**TUGAS PENGGANTI PRAKTIKUM**

**ANALISIS ALGORITMA**

****

**ILHAM MUHARAM**

**140810170046**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PADJADJARAN**

**TAHUN AJARAN 2018/2019**

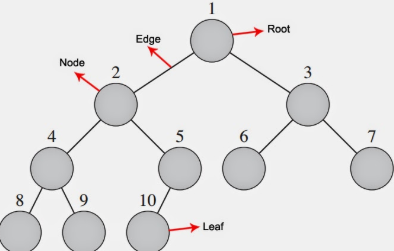
**Tugas:**

Heap Sort

* Hitung kompleksitas waktu dan big O
* Jelaskan Step by Step
* Contoh soal minimal 6 inputan
* Running Time

**Heap Sort**

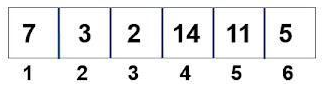
Heap sort adalah sebuah metode sorting (pengurutan) angka pada sebuah array dengan cara menyerupai binary tree, yaitu dengan cara memvisualisasikan sebuah array menjadi sebuah binary tree yang nantinya pada binary tree tersebut nilai pada masing-masing index array akan diurutkan. Pada heap sort terdapat 3 bagian yaitu Node, Edge, dan leaf dimana node itu adalah setiap index yang berada pada array, edge adalah garis yang menghubungkan tiap node dan leaf adalah setiap node yang tidak memiliki child node (node turunan). Selain itu juga ada yang bernama root yaitu node awal pada sebuah heap, berikut adalah ilustrasi dari bagian yang dimiliki oleh heap:



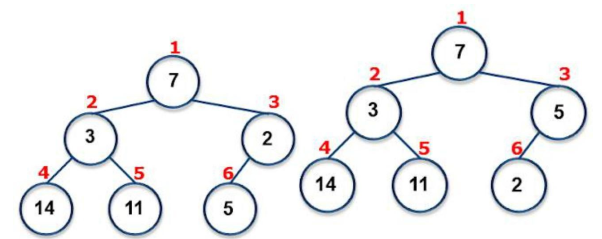
Heap tree terbagi menjadi 2 jenis yaitu ***Max-Heap*** dan ***Min-Heap***, dimana max-heap adalah kondisi heap tree yang memiliki nilai tertinggi berada di node root dan setiap child node memiliki nilai yang lebih kecil dari nilai yang dimiliki parent nodenya. Sedangkan pada min-heap adalah kondisi kebalikan dengan max-heap, pada min-heap nilai terkecil berada di node root dan setiap child node memiliki nilai yang lebih besar dari nilai yang dimiliki parent nodenya. Pada metode heap sort jenis heap tree yang digunakan adalah ***Max-Heap***.

Adapun proses dari Heap Sort:

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



Dari sebuah array dibuat menjadi Complete Binary Tree, lalu lakukan proses pengurutan secara max heap dengan cara banyak nya 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 disbanding kan 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

1. Hapus atau "Pecat" root dan tukarkan dengan simpul pada posisi terakhir.

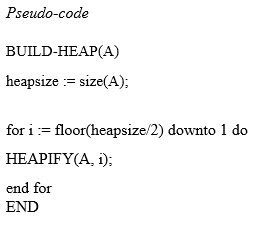
1. Banyaknya simpul dikurangi 1

1. Jika n lebih dari 1, maka lakukan reorganisasi heap

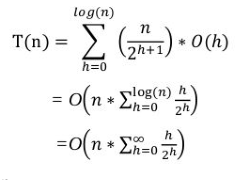
1. Lakukan langkah ke-2 hingga ke-5 sampai n = 0

**Kompleksitas waktu dan Big-O**

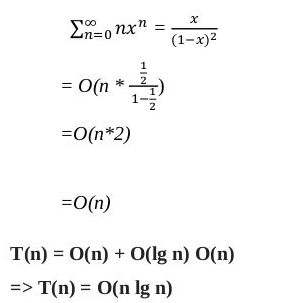
Algoritma pengurutan Heap Sort merupakan salah satu metode pengurutan tercepat setelah Merge Sort dan Quick Sort dengan kompleksitas ​O(n log n)



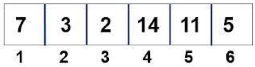
**Kompleksitas waktu**



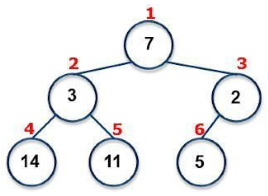
Big-O Notation



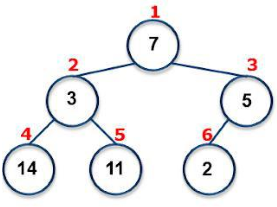
**Step-Step Heap sort 6 inputan**

Input yang di masukkan:  


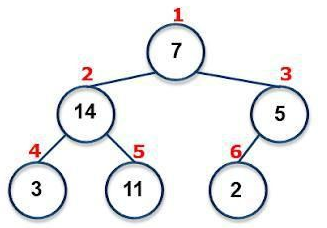
Konversi kedalam bentuk binary tree:



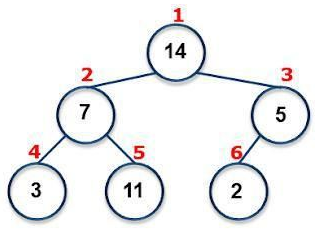
Jika sudah menjadi CBT lalu lakukan proses pengurutan secara max heap dengan cara banyaknya simpul dibagi dua untuk mencarinilaitengahdarisebuaharray,sebagaicontoh N = 6, Tengah = 6/2 = 3. Lalu lakukan reorganisasi pada simpul atau node ke-3. Dengan cara jika angka yang sekarang dibandingkan dengan angka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

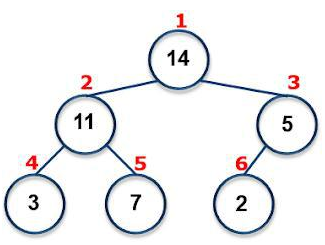


Lalu, lakukan reorganisasi pada simpul ke-2.

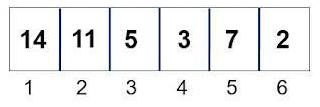


Lalu lakukan juga pada simpul ke-1

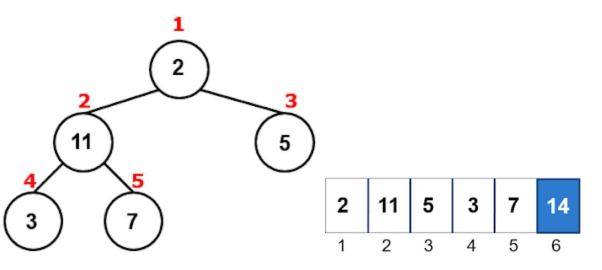




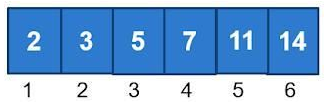
Terjadi perulangan karena angka 7 itu lebih kecil dari 14 dan 11, tetapi tidak lebih kecil dari 3. Maka dari itu dipindah posisikan sebanyak dua kali. Dan hasilnya adalah sebagai berikut :



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.

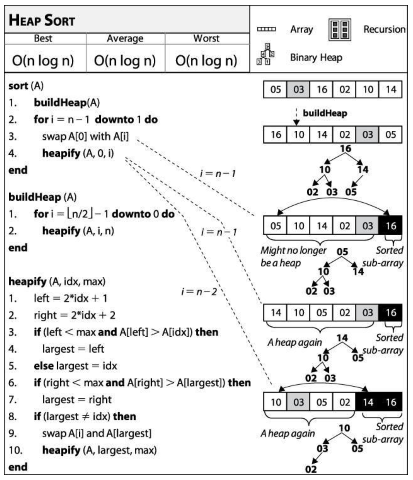


Karena datanya Binary Tree tidak dalam keadaan Max Heap, maka harus dilakukan lagi pembentukanheapagarmenjadimaxHeap.Lakukanterushinggan=0.Sehinggahasilnya dapat dilihat sebagai berikut.



**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