

# Checkpoint: Binary Heap

Monday, November 14, 2022 1:59 PM  
Nazimer

Diketahui sebuah array yang merepresentasikan sebuah **binary heap** dengan kaidah **min-heap**, dengan nama **myBinHeap**, sebagai berikut (indeks array dari 0, 1, ....).

Indeks	0	1	2	3	4	5	6
data	11	12	13	14	15	16	17

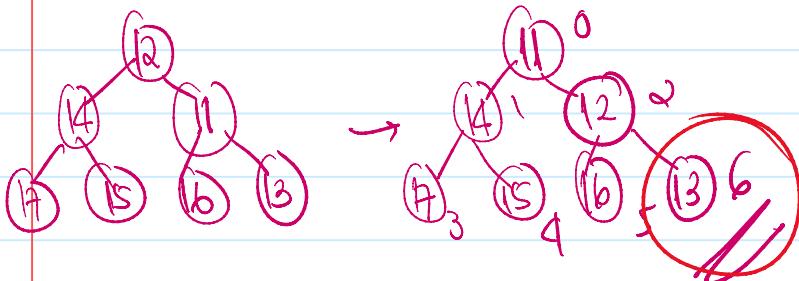
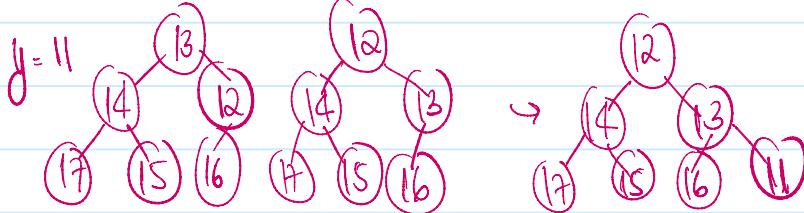
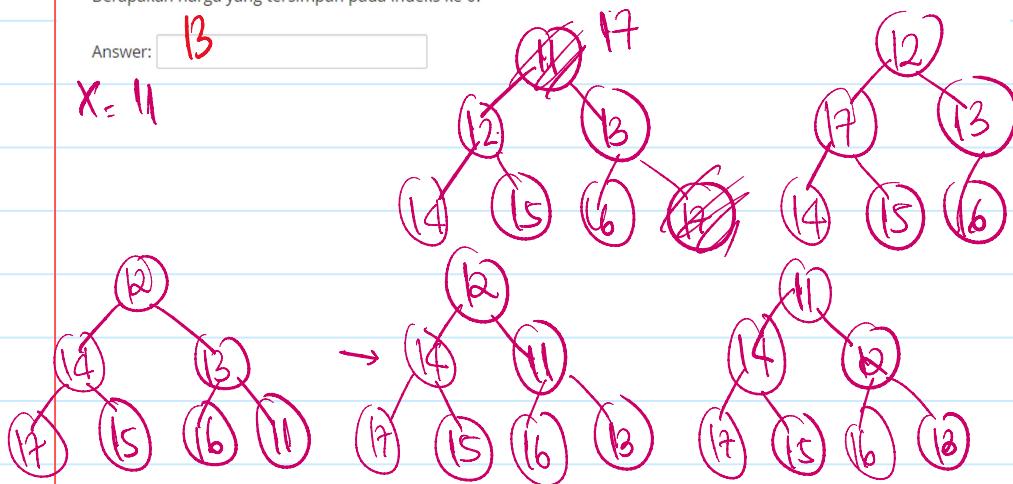
Jika dilakukan beberapa operasi sebagai berikut:

```
int x = myBinHeap.getMin(); //operasi mendapatkan min dan remove  
myBinheap.add(x); //operasi insert  
int y = myBinHeap.getMin(); //operasi mendapatkan min dan remove  
myBinheap.add( y ); //operasi insert
```

Berapakah harga yang tersimpan pada indeks ke 6?

Answer:

X = 11



Diketahui sebuah array yang merepresentasikan sebuah **binary heap** dengan kaidah **max-heap** sebagai berikut (indeks array dari 0, 1, ....)

Indeks	0	1	2	3	4	5	6	7	8	9
data	30	22	25	11	16	21	10	5	9	3

Jika dilakukan penghapusan maksimum (remove max) dua kali, pada indeks berapakah kemudian 9 berada?

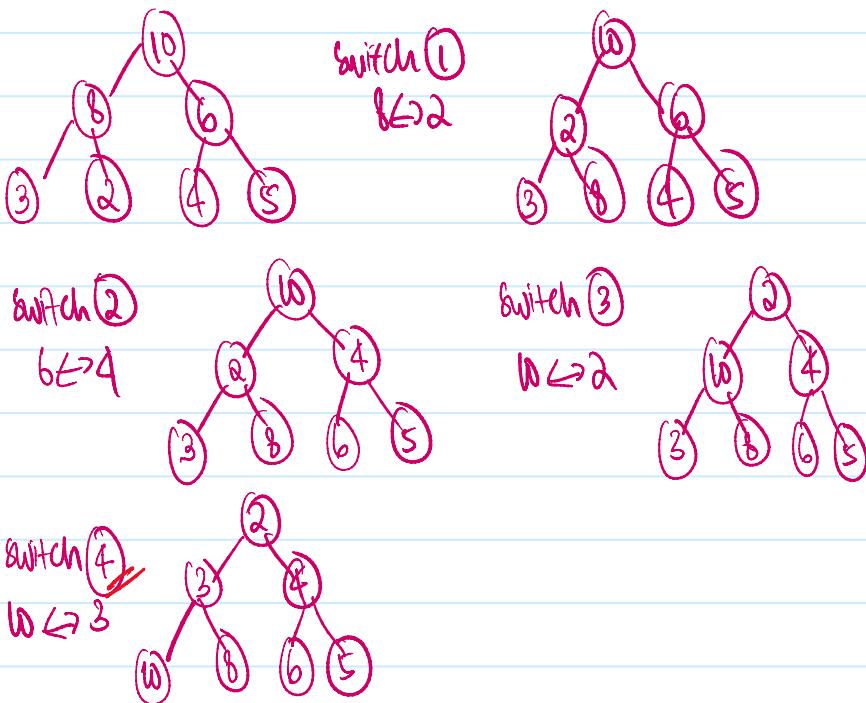
Answer: 4



Diketahui sebuah **binary heap** dengan kaidah **max-heap** yang direpresentasikan dalam array sebagai berikut.

10	8	6	3	2	4	5
----	---	---	---	---	---	---

Algoritma Fix-heap (Heapify) digunakan untuk mengubah kaidah **max-heap** menjadi **min-heap**. Berapa banyak proses pertukaran/pergeseran elemen yang terjadi? (*Tulis angkanya saja*)



Manakah pernyataan berikut yang salah mengenai algoritma pengurutan heap sort?

Select one:

- Implementasi algoritma heap sort dapat dilakukan secara in-place, yaitu dengan menukar-nukar posisi elemen pada array (tidak membutuhkan array baru). ✓
- Pengurutan *ascending* (terurut menaik) memanfaatkan min-heap, sedangkan pengurutan *descending* (terurut menurun) memanfaatkan max-heap. ✗
- Meskipun kompleksitas heapify bisa mencapai  $O(N)$ , kompleksitas heap sort tidak akan lebih baik dari  $O(N \log N)$ . ✓
- Heap-sort pada dasarnya merupakan algoritma yang not-stable, yaitu elemen dengan nilai yang sama mungkin saja muncul dengan

lon 1

Heap sort

⇒ in-place

⇒ not stable

Ascending → MaxHeap

Descending → MinHeap

Operasi remove

Heap sort:  $N \log N$

↳ Heapify

- Heap-sort pada dasarnya merupakan algoritma yang not-stable, yaitu elemen dengan nilai yang sama mungkin saja muncul dengan urutan yang berbeda antara sebelum dan sesudah pengurutan.

Heapsort digunakan untuk melakukan sorting secara **ascending** data berikut:

indeks	0	1	2	3	4	5	6
data	99	88	77	66	55	44	33

Setelah dilakukan heapify (fix-heap) kemudian iterasi memindahkan max ke sorted area sebanyak dua kali, berapakah data yang berada pada indeks 3?

Select one:

- 55
- 66
- 33
- 44



remove 1 99 88 77 66 55 44 33  
 33 88 77 66 55 44 | 99  
 88 33 77 66 55 44 | 99  
 88 66 77 33 55 44 | 99

remove 2 44 66 77 33 55 | 88 99  
 77 66 44 | 33 55 | 88 99  
 0 1 2 3

Diketahui sebuah array yang merepresentasikan **binary heap** dengan kaidah **min-heap** sebagai berikut di mana setiap elemennya berbeda satu sama lain (unik).

A	B	C	D	E	F	G	H	I
---	---	---	---	---	---	---	---	---

Manakah pernyataan yang belum pasti benar?

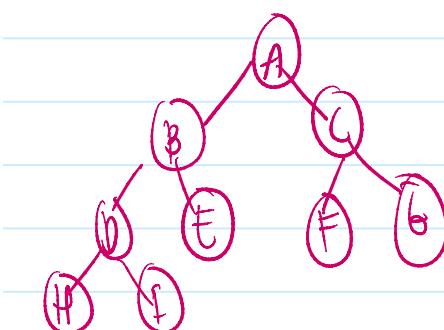
(Jawaban mungkin lebih dari satu)

Select one or more:

- Nilai elemen G lebih besar daripada elemen D. ↗ belum tentu

U

Ascending → MaxHeap



- Nilai elemen G lebih besar daripada elemen D. *belum tentu*
- Elemen A merupakan elemen dengan nilai terkecil. *g*
- Jika elemen C terkecil kedua, maka B terkecil ketiga. *belum tentu*
- Elemen H merupakan anak kiri dari elemen D. *✓*
- Elemen E merupakan leaf. *✓*
- Elemen I merupakan elemen dengan nilai terbesar. *X* *belum tentu*

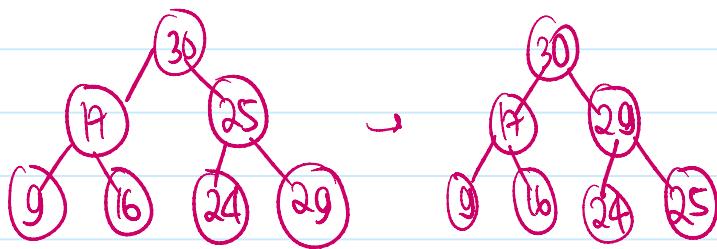
Diketahui sebuah array yang merepresentasikan sebuah **binary heap** dengan kaidah **max-heap** sebagai berikut (indeks array dari 0, 1, ....)

Indeks	0	1	2	3	4	5	6	7
data	30	17	25	9	16	24		

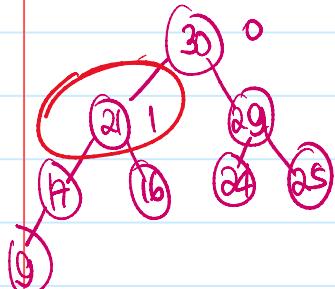
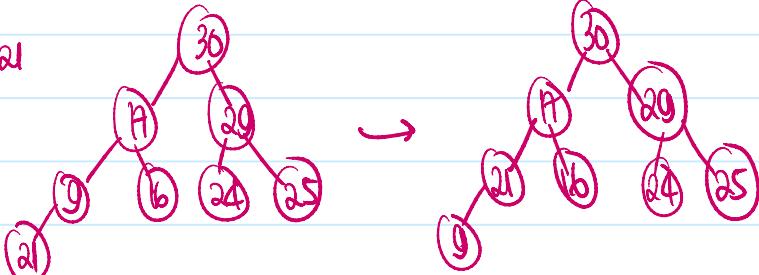
Jika dilakukan penambahan (*insert*) dua elemen yaitu 29 dan 21 berturut-turut, pada indeks berapakah kemudian 21 ditempatkan?

Answer:

*max heap*



*add 21*



Diketahui sebuah complete binary tree terdiri dari

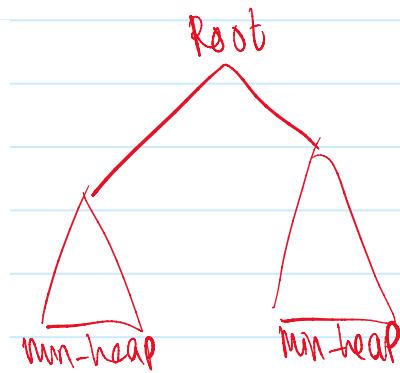
*Root*

Diketahui sebuah complete binary tree terdiri dari  $N$  elemen. Subtree kiri dan subtree kanan dari binary tree tersebut masing-masing dipastikan merupakan min-heap.

Untuk mengubah binary tree tersebut menjadi sebuah min-heap, dengan prosesnya dimulai dari root, kompleksitasnya adalah ...

Select one:

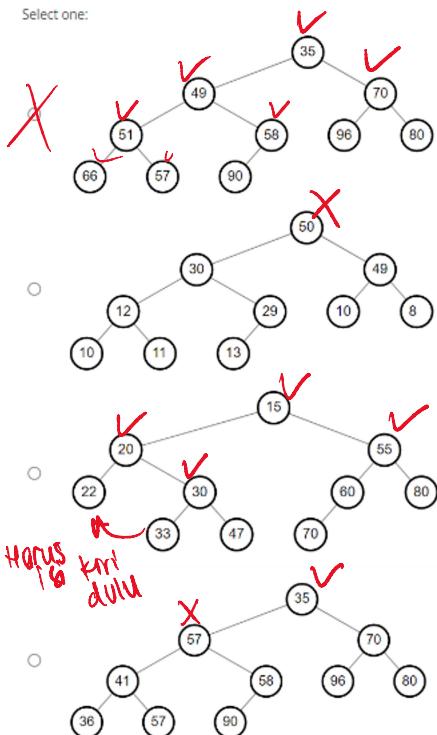
- O(1)
- O( $N$ )
- O( $N \log N$ )
- O( $\log N$ )



root  
percolate down sebanyak  $\log N$ ?  
Worst

Manakah di antara tree berikut yang merepresentasikan sebuah **binary heap** dengan kaidah **min-heap**?

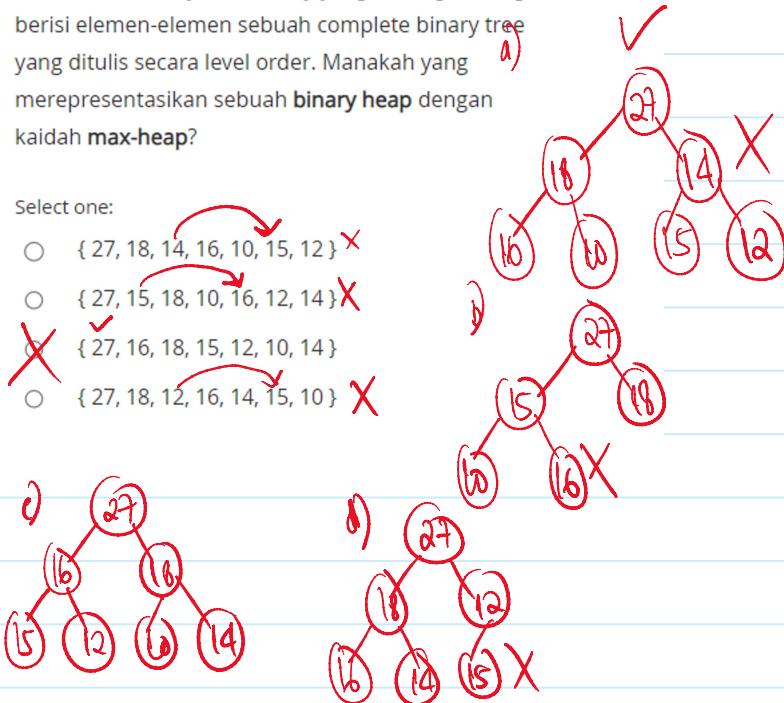
Select one:



Berikut adalah sejumlah array yang masing-masing berisi elemen-elemen sebuah complete binary tree yang ditulis secara level order. Manakah yang merepresentasikan sebuah **binary heap** dengan kaidah **max-heap**?

Select one:

- { 27, 18, 14, 16, 10, 15, 12 } X
- { 27, 15, 18, 10, 16, 12, 14 } X
- { 27, 16, 18, 15, 12, 10, 14 }
- { 27, 18, 12, 16, 14, 15, 10 } X



Manakah pernyataan yang salah mengenai kompleksitas binary heap?

Select one:

- Kompleksitas operasi heapify pada suatu min-heap adalah  $O(\log N)$  →  $O(N)$  X
- Kompleksitas operasi findMin pada suatu min-heap adalah  $O(1)$  →  $O(1)$  ✓
- Kompleksitas operasi insert pada suatu min-heap adalah  $O(\log N)$  →  $O(\log N)$  ✓
- Kompleksitas operasi remove pada suatu min-heap adalah  $O(\log N)$  →  $O(\log N)$  ✓ *y recolate  
shift*

Diberikan array yang berisi data berikut.

indeks	0	1	2	3	4	5	6
data	99	88	77	66	55	44	33

Jika pada data akan dilakukan **heapify (fix-heap)** dengan kaidah **min-heap**, isi array tersebut (setelah heapify selesai!) adalah:

Select one:

- 33, 66, 44, 99, 77, 88, 55
- 33, 44, 55, 66, 77, 88, 99
- 33, 55, 44, 77, 66, 99, 88
- 33, 55, 44, 66, 88, 99, 77

*dr bawah?*

