Heap Sort

https://www.youtube.com/watch?v=2DmK_H7IdTo

INSTITUT
TEKNOLOGI
HARAPAN
BANGSA
Veritas vos liberabit

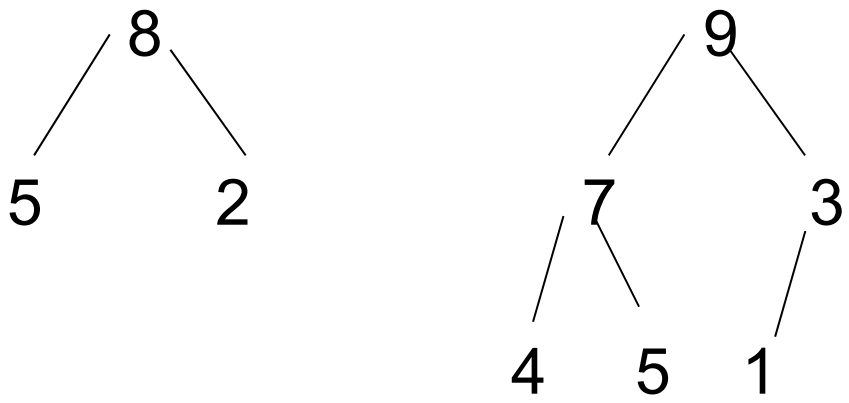
Heap Sort

- Kecepatannya : n $\log_2$ n (rata-rata dan kasus terburuk)
- Quicksort juga mempunyai kecepatan yang sama, tetapi pada kasus rata-rata, tidak pada kasus terburuk
- Langkah-langkah utama HEAP SORT
  - Buat sebuah heap yang berisi data yang akan diurutkan
  - Pindahkan elemen maksimumnya dan buat heap baru yang berisi elemen sisanya

Heap Sort

- Heap adalah sebuah binary tree (pada umumnya complete) yang setiap nodenya mempunyai nilai yang lebih besar atau sma dengan nilai dari kedua anaknya (jika ada)

Contoh :



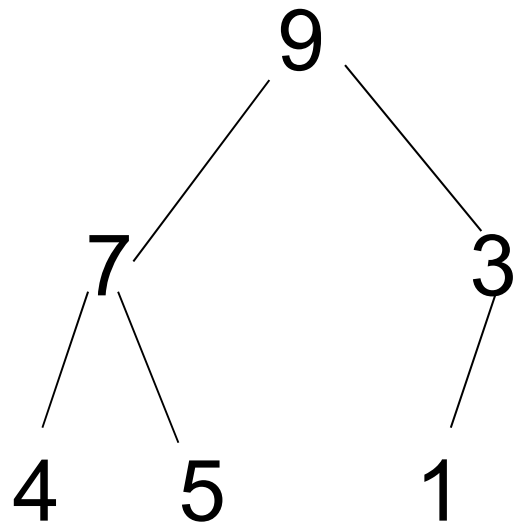
Nama yang lebih cocok adalah

Maxitree (bukan heap)

Heap Sort

Bagaimana cara pengurutan sebuah heap ?

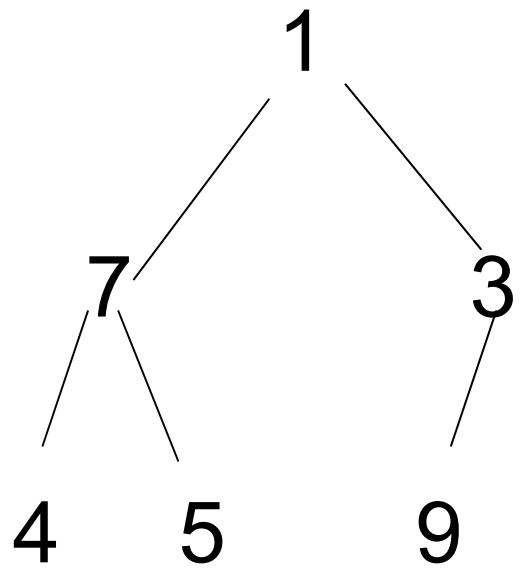
Heap yang kita miliki :



Sesuai dengan definisi, root berisi nilai terbesar => mencari nilai terbesar adalah operasi yang mudah

Heap Sort

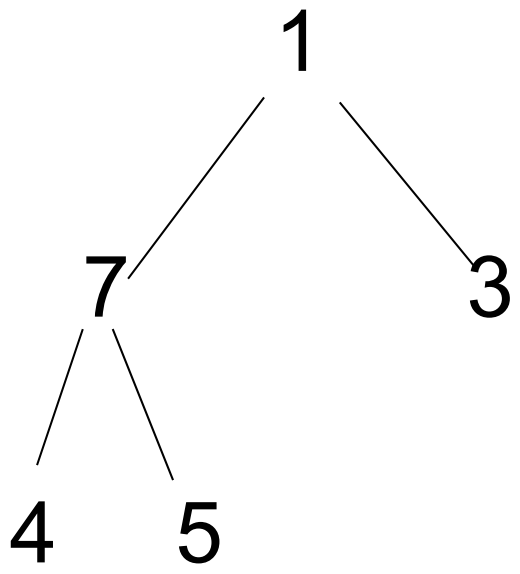
Nilai maksimum diletakkan di akhir heap, diperoleh :



Elemen terakhir (9) tidak akan diproses lagi sebab tempatnya sudah benar

Heap Sort

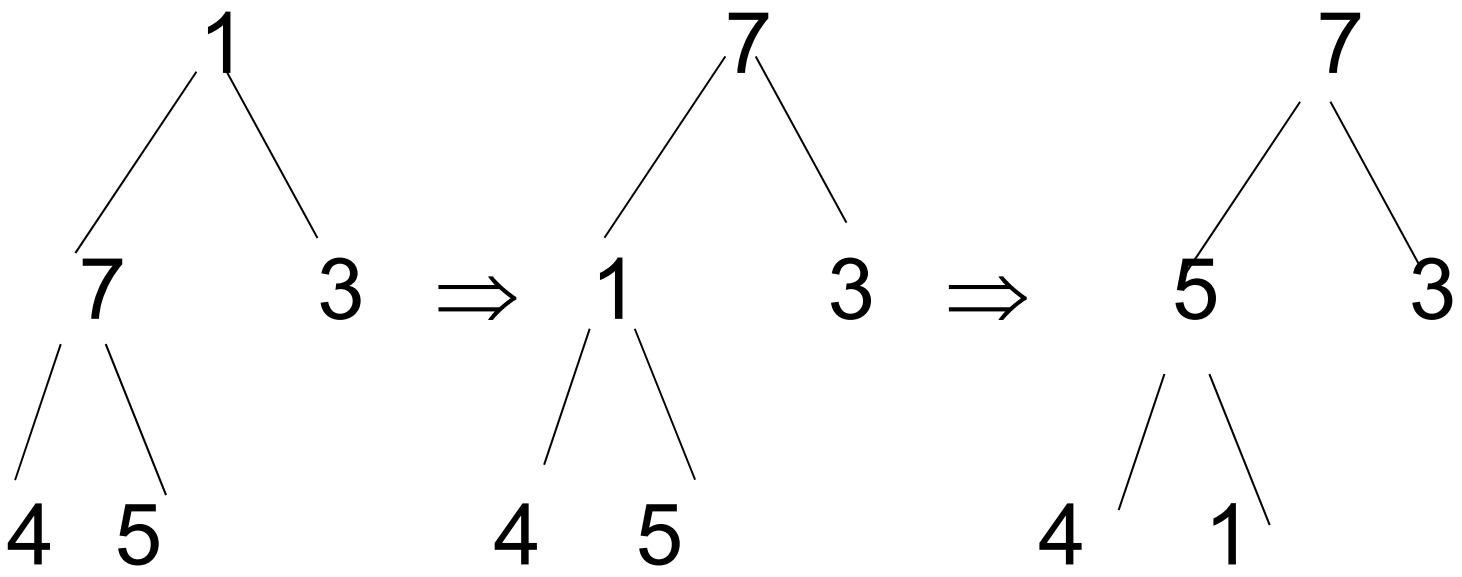
Kita akan memproses tree :



Tree tersebut bukan heap sebab rootnya mempunyai nilai baru, tetapi kedua subtreenya adalah heap

Heap Sort

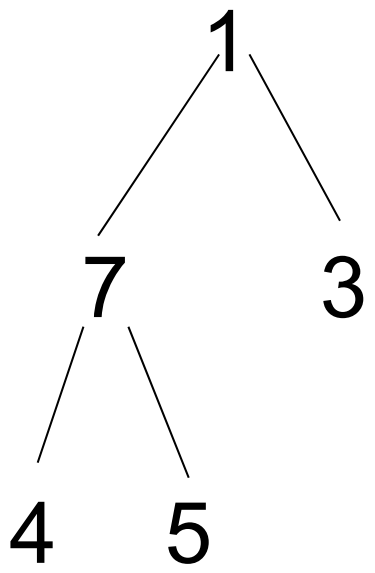
Bentuk heap baru dari tree yg sudah ada:



Kecepatan : Paling banyak $\log_2$ (n) pertukaran

Proses diatas diulang untuk heap yang lebih kecil

Heap Sort



→ nilai baru

- Tree tsb bukan heap, tetapi kedua subtreenya adalah heap

Heap Sort

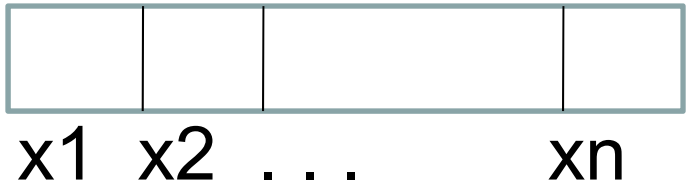
- Cara reorganisasi (adjust) => memindahkan nilai root ke bawah dengan cara menukarkannya dengan nilai son-nya yang terbesar (proses ini diulang terus sampai memperoleh HEAP)

Heap Sort

```
ADJUST(T) { T adalah sebuah tree yang subtree kiri dan
    kanannya adalah heap}
    X = T
    V = nilai dari root T
    while (x mempunyai sebuah son yang
                    nilainya lebih besar dari V)
    do
        Y = son dari x yang nilainya lebih besar
        nilai root dari x = nilai root dari y
        x = y
    end
    nilai root dari x = v
END ADJUST
```

Struktur data yang dipakai

- Sebuah complete binary tree dapat direpresentasikan dengan array satu dimensi



- Kedua son dari sebuah elemen berada di posisi 2i dan 2i+1
- Jika 2i atau 2i+1 lebih besar dari N, elemen i mempunyai 1 son atau tidak mempunyai son

Algoritme ADJUST

```
ADJUST(I,N)
    K=I; V = X(I); J=2K; HEAP=false
    while (J<=N) and (not HEAP)
    do
          { cari max dari son kiri-kanan }
          if J<N { right son ada ?}
          then if x(j) < x(j+1) then j = j+1

          { bandingkan max son dng V=> bergerak ke bawah/
            berhenti}
          if x(j) <= V
          then HEAP = TRUE
          else begin x(K) = x(J) ; K = j; J = 2K end ;
    end while
    x (K) = v
END ADJUST
```

INSTITUT TEKNOLOGI HARAPAN BANGSA

Veritas vos liberabit

Algoritme HEAPSORT pada level tertinggi

HEAPSORT (X,N)

    ProduceHeap(X,N)

    SortHeap(X,N)

Algoritme ProduceHeap

- Dalam sebuah HEAP, elemen-elemen pada level terendah tidak mempunyai aturan khusus
- Semua nilai di dalam array x dari $\lfloor n/2 \rfloor + 1$ sampai N adalah elemen pada level terendah
- Algoritma ADJUST dipakai secara iteratif dan berjalan mundur, dengan cara menyisipkan x(n/2), x(n/2 -1) … x(1)

```
PRODUCEHEAP(X,N)
      for I = ⌊n/2⌋ to 1 step -1
      do  ADJUST(I,N)
end PRODUCEHEAP
```

Algoritme SortHeap

- Setelah menjalankan PRODUCEHEAP, x(1) adalah nilai maksimum
- Selanjutnya, x(1) ditukar dengan x(n) dan reorganisasi x(1:n-1) agar menjadi heap
- Proses tersebut diulang sampai x(1:2)

Algoritme SortHeap

```
SORTHEAP(X,N)
    I=N
    while I >= 2
    do   { loop invariant x(I+1:N) sudah terurut dan
             berisi nilai terbesar dari x; x(1:i) adalah heap}

         TUKAR x(1) dengan x(i)
         ADJUST(1,I-1)
         I = I -1
    end while
End SORTHEAP
```

INSTITUT TEKNOLOGI HARAPAN BANGSA

Veritas vos liberabit