

TUGAS PENGANTI LIBURAN

Untuk Memenuhi Tugas

Mata Kuliah Praktikum Analisis Algoritma



Disusun oleh :

Felia Sri Indriyani

140810170018

TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2019

Tugas 1 : Buatlah program heap sort dalam c++, hitunglah kompleksitas waktu dan big-O, jelaskan step by step heap sort dengan contoh soal minimal 6 inputan, running time. Kumpulkan dalam format npm_HeapSort.pdf + cpp nya

Program :

```
/*
Nama      : Felia Sri Indriyani
NPM       : 140810170018
Program   : HeapSort
*/
// C++ program for implementation of Heap Sort
#include <iostream>

using namespace std;

// To heapify a subtree rooted with node i which is
// an index in arr[]. n is size of heap
void heapify(int arr[], int n, int i)
{
    int largest = i; // Initialize largest as root
    int l = 2*i + 1; // left = 2*i + 1
    int r = 2*i + 2; // right = 2*i + 2

    // If left child is larger than root
    if (l < n && arr[l] > arr[largest])
        largest = l;

    // If right child is larger than largest so far
    if (r < n && arr[r] > arr[largest])
        largest = r;

    // If largest is not root
    if (largest != i)
    {
        swap(arr[i], arr[largest]);

        // Recursively heapify the affected sub-tree
        heapify(arr, n, largest);
    }
}

// main function to do heap sort
void heapSort(int arr[], int n)
{
    // Build heap (rearrange array)
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // One by one extract an element from heap
    for (int i=n-1; i>=0; i--)
    {
        // Move current root to end
```

```

        swap(arr[0], arr[i]);

        // call max heapify on the reduced heap
        heapify(arr, i, 0);
    }
}

/* A utility function to print array of size n */
void printArray(int arr[], int n)
{
    for (int i=0; i<n; ++i)
        cout << arr[i] << " ";
    cout << "\n";
}

// Driver program
int main()
{
    cout << "Program Heap Sort" << endl;
    cout << "=====" << endl;
    cout << "Soal : 12, 11, 13, 5, 6, 7, 1" << endl;
    int arr[] = {12, 11, 13, 5, 6, 7, 1};
    int n = sizeof(arr)/sizeof(arr[0]);

    heapSort(arr, n);

    cout << "Array yang telah terurut : \n";
    printArray(arr, n);
}

```

Kompleksitas waktu dan big-O :

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

Pseudo-code

```

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

```

Kompleksitas Waktu

$$\begin{aligned}T(n) &= \sum_{h=0}^{\log(n)} \left(\frac{n}{2^{h+1}} \right) * O(h) \\&= O\left(n * \sum_{h=0}^{\log(n)} \frac{h}{2^h}\right) \\&= O\left(n * \sum_{h=0}^{\infty} \frac{h}{2^h}\right)\end{aligned}$$

Big-O notation

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

$$\begin{aligned}\sum_{n=0}^{\infty} nx^n &= \frac{x}{(1-x)^2} \\&= O\left(n * \frac{\frac{1}{2}}{1-\frac{1}{2}}\right) \\&= O(n * 2)\end{aligned}$$

$$= O(n)$$

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

$$\Rightarrow T(n) = O(n \lg n)$$

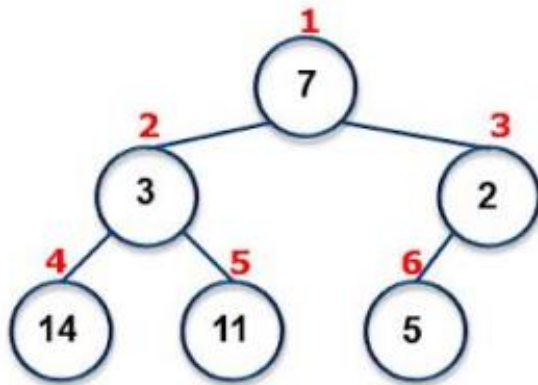
Step by step :

Misal Input yang dimasukkan adalah :

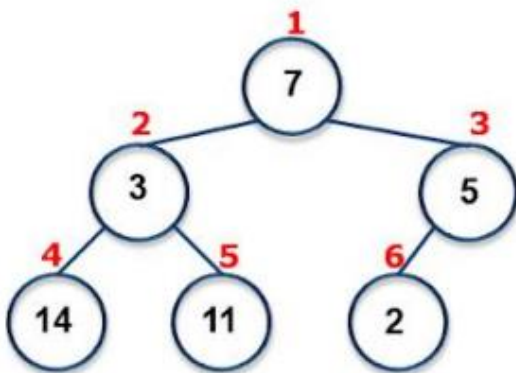
Misal Input yang dimasukkan adalah :

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

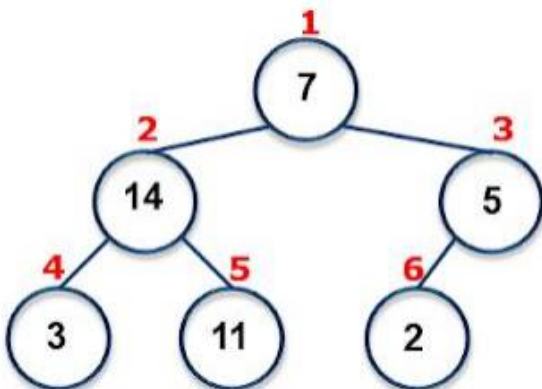
Konversi kedalam bentuk binary tree seperti ini :



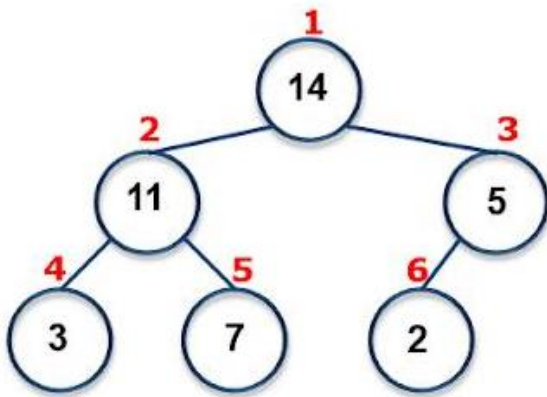
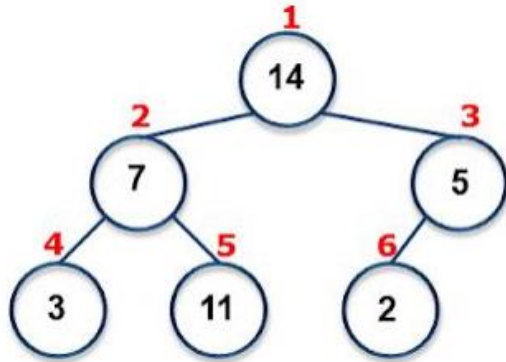
Jika sudah menjadi CBT lalu lakukan proses pengurutan secara max heap dengan cara banyaknya simpul dibagi dua untuk mencari nilai tengah dari sebuah array, sebagai contoh $N = 6$, $Tengah = 6/2 = 3$. Lalu lakukan reorganisasi pada simpul atau node ke-3. Dengan cara jika angka yang sekarang dibandingkan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



Lalu, lakukan reorganisasi pada simpul ke-2.



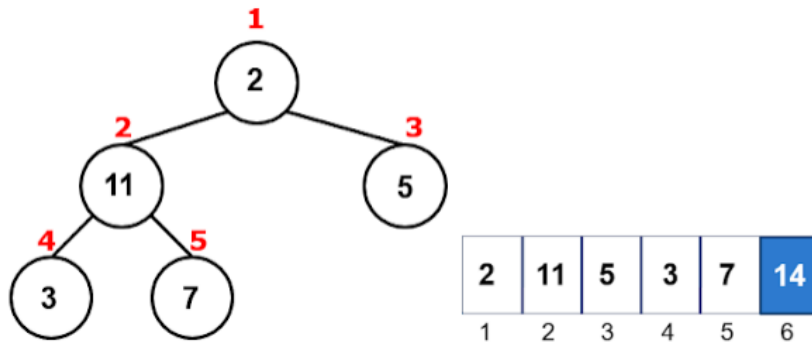
Lalu lakukan juga pada simpul ke-1



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 :

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

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 $n = 0$.

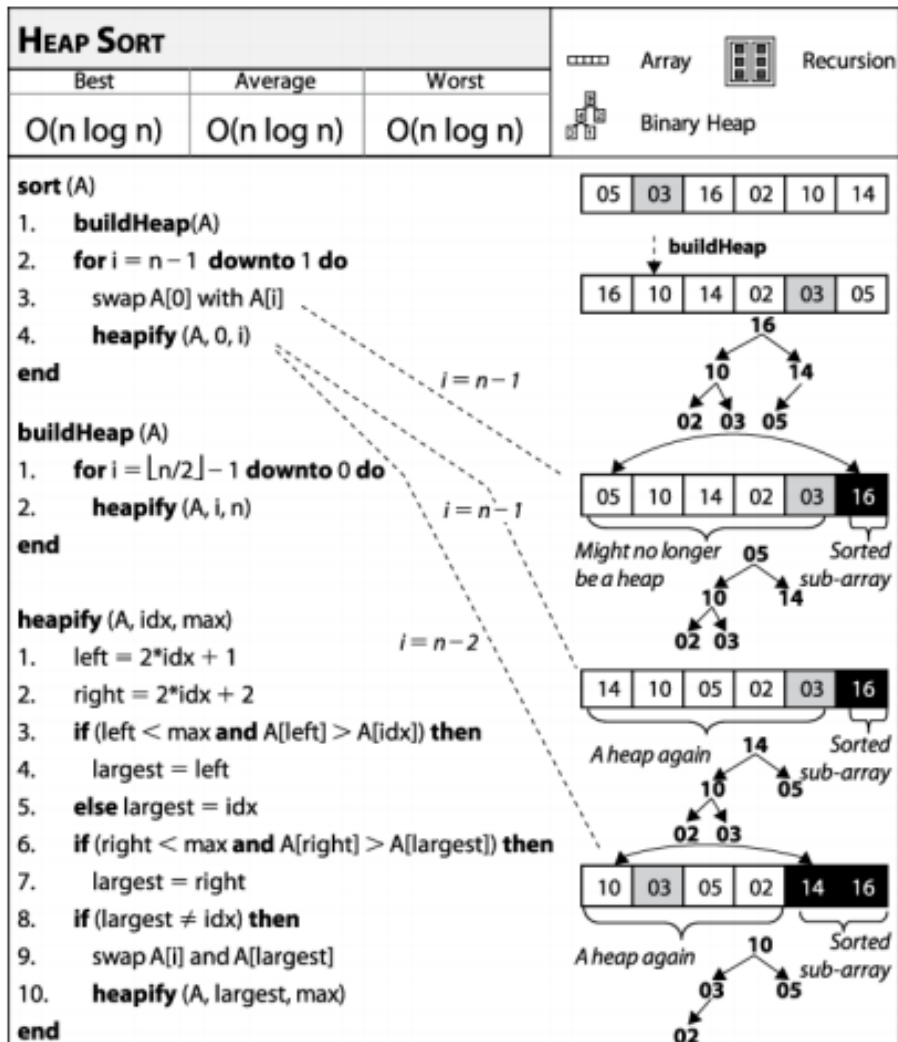


Karena datanya Binary Tree tidak dalam keadaan Max Heap, maka harus dilakukan lagi pembentukan heap agar menjadi max Heap. Lakukan terus hingga $n = 0$. Sehingga hasilnya dapat dilihat sebagai berikut.



Contoh soal dan running time :

$$T(n) = O(n) + O(\lg n) \quad O(n) \Rightarrow T(n) = O(n \lg n)$$



Quick sort: Time taken for execution: 0.005288

Heap sort: Time taken for execution: 0.234245

"D:\Computer Science\SEMESTER 4\Analisa Algoritma\Praktikum\TugasPengganti\HeapSort\t

Program Heap Sort

=====


Soal : 12, 11, 13, 5, 6, 7, 1

Array yang telah terurut :

1 5 6 7 11 12 13

Process returned 0 (0x0) execution time : 0.527 s

Press any key to continue.

 "D:\Computer Science\SEMESTER 4\Analisa Algoritma\Praktikum\TugasPengg

```
Program Heap Sort
```

```
=====
```

```
Soal : 7, 3, 2, 14, 11, 5
```

```
Array yang telah terurut :
```

```
2 3 5 7 11 14
```

```
Process returned 0 (0x0)   execution time : 0.525 s
```

```
Press any key to continue.
```