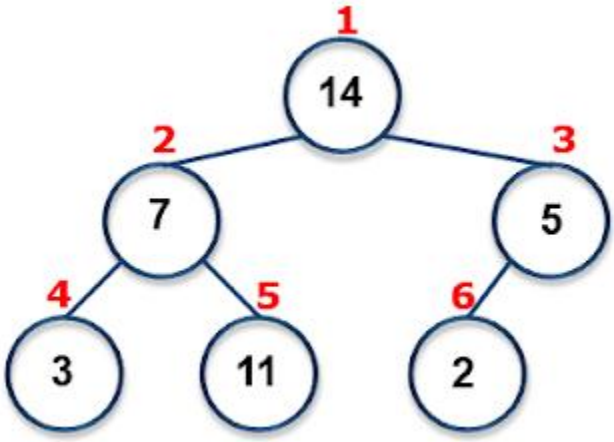
Jika sudah menjadi CBT lalu lakukan proses pengurutan secara max heap dengan cara banyaknya simpul dibagi dua untuk mencari nilai tengah dari sebuah array, sebagai contoh N = 6, Tengah = 6/2 = 3. Lalu lakukan reorganisasi pada simpul atau node ke-3. Dengan cara jika angka yang sekarang dibandingkan dengan angka selanjutnya yang ada di node turunannya itu lebih kecil maka tukar posisi.

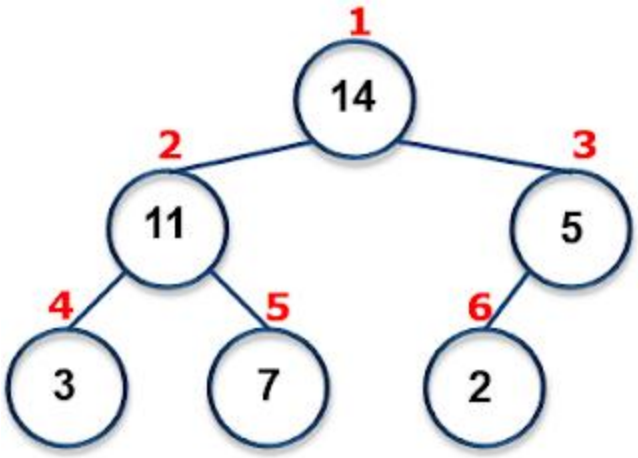


Lalu, lakukan reorganisasi pada simpul ke-2.



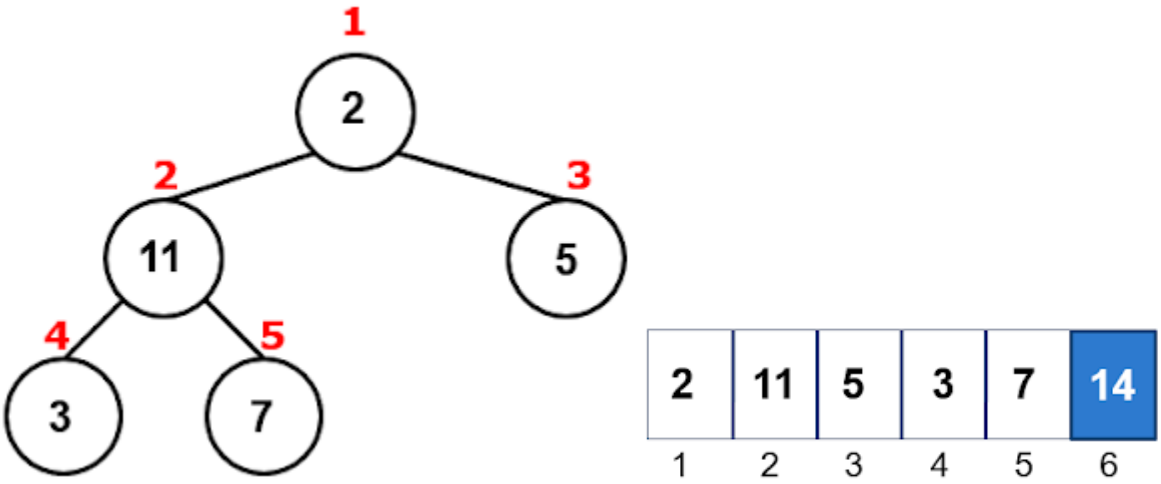
Lalu lakukan juga pada simpul ke-1





Terjadi perulangan karena angka 7 itu lebih kecil dari 14 dan 11, tetapi tidak lebih kecil dari 3. Maka dari itu dipindah posisikan sebanyak dua kali. Dan hasilnya adalah sebagai berikut :

Hapus atau "Pecat" root dan tukarkan dengan simpul pada posisi terakhir. Banyaknya simpul dikurangi 1. Jika n lebih dari 1, maka lakukan reorganisasi heap. Lakukan langkah ke-2 hingga ke-5 sampai n = 0.

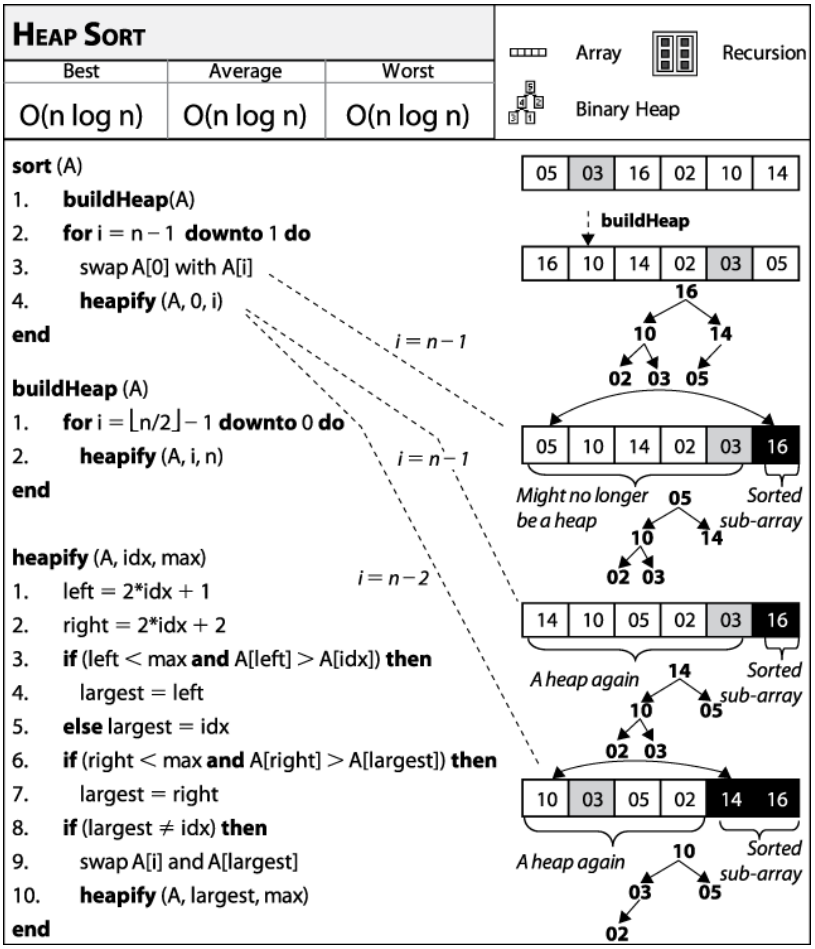


Karena datanya Binary Tree tidak dalam keadaan Max Heap, maka harus dilakukan lagi pembentukan heap agar menjadi max Heap. Lakukan terus hingga n = 0. Sehingga hasilnya dapat dilihat sebagai berikut.

- **Running Time**

T(n) = O(n) + O(lg n) O(n)

=> T(n) = O(n lg n)



Quick sort: Time taken for execution: 0.005288
Heap sort: Time taken for execution: 0.234245