Heapify & Heap sort

Alfan F. Wicaksono

Heapify

- Proses konversi sebuah array (*arbitrary*) menjadi sebuah min/max heap.

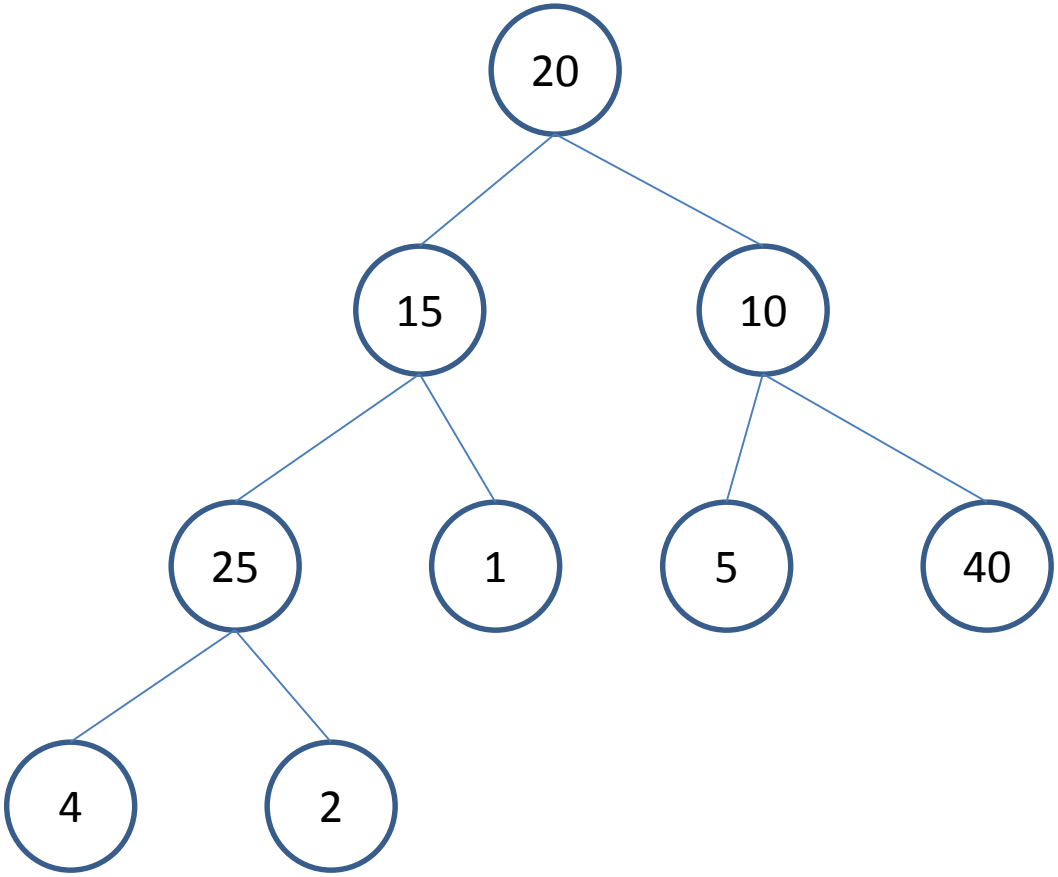
Heapify

- Algoritma yang efisien bekerja dengan prinsip **bottom-up**, dan melakukan **percolate down** berkali-kali hingga root.
- Proses dimulai dari **node bukan daun** yang ada di level **paling bawah** – **paling kanan**. Lalu, bergerak secara **reversed level order**.
- Kompleksitas algoritma ini adalah **O(n)**, dimana **n** adalah ukuran heap.

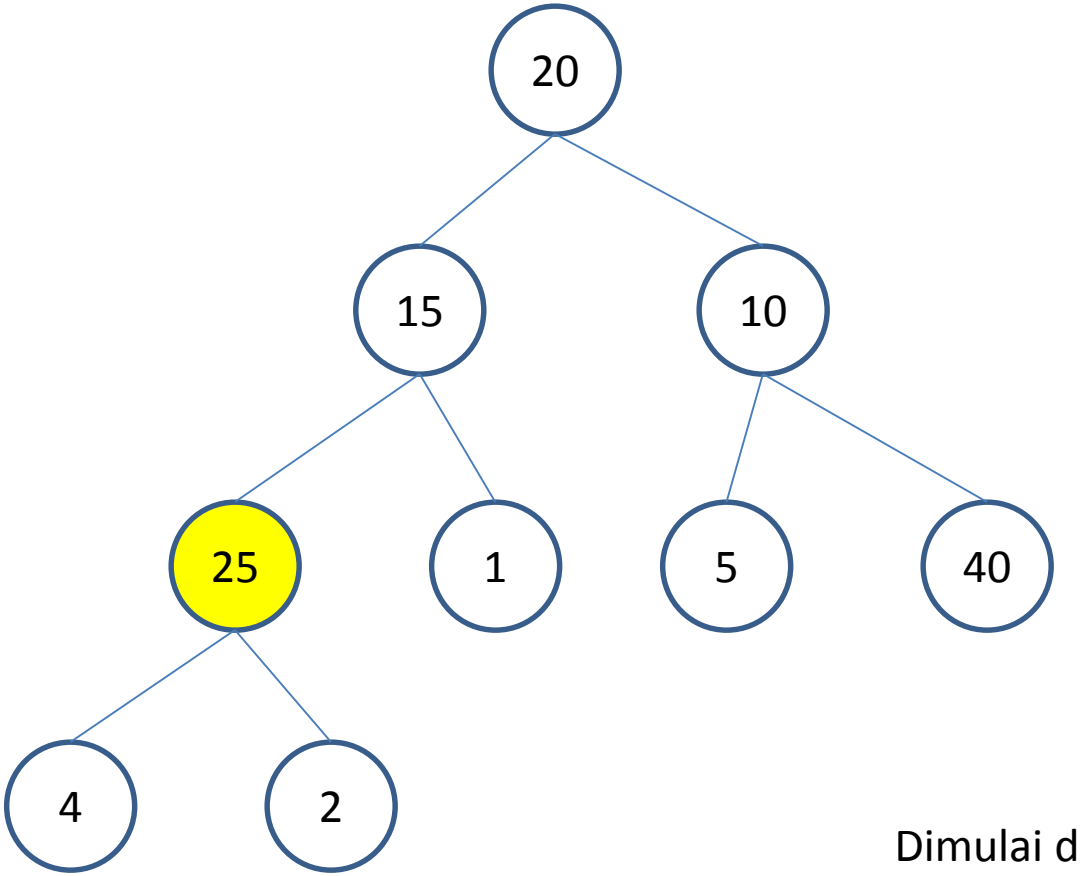
Heapify

```
heapify(A):
   A.heapSize = A.length
   for i = floor(n/2)-1 downto 0:
      percolateDown(A, i)
```

n : banyak elemen di array/heap size

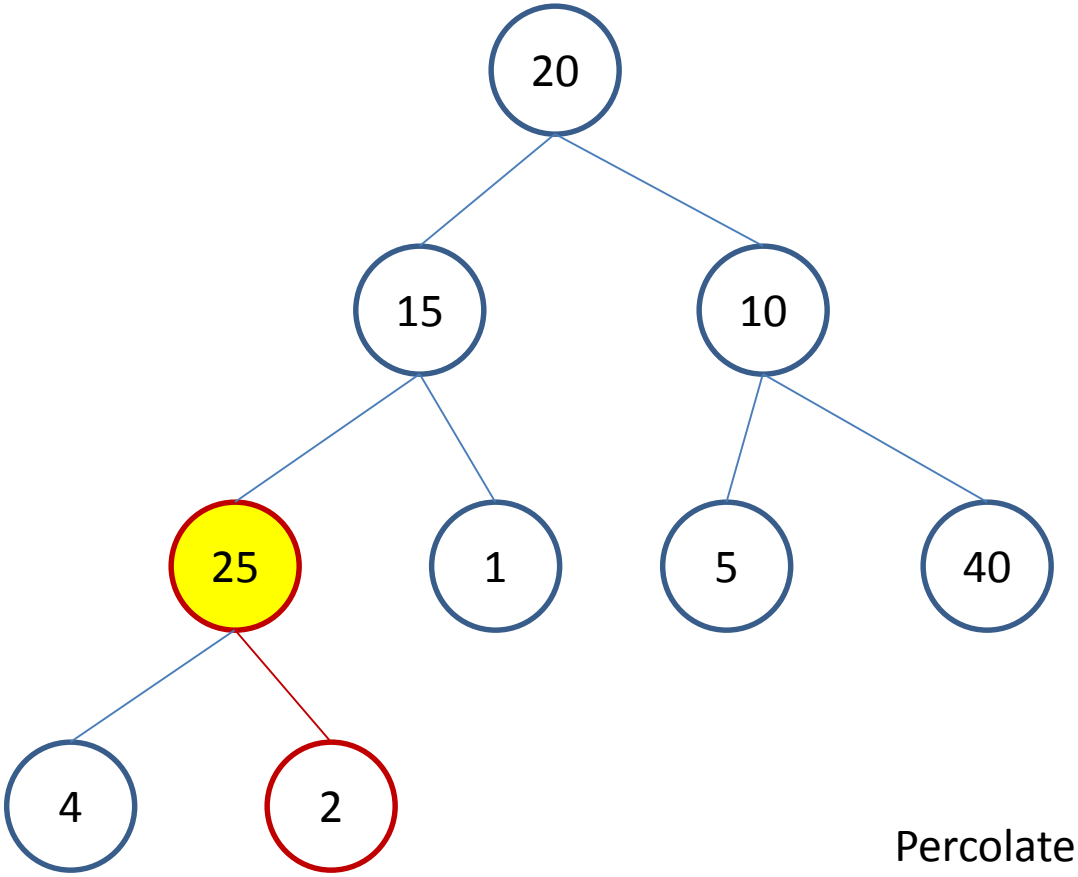


20	**15**	**10**	**25**	**1**	**5**	**40**	**4**	**2**
0	1	2	3	4	5	6	7	8

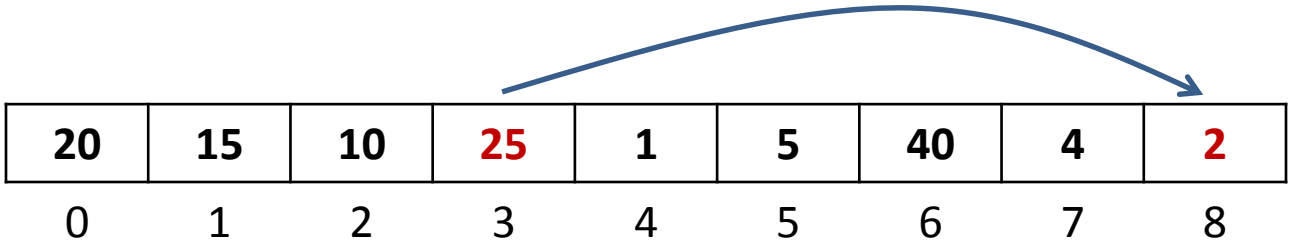


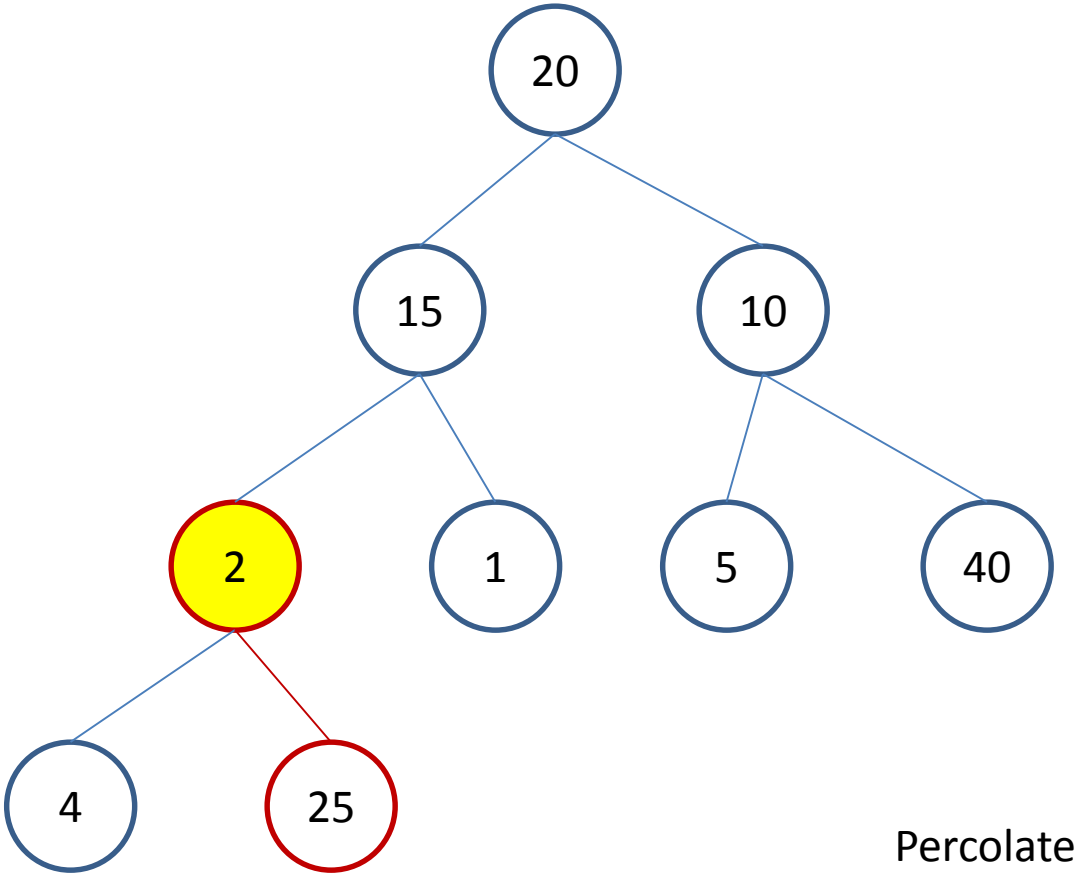
Dimulai dari **floor(9/2) = 3**

20	**15**	**10**	**25**	**1**	**5**	**40**	**4**	**2**
0	1	2	3	4	5	6	7	8

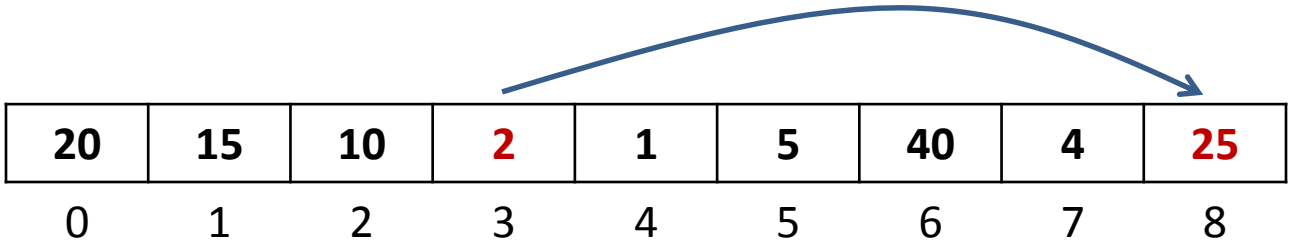


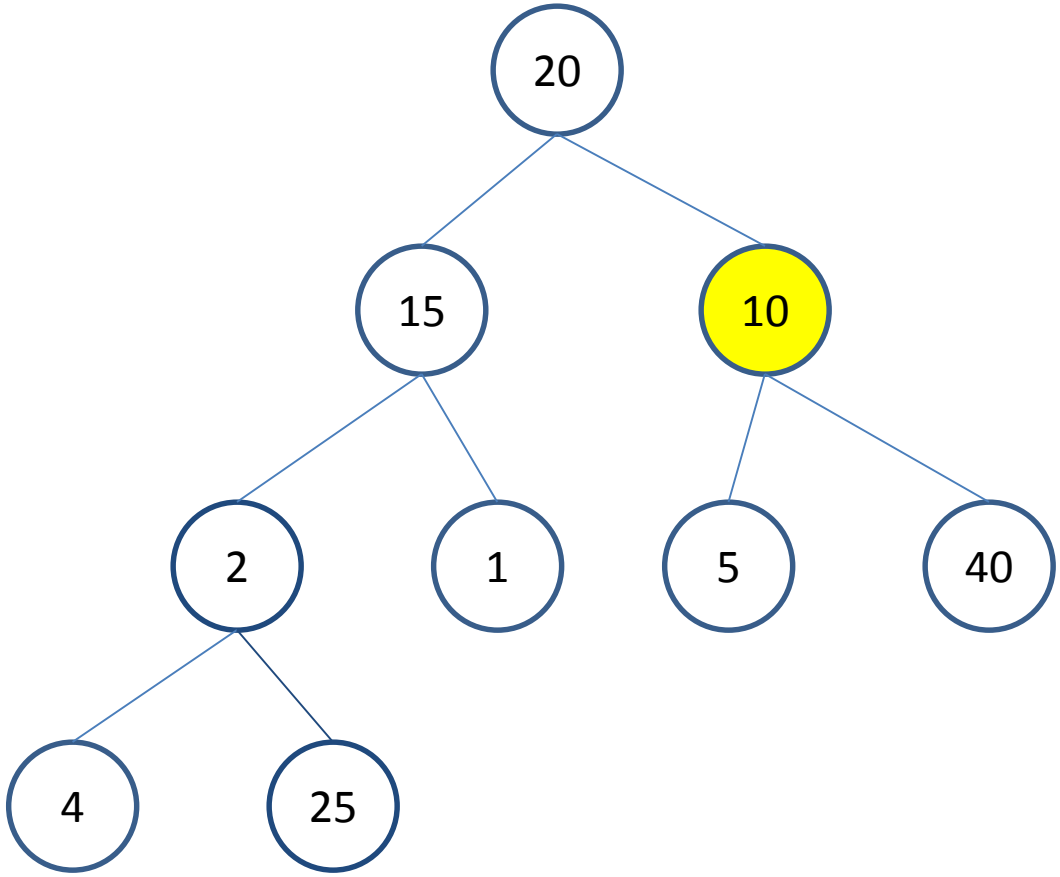
Percolate down !

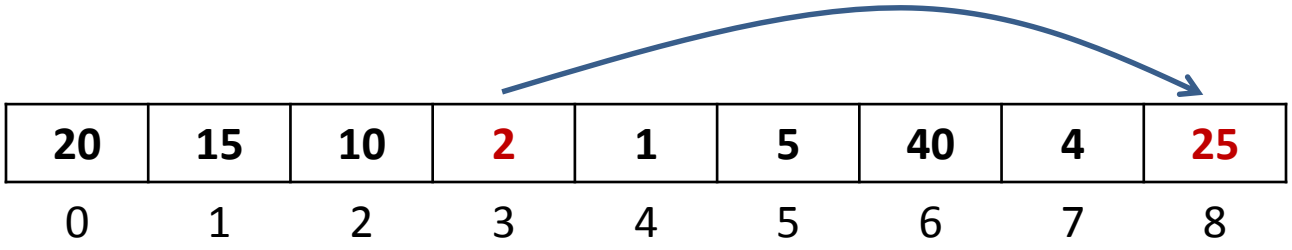


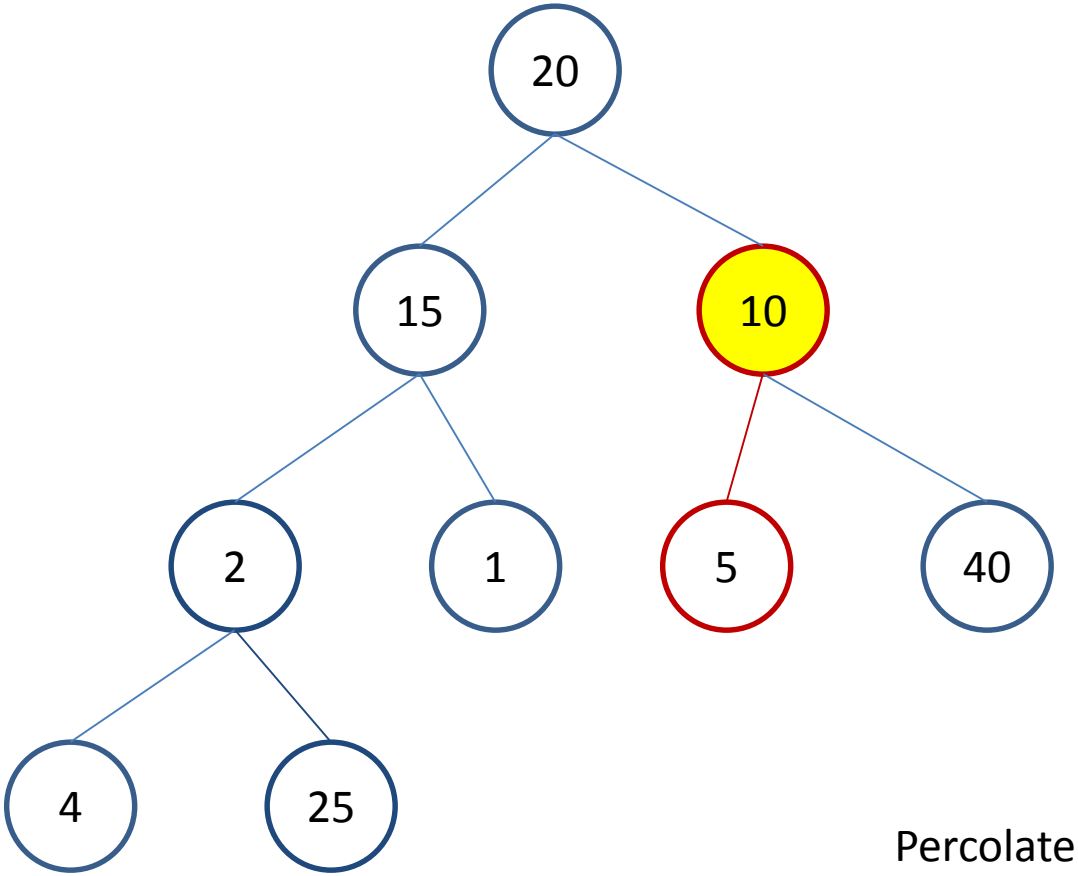


Percolate down !

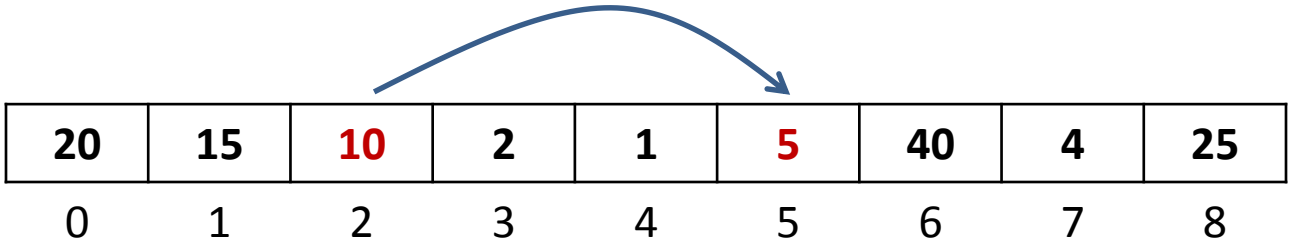


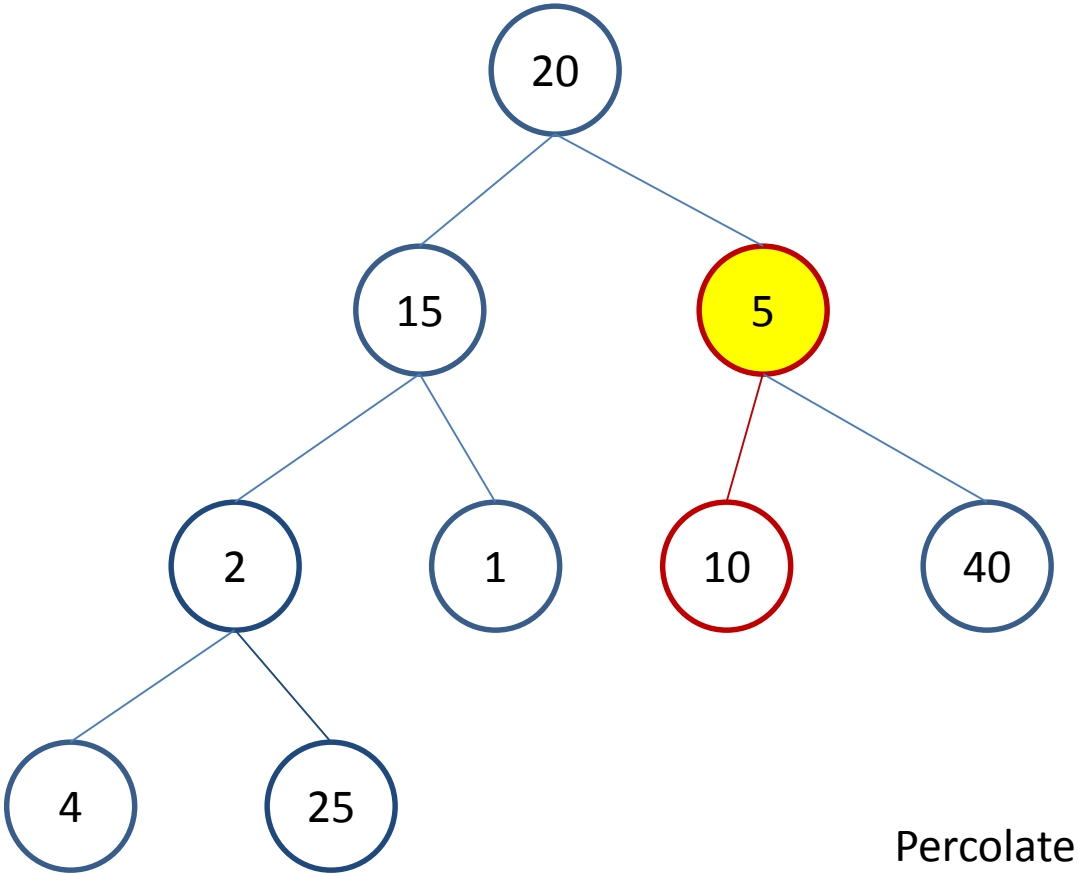

20
15
10
2
1
5
40
4
25


20 15 10 2 1 5 40 4 25
0 1 2 3 4 5 6 7 8

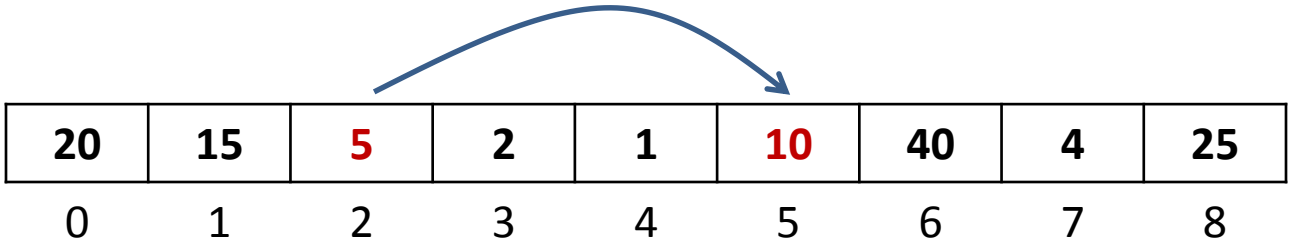


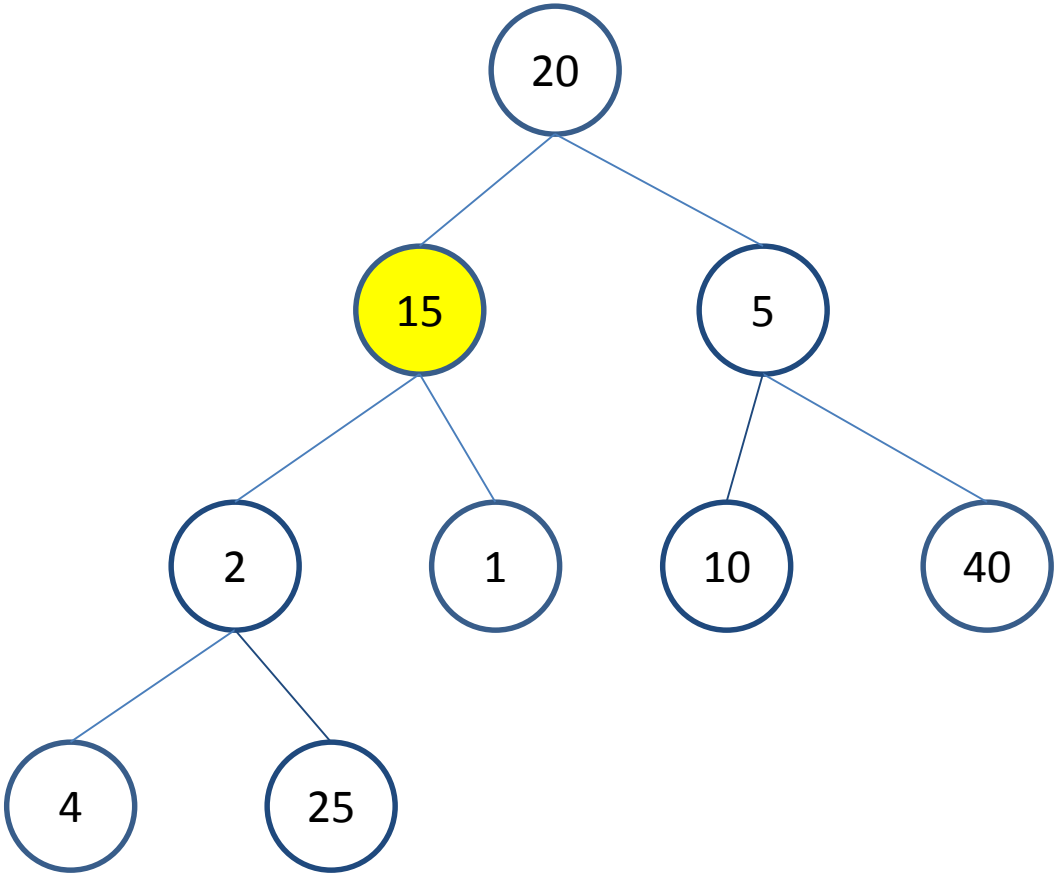
Percolate down !



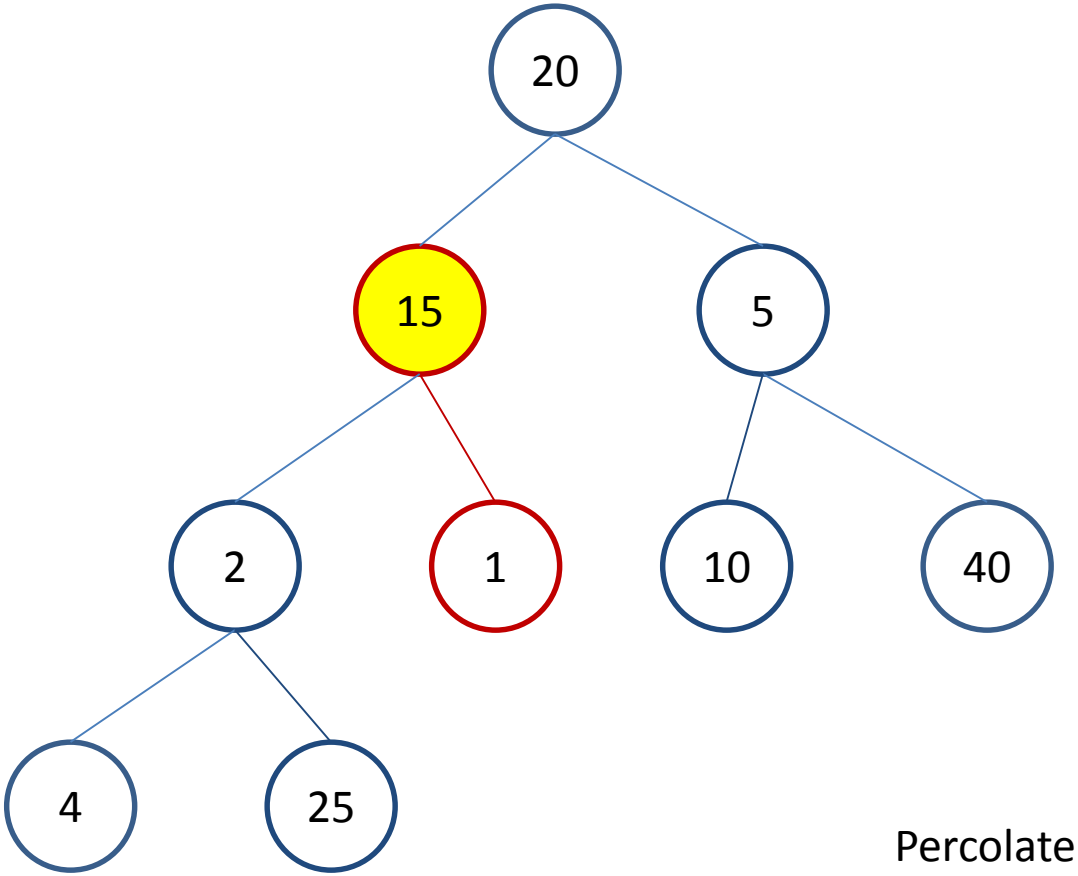


Percolate down !

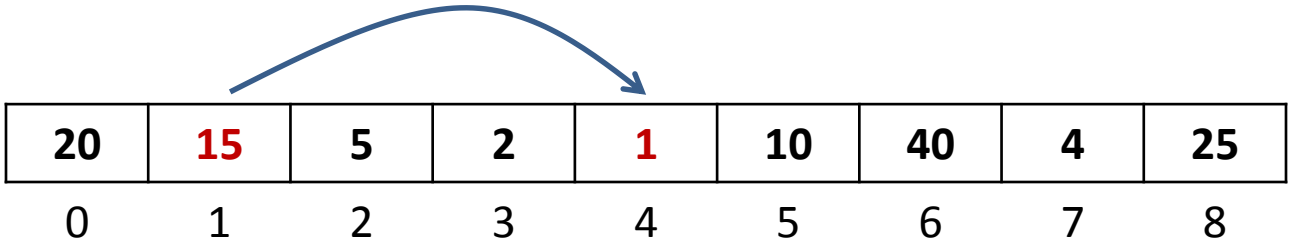


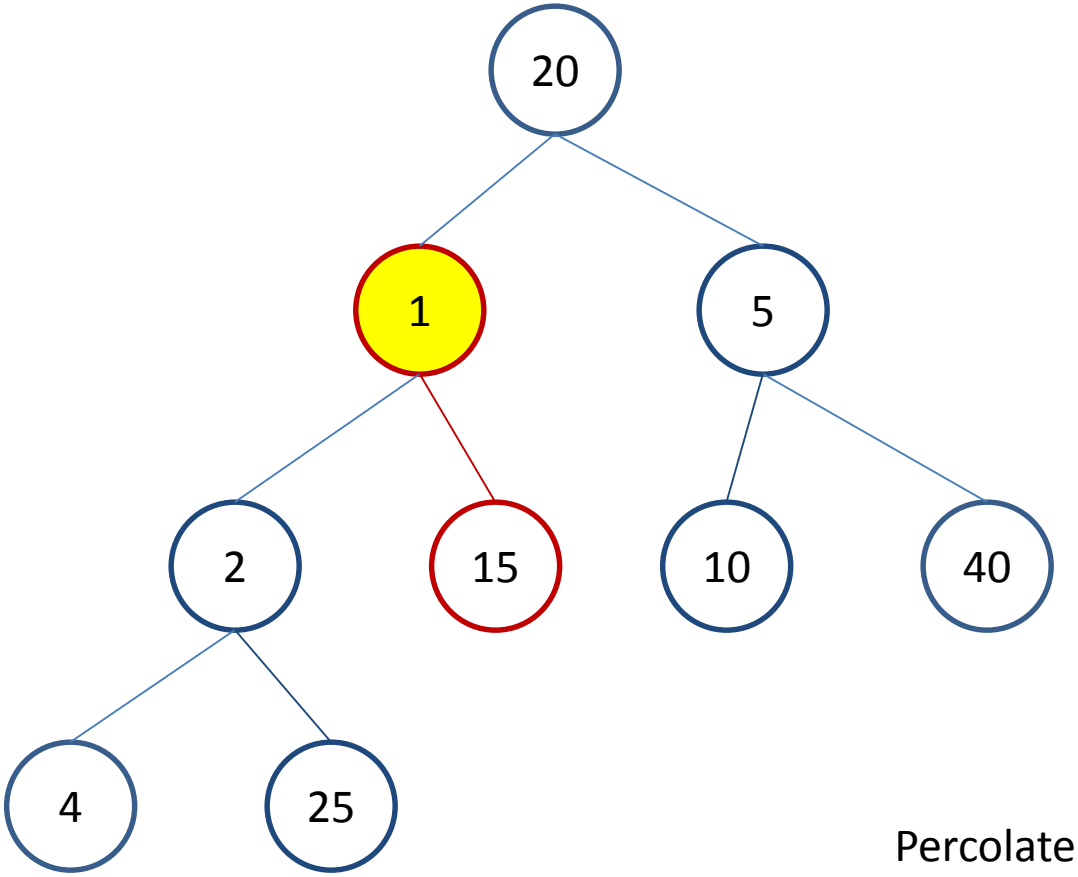


20	15	5	2	1	10	40	4	25
0	1	2	3	4	5	6	7	8

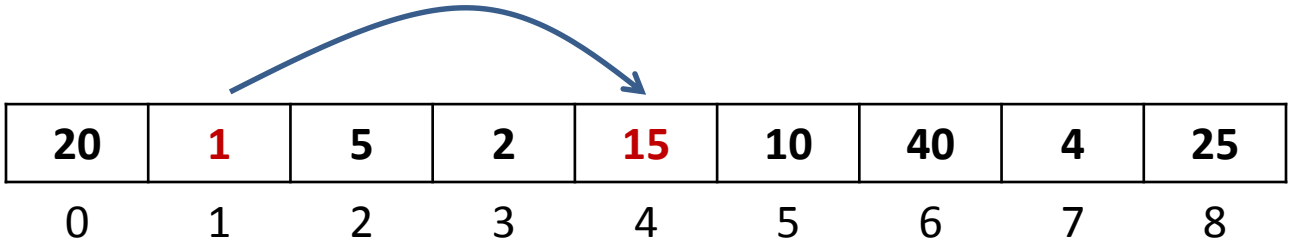


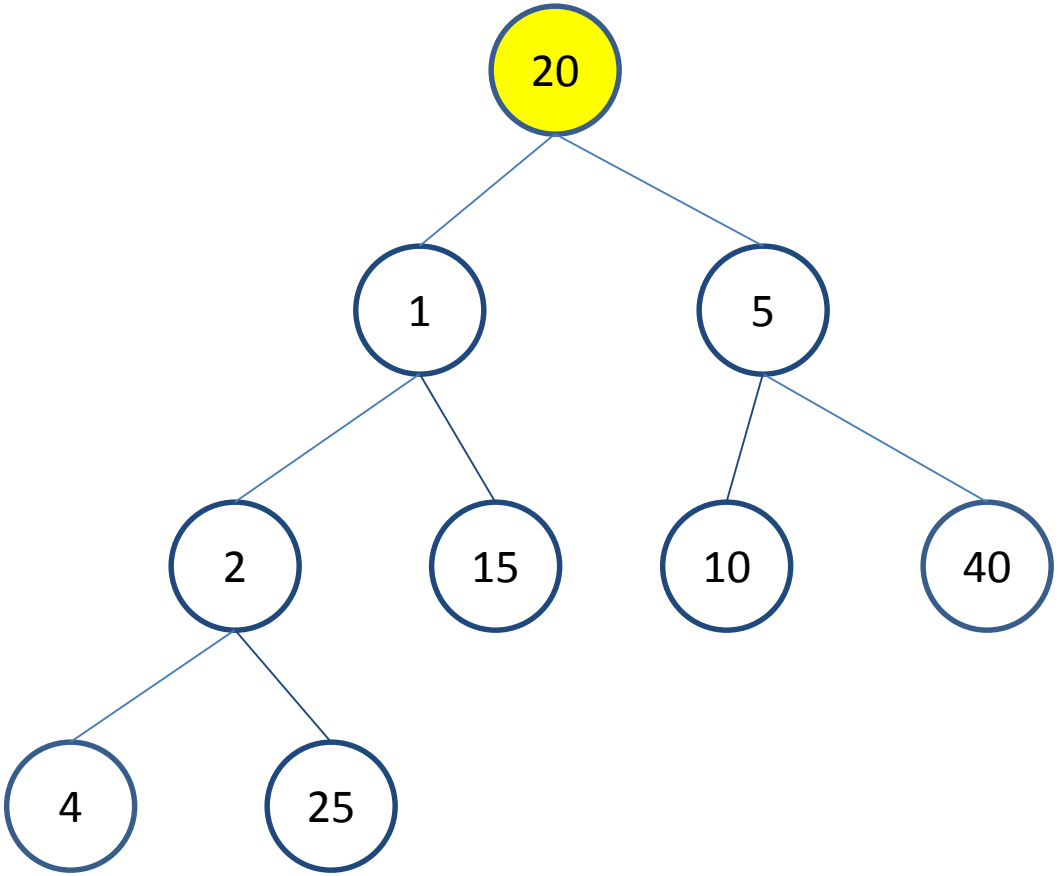
Percolate down !



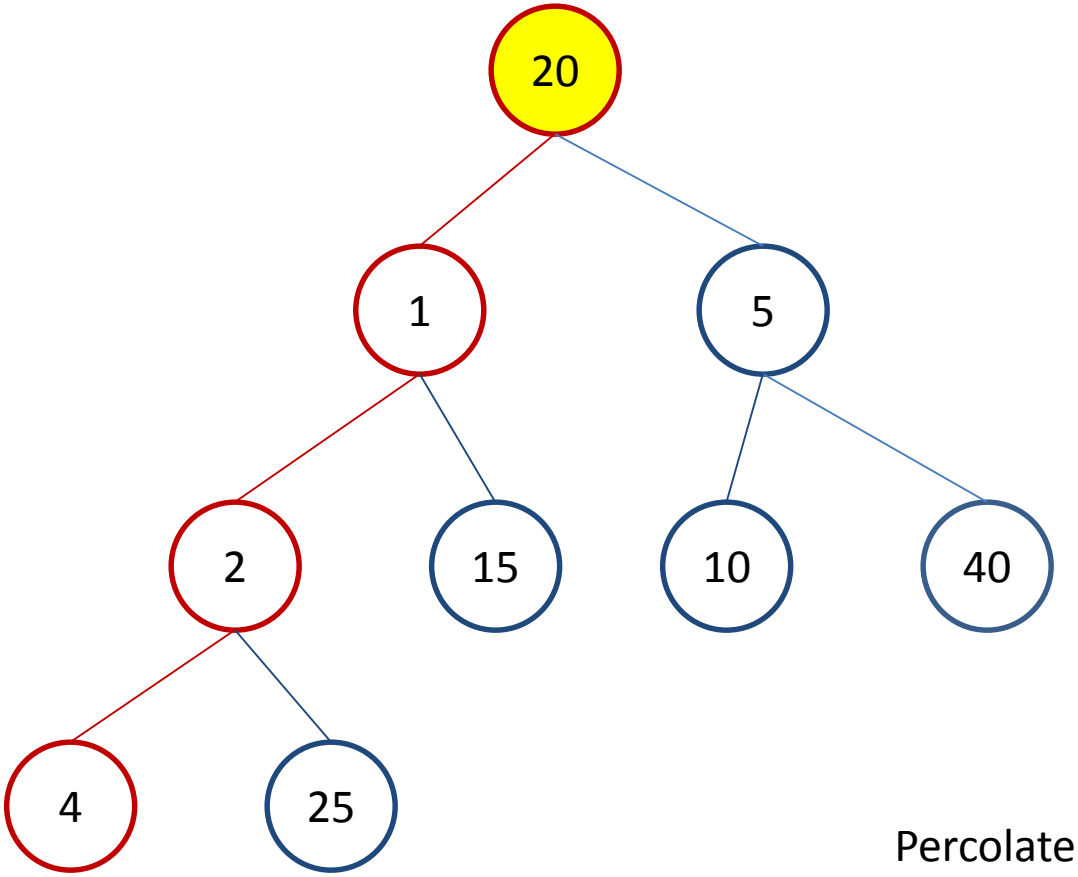


Percolate down !

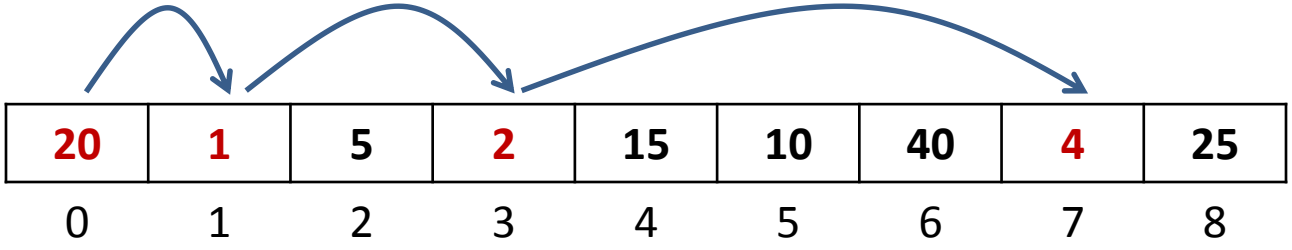


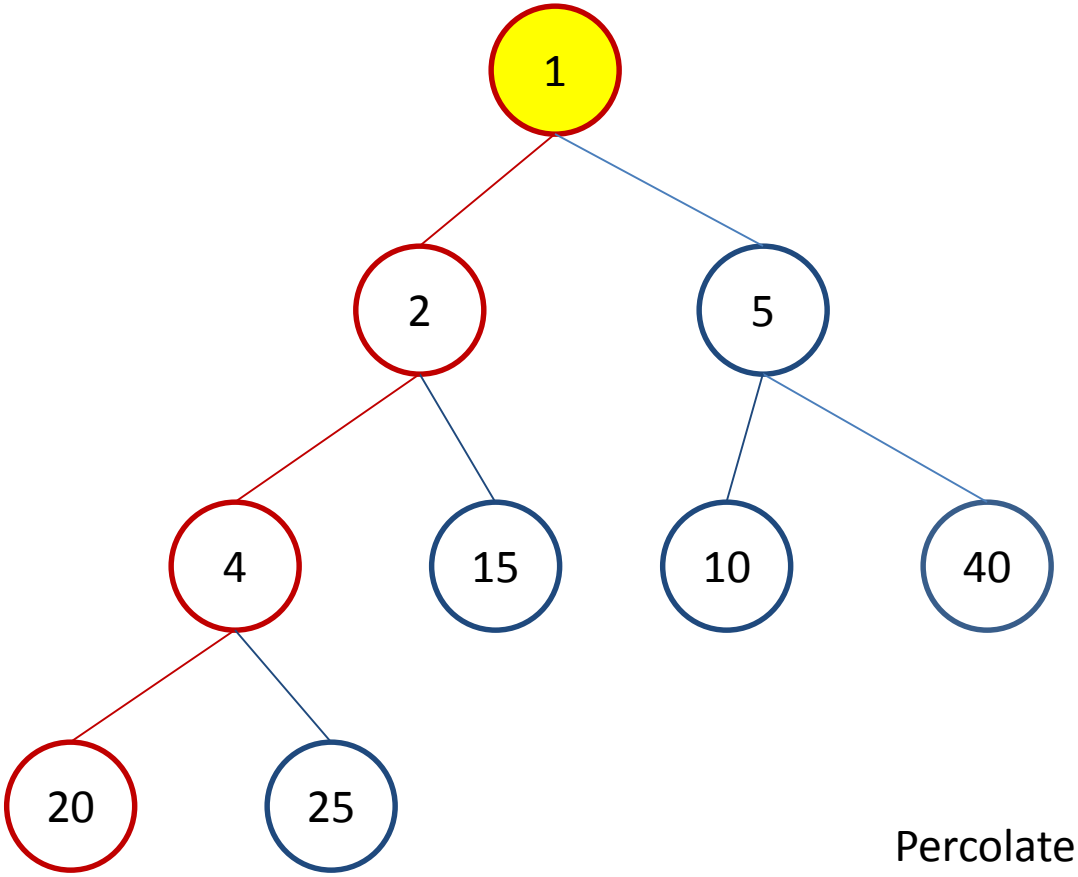


20	1	5	2	15	10	40	4	25
0	1	2	3	4	5	6	7	8

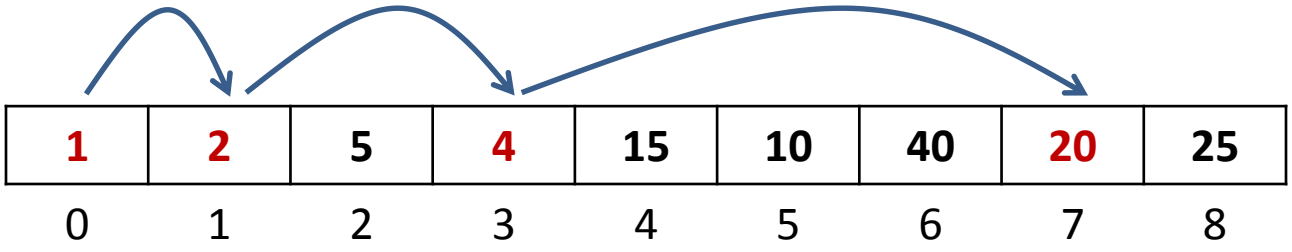


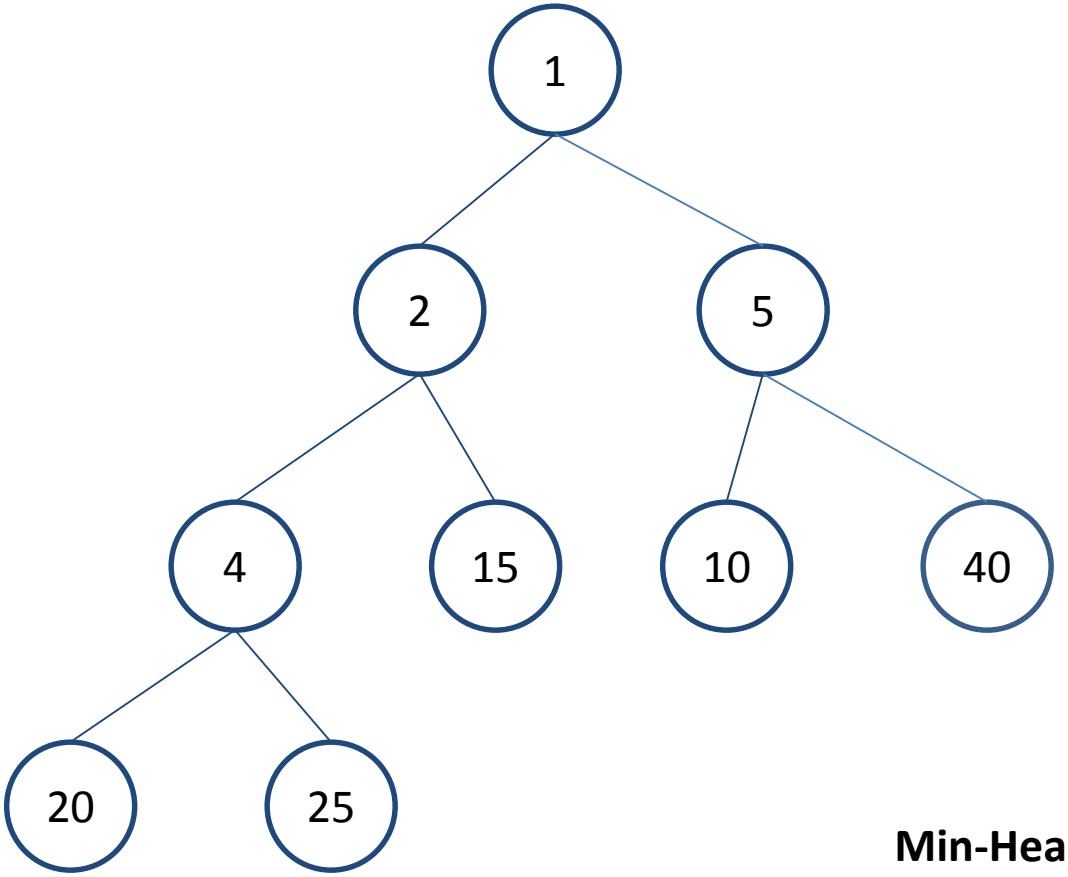
Percolate down !





Percolate down !





Min-Heap terbentuk !

1	**2**	**5**	**4**	**15**	**10**	**40**	**20**	**25**
0	1	2	3	4	5	6	7	8

Heap Sort

Input: arbitrary array

- Bangun heap dengan array tersebut (heapify)
- Lakukan removeMin/Max terus-menerus hingga array terurut

- Kompleksitas Waktu O(n log n)

Heap Sort

- Gunakan Min-Heap untuk Descending Sorting
- Gunakan Max-Heap untuk Ascending Sorting

Heap Sort

- Urutkan array berikut (descending) !

20	15	10	25	1	5	40	4	2
0	1	2	3	4	5	6	7	8

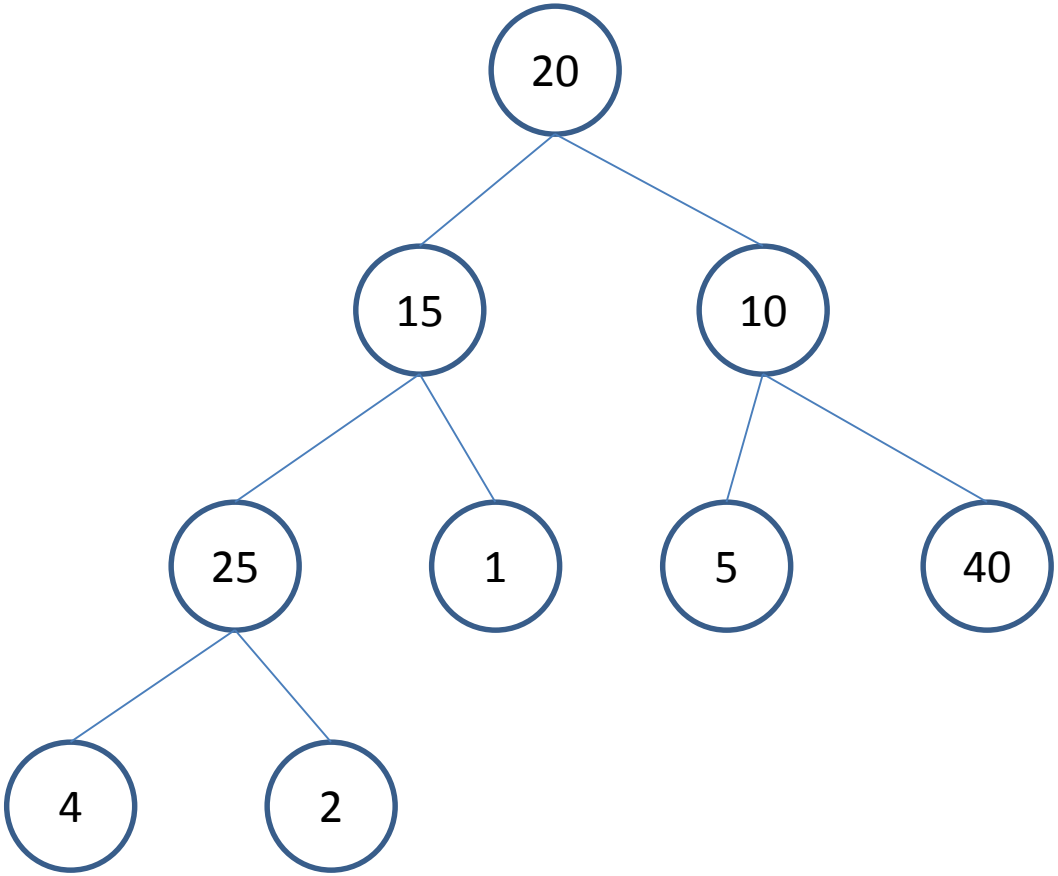
*Gunakan **Min-Heap**
*Array yang sama dengan simulasi heapify di slide sebelumnya !

- #1 lakukan heapify !

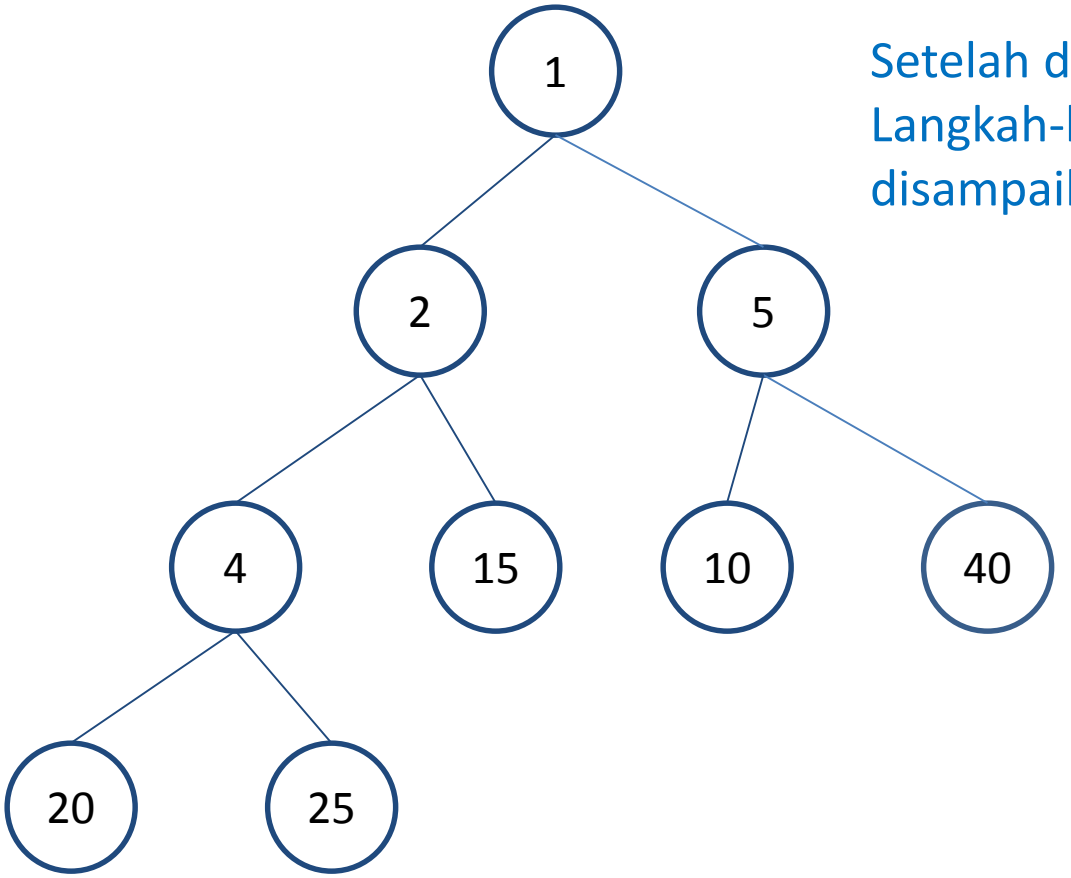
20	**15**	**10**	**25**	**1**	**5**	**40**	**4**	**2**
0	1	2	3	4	5	6	7	8

*Gunakan **Min-Heap**
*Array yang sama dengan simulasi heapify di slide sebelumnya !

Kondisi Sebelum heapify !

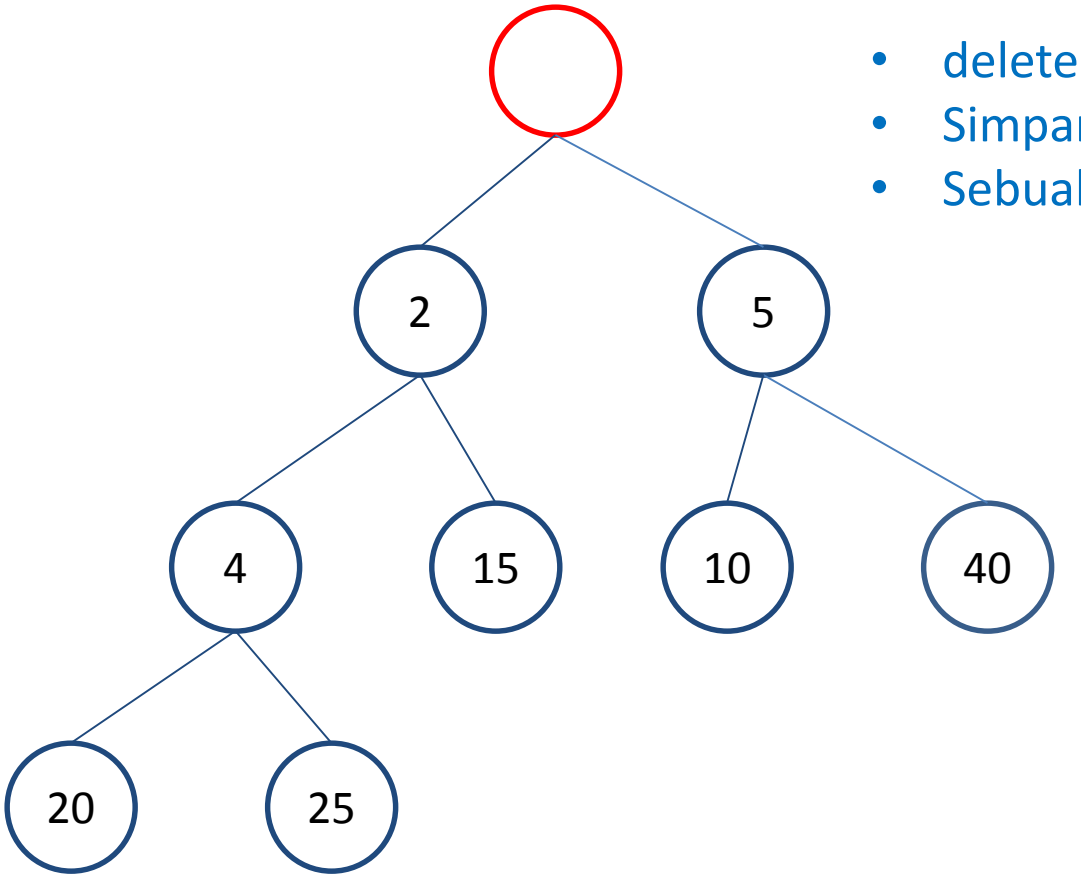


20	15	10	25	1	5	40	4	2
0	1	2	3	4	5	6	7	8



Setelah dilakukan Heapify !
Langkah-langkah sama dengan yang sudah disampaikan di slide-slide sebelumnya !

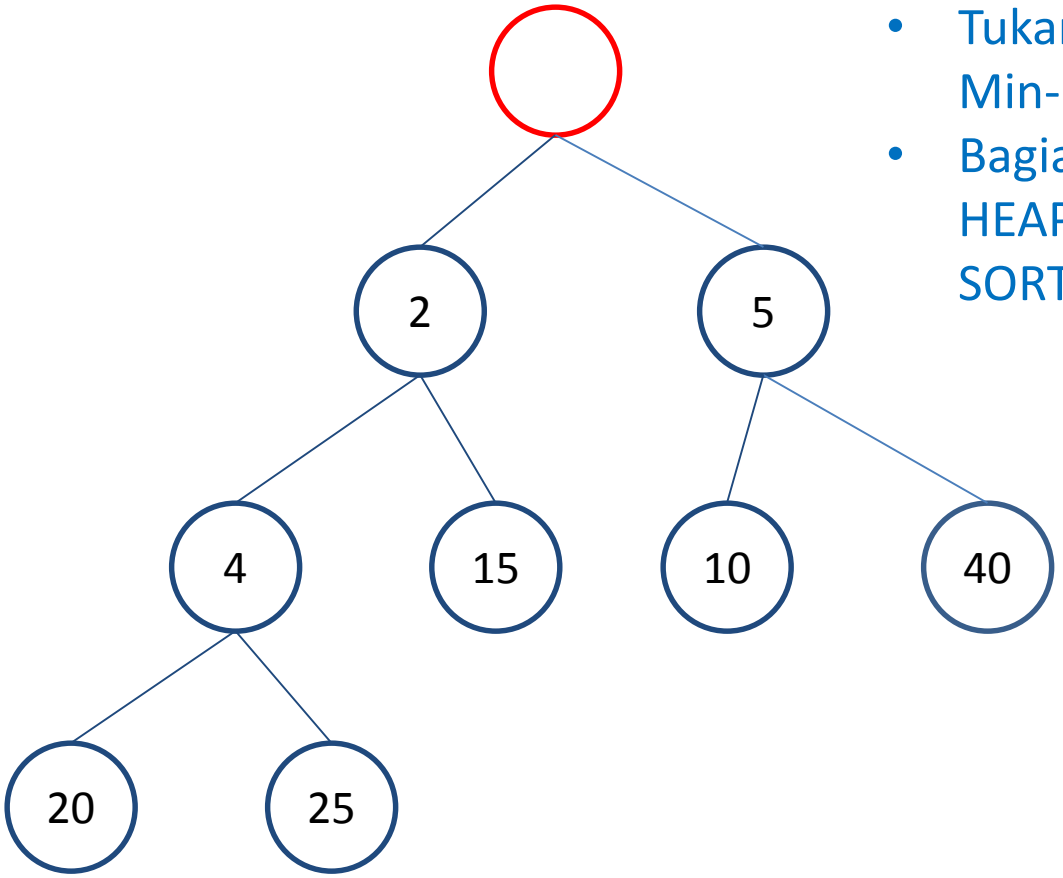
1	**2**	**5**	**4**	**15**	**10**	**40**	**20**	**25**
0	1	2	3	4	5	6	7	8



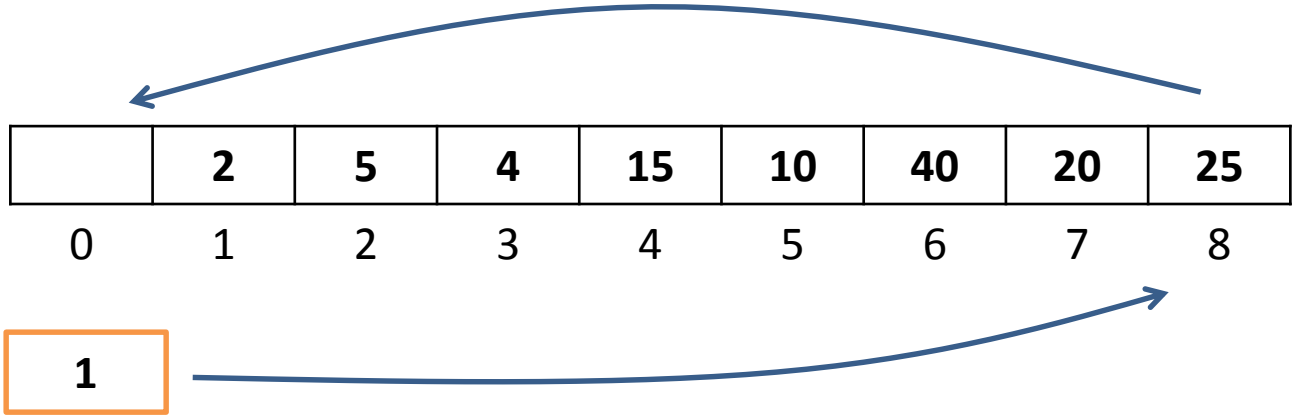
- deleteMin() atau top element 1
- Simpan 1 di temporary space
- Sebuah “lubang” terbentuk di posisi top

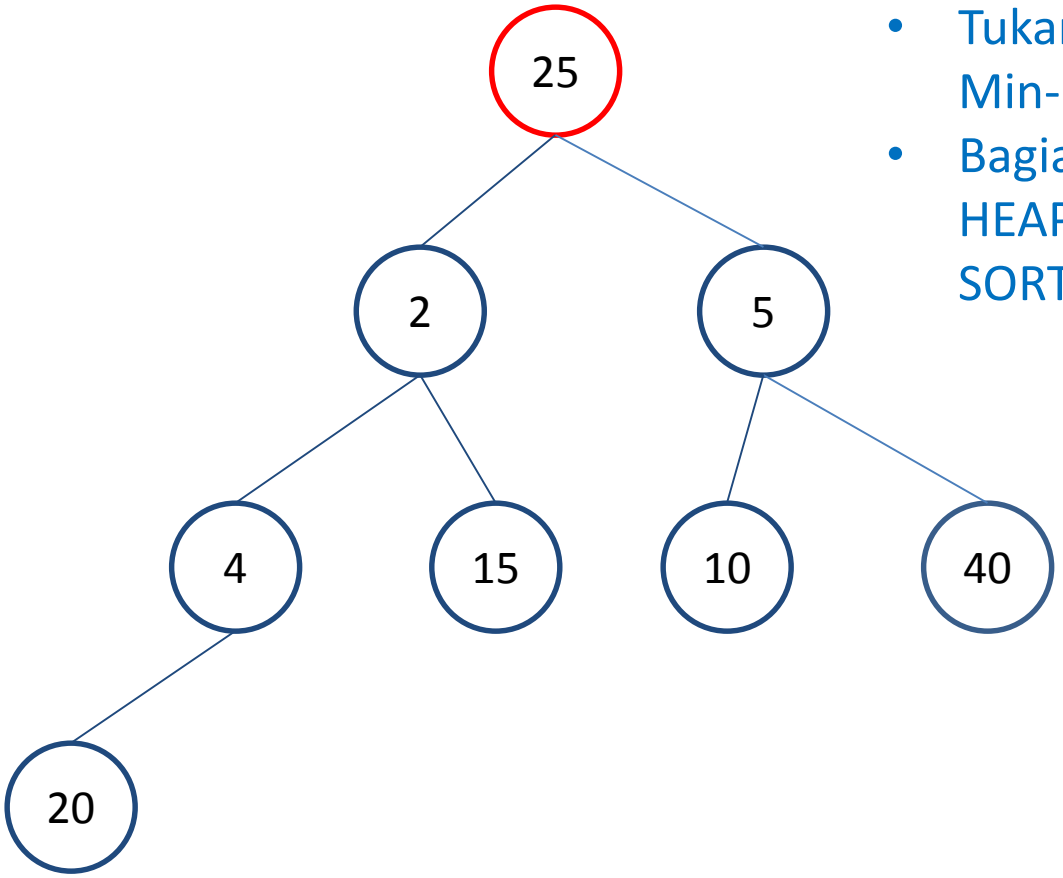
	2	**5**	**4**	**15**	**10**	**40**	**20**	**25**
0	1	2	3	4	5	6	7	8

1



- Tukar 1 dengan elemen terakhir pada Min-Heap
- Bagian akhir array sudah BUKAN bagian HEAP, tetapi sudah menjadi bagian SORTED ARRAY !

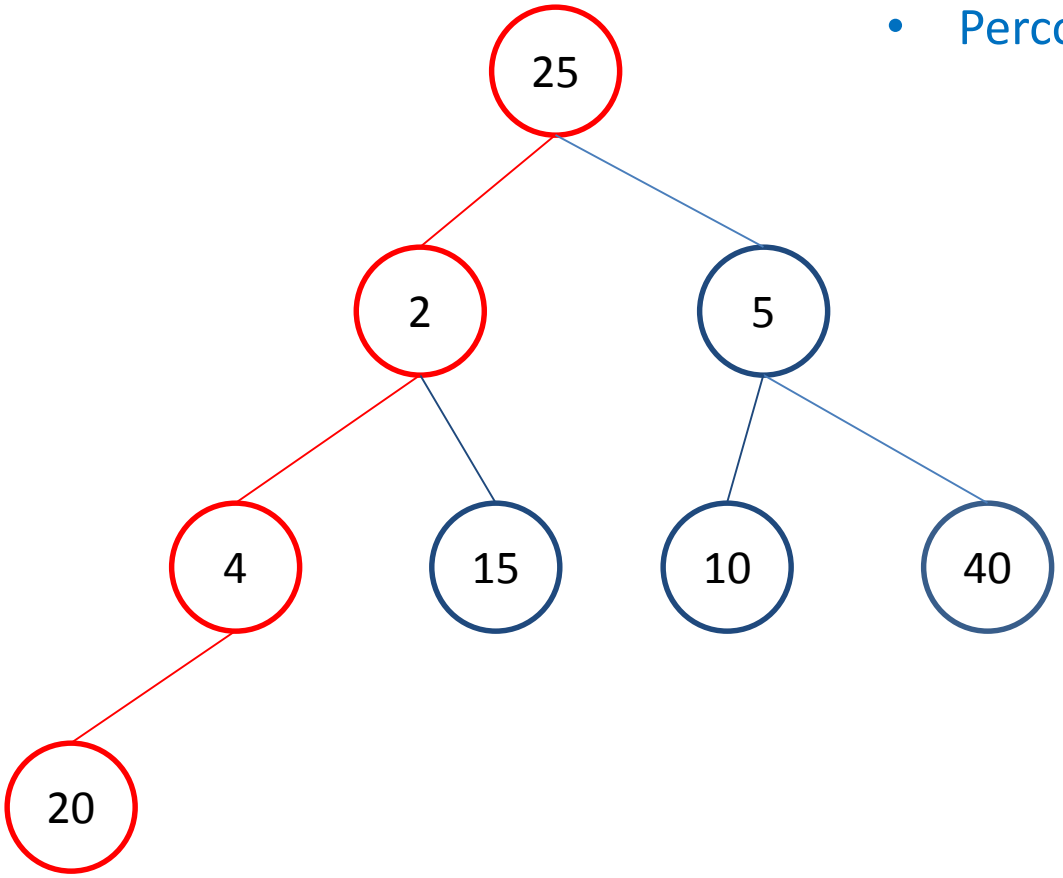


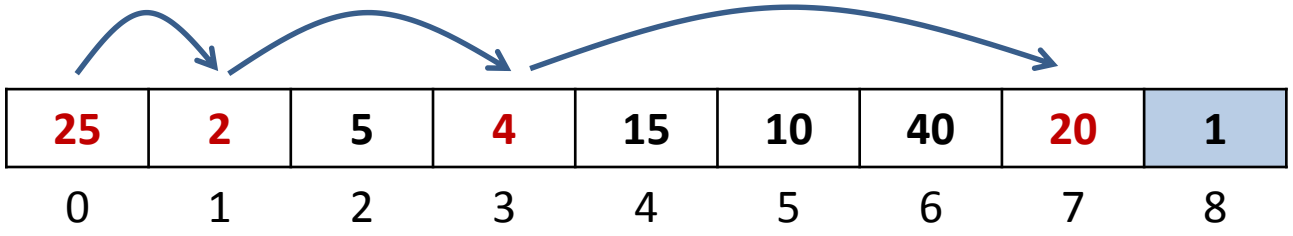


- Tukar 1 dengan elemen terakhir pada Min-Heap
- Bagian akhir array sudah BUKAN bagian HEAP, tetapi sudah menjadi bagian SORTED ARRAY !

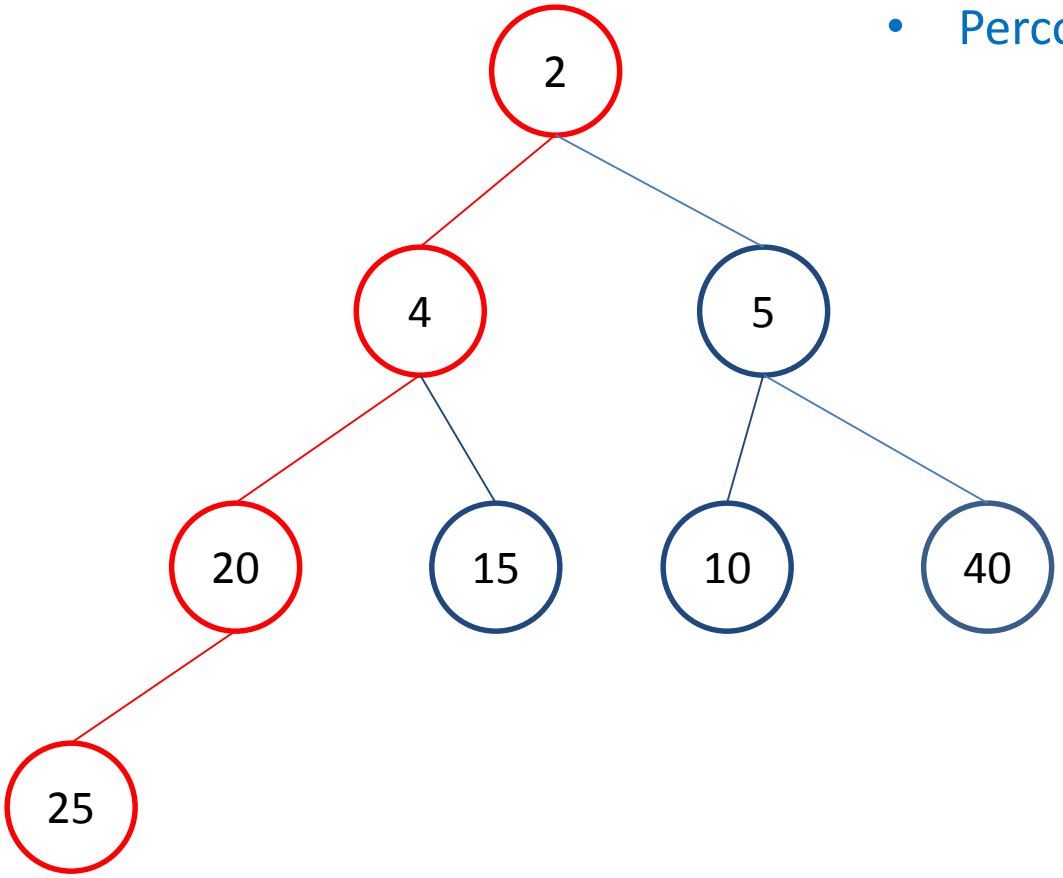
25	**2**	**5**	**4**	**15**	**10**	**40**	**20**	**1**
0	1	2	3	4	5	6	7	8

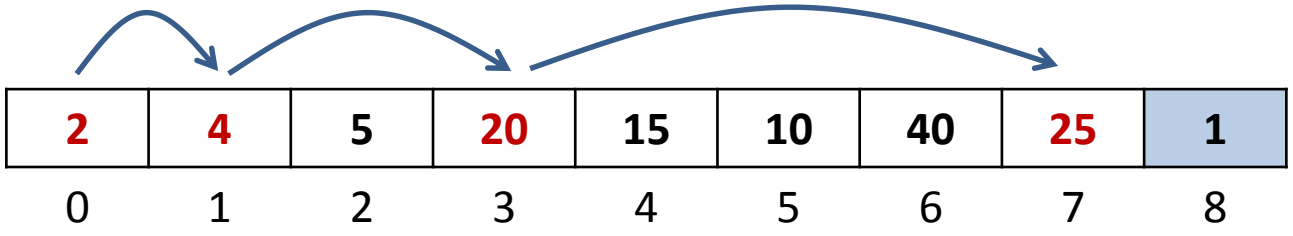
- Percolate down dari root



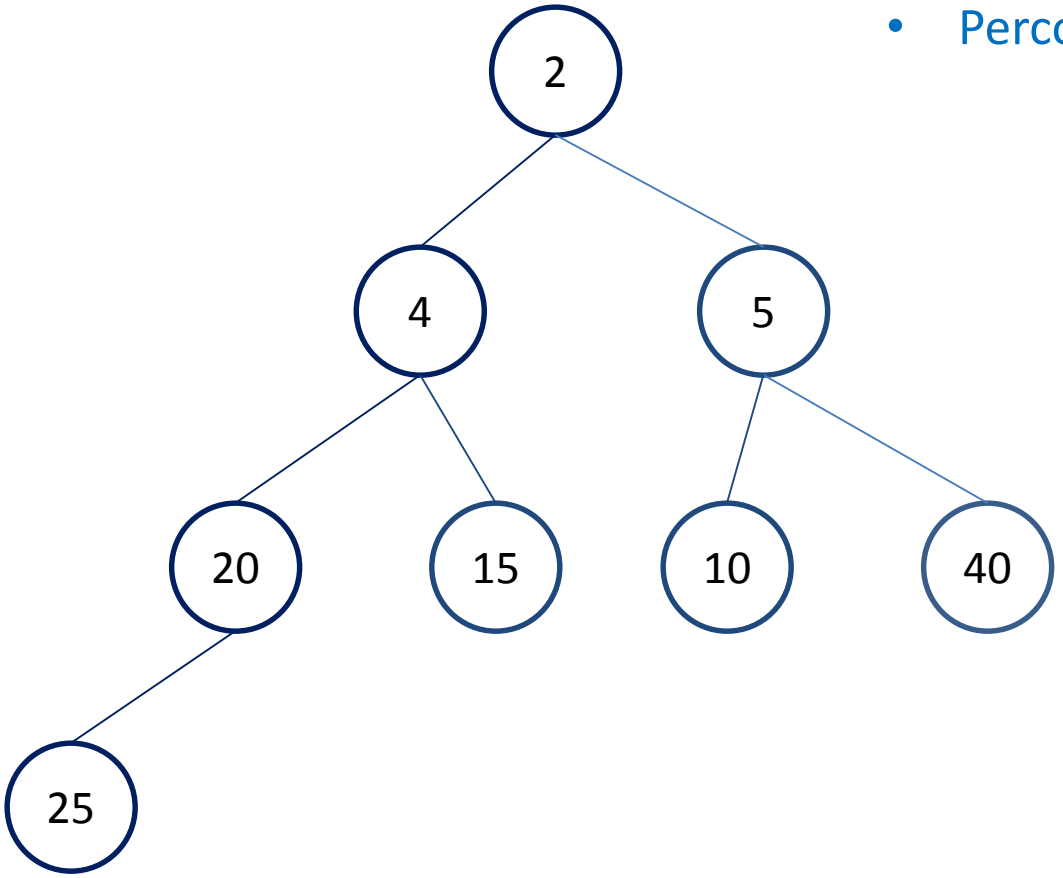


- Percolate down dari root

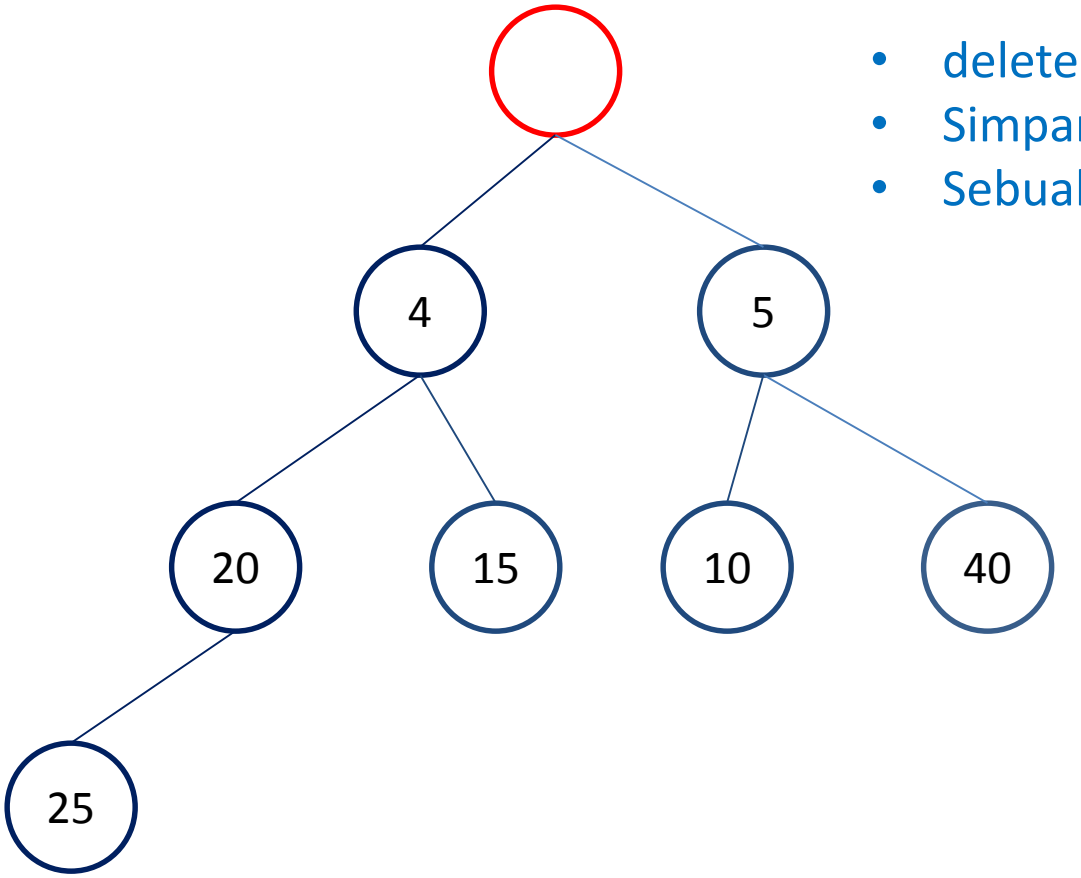




- Percolate down dari root



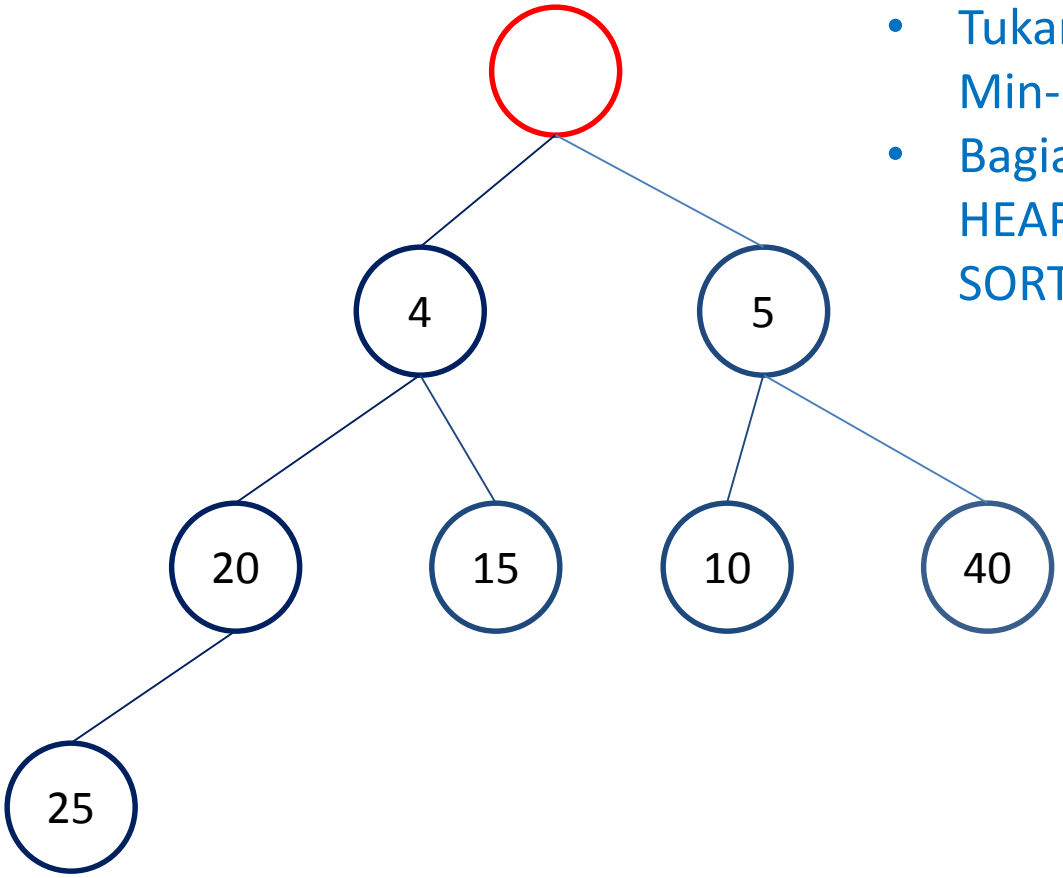
2	4	5	20	15	10	40	25	1
0	1	2	3	4	5	6	7	8



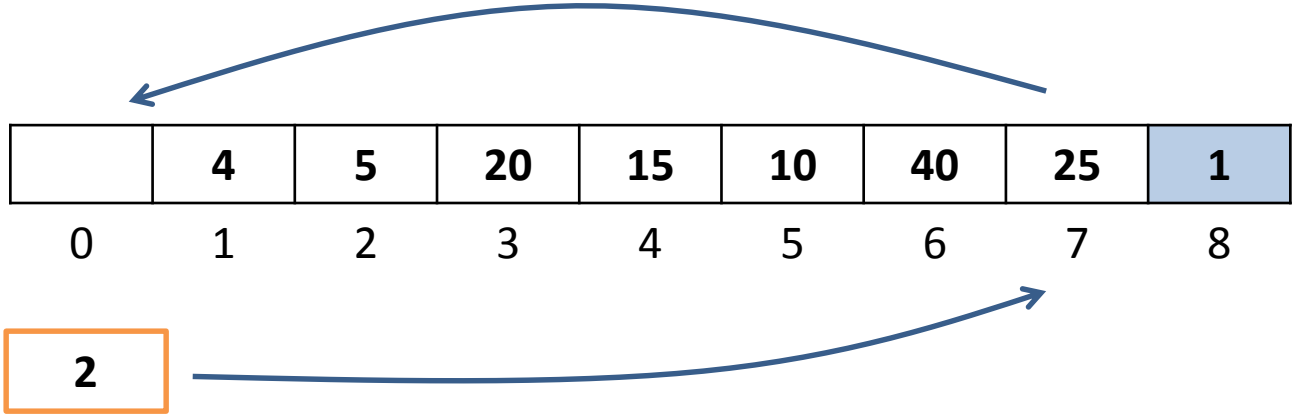
- deleteMin() atau top element 2
- Simpan 2 di temporary space
- Sebuah “lubang” terbentuk di posisi top

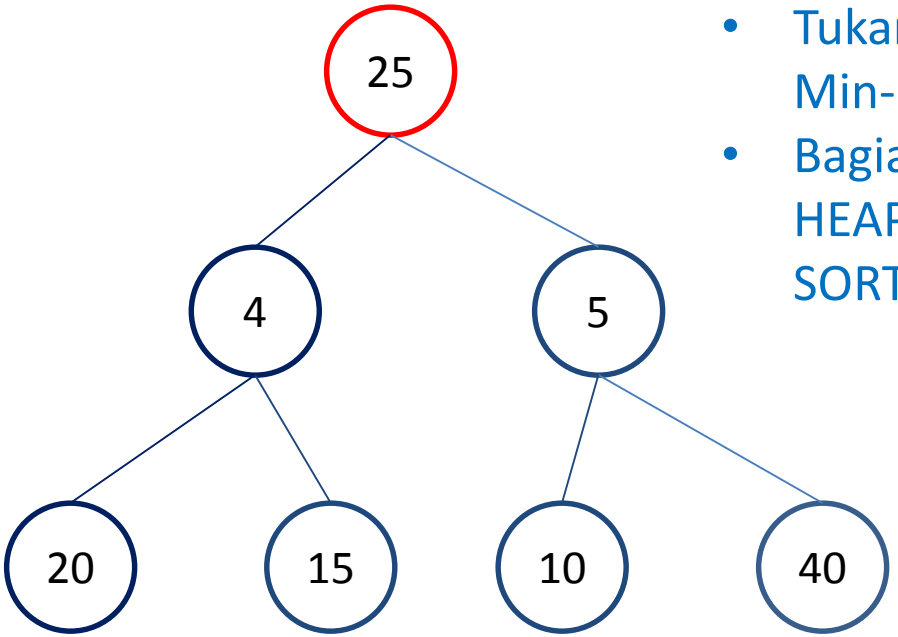
	4	**5**	**20**	**15**	**10**	**40**	**25**	**1**
0	1	2	3	4	5	6	7	8

2



- Tukar 2 dengan elemen terakhir pada Min-Heap
- Bagian akhir array sudah BUKAN bagian HEAP, tetapi sudah menjadi bagian SORTED ARRAY !

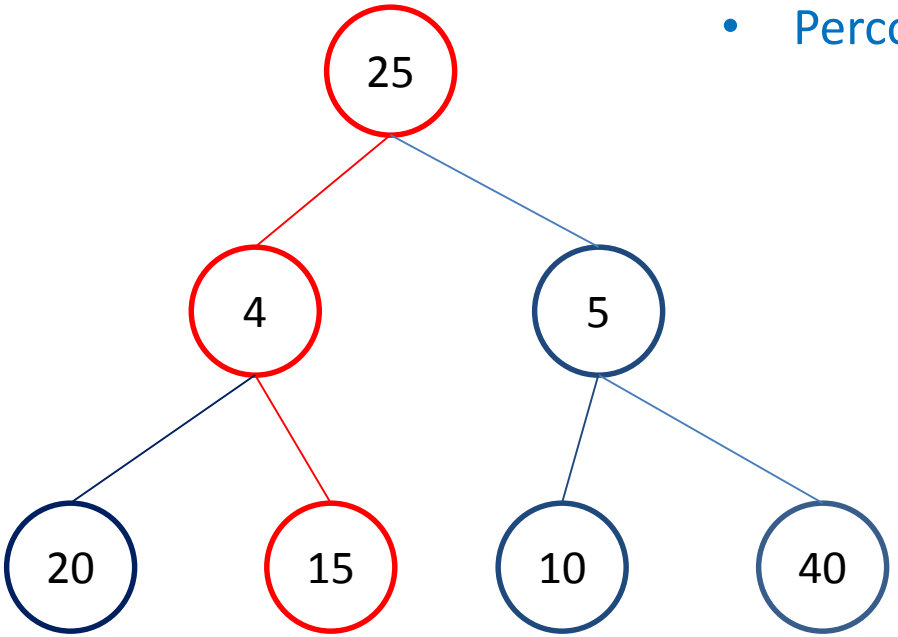


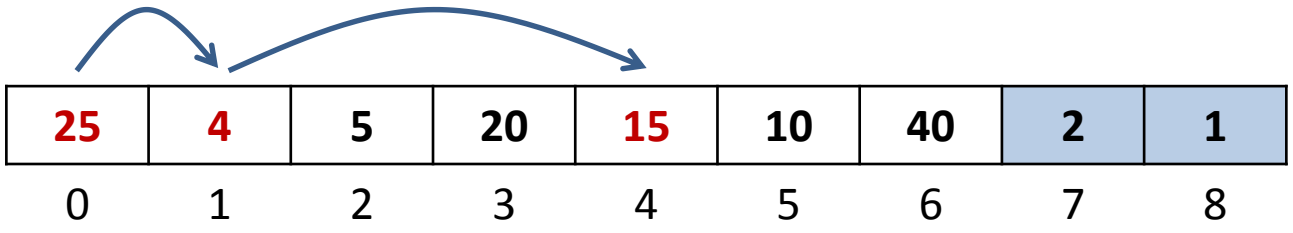


- Tukar 2 dengan elemen terakhir pada Min-Heap
- Bagian akhir array sudah BUKAN bagian HEAP, tetapi sudah menjadi bagian SORTED ARRAY !

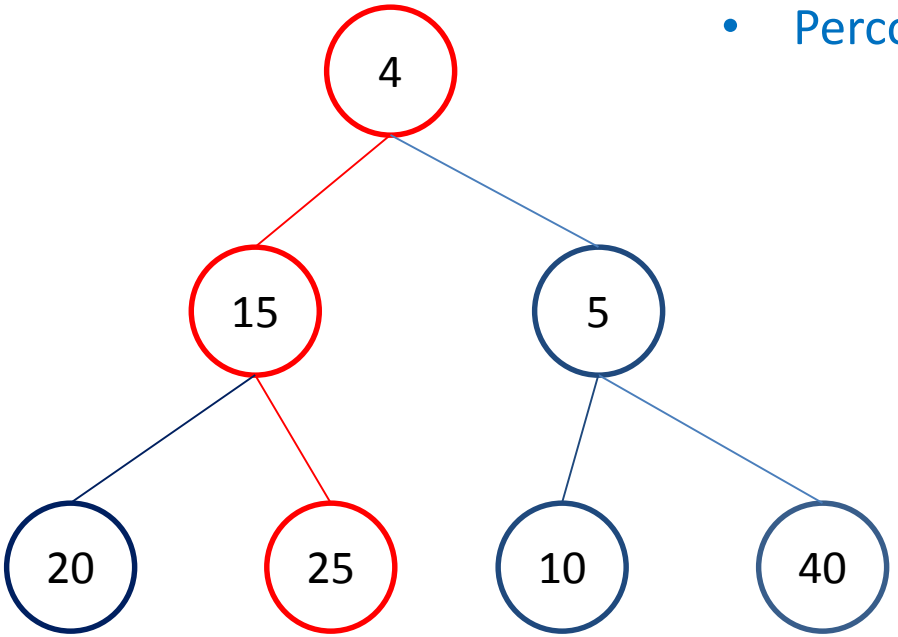
25	**4**	**5**	**20**	**15**	**10**	**40**	**2**	**1**
0	1	2	3	4	5	6	7	8

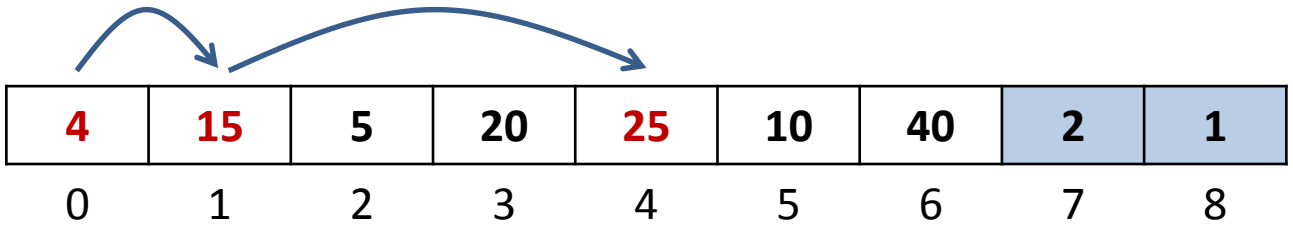
- Percolate down dari root



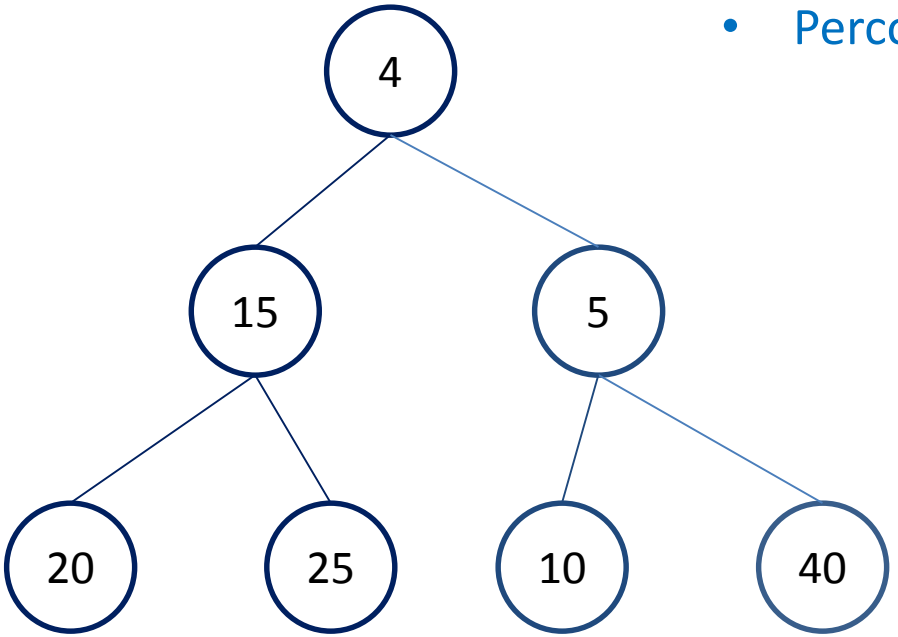


- Percolate down dari root

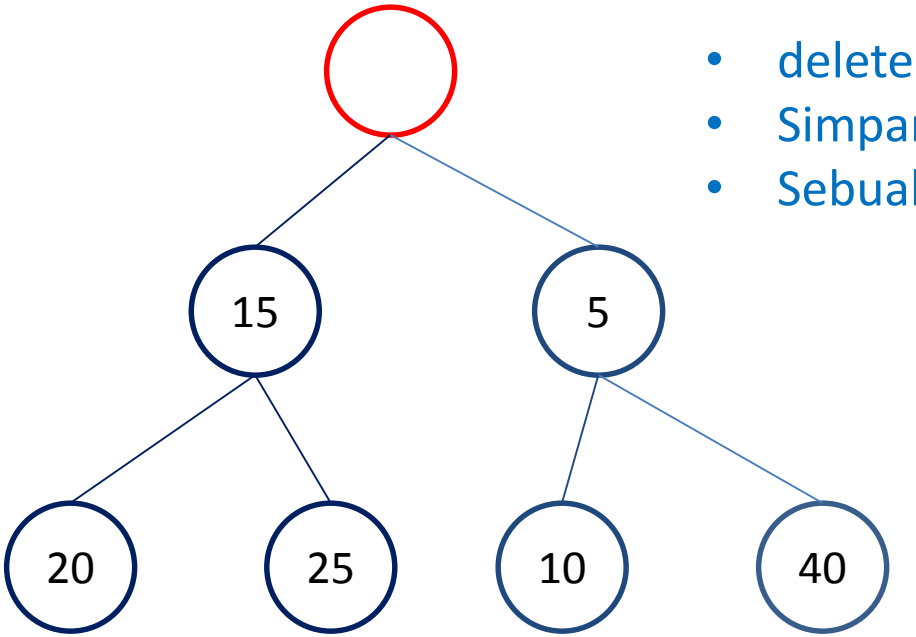




- Percolate down dari root



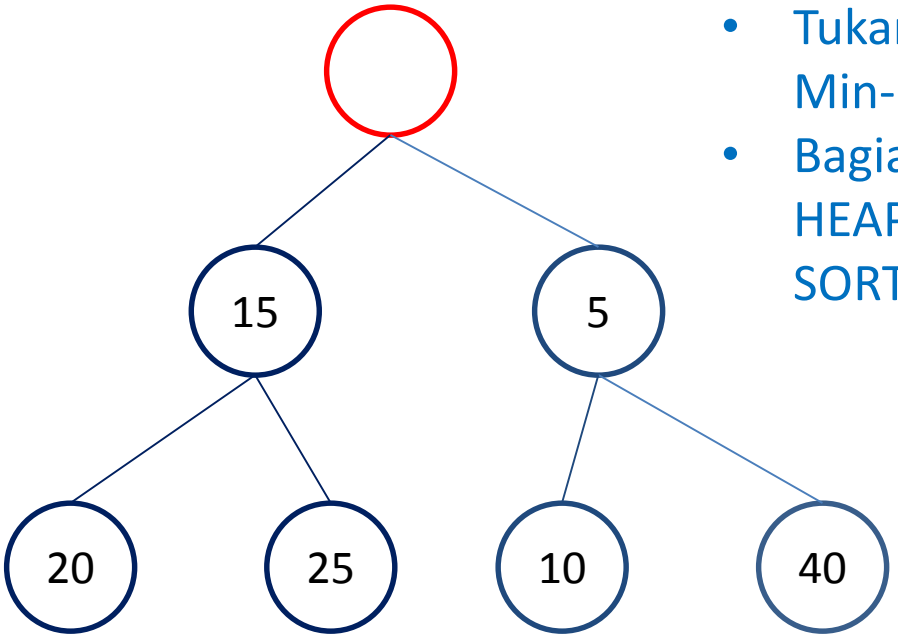
4	15	5	20	25	10	40	2	1
0	1	2	3	4	5	6	7	8



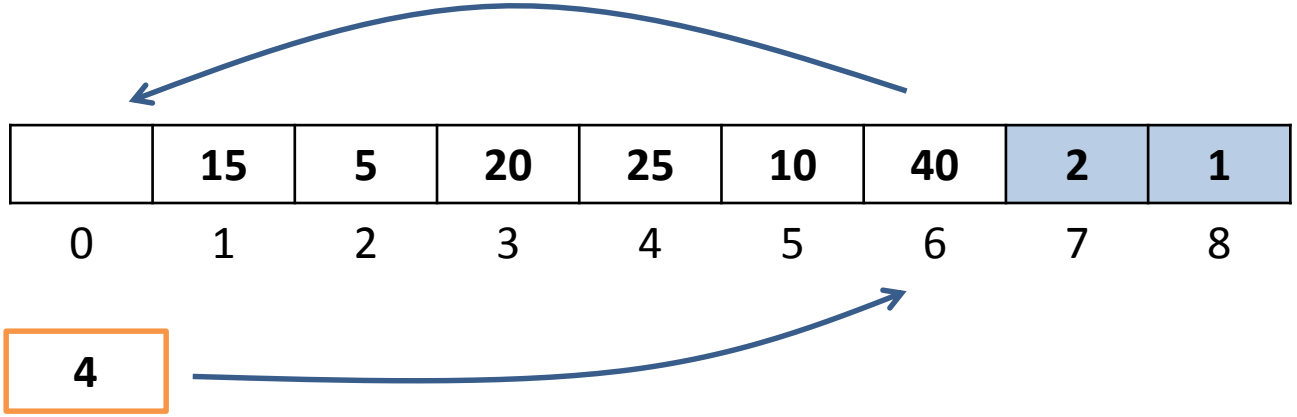
- deleteMin() atau top element 4
- Simpan 4 di temporary space
- Sebuah “lubang” terbentuk di posisi top

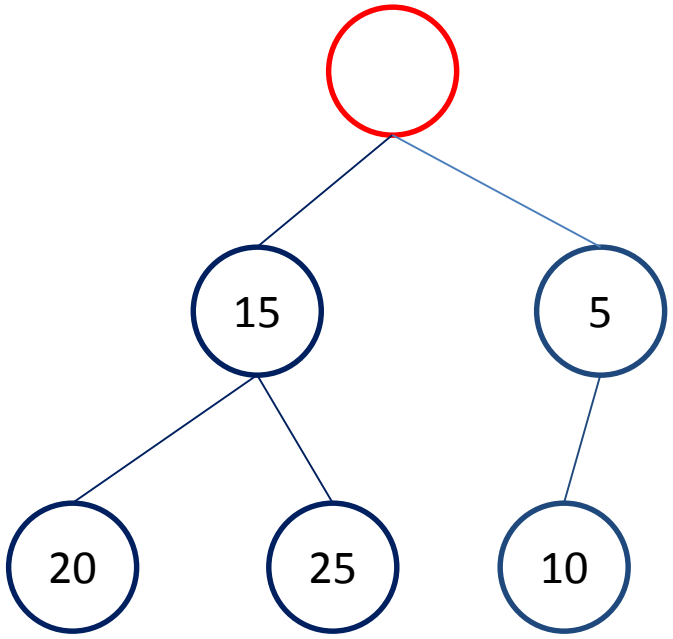
	15	**5**	**20**	**25**	**10**	**40**	**2**	**1**
0	1	2	3	4	5	6	7	8

4



- Tukar 4 dengan elemen terakhir pada Min-Heap
- Bagian akhir array sudah BUKAN bagian HEAP, tetapi sudah menjadi bagian SORTED ARRAY !





- Tukar 4 dengan elemen terakhir pada Min-Heap
- Bagian akhir array sudah BUKAN bagian HEAP, tetapi sudah menjadi bagian SORTED ARRAY !

40	**15**	**5**	**20**	**25**	**10**	**4**	**2**	**1**
0	1	2	3	4	5	6	7	8

- Lakukan hal yang sama hingga semua elemen pada Heap menjadi terurut ☺