Data Structures & Algorithms

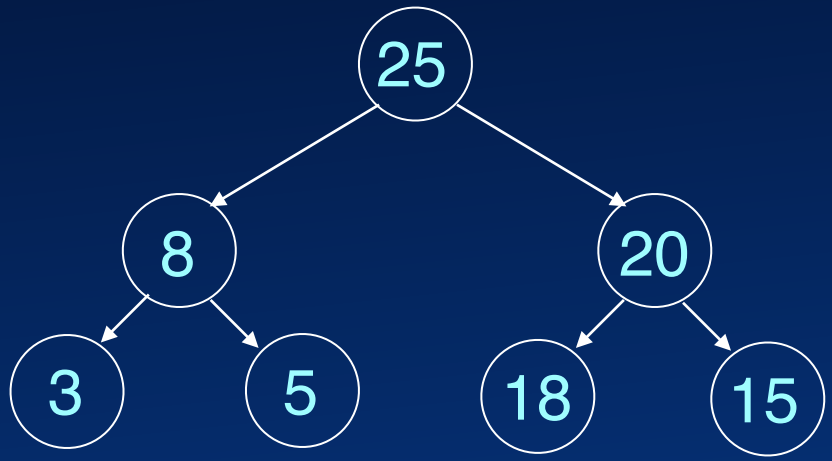
Subodh Kumar

(subodh@iitd.ac.in, Bharti 422)

Dept of Computer Sc. & Engg.

Heap

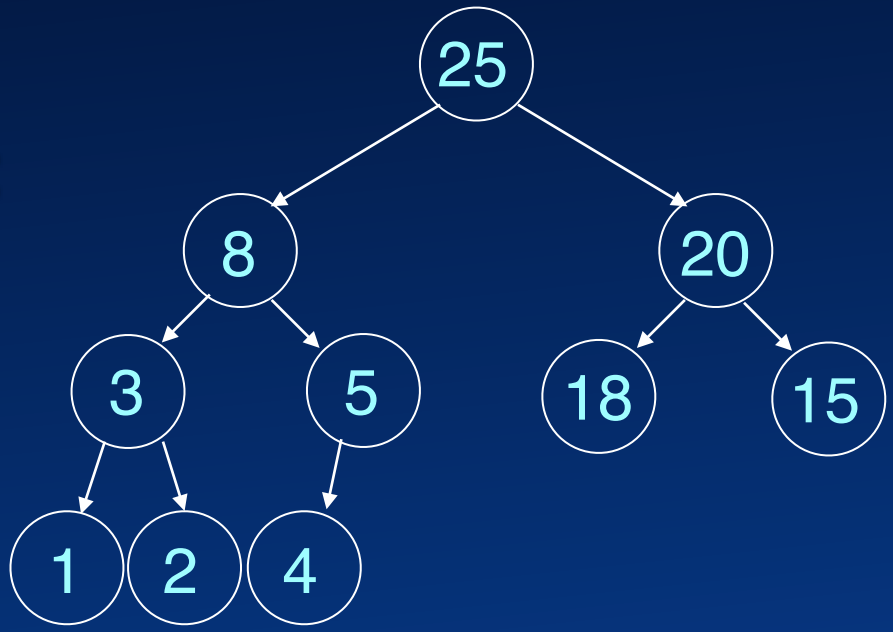
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

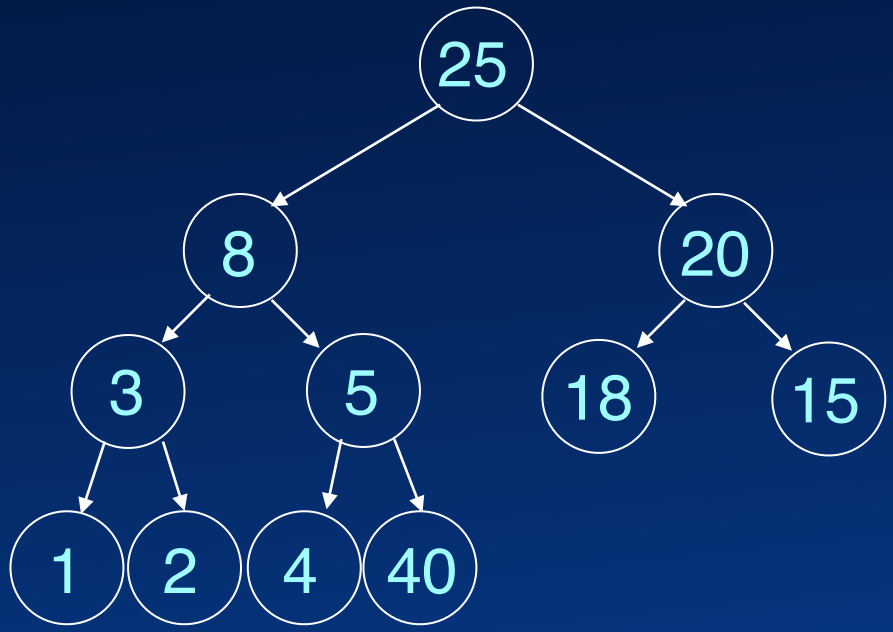
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

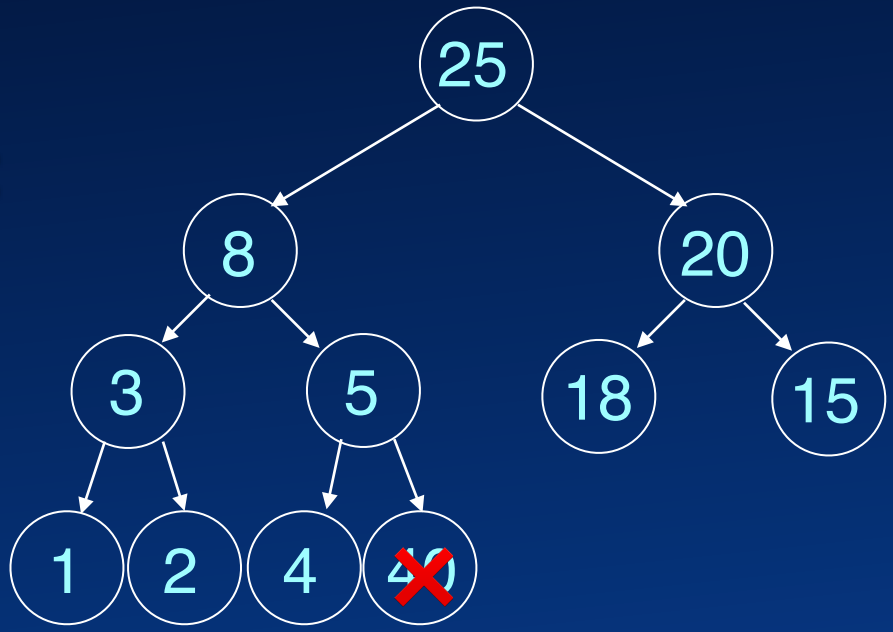
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

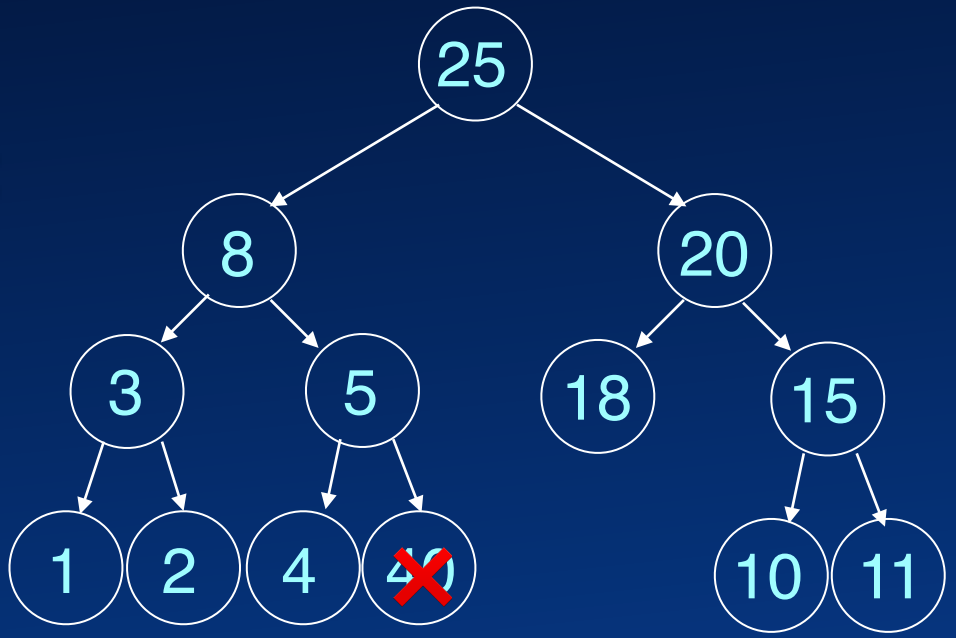
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

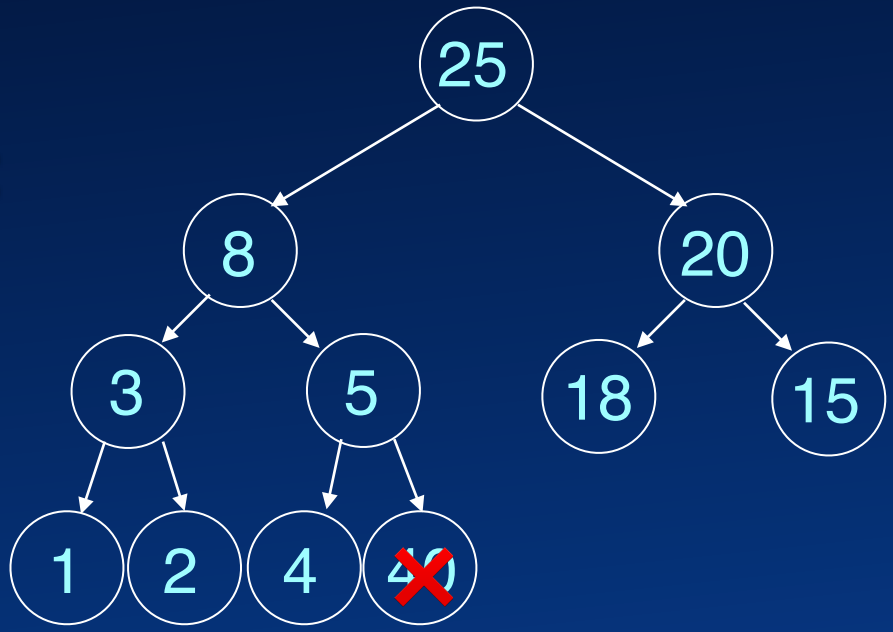
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

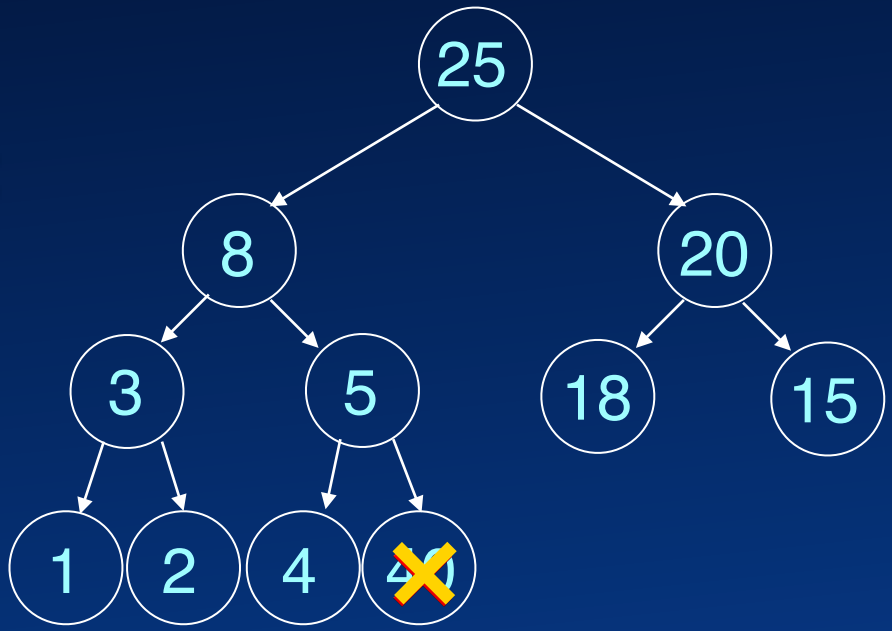
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

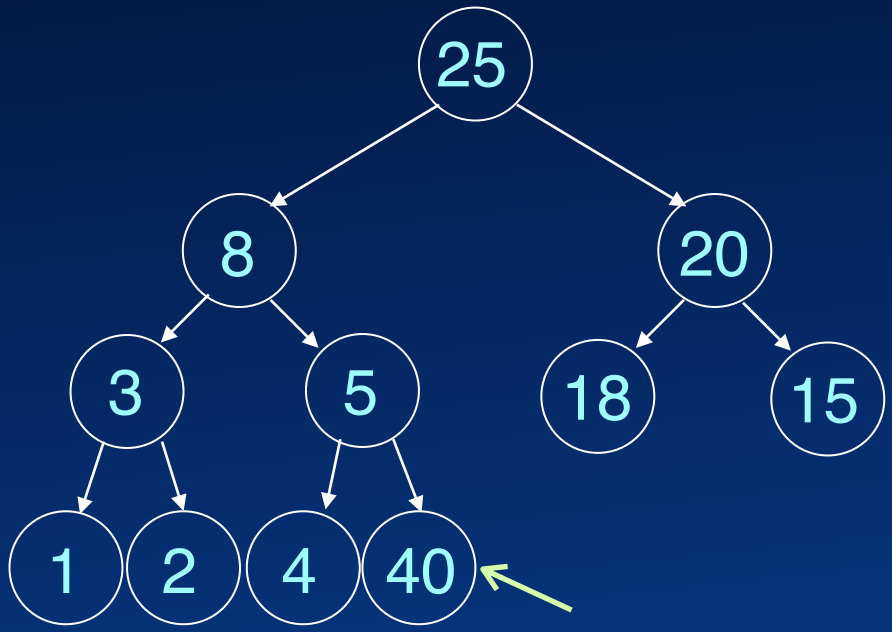
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

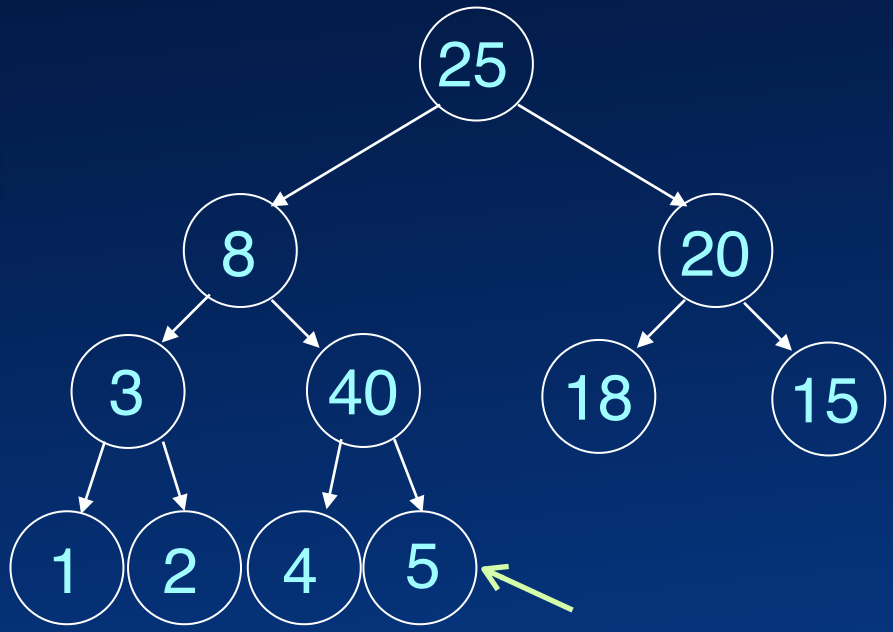
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

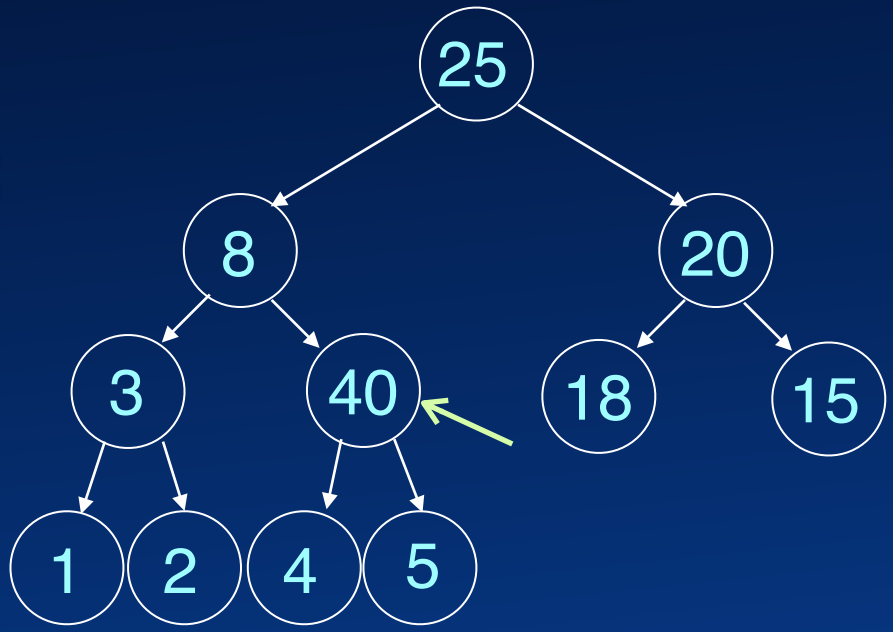
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

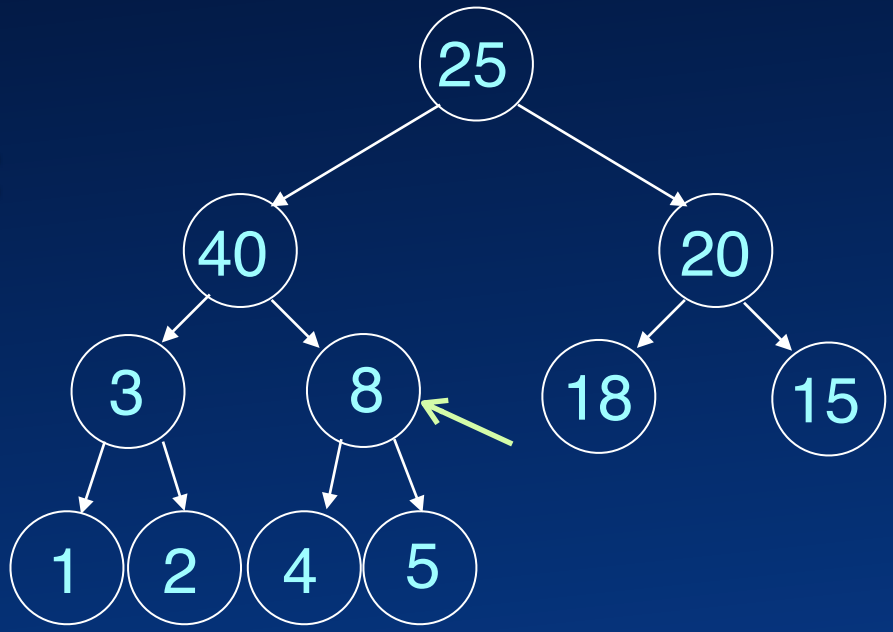
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

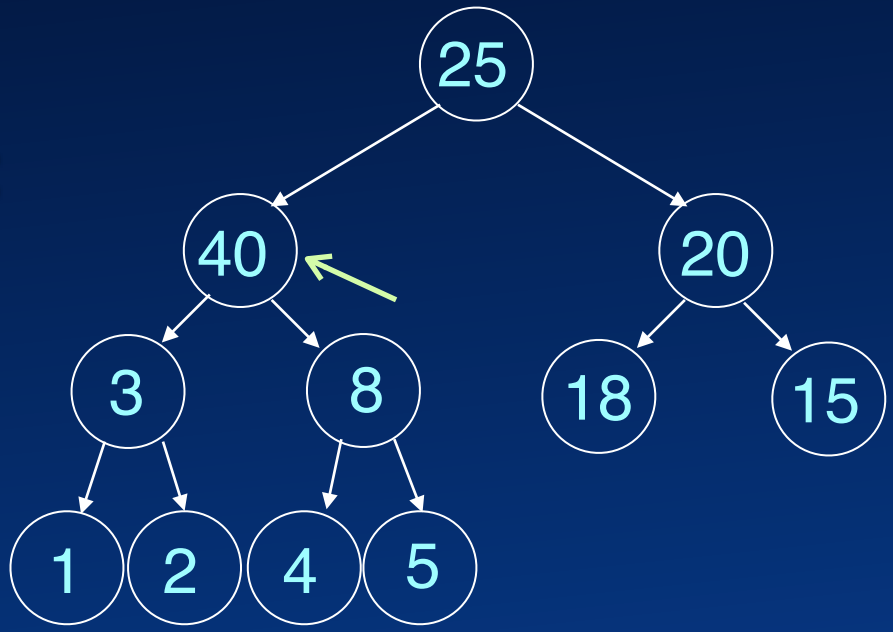
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

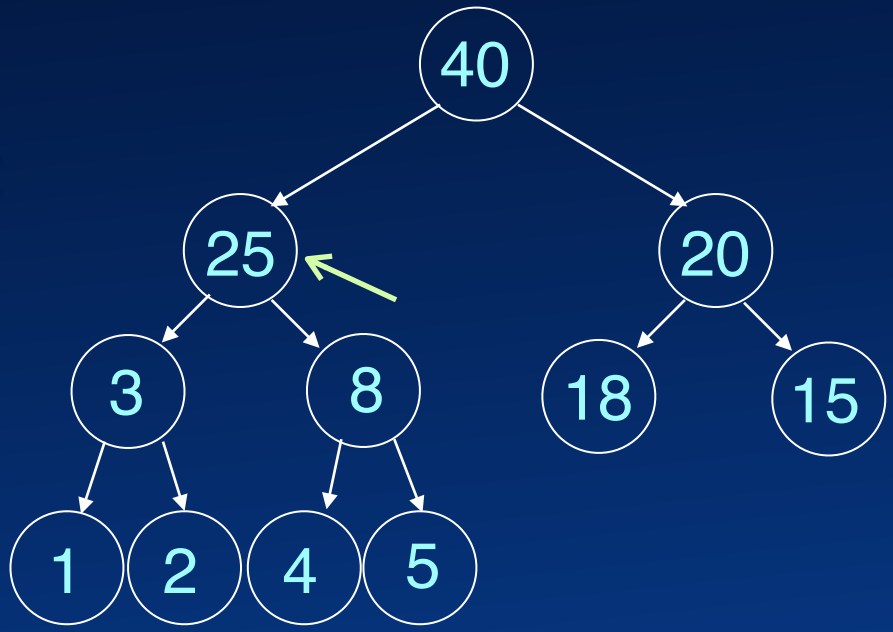
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

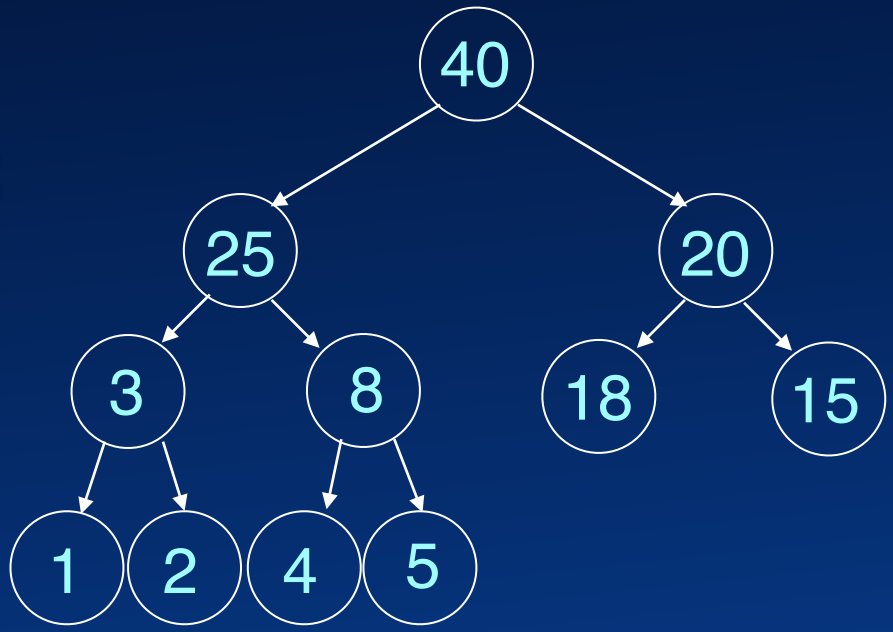
- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree

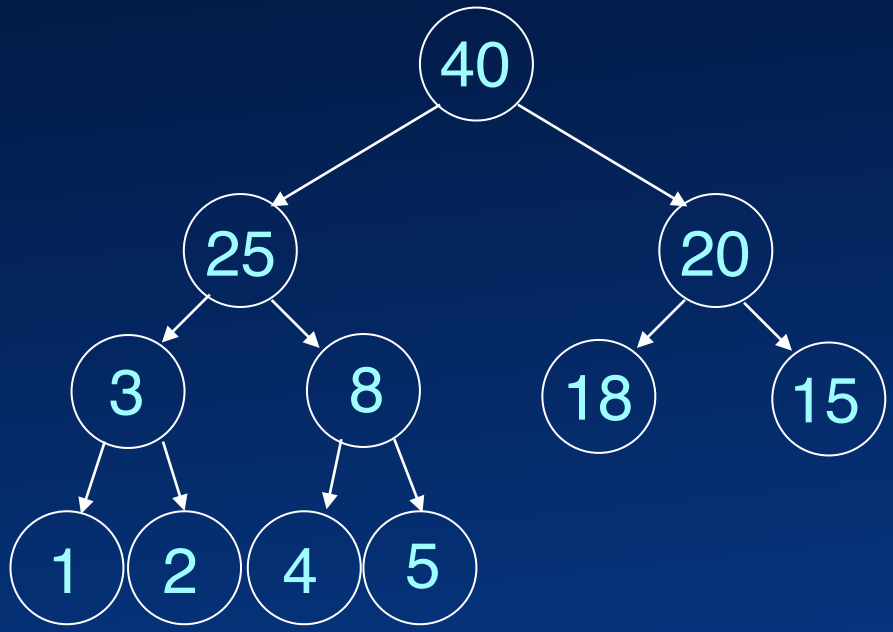


Heap

Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree

Inserted 40



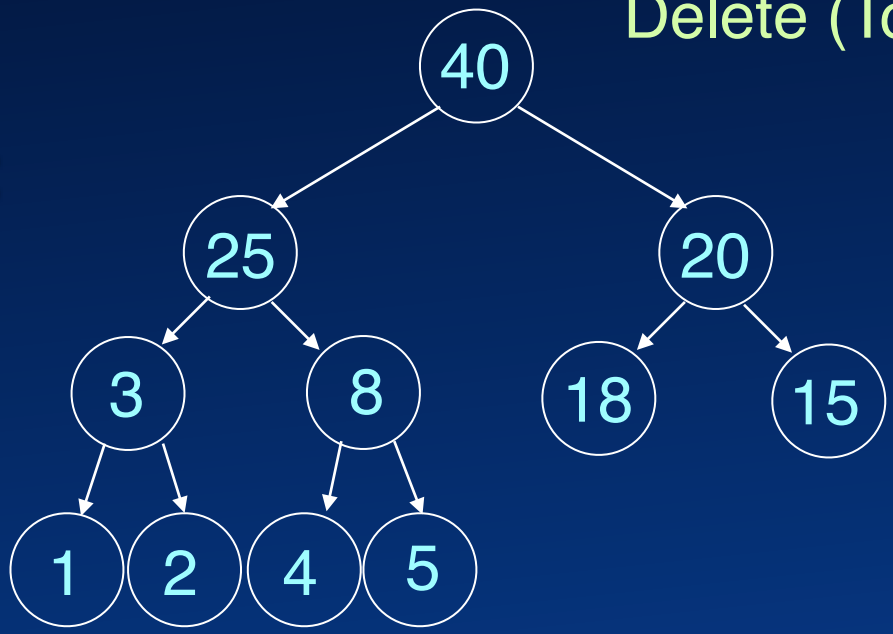
Heap

Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree

Inserted 40

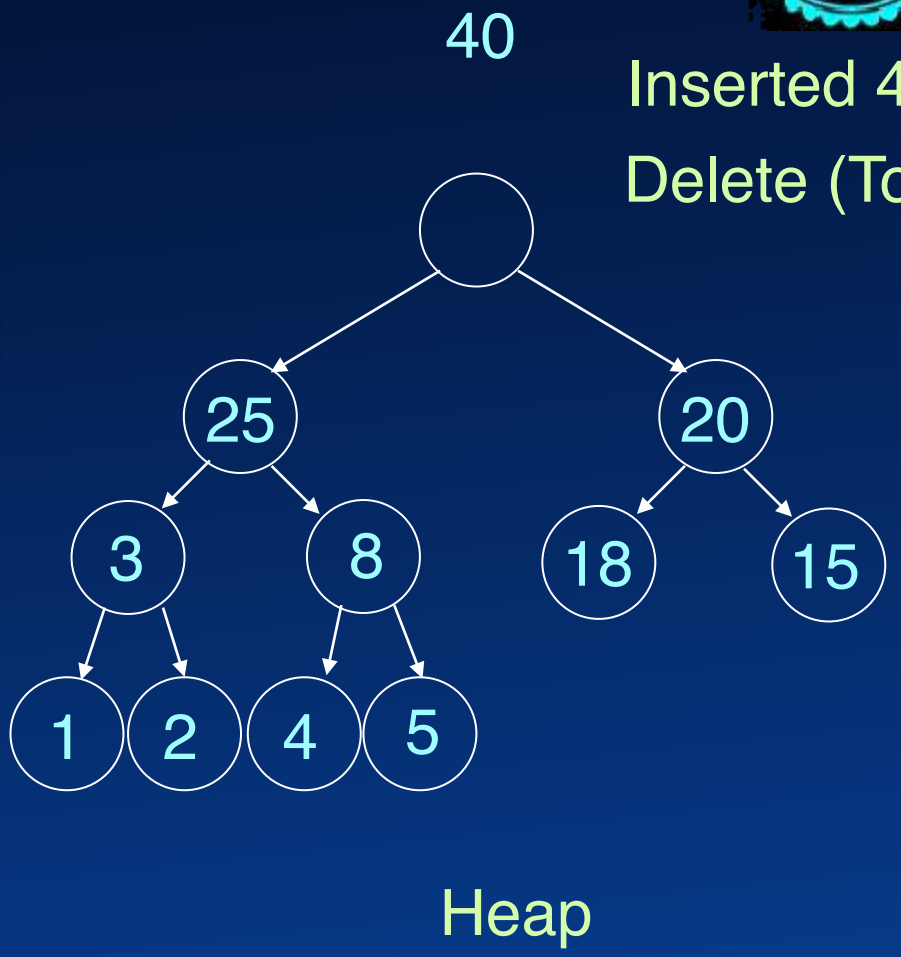
Delete (Top)



Heap

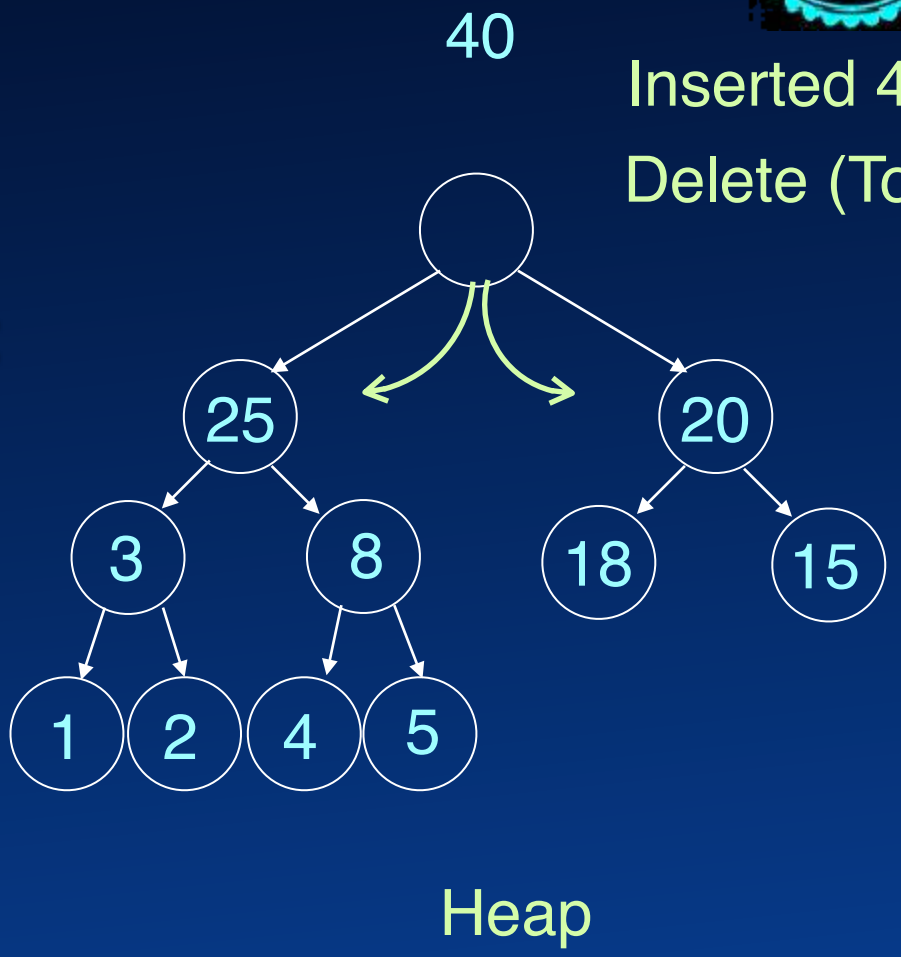
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



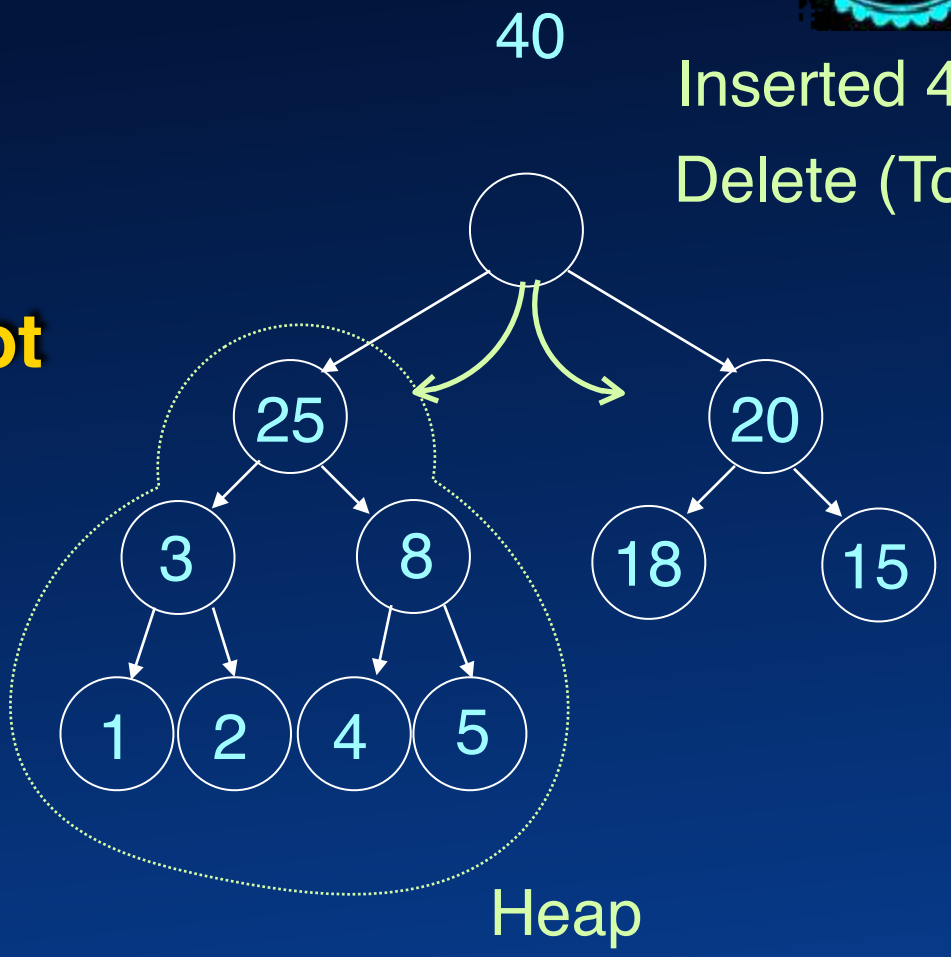
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



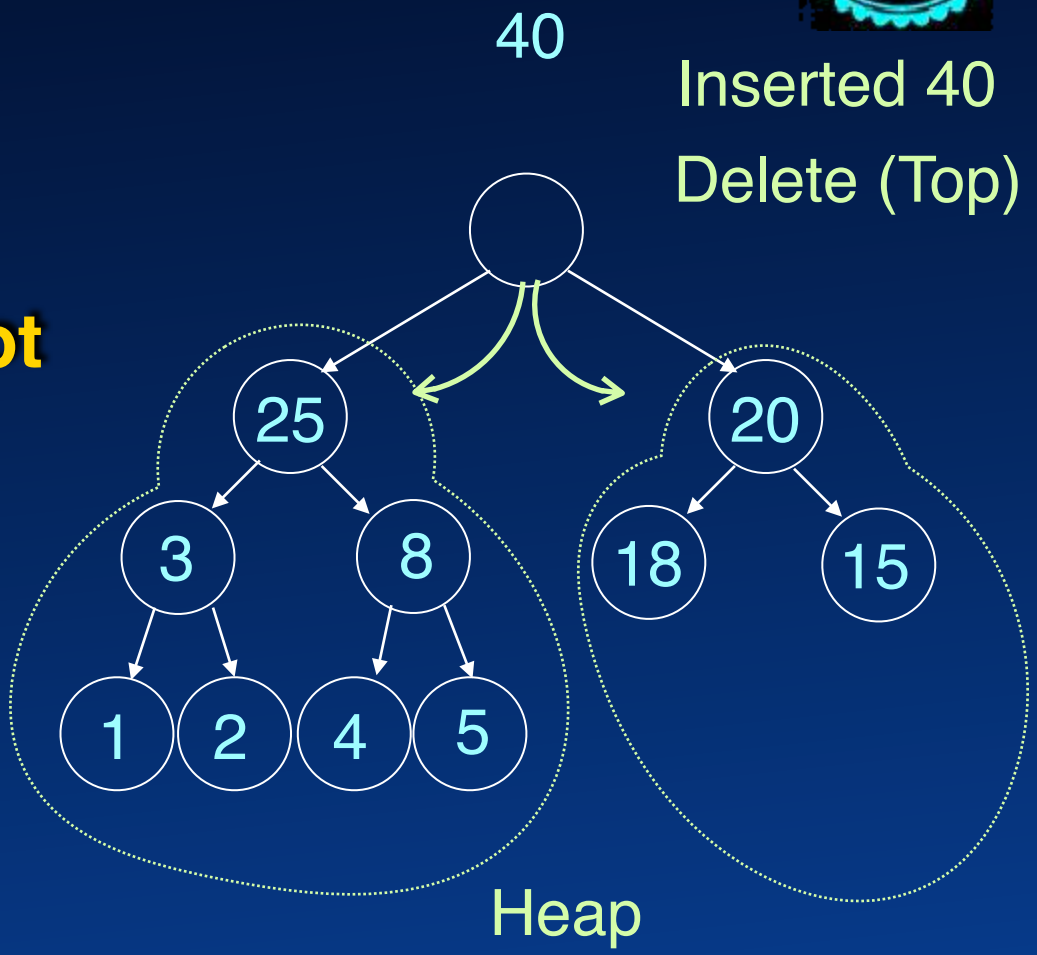
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



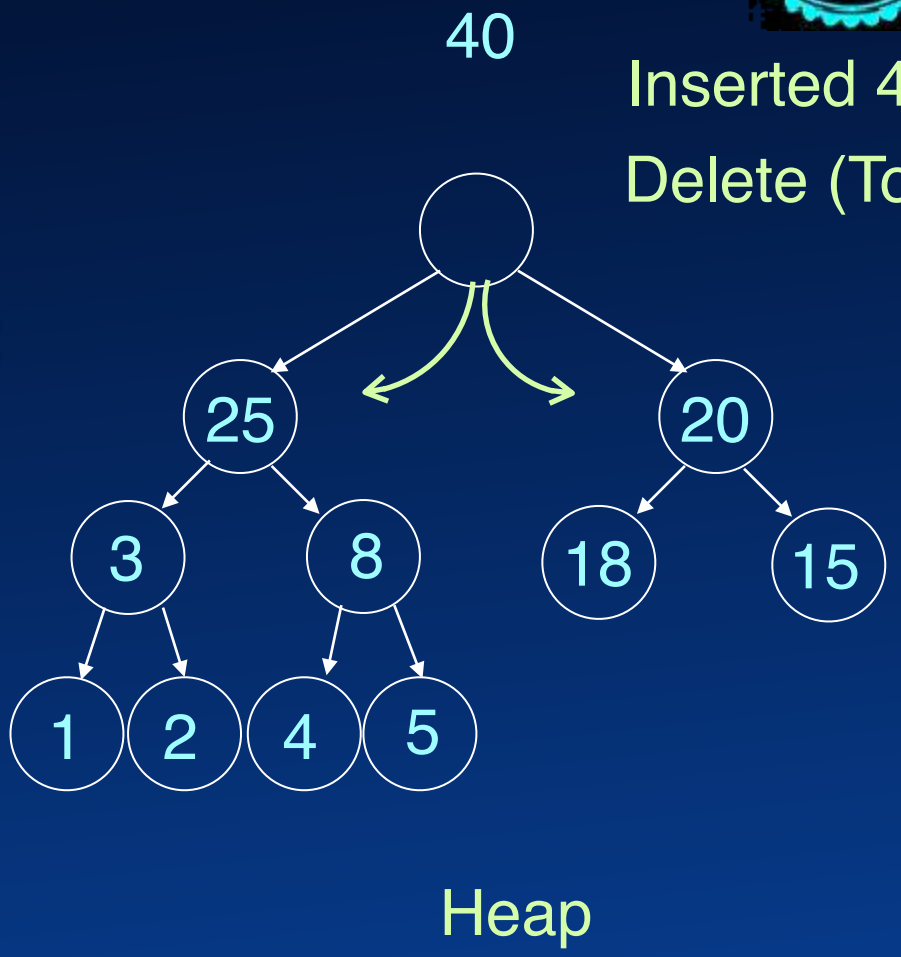
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



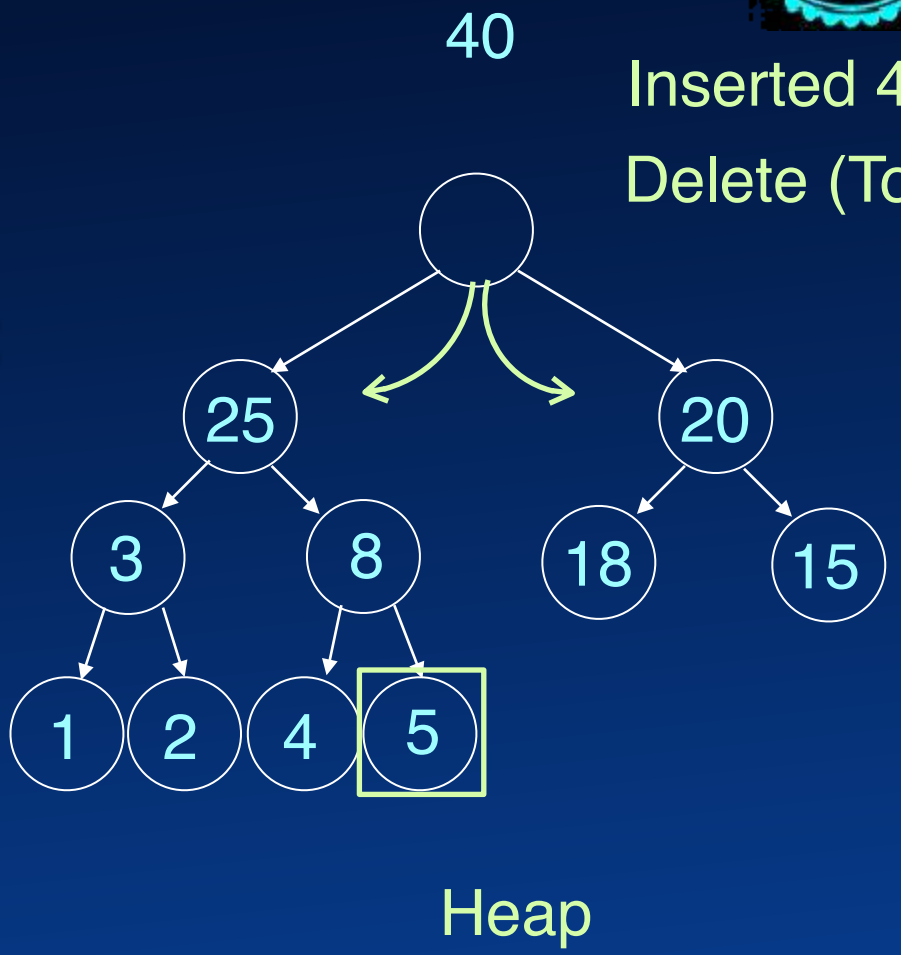
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



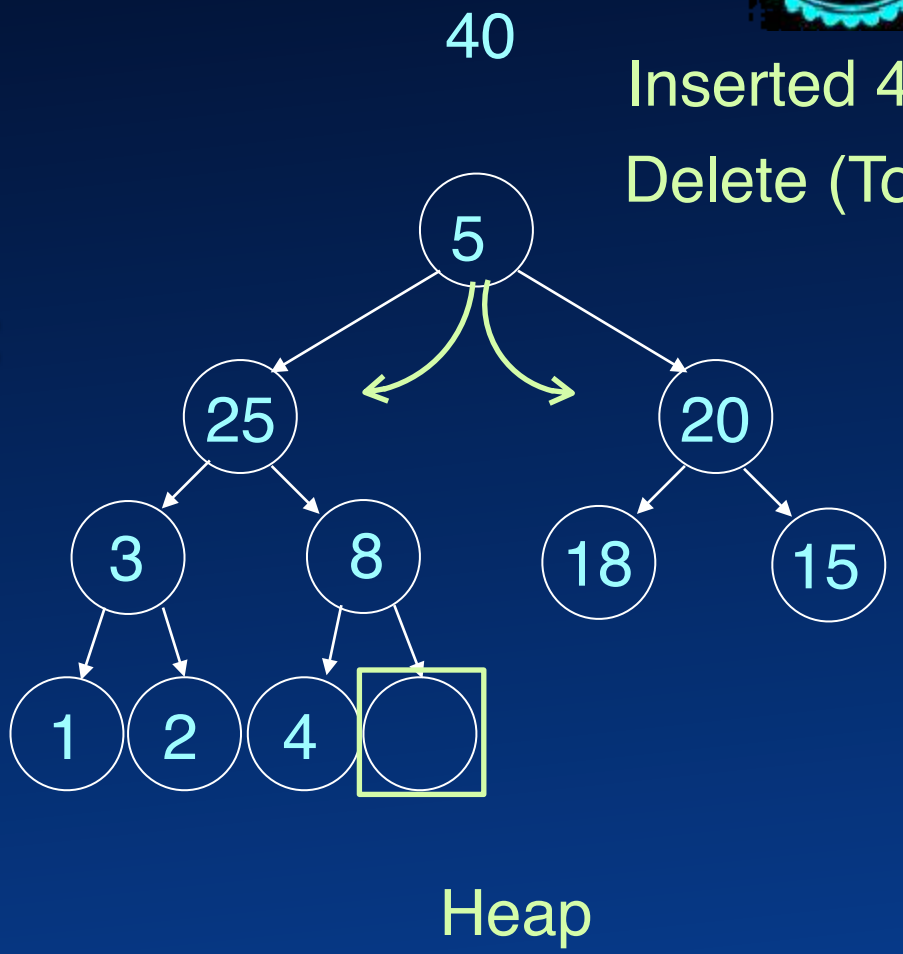
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



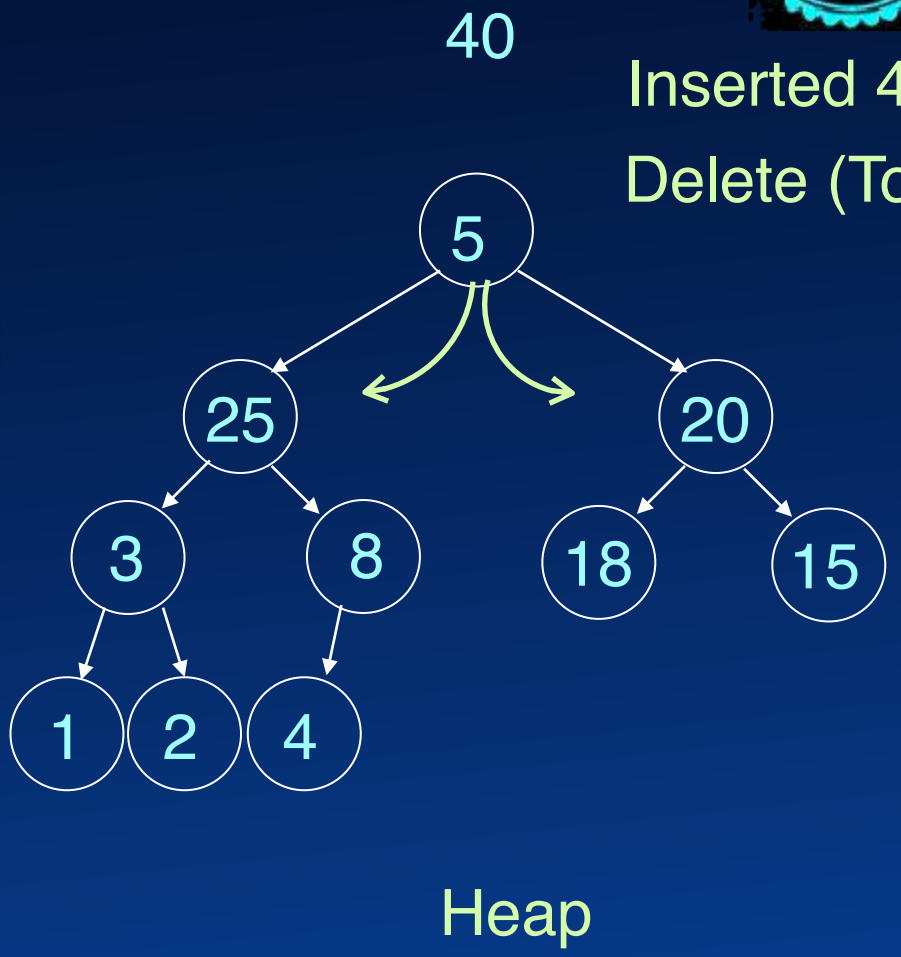
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



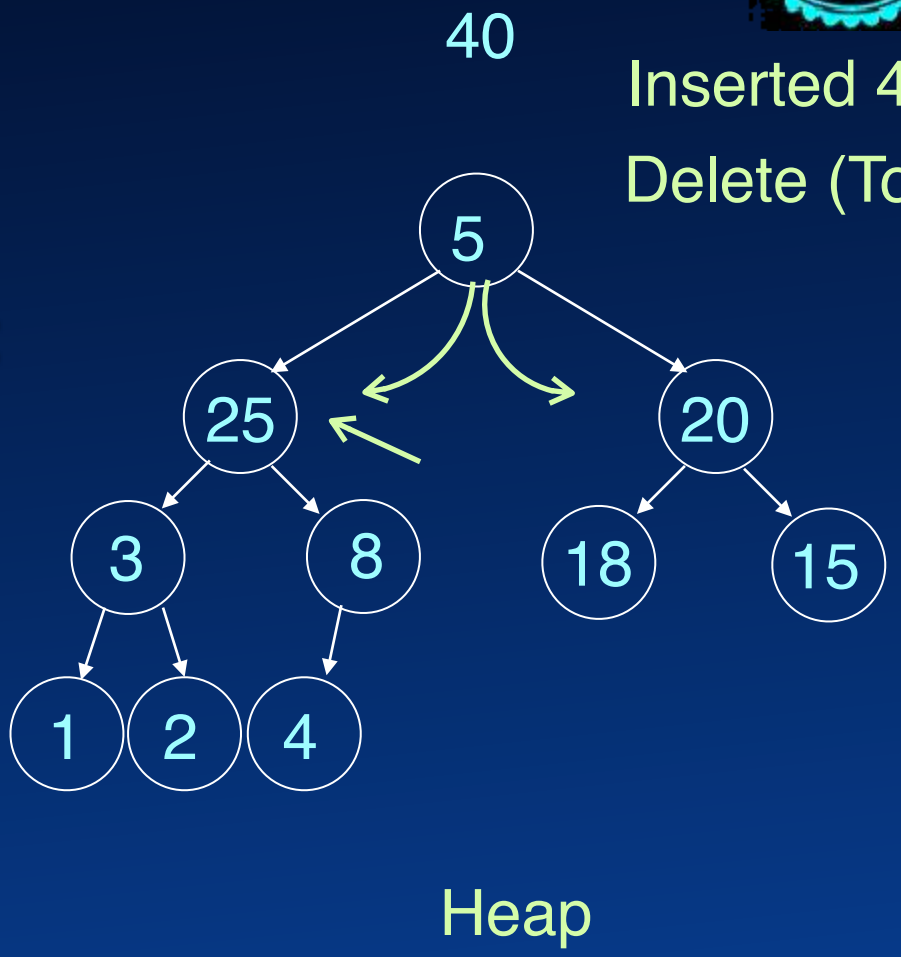
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



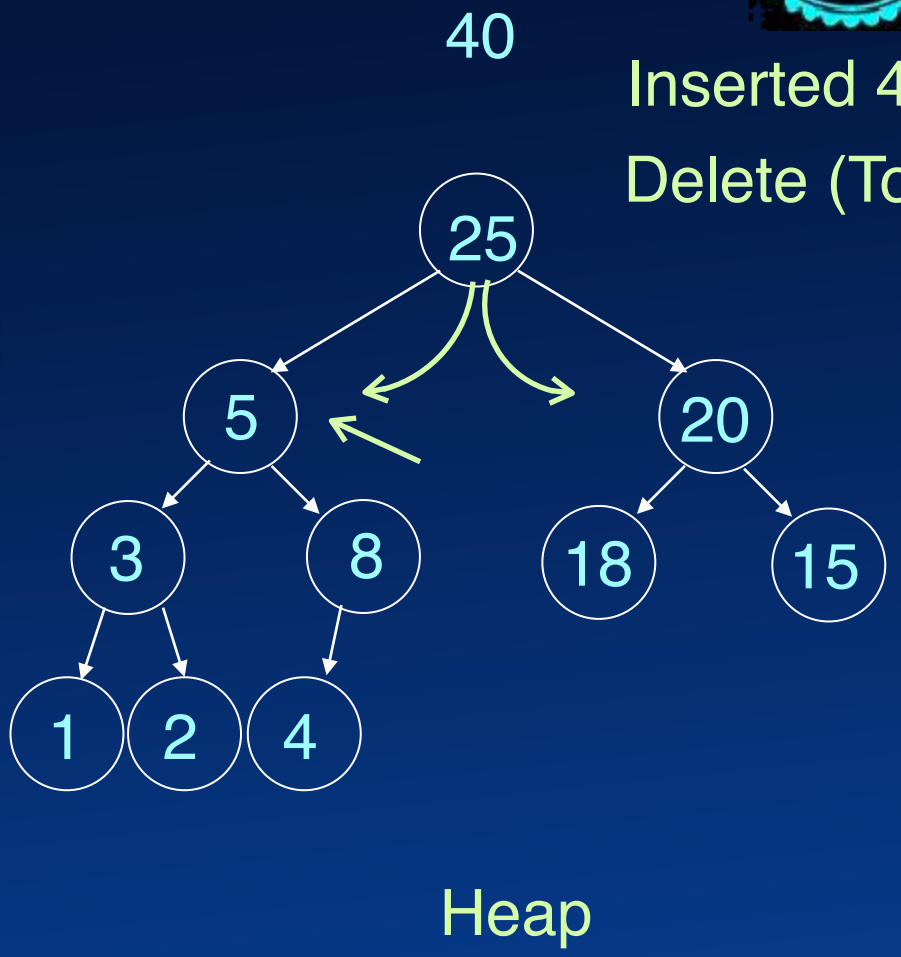
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



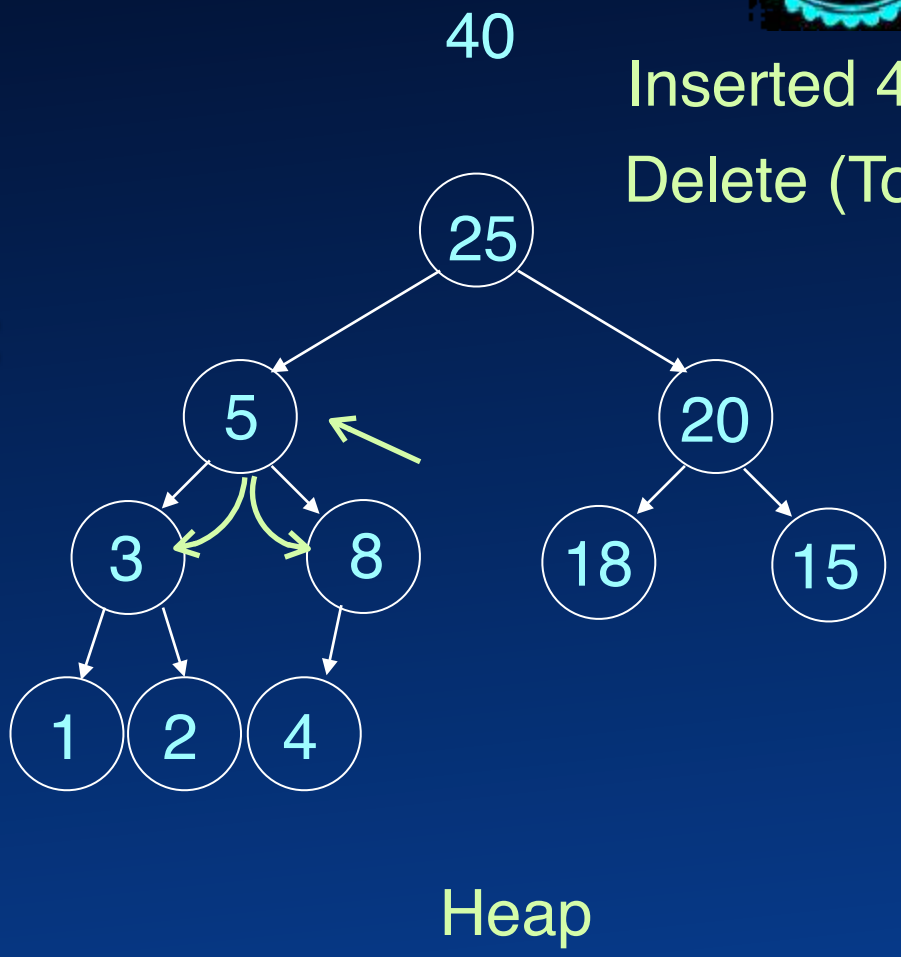
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



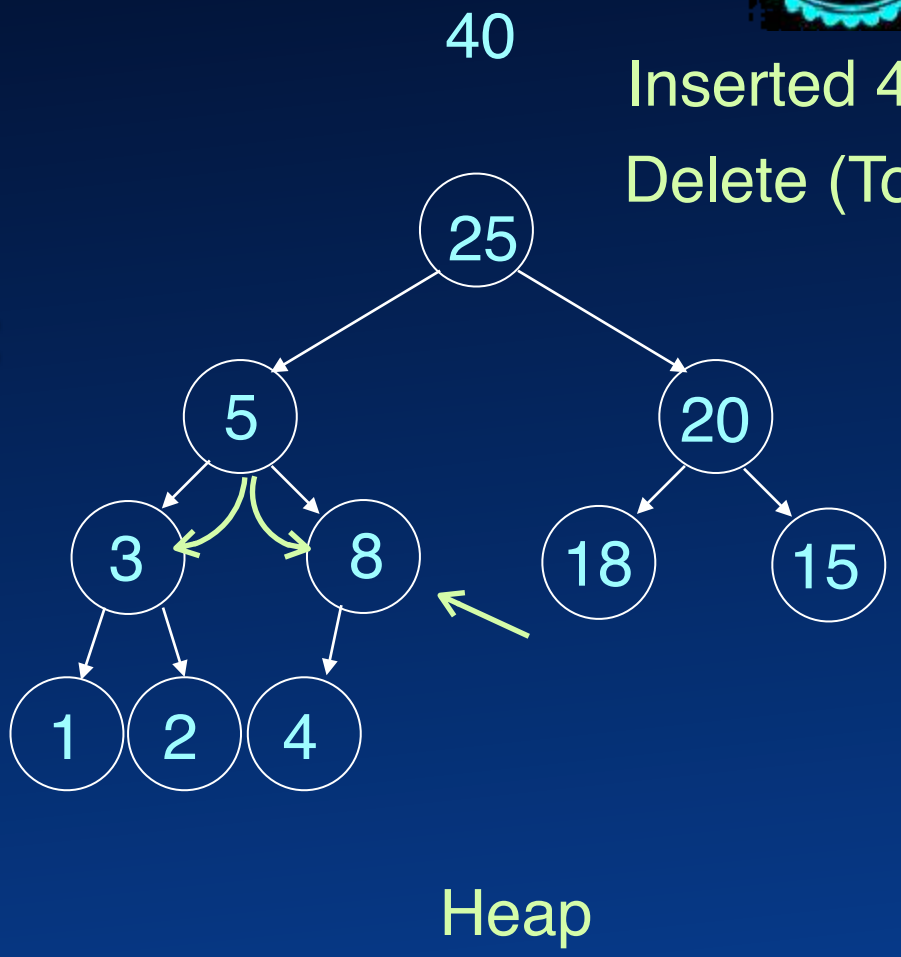
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



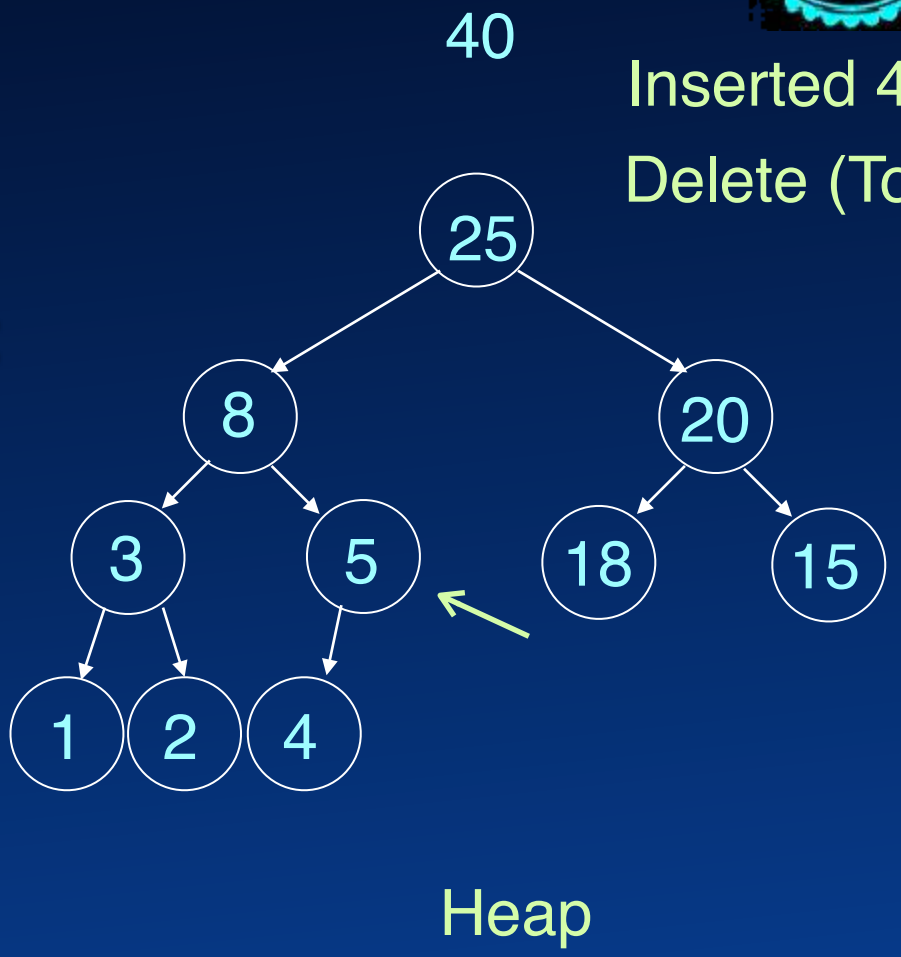
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



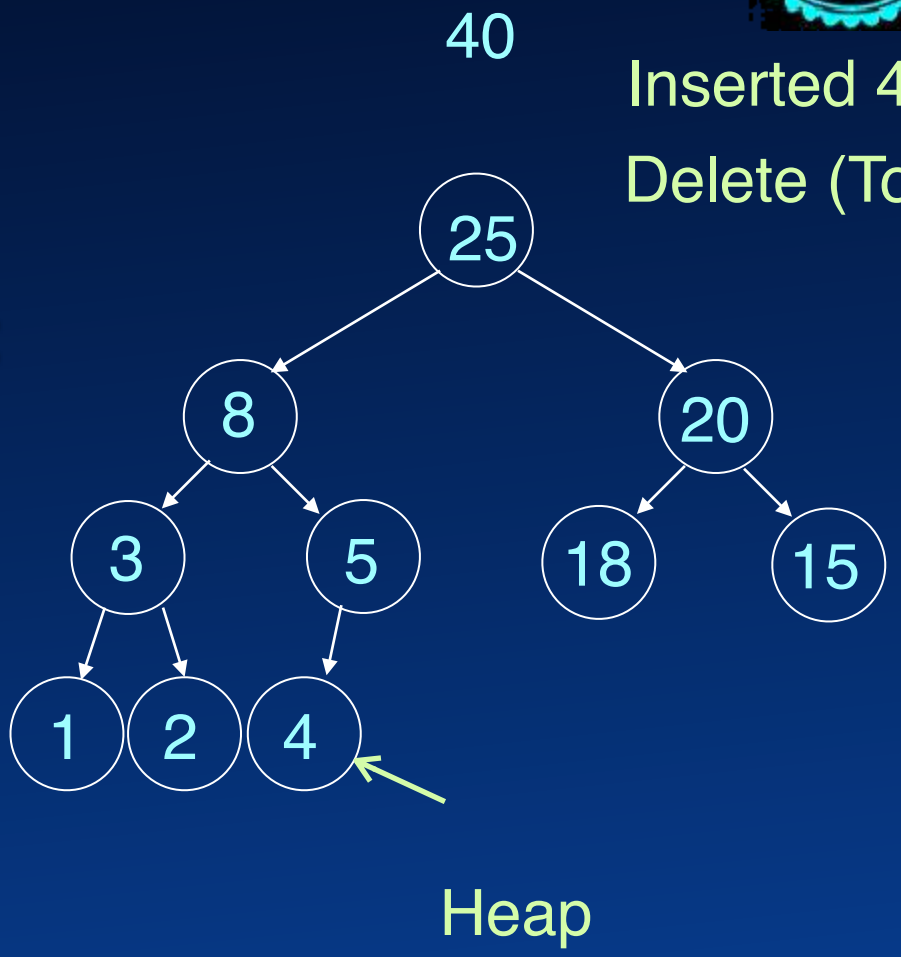
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



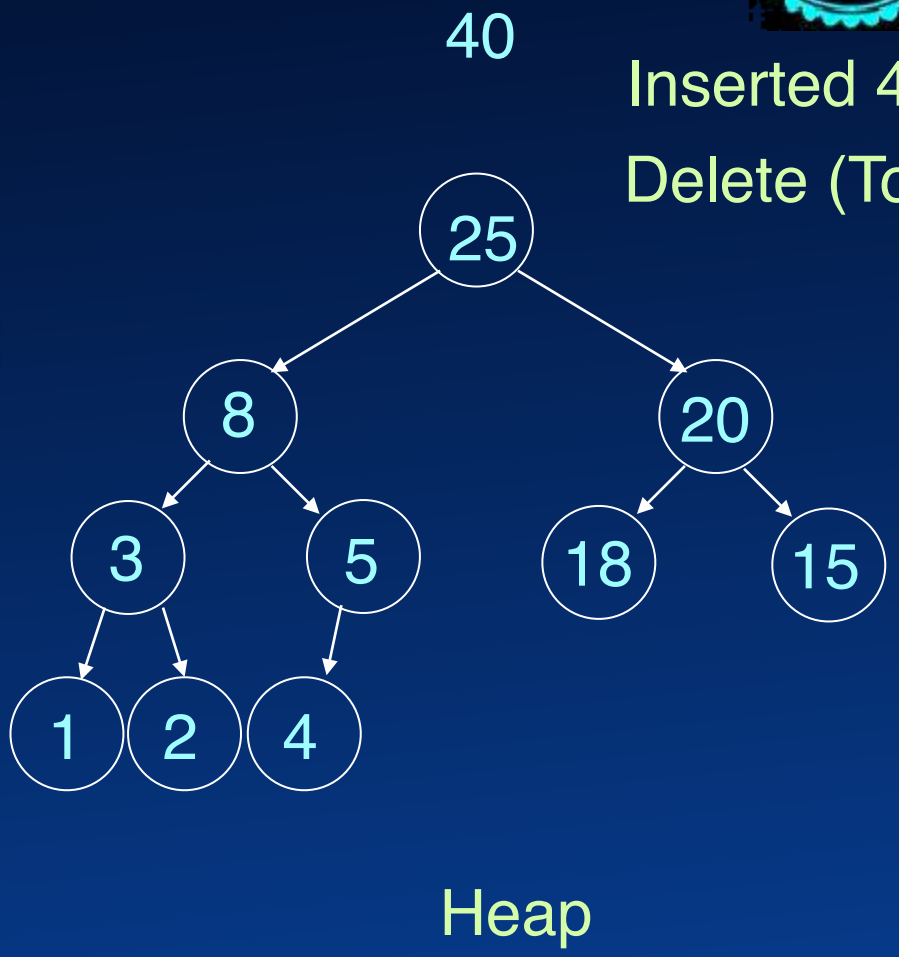
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



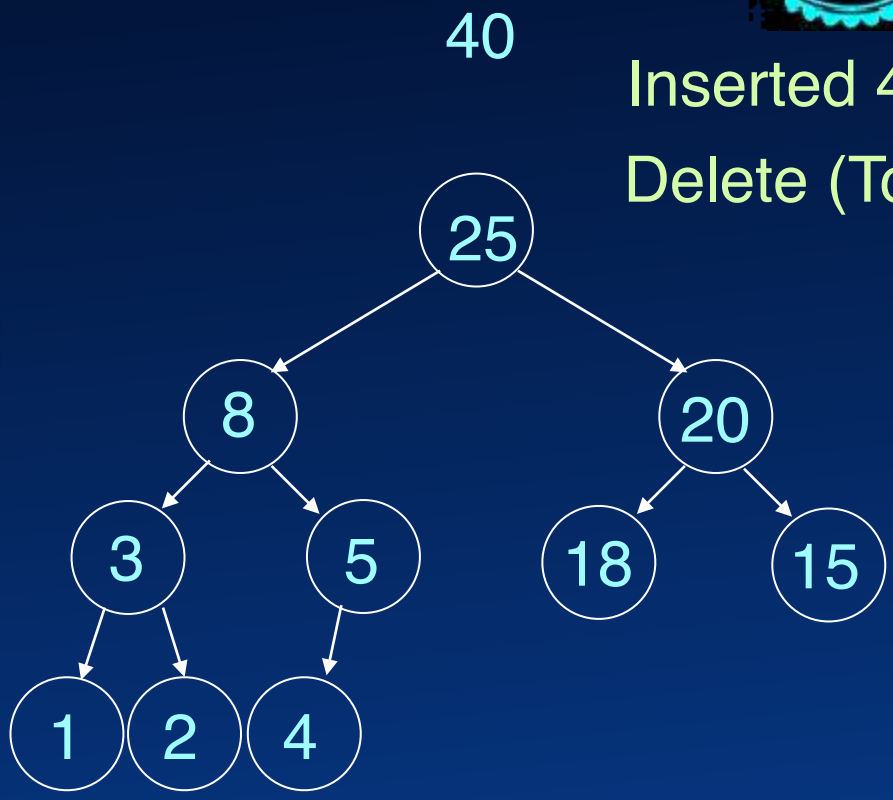
Heap

- Left-complete tree
- Comparable keys
- "Top" key in the root
  - For every subtree



Heap

- **Left-complete tree**
- **Comparable keys**
- **"Top" key in the root**
  - **For every subtree**



Heap

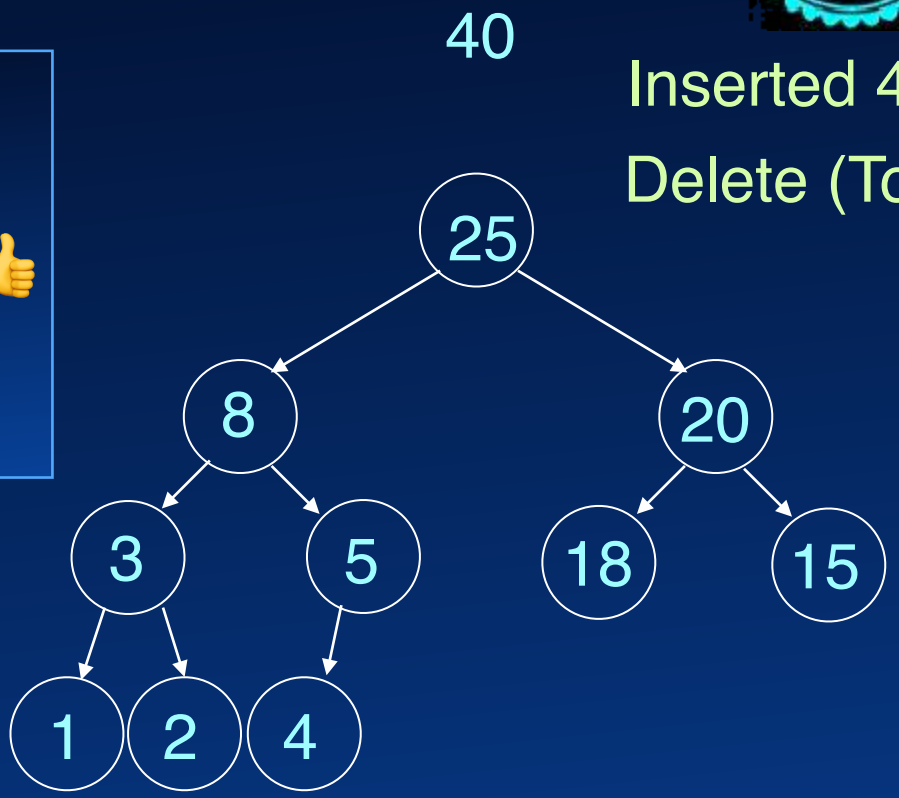
Insert:
Add node at next spot
Bubble-up
Delete:
Remove root
Replace with last spot
Bubble-down

Heap

Bubble up:
- if no parent 👍
- if(key lower-than parent.key) 👍
- swapwith(parent)
- parent.bubbleup()

Insert:
- Add node at next spot
- Bubble-up

Delete:
- Remove root
- Replace with last spot
- Bubble-down



Heap

Heap

Bubble down:

Bubble up:
- if no parent 👍
- if(key lower-than parent.key) 👍
- swapwith(parent)
- parent.bubbleup()

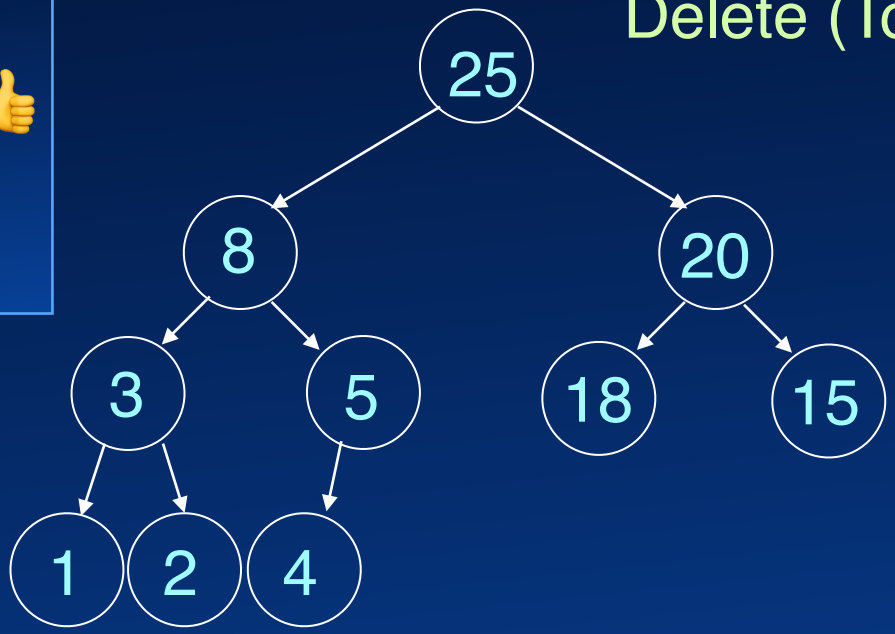
Insert:
- Add node at next spot
- Bubble-up

Delete:
- Remove root
- Replace with last spot
- Bubble-down

40

Inserted 40

Delete (Top)



Heap

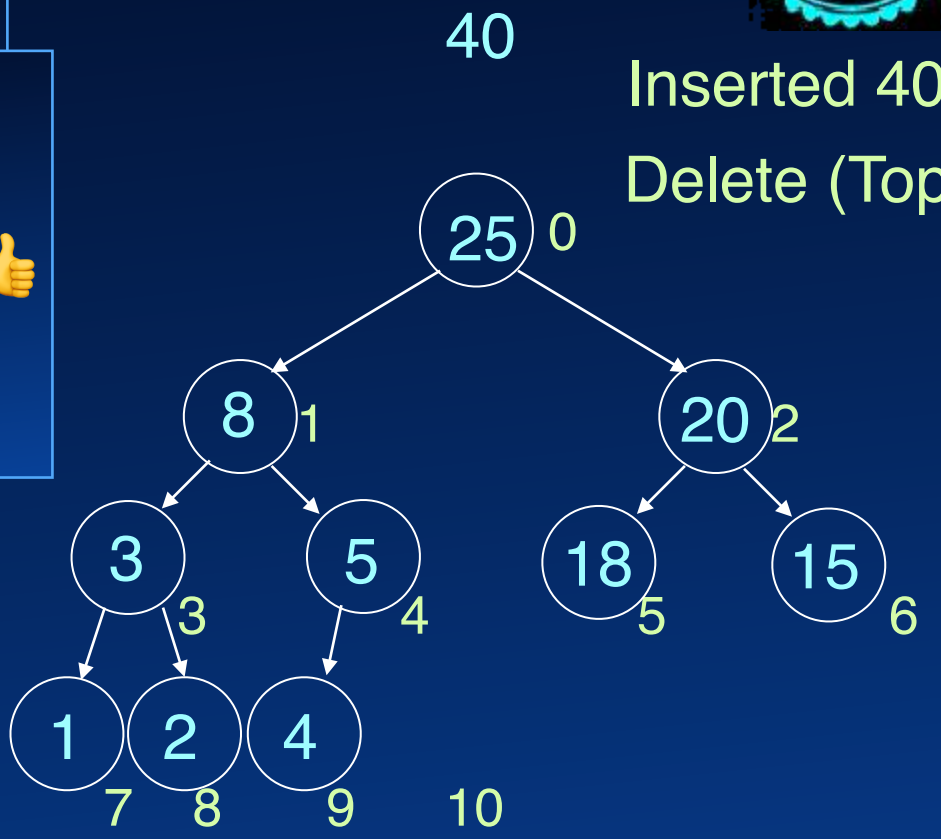
Heap

Bubble down:

Bubble up:
- if no parent 👍
- if(key lower-than parent.key) 👍
- swapwith(parent)
- parent.bubbleup()

Insert:
- Add node at next spot
- Bubble-up

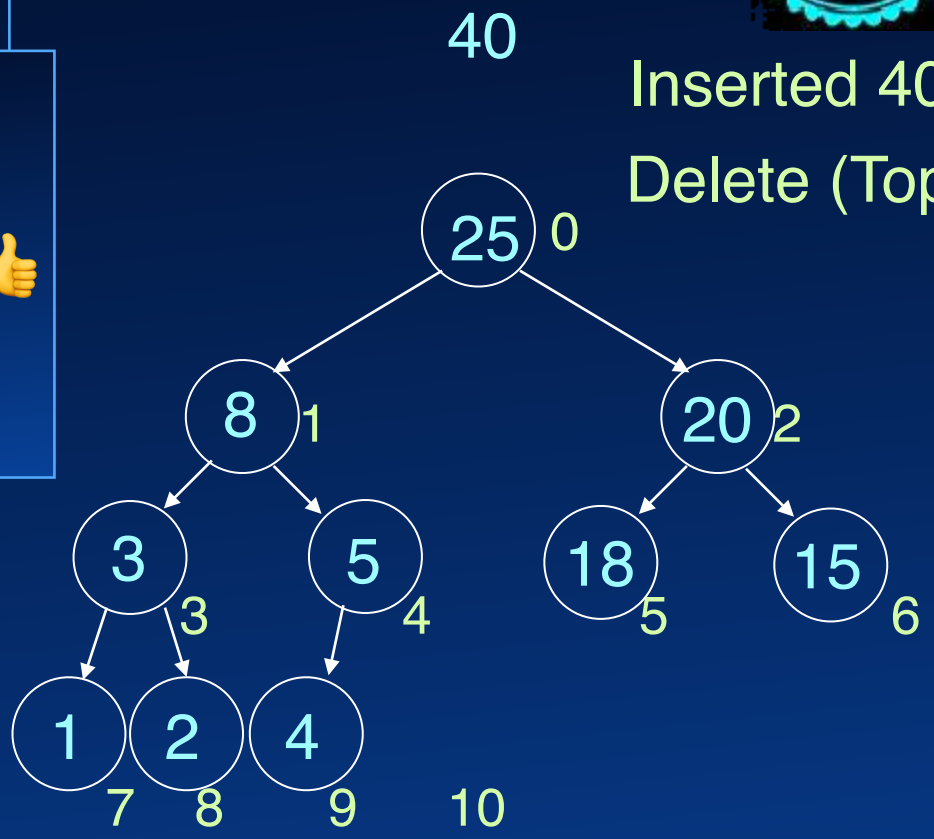
Delete:
- Remove root
- Replace with last spot
- Bubble-down



Heap

Bubble down:

Bubble up:
- if no parent 👍
- if(key lower-than parent.key) 👍
- swapwith(parent)
- parent.bubbleup()



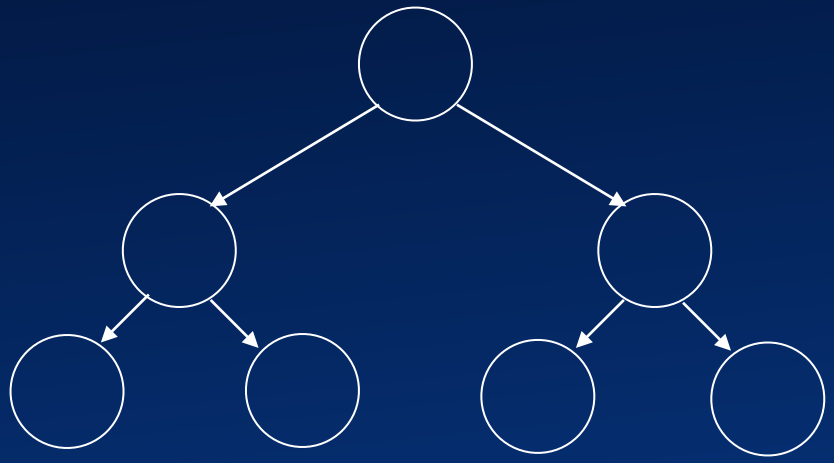
Insert:
- Add node at next spot
- Bubble-up

Delete:
- Remove root
- Replace with last spot
- Bubble-down

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

In which node may the third largest element of a heap be?

mail: col106quiz@cse.iitd.ac.in
format: 1,2,3,4,5,6

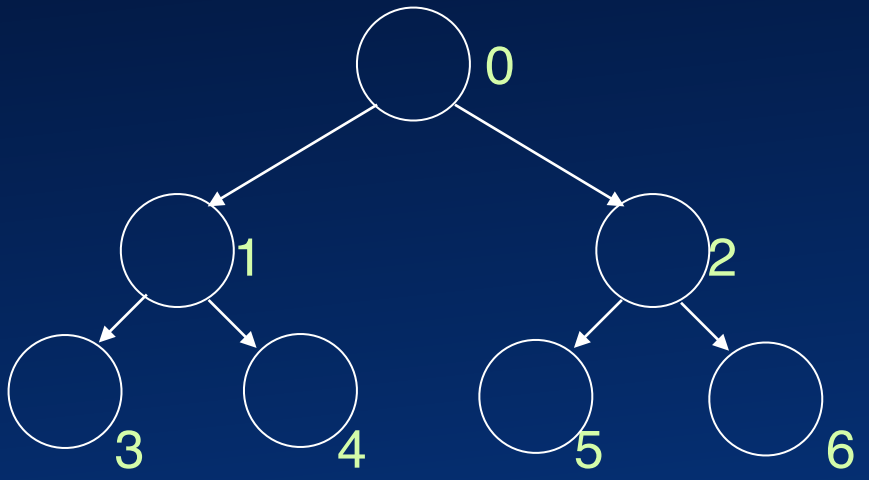


- 1
- 2
- 3
- 4
- 5
- 6

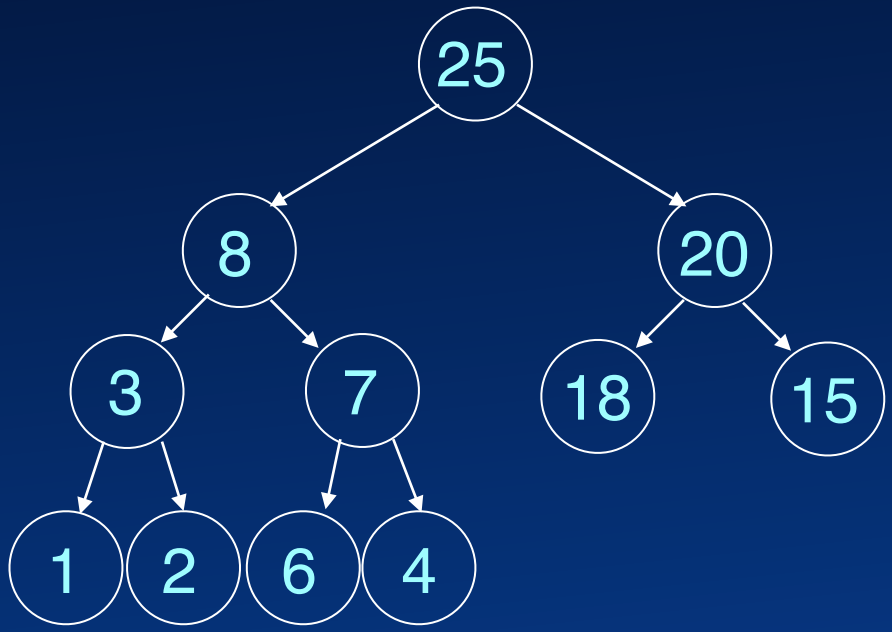
In which node may the third largest element of a heap be?

mail: col106quiz@cse.iitd.ac.in
format: 1,2,3,4,5,6

- 1
- 2
- 3
- 4
- 5
- 6

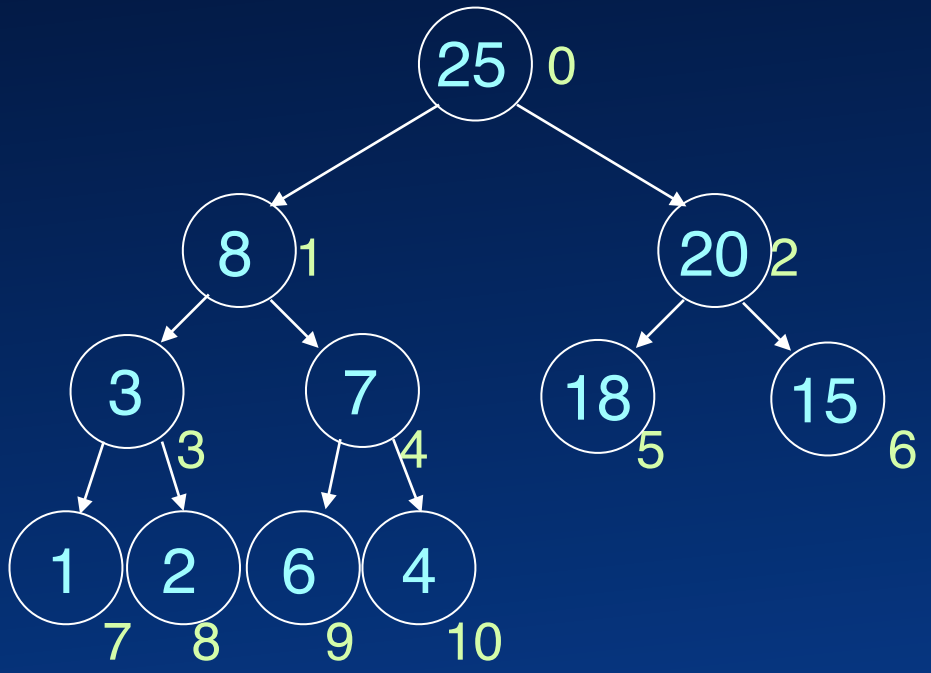


Heap



Heap

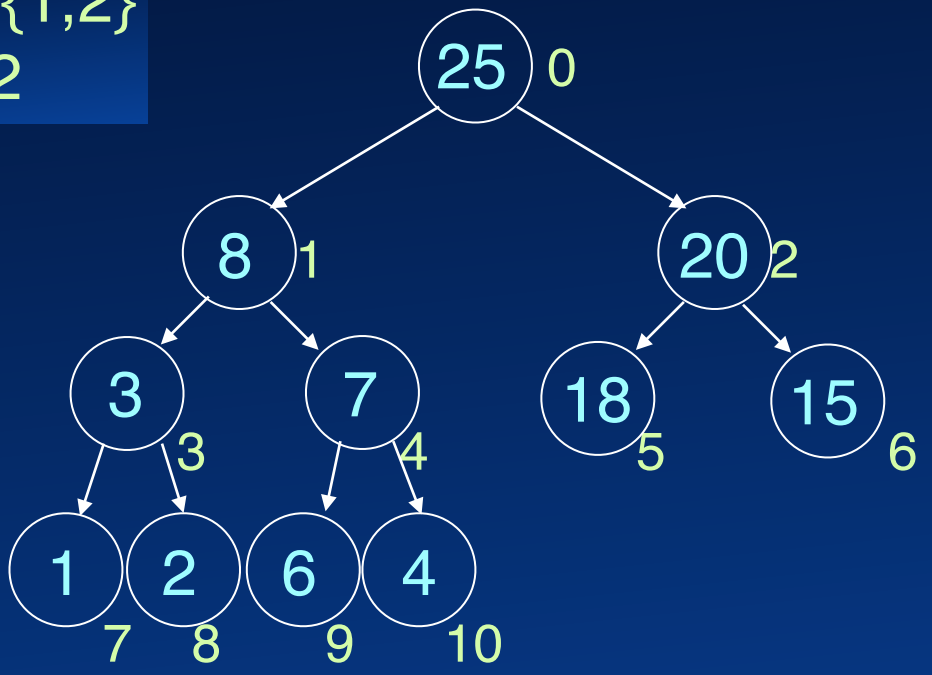
Heap



Heap

Heap

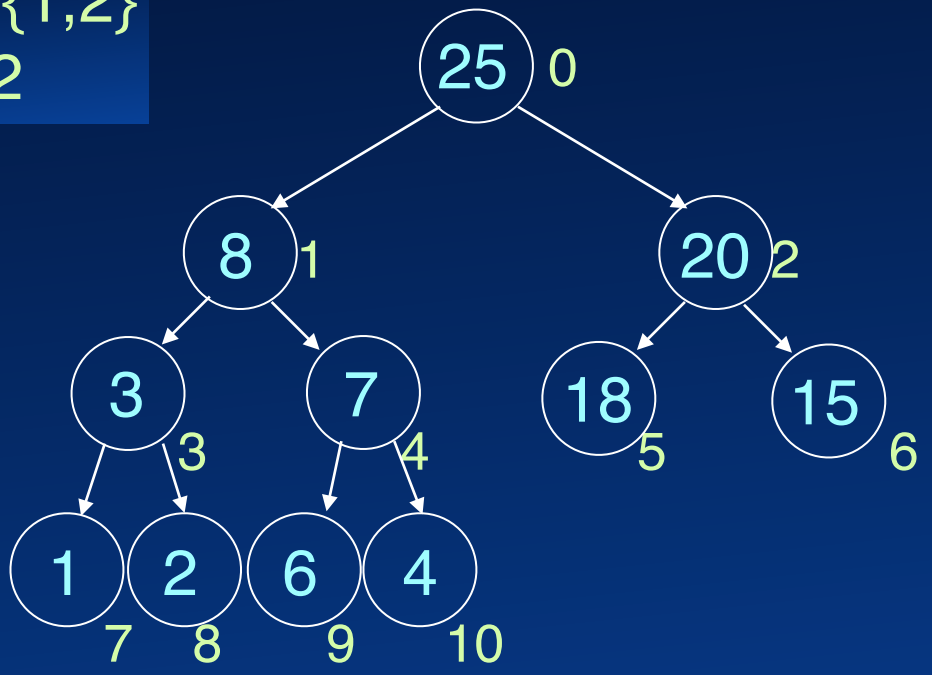
child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2



Heap

Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

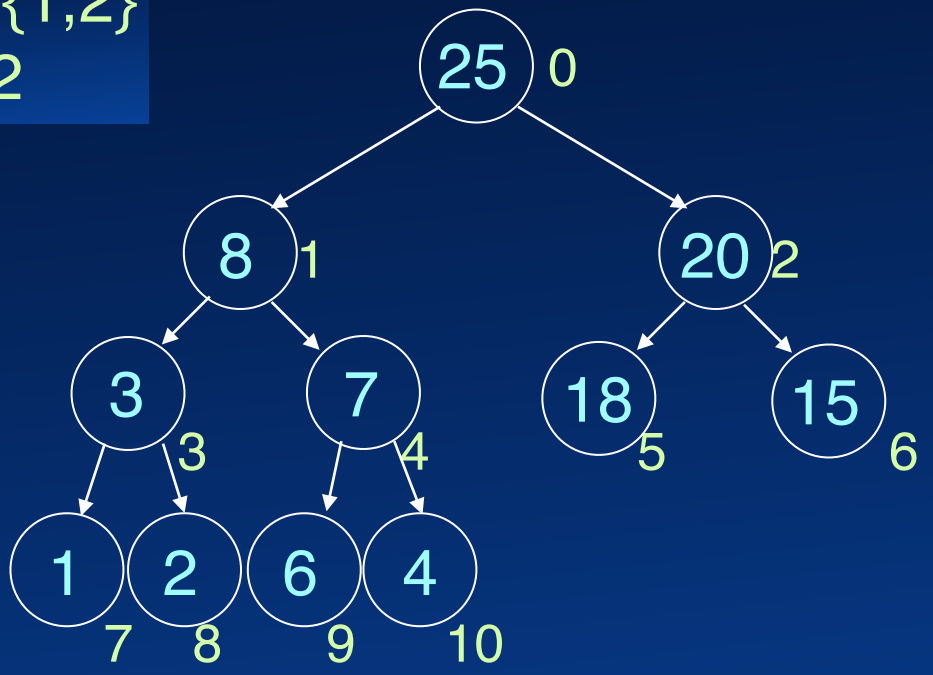


25	8	20	3	7	18	15	1	2	6	4		

Heap

Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

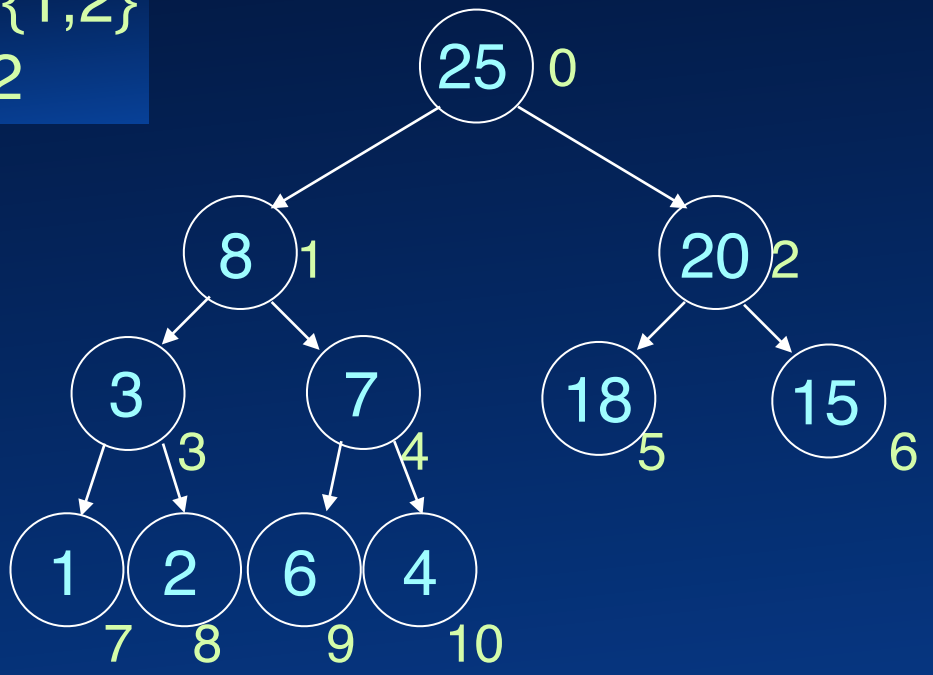


25	8	20	3	7	18	15	1	2	6	4		
0	1	2	3	4	5	6	7	8	9	10	11	

Heap

Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

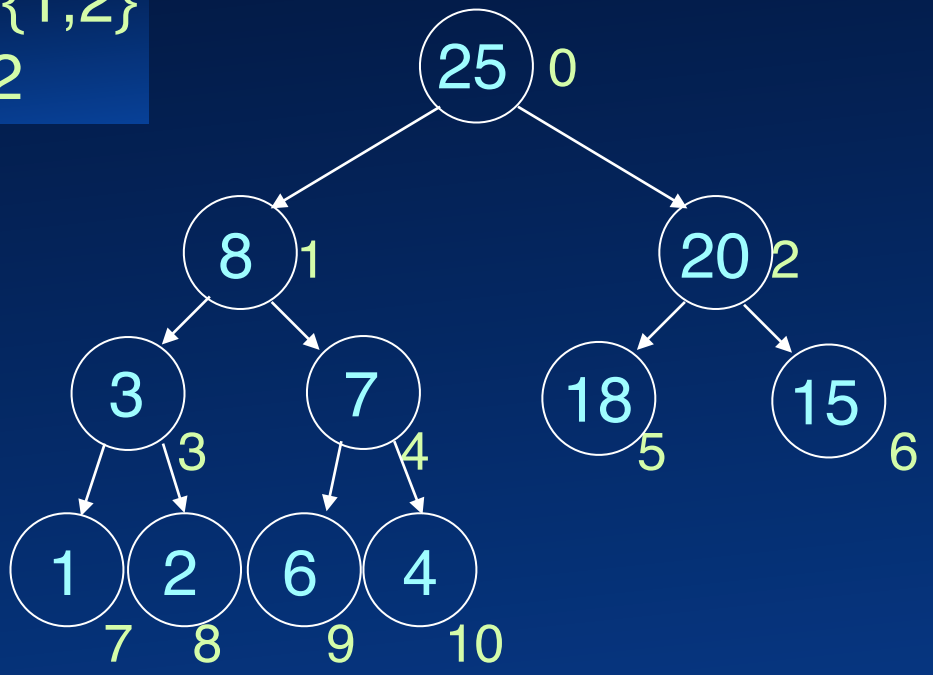


25	8	20	3	7	18	15	1	2	6	4		
0	1	2	3	4	5	6	7	8	9	10	11	

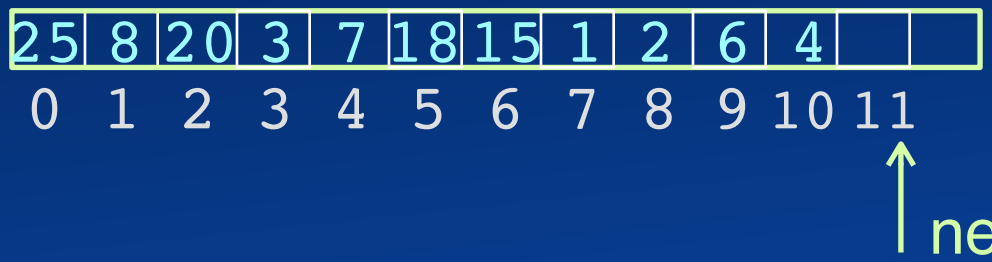
Heap

Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

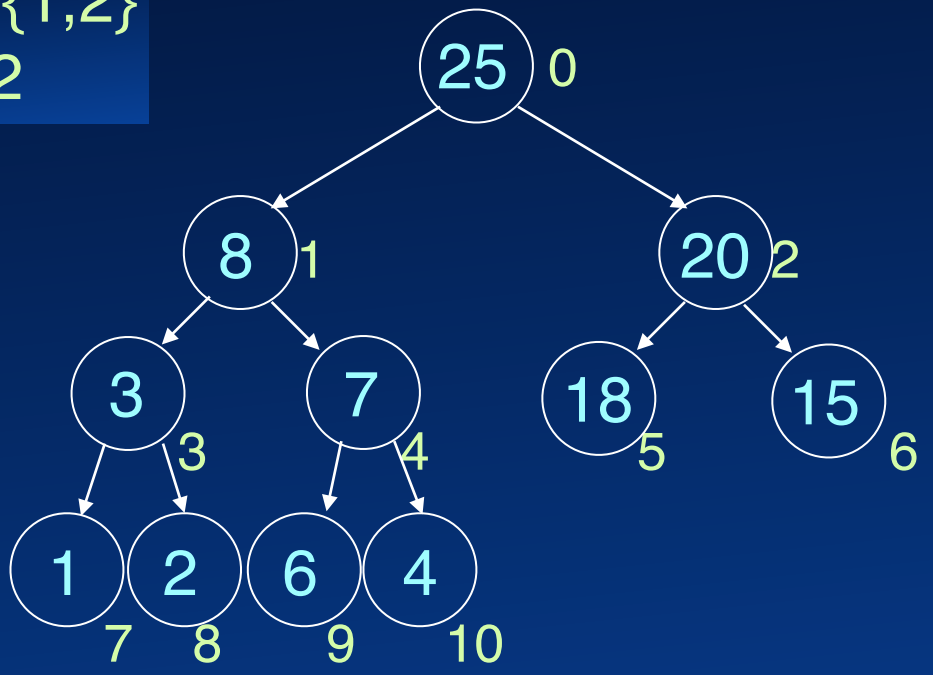


Heap

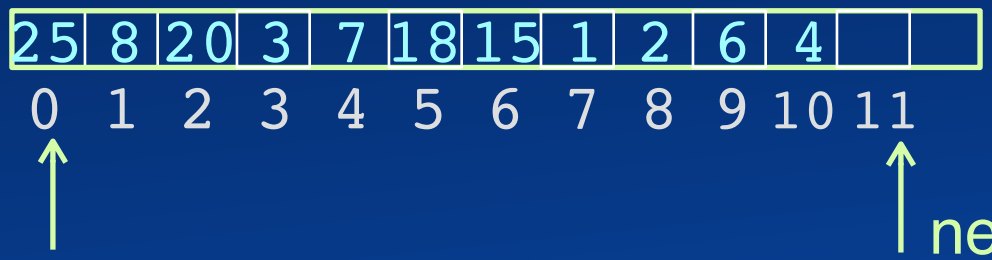


Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

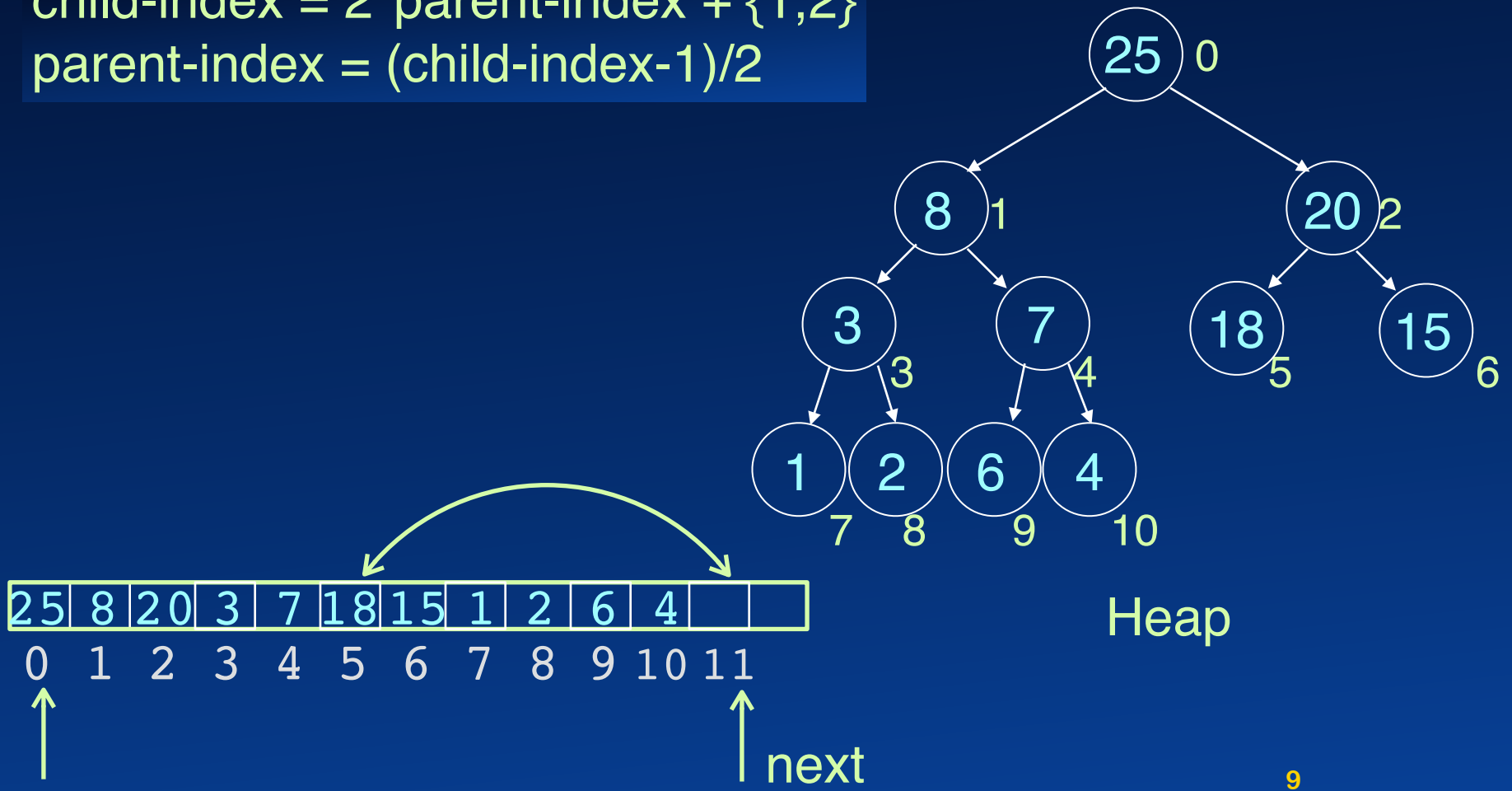


Heap



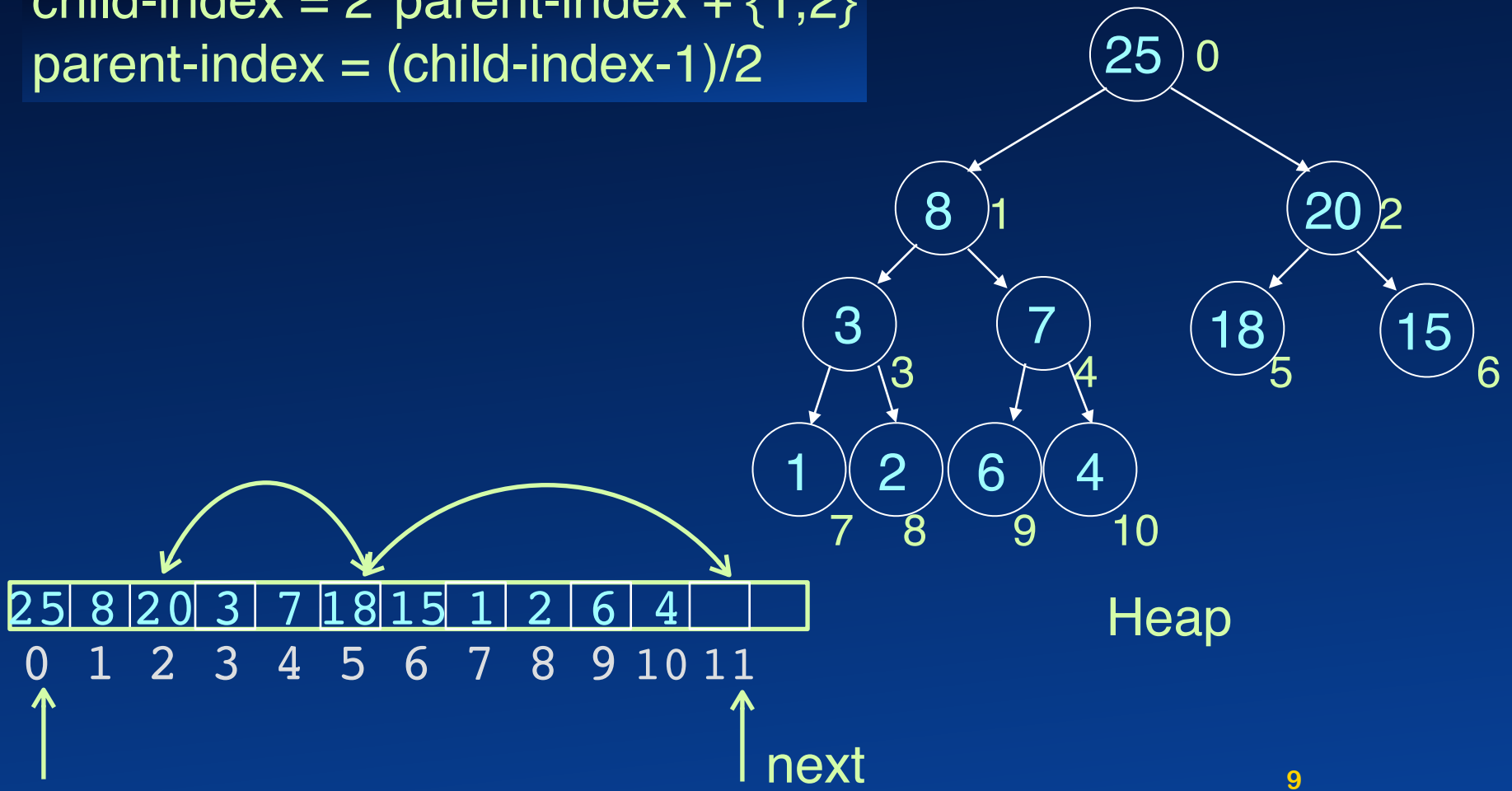
Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2



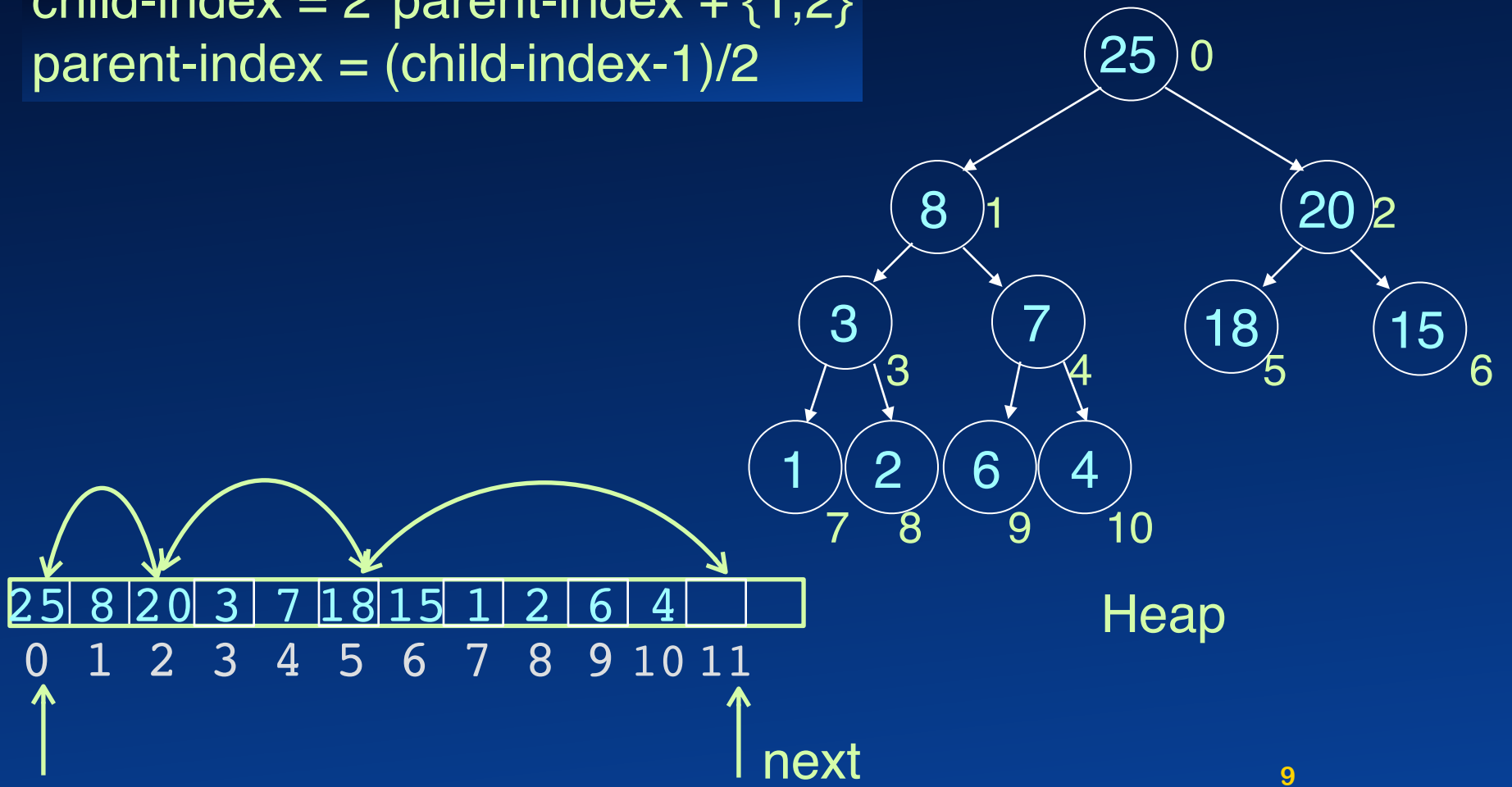
Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2



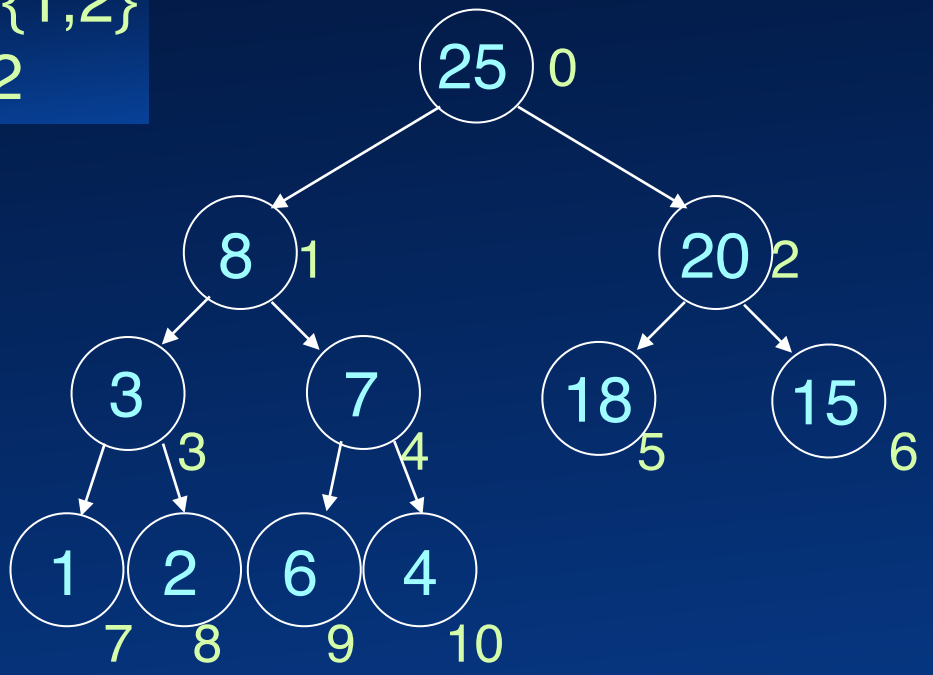
Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

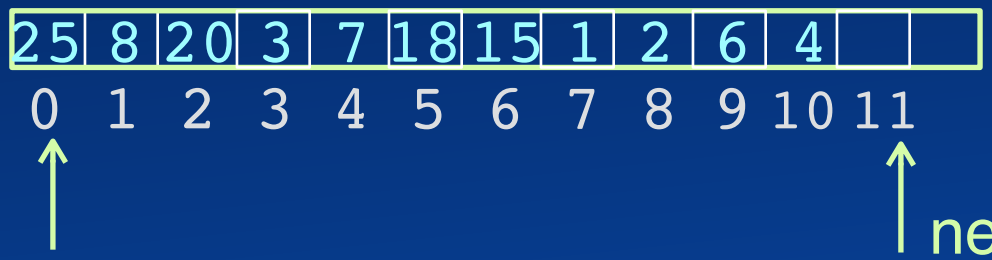


Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

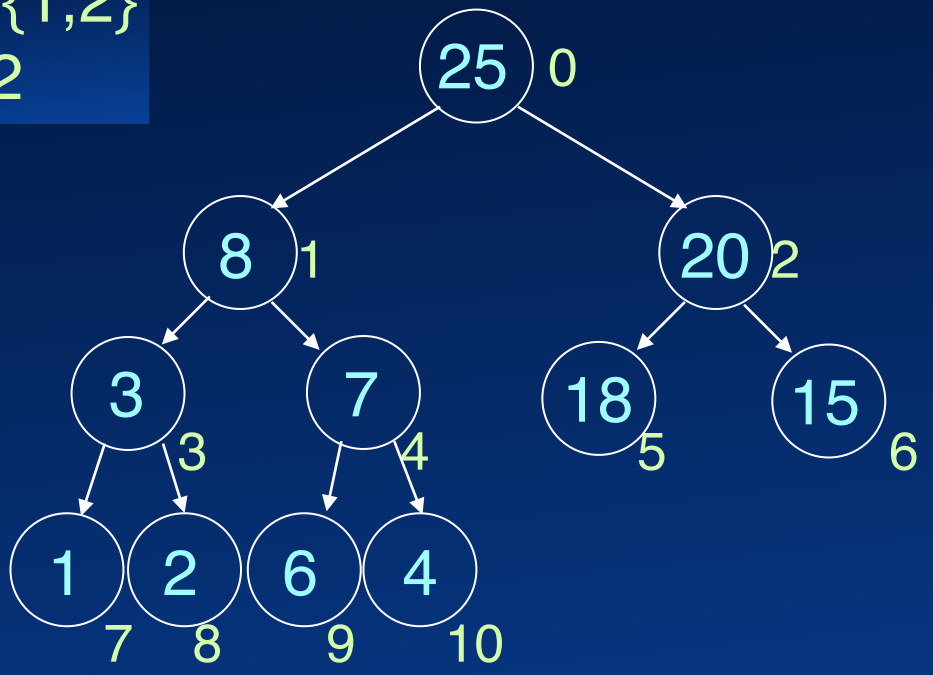


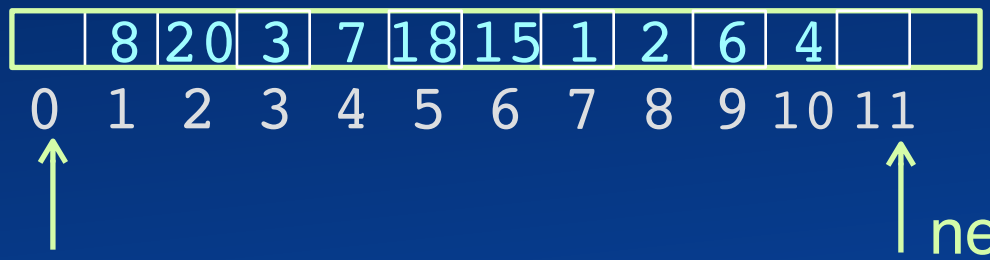
Heap



Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

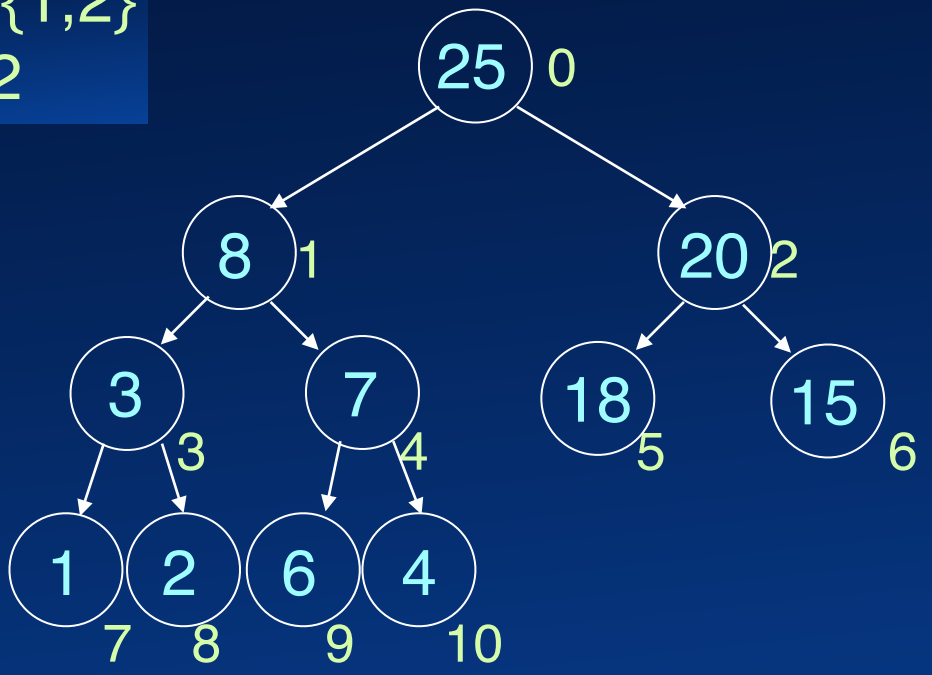




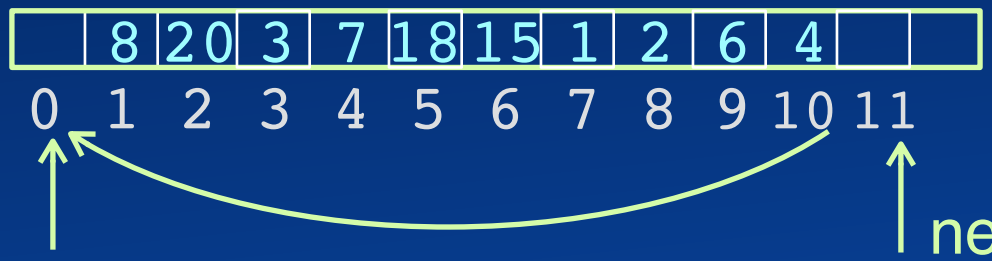
Heap

Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

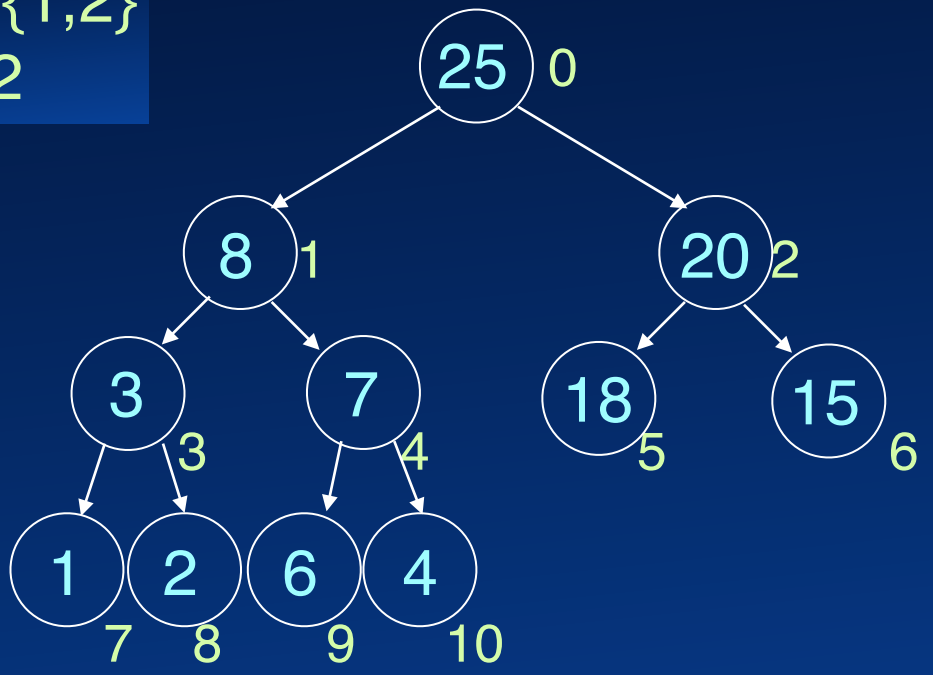


Heap

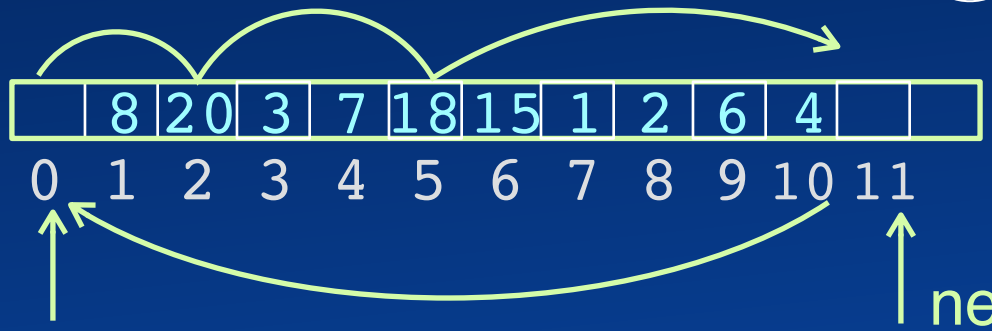


Heap

child-index = 2*parent-index + {1,2}
parent-index = (child-index-1)/2

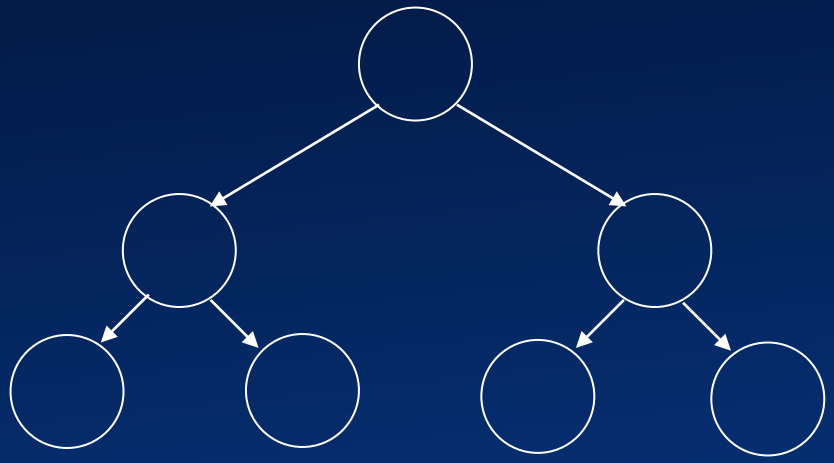


Heap



In which node may the third largest element of a heap be?

mail: col106quiz@cse.iitd.ac.in
format: 1,2,3,4,5,6

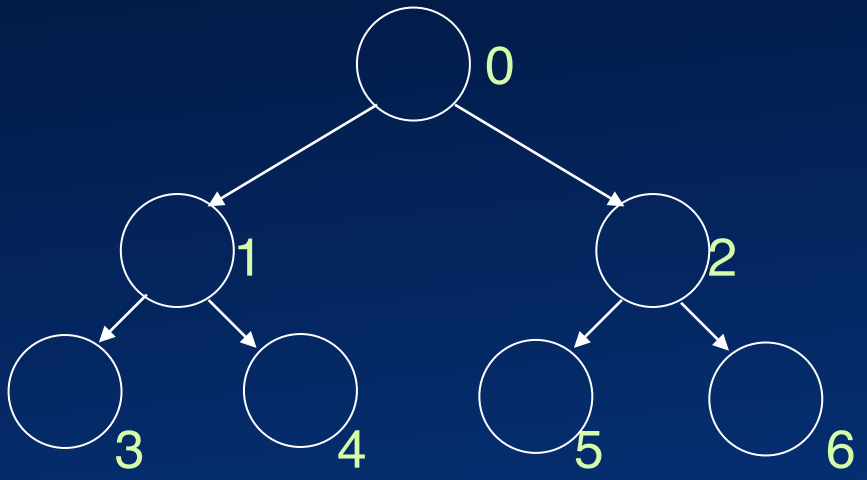


- 1
- 2
- 3
- 4
- 5
- 6

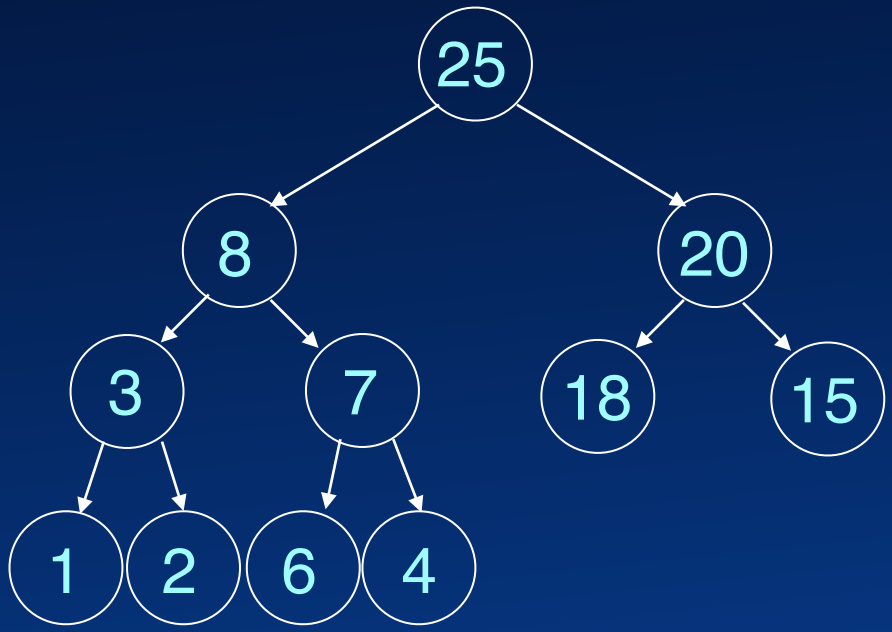
In which node may the third largest element of a heap be?

mail: col106quiz@cse.iitd.ac.in
format: 1,2,3,4,5,6

- 1
- 2
- 3
- 4
- 5
- 6



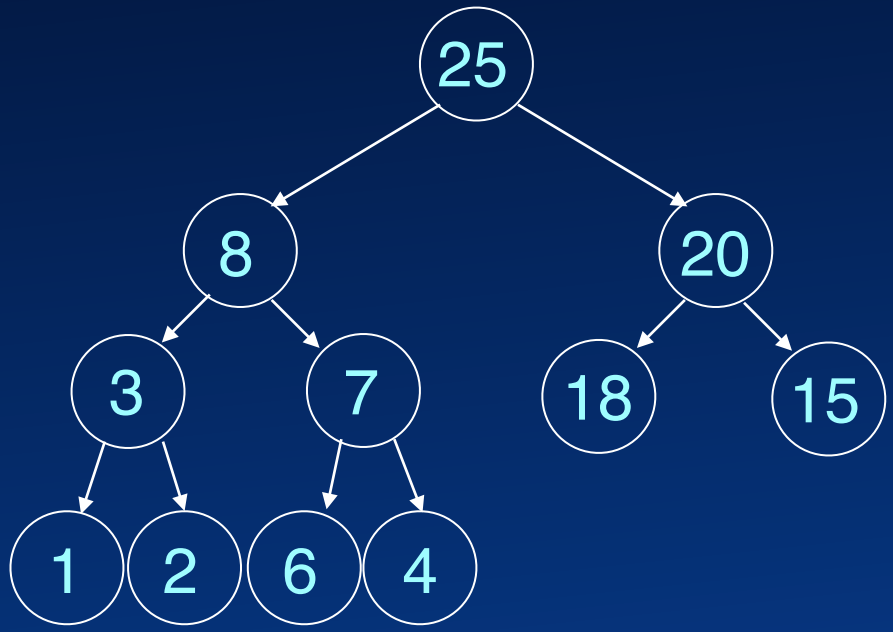
Heap Construction



Heap

Heap Construction

$? \leq \lg 1 + \lg 2 + \lg 3 + \ldots \leq ?$

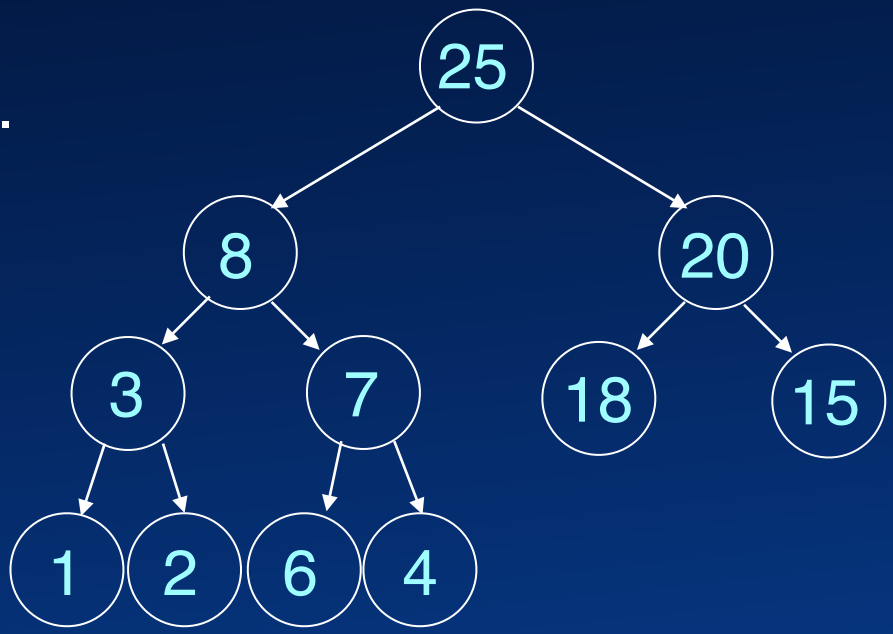


Heap

Heap Construction

$? \leq \lg 1 + \lg 2 + \lg 3 + \ldots \leq ?$

$\lg 1 + 1 + \lg 2 + 1 + \lg 3 + 1 \ldots$



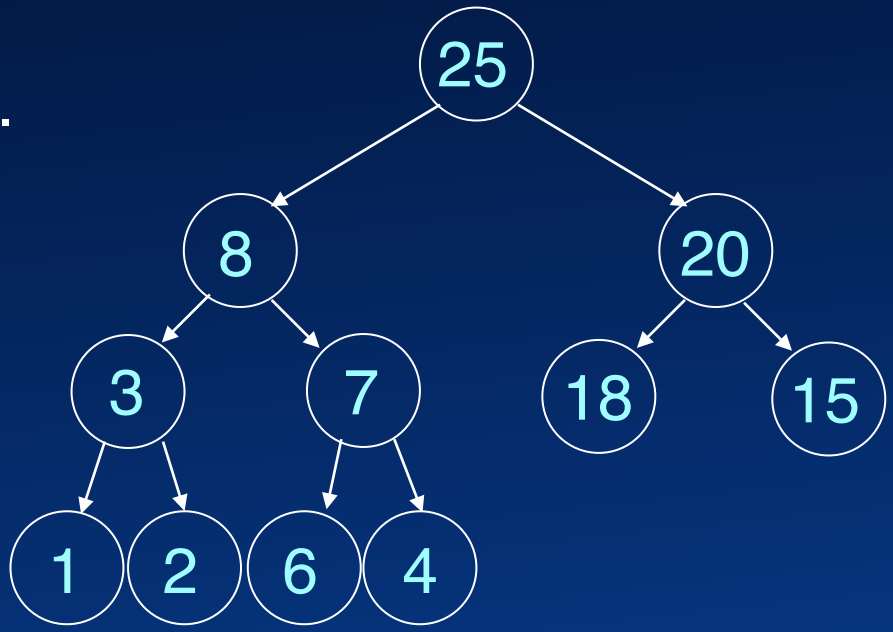
Heap

Heap Construction

$? \leq \lg 1 + \lg 2 + \lg 3 + \ldots \leq ?$

$\lg 1 + 1 + \lg 2 + 1 + \lg 3 + 1 \ldots$

$\lg (1 * 2 * 3 \ldots n\text{-}1) = \lg (n\text{-}1)!$



Heap

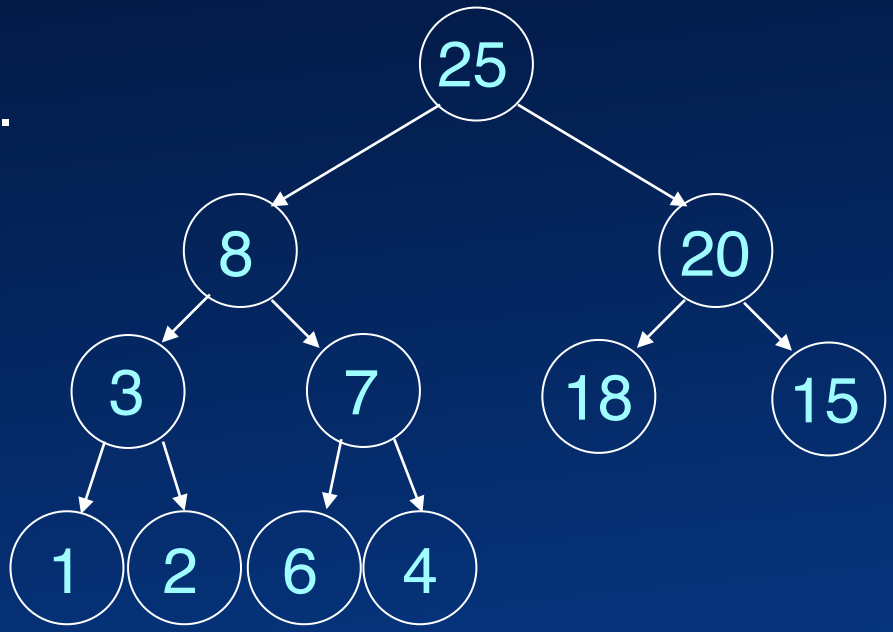
Heap Construction

? ≤ lg 1 + lg 2 + lg 3 + … ≤ ?

lg 1 + 1 + lg 2 + 1 + lg 3 + 1 ..

lg (1 * 2 * 3 … n-1) = lg (n-1)!

$\Theta(n \lg n)$



Heap

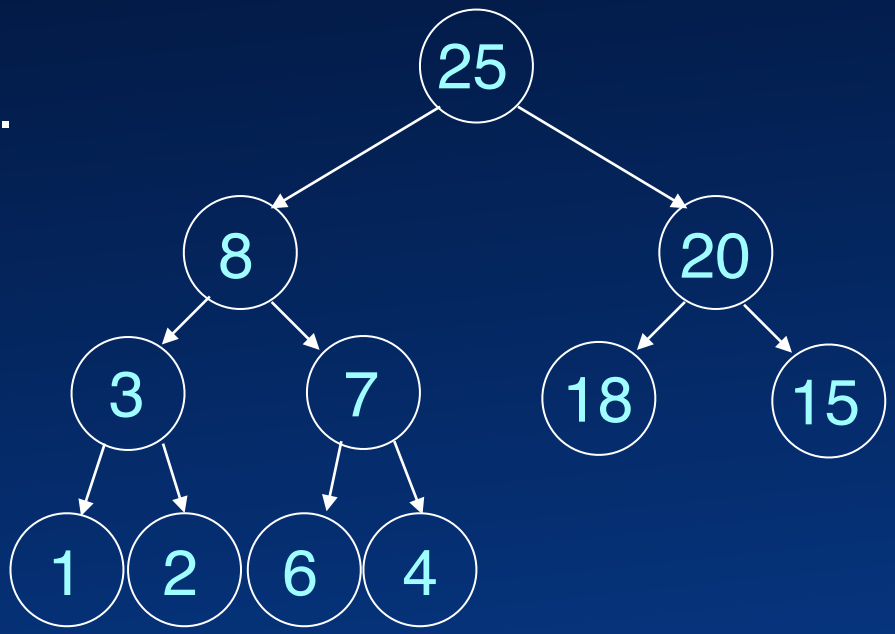
Heap Construction

? ≤ lg 1 + lg 2 + lg 3 + … ≤ ?

lg 1 + 1 + lg 2 + 1 + lg 3 + 1 ..

lg (1 * 2 * 3 … n-1) = lg (n-1)!

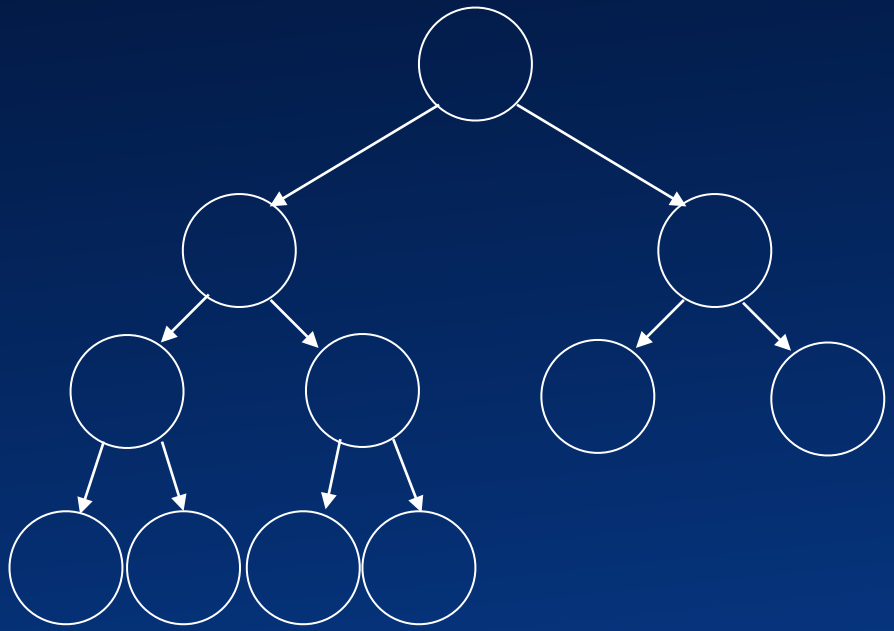
$\Theta(n \lg n)$



Heap

$(n/2)^{n/2} < n! < n^n, n > 1$

Heap Construction

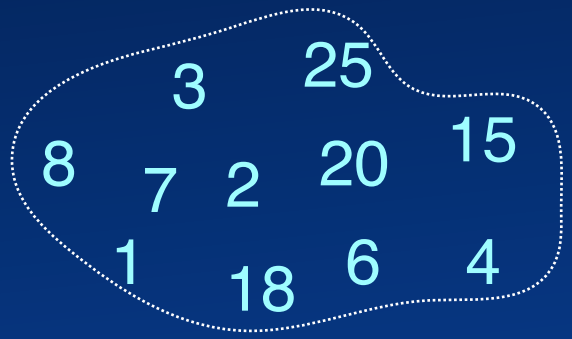


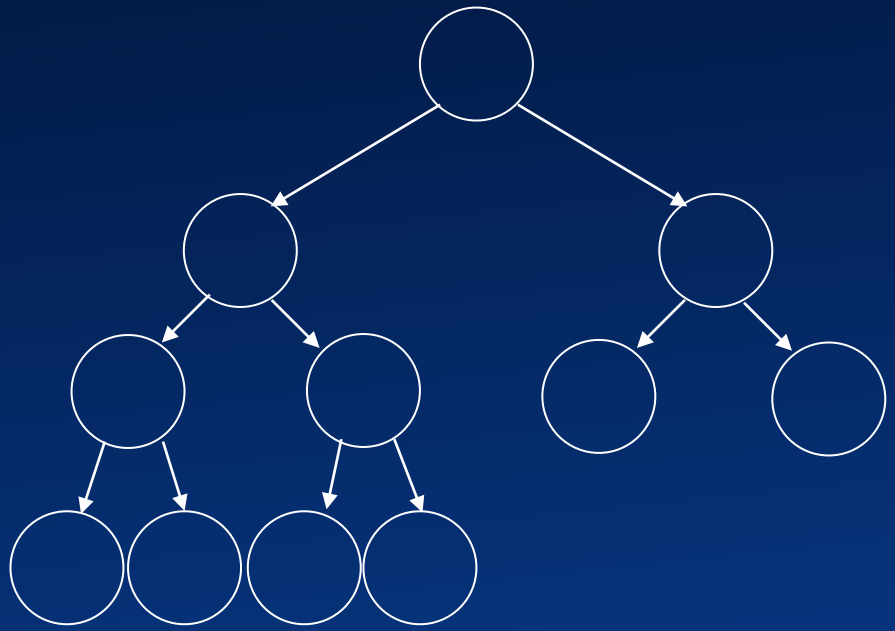
3 25

8 7 2 20 15

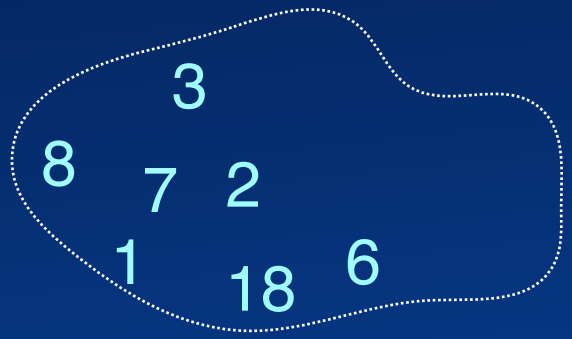
1 18 6 4

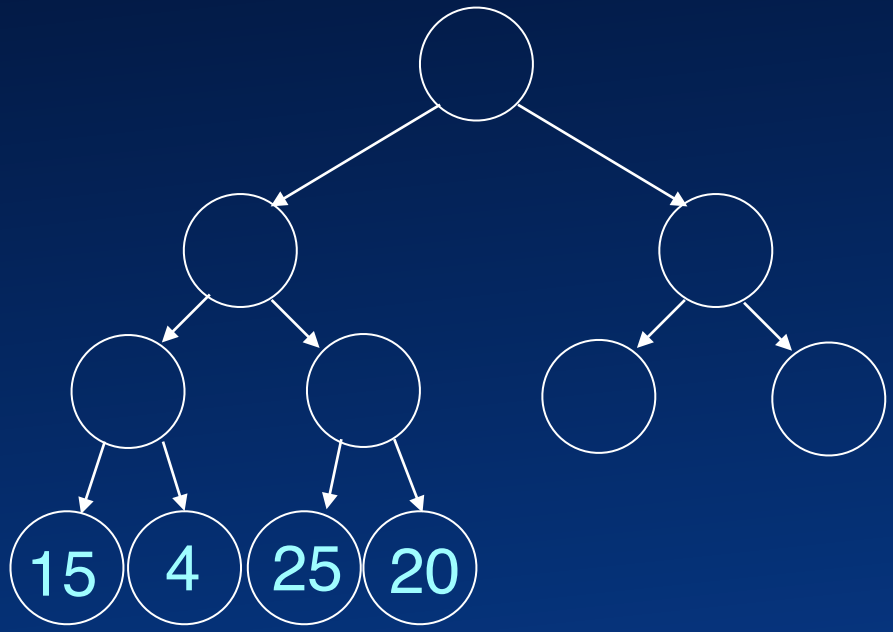
Heap Construction



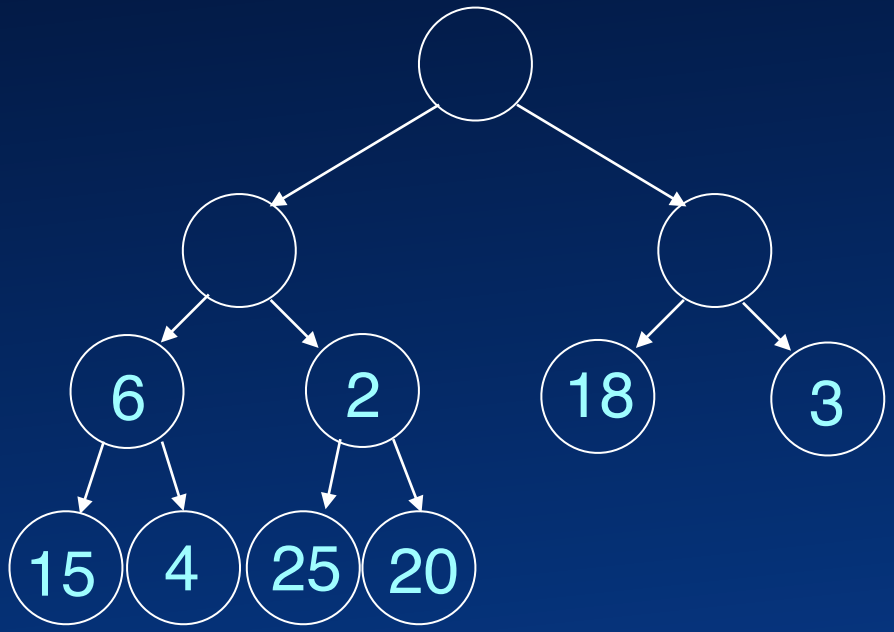


Heap Construction

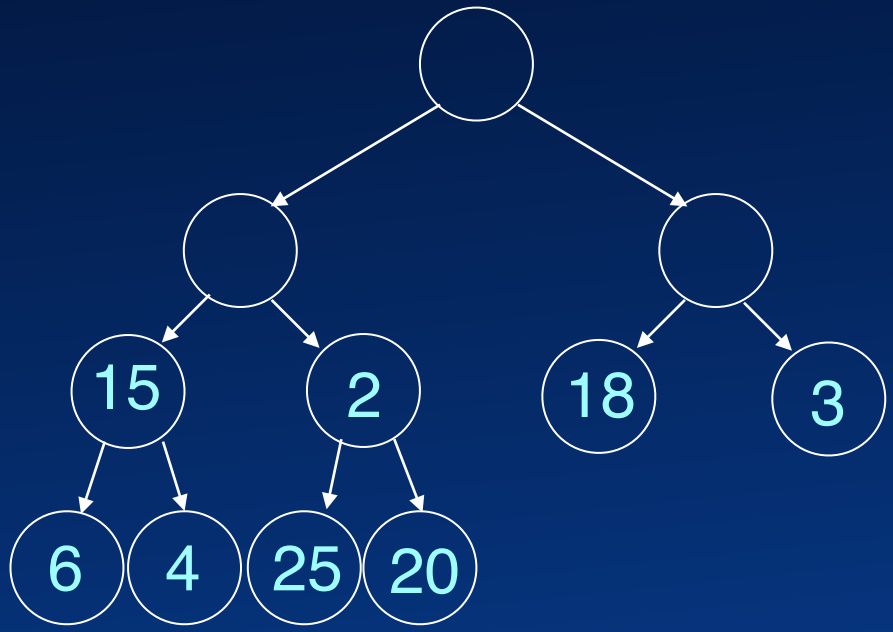




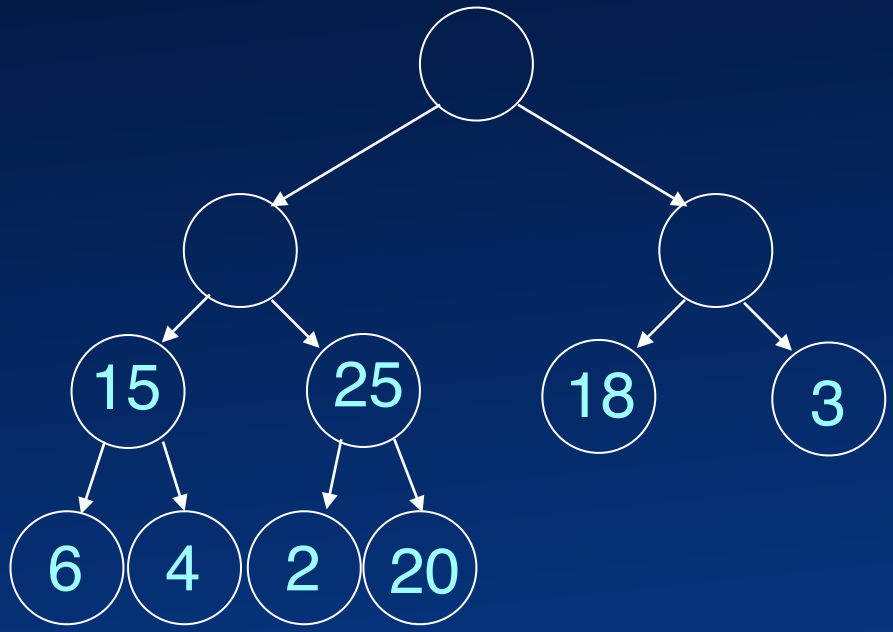
Heap Construction



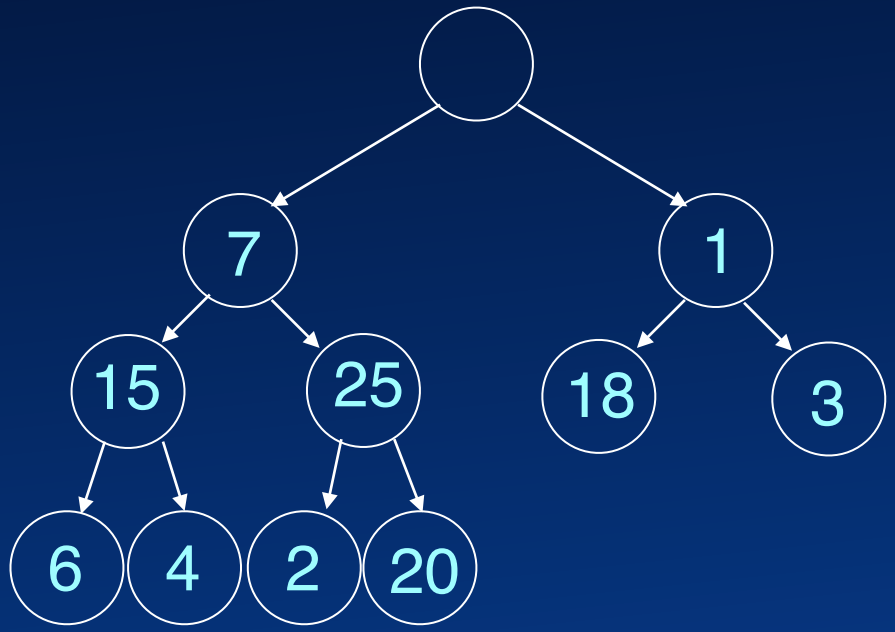
Heap Construction



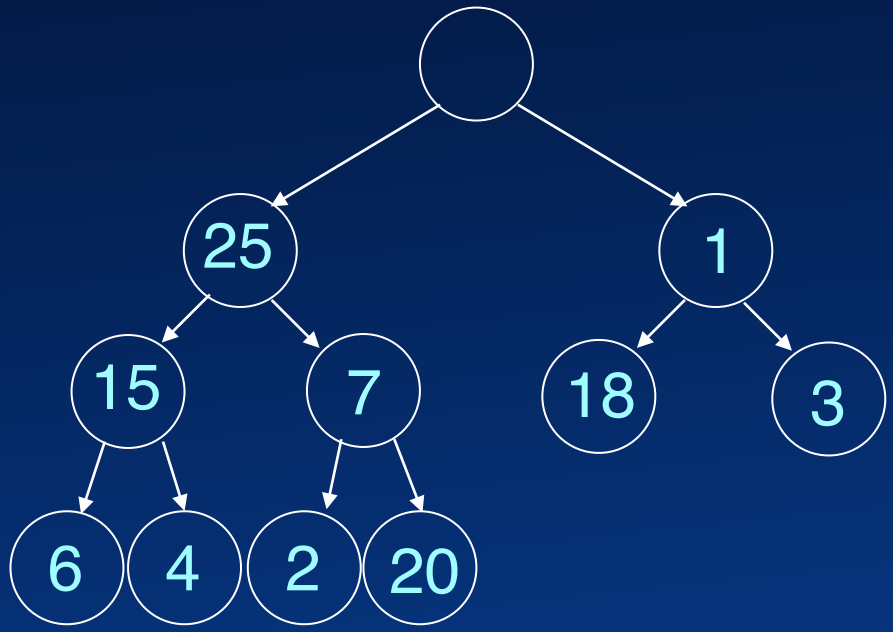
Heap Construction



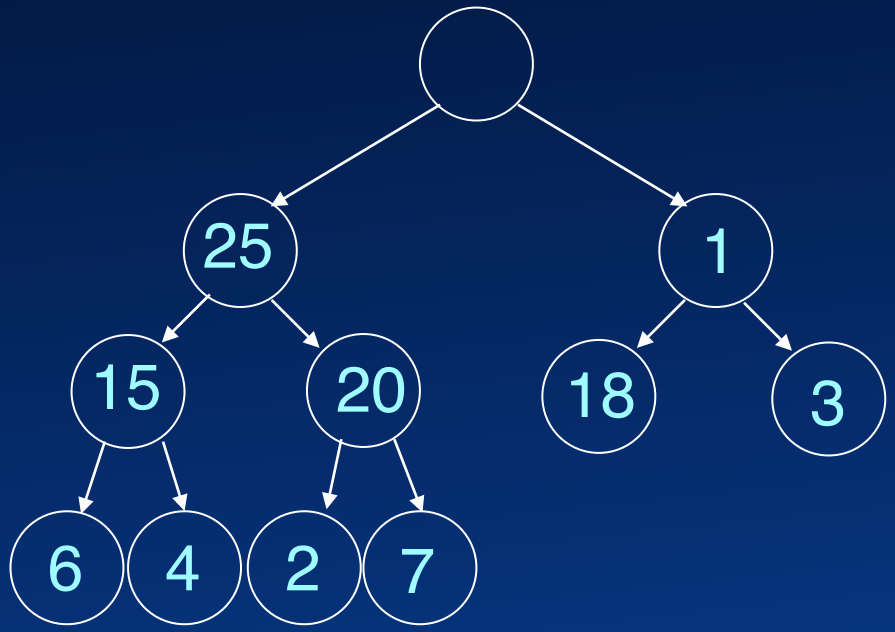
Heap Construction



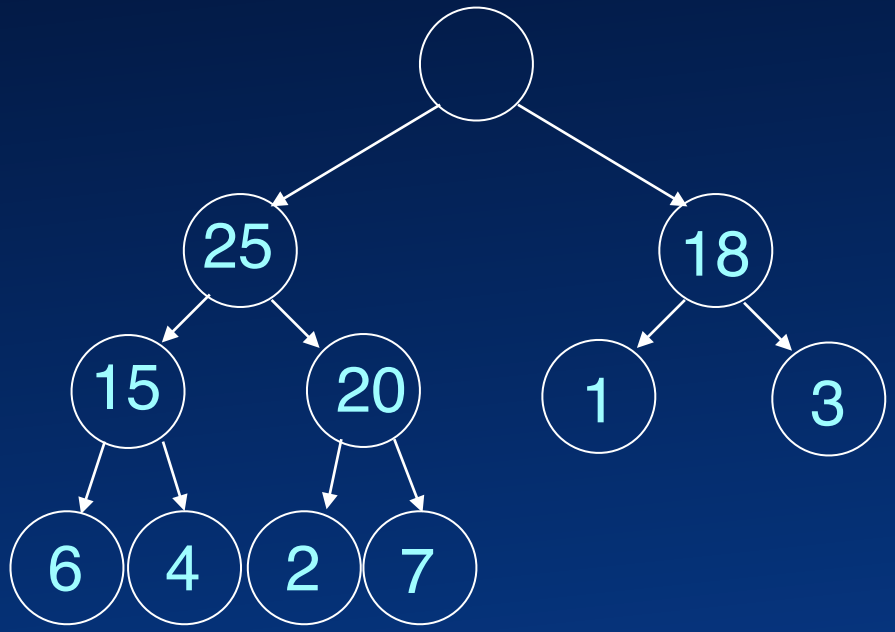
Heap Construction



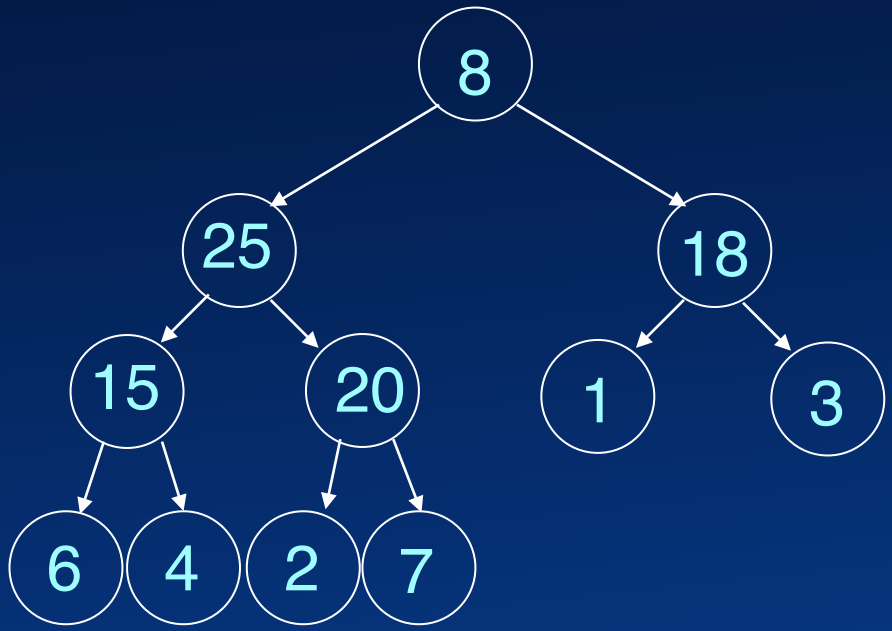
Heap Construction



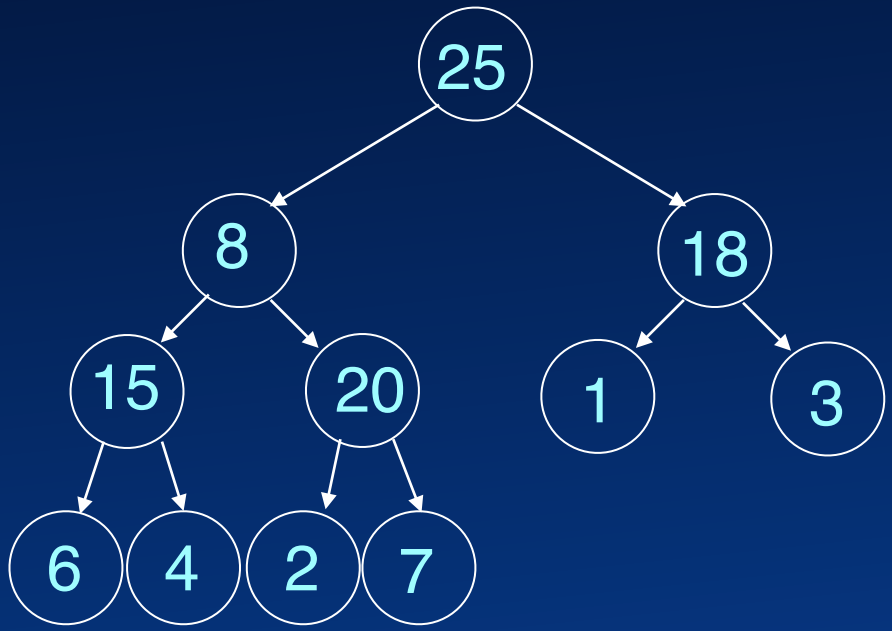
Heap Construction



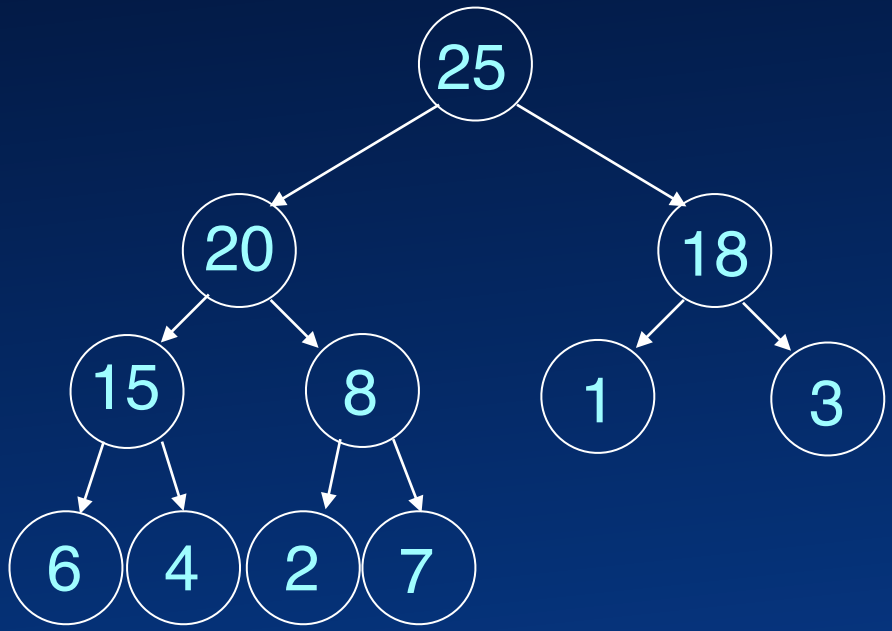
Heap Construction



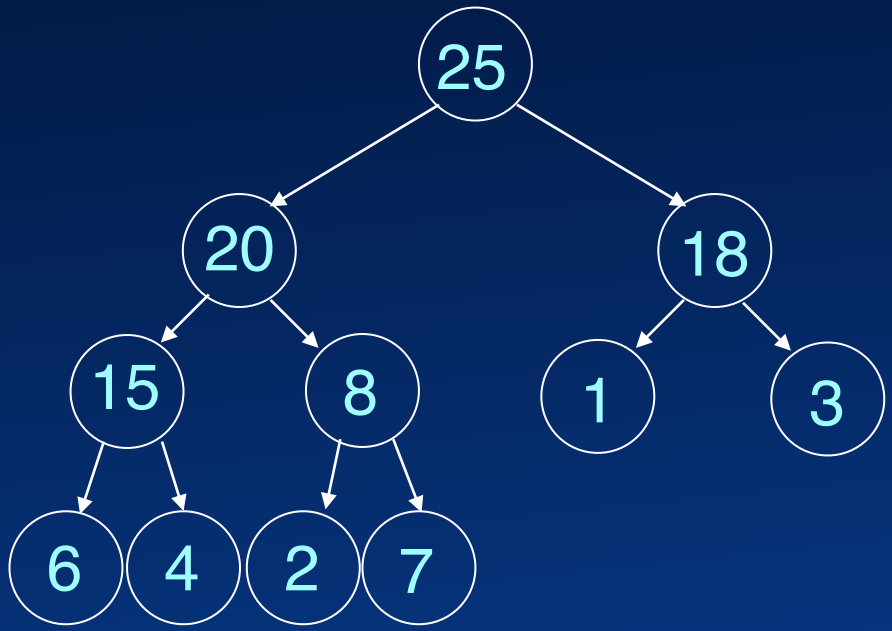
Heap Construction



Heap Construction



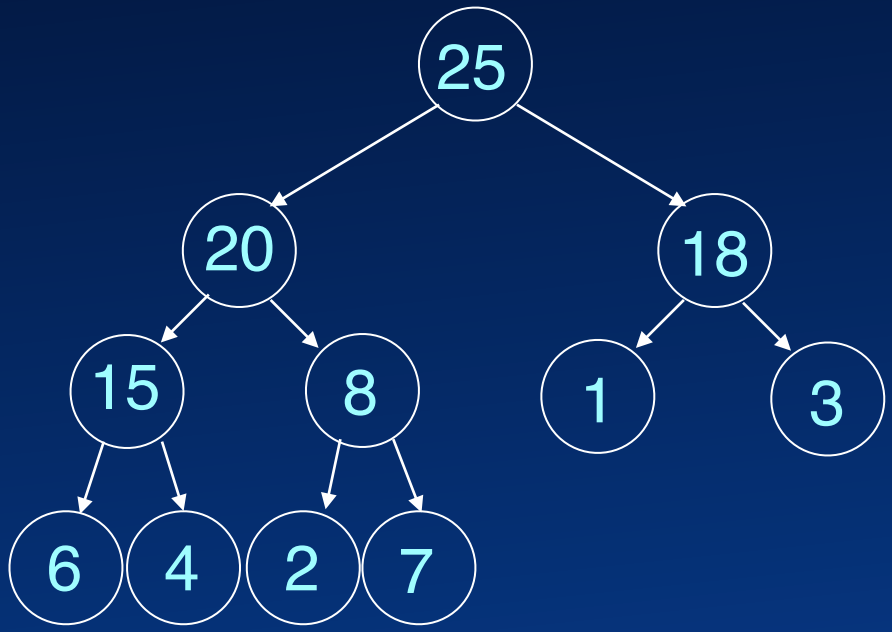
Heap Construction



Heap

Heap Construction

$$T(n) \leq (n/2) + 2(n/4) + 3(n/8) + 4(n/16)$$

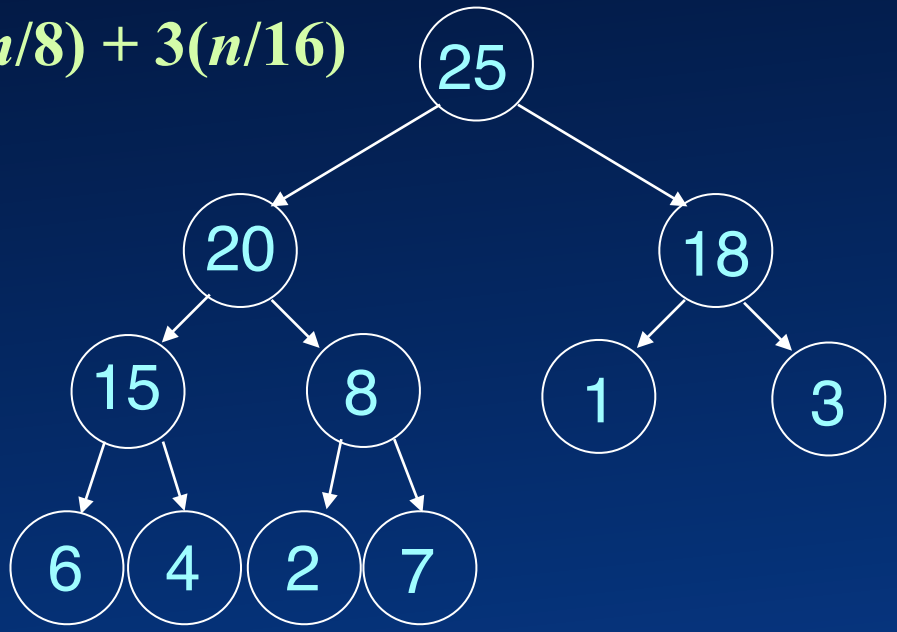


Heap

Heap Construction

$$T(n) \leq (n/2) + 2(n/4) + 3(n/8) + 4(n/16)$$

$$= n/2 + n/4 + n/8 + .. (n/4) + 2(n/8) + 3(n/16)$$

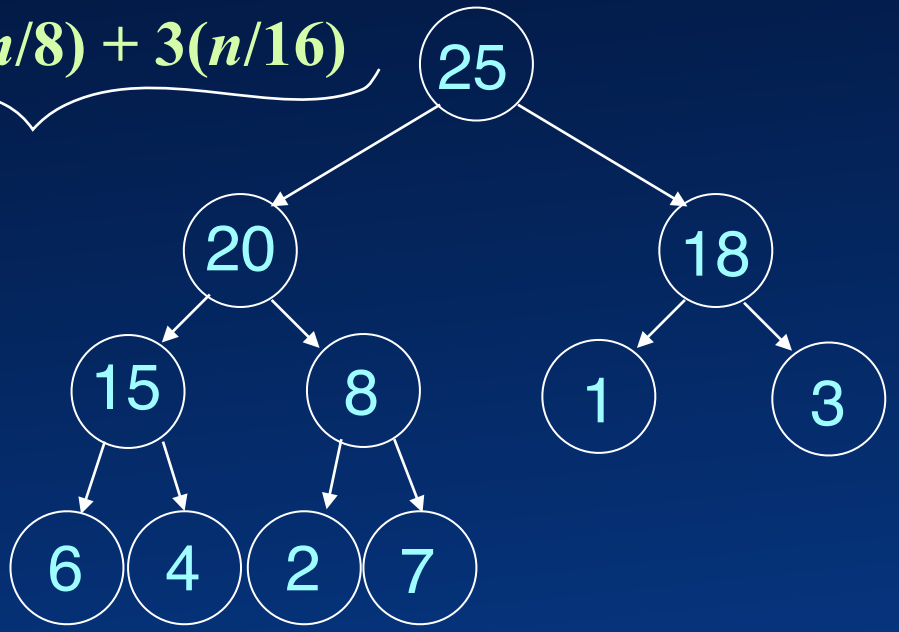


Heap

Heap Construction

$$T(n) \leq (n/2) + 2(n/4) + 3(n/8) + 4(n/16)$$

$$= n/2 + n/4 + n/8 + .. (n/4) + 2(n/8) + 3(n/16)$$



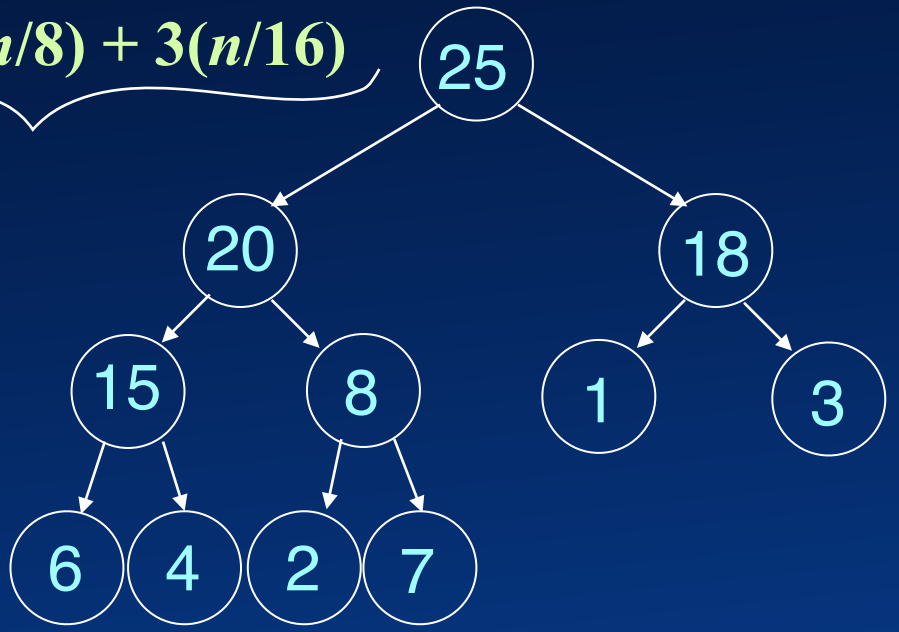
Heap

Heap Construction

$T(n) \leq (n/2) + 2(n/4) + 3(n/8) + 4(n/16)$

$= n/2 + n/4 + n/8 + .. (n/4) + 2(n/8) + 3(n/16)$

$T(n) \leq (n\text{-}1) + T(n/2)$



Heap

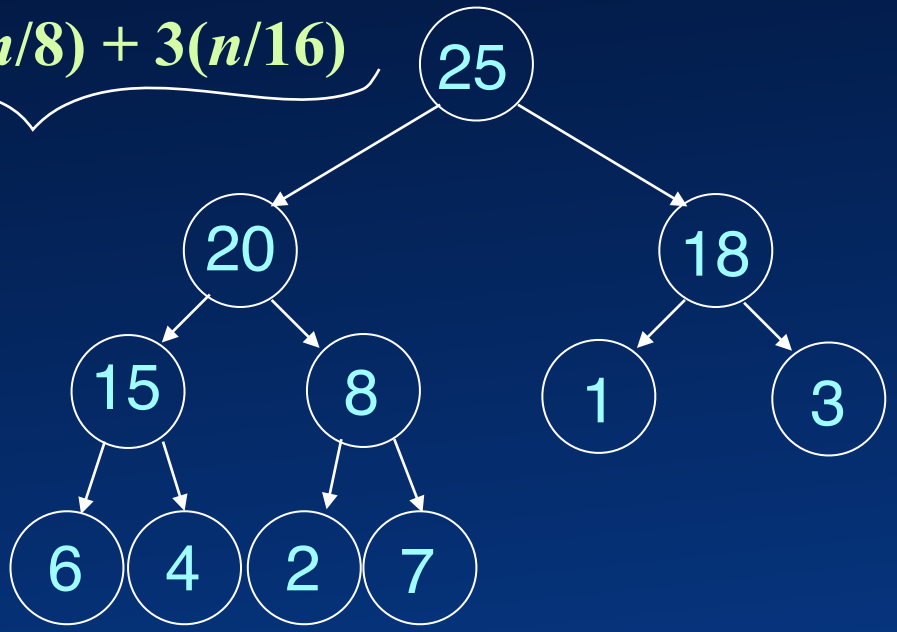
Heap Construction

$T(n) \leq (n/2) + 2(n/4) + 3(n/8) + 4(n/16)$

$= n/2 + n/4 + n/8 + .. (n/4) + 2(n/8) + 3(n/16)$

$T(n) \leq (n\text{-}1) + T(n/2)$

$= (n\text{-}1) + (n/2 - 1) ..$



Heap

Heap Construction

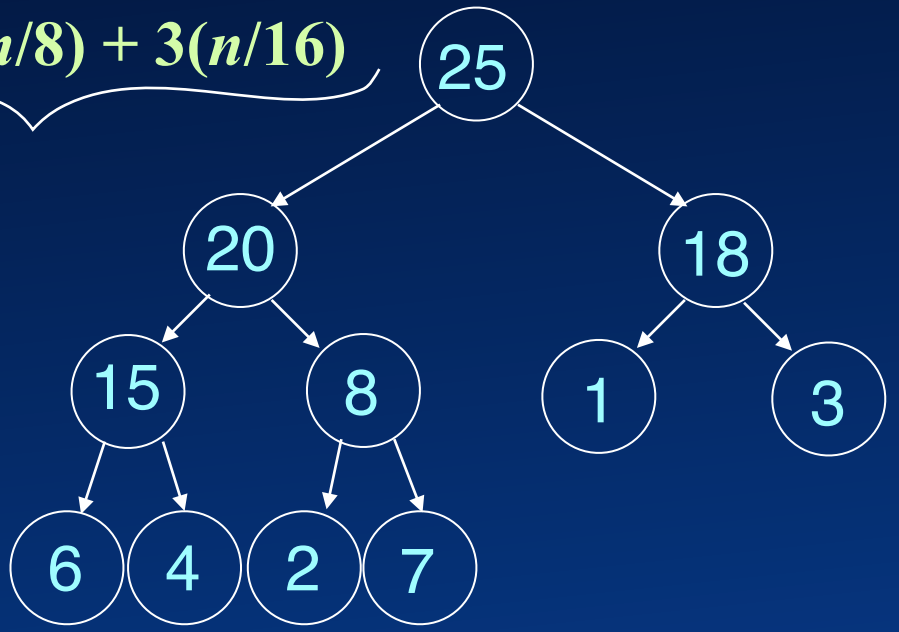
$T(n) \leq (n/2) + 2(n/4) + 3(n/8) + 4(n/16)$

$= n/2 + n/4 + n/8 + .. (n/4) + 2(n/8) + 3(n/16)$

$T(n) \leq (n\text{-}1) + T(n/2)$

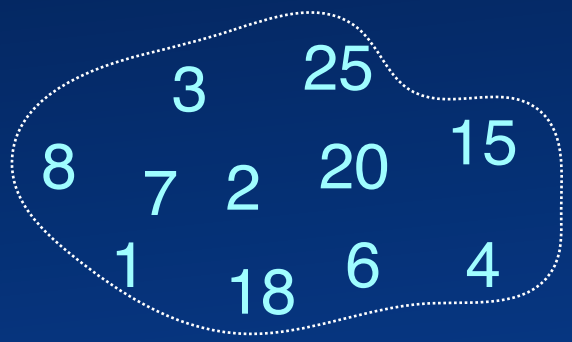
$= (n\text{-}1) + (n/2 - 1) ..$

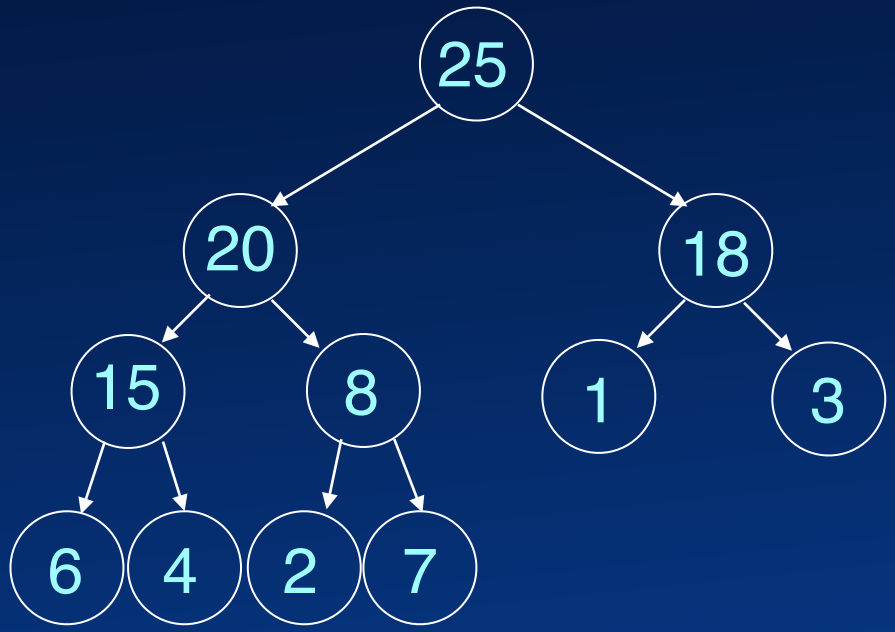
$= \Theta(n)$



Heap

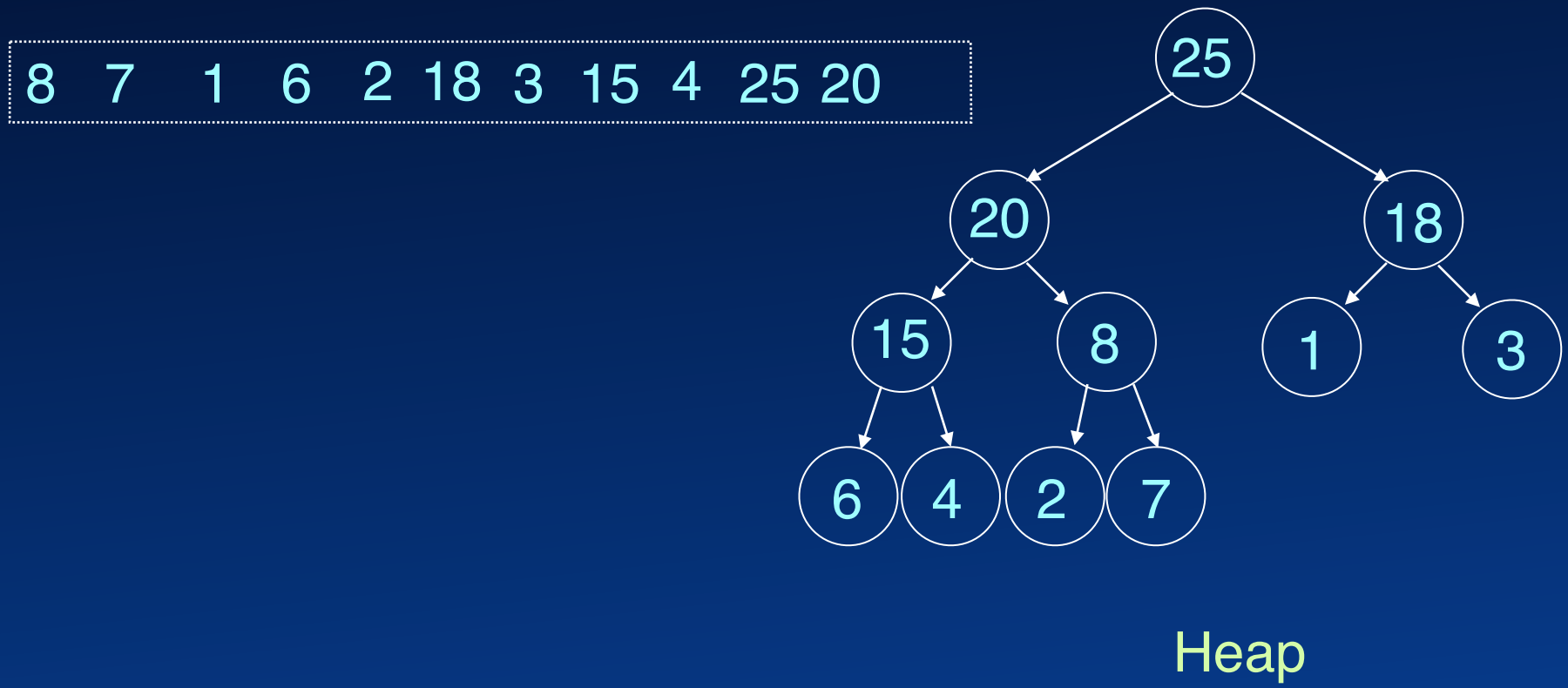
Heap Construction





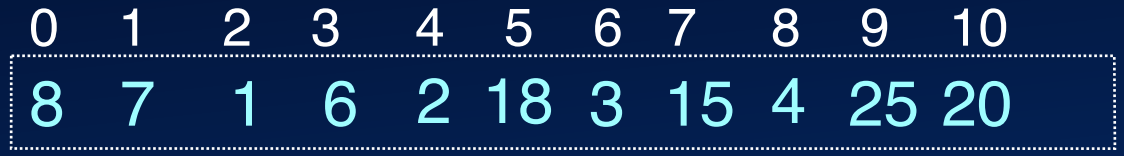
Heap

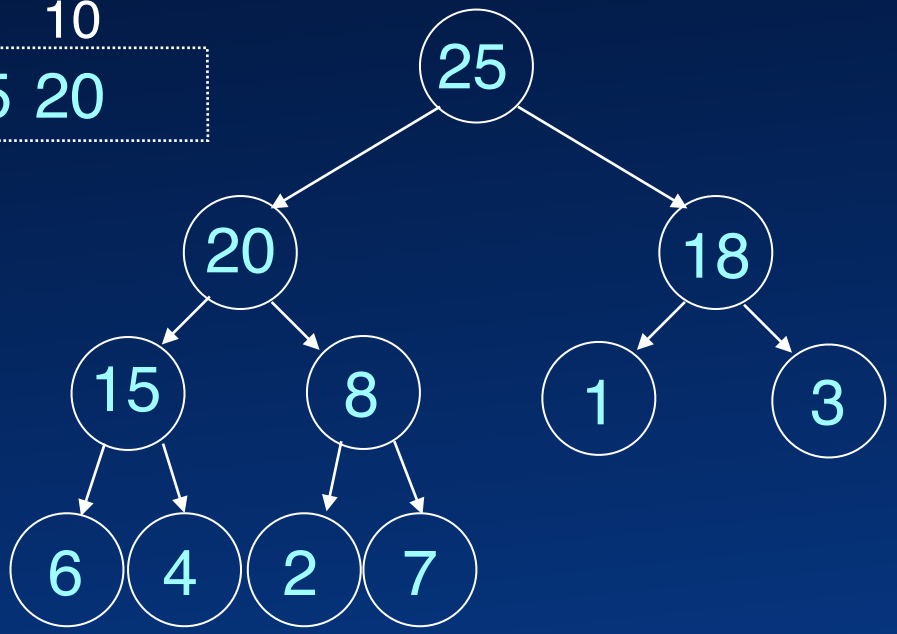
Heap Construction



Heap

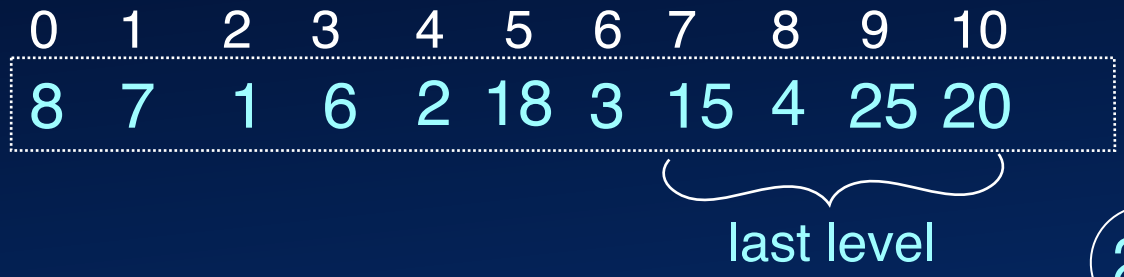
Heap Construction

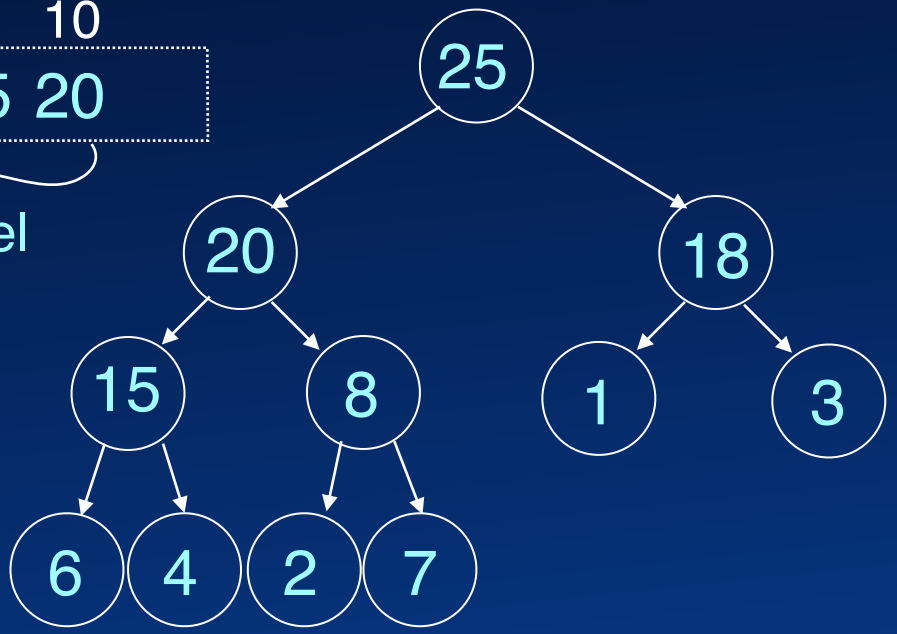




Heap

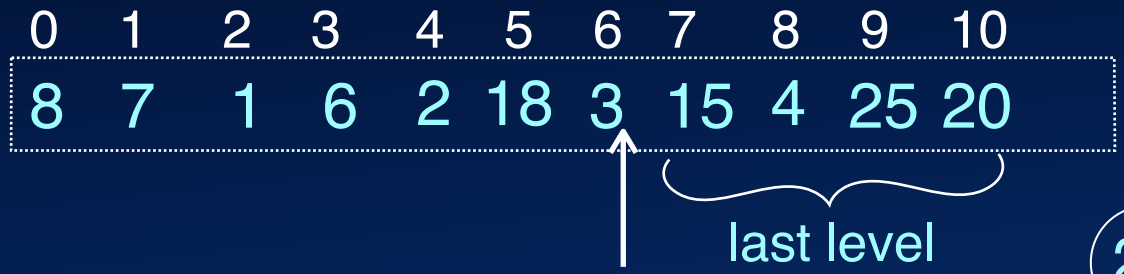
Heap Construction

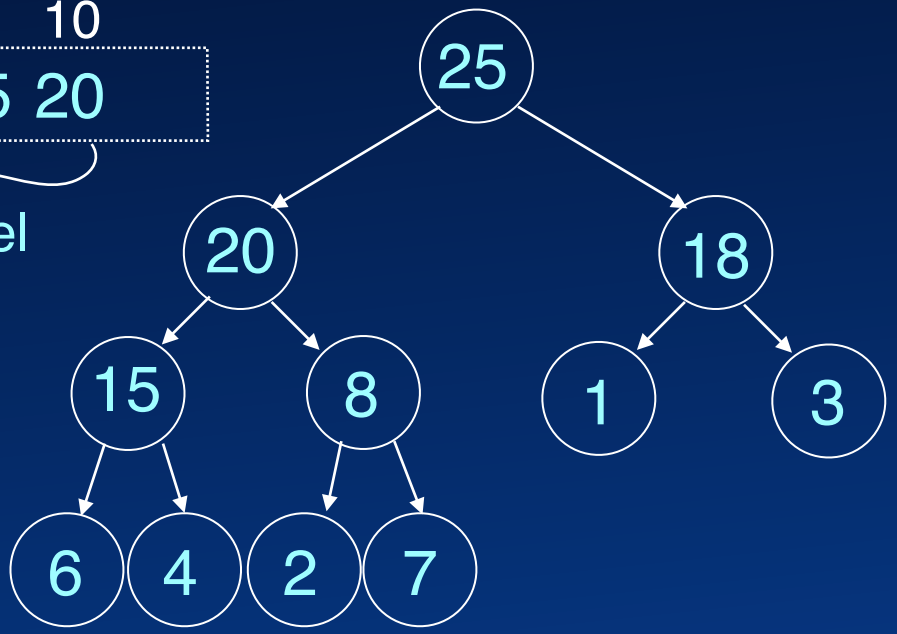




Heap

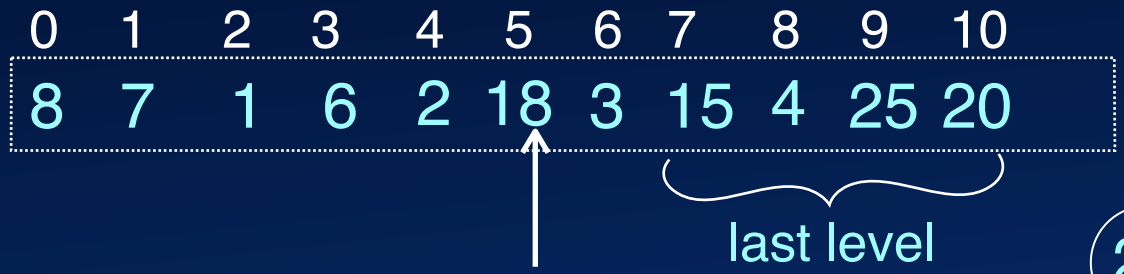
Heap Construction

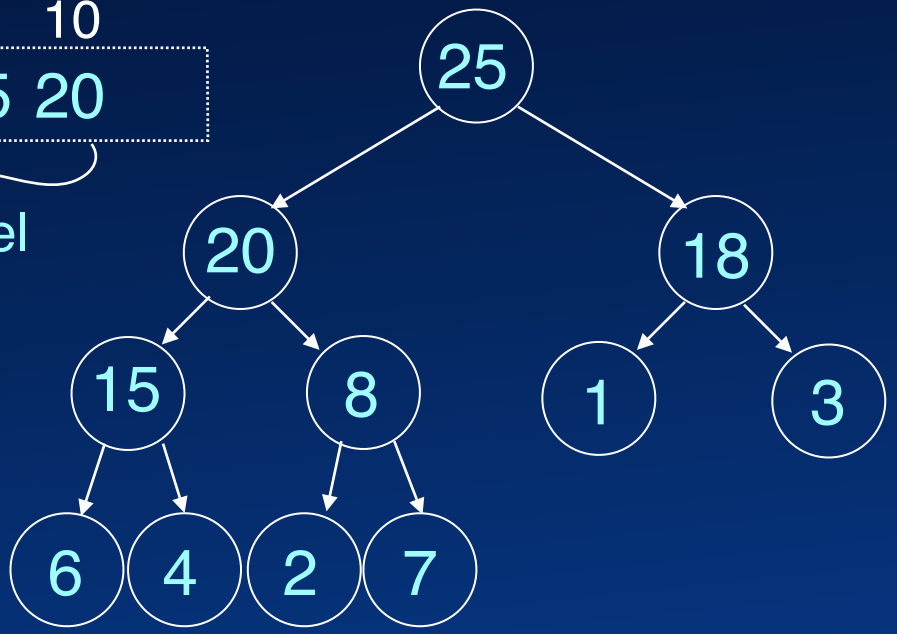




Heap

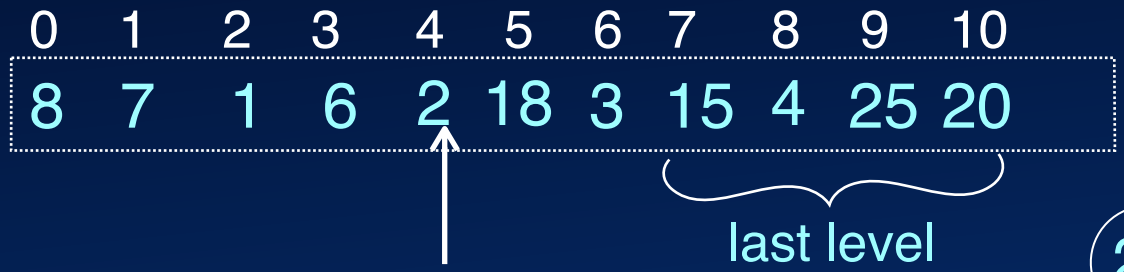
Heap Construction

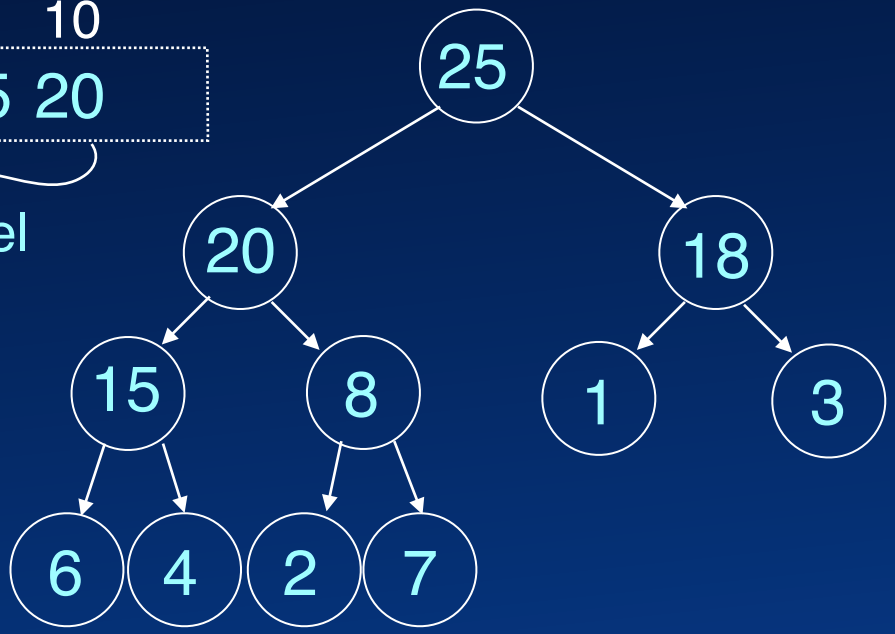




Heap

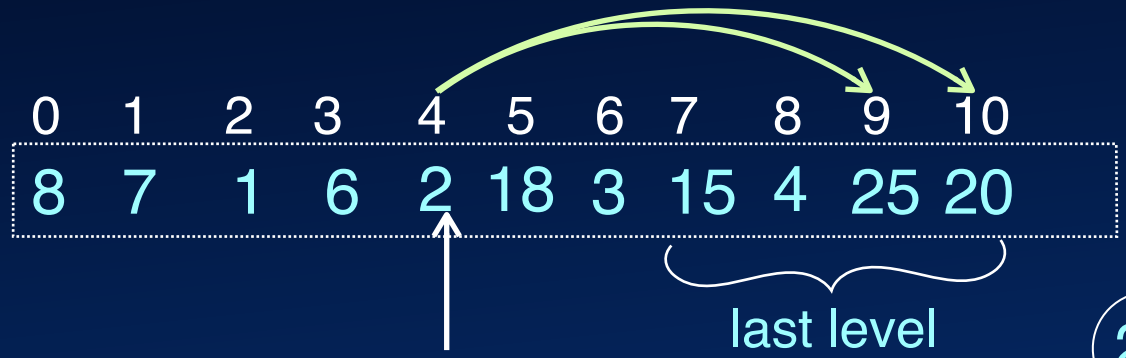
Heap Construction

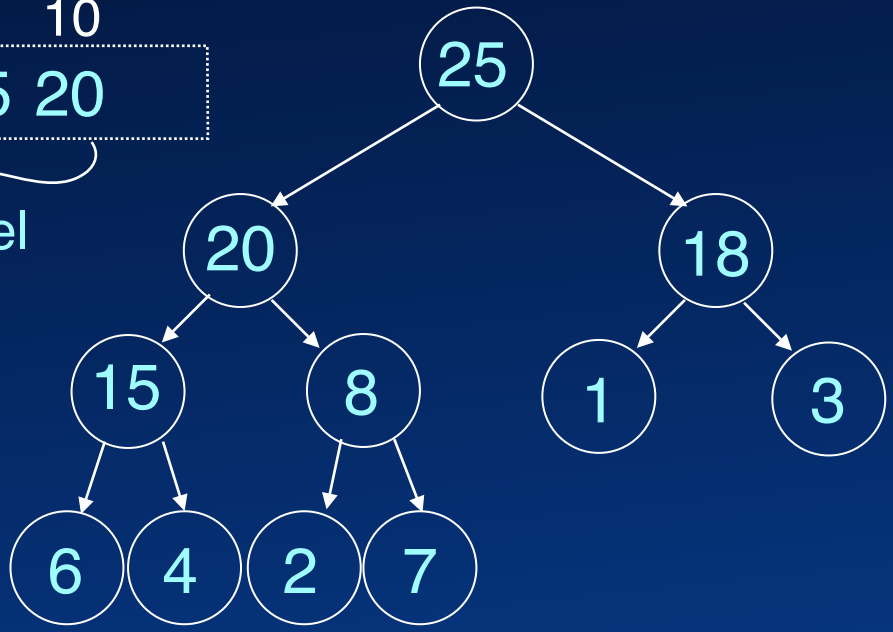




Heap

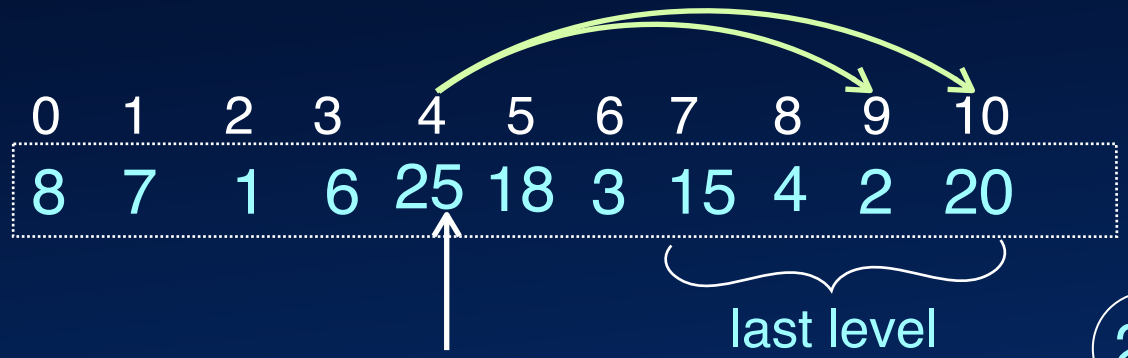
Heap Construction

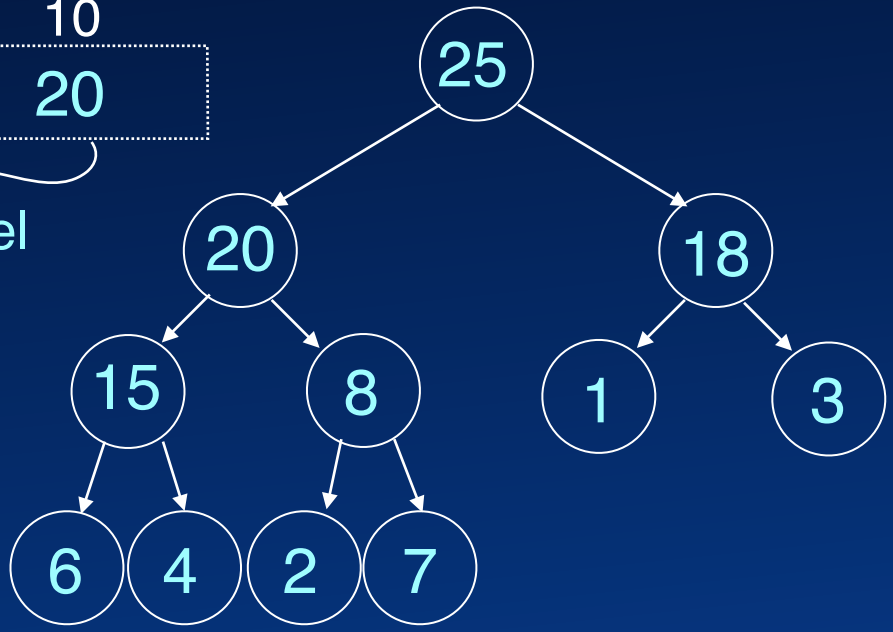




Heap

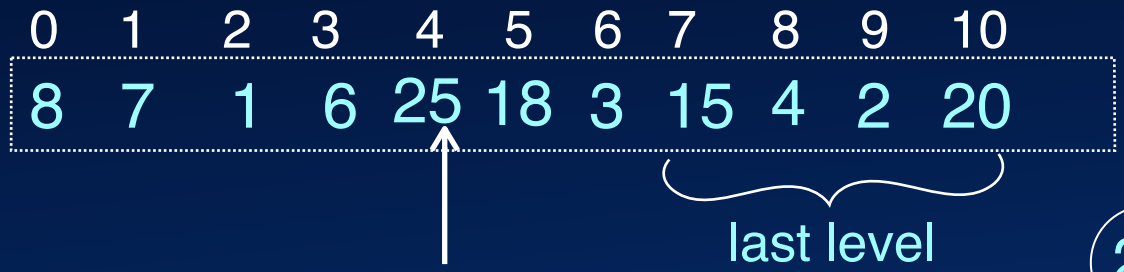
Heap Construction

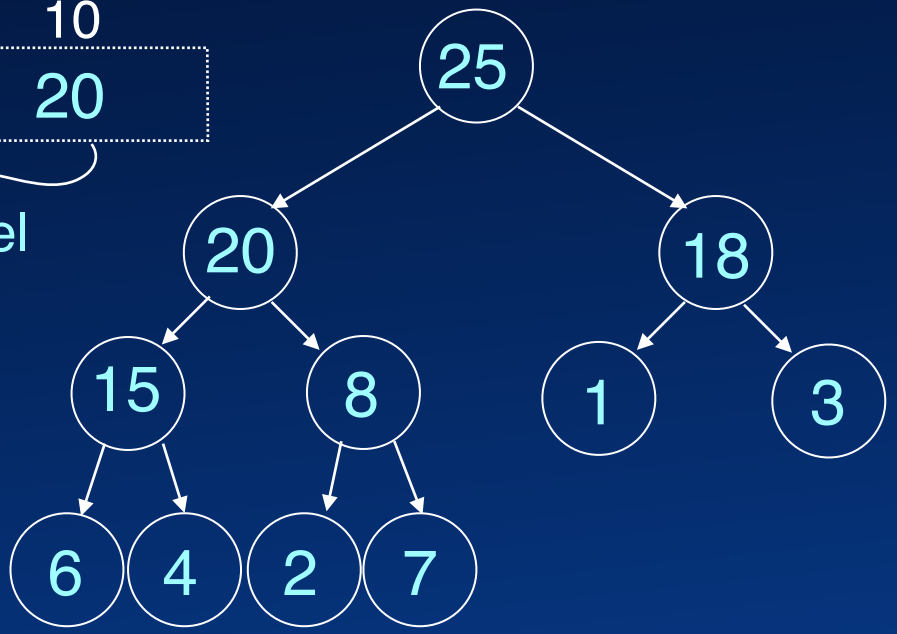




Heap

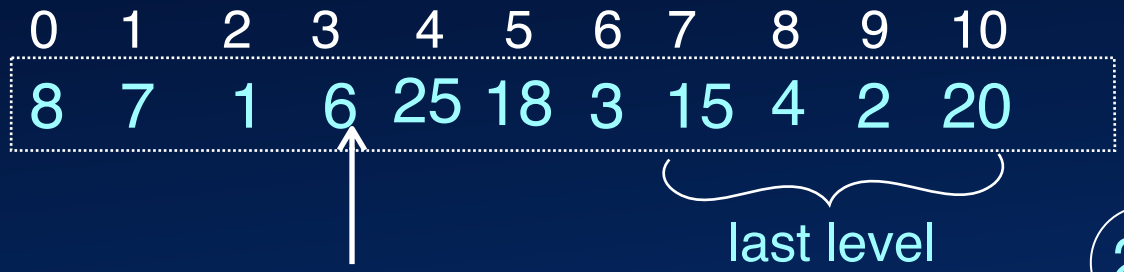
Heap Construction

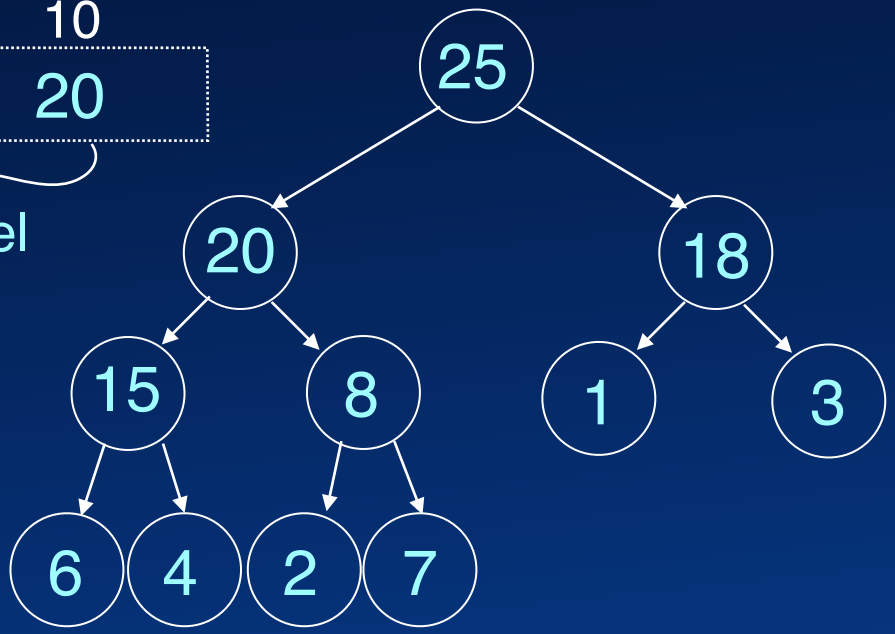




Heap

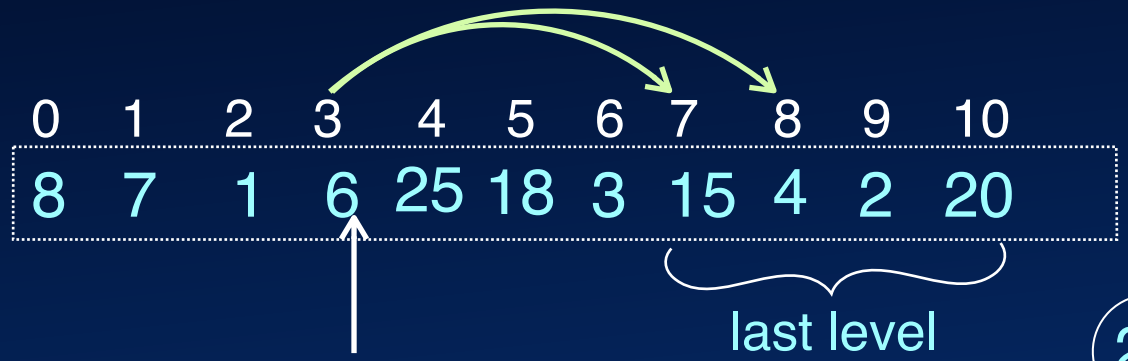
Heap Construction

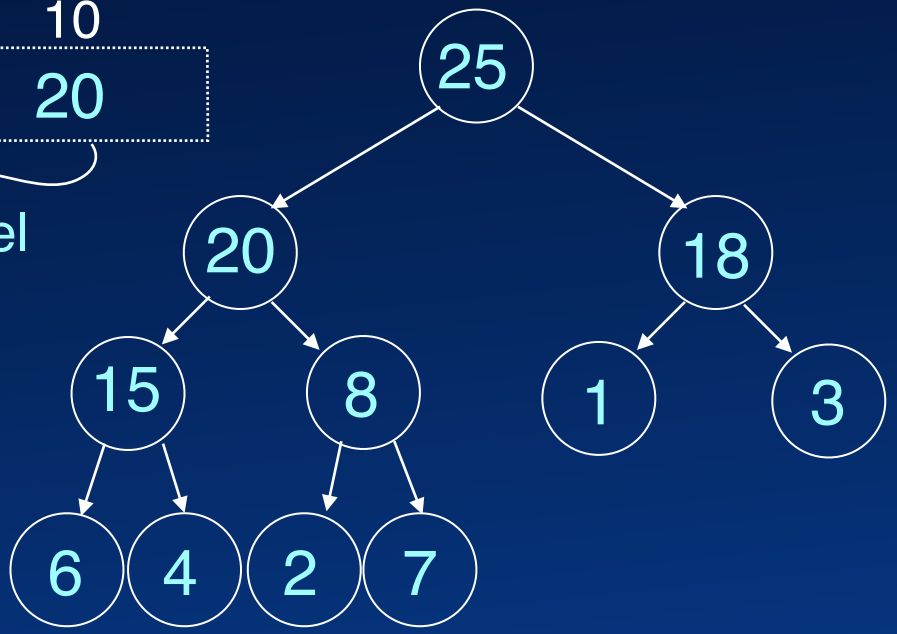




Heap

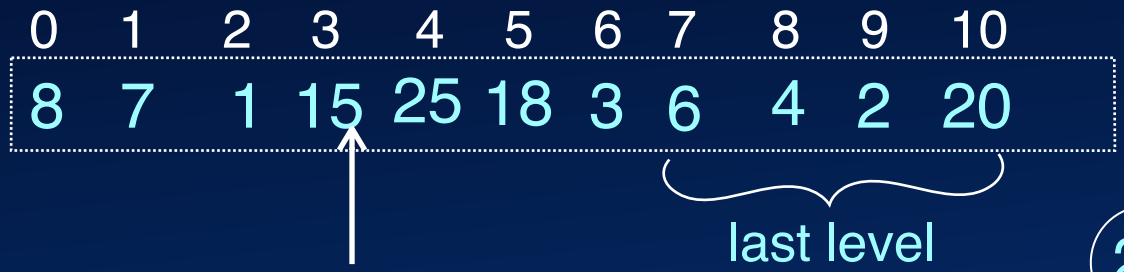
Heap Construction

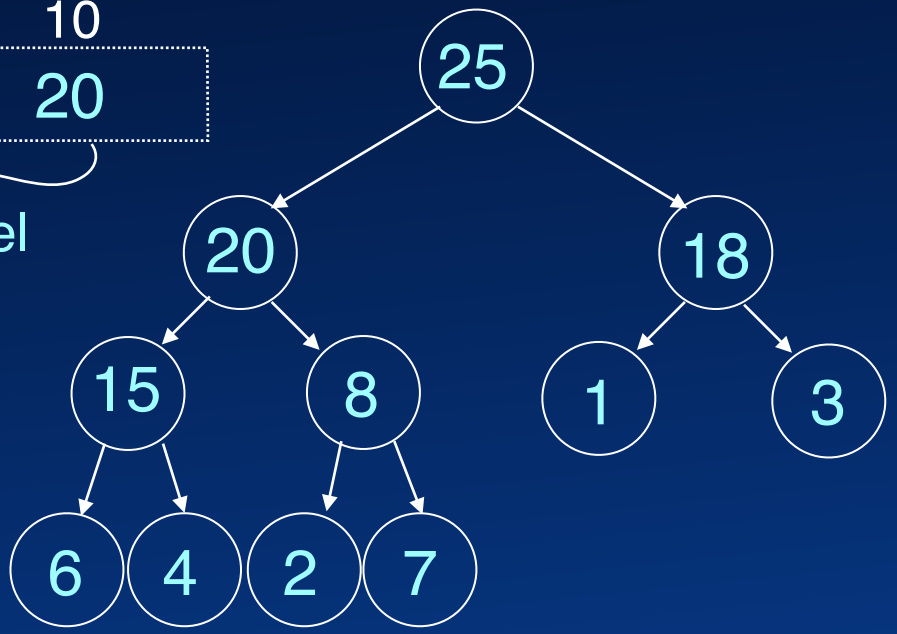




Heap

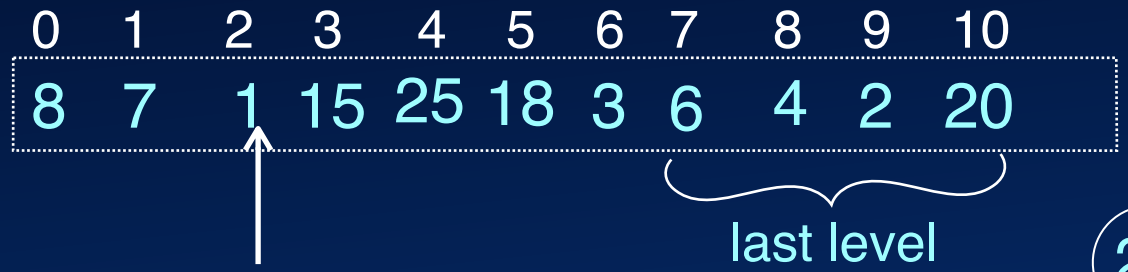
Heap Construction

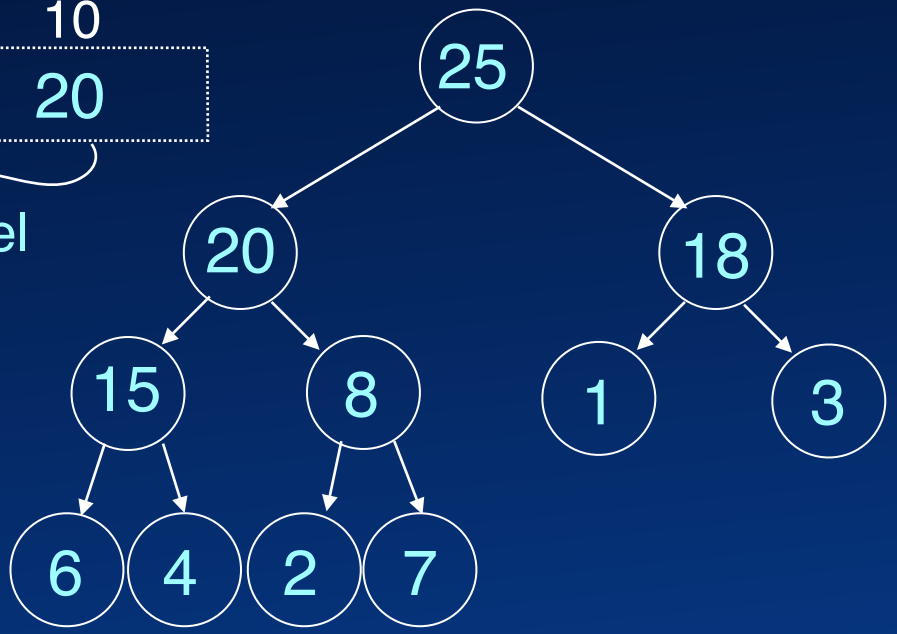




Heap

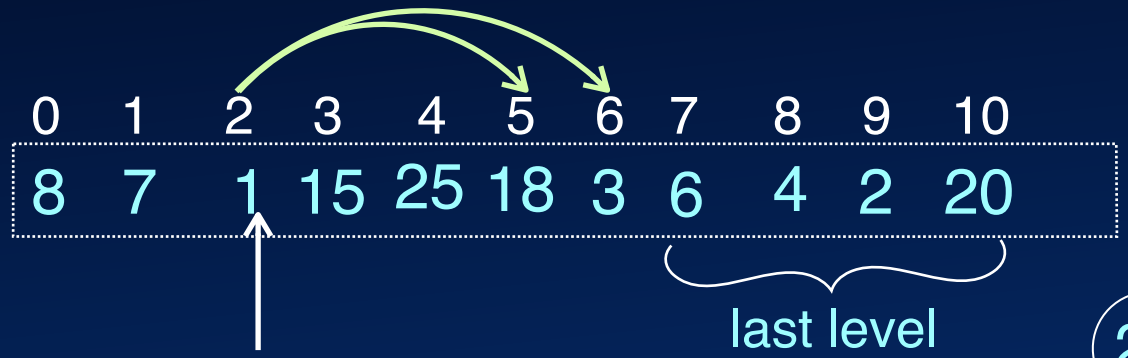
Heap Construction

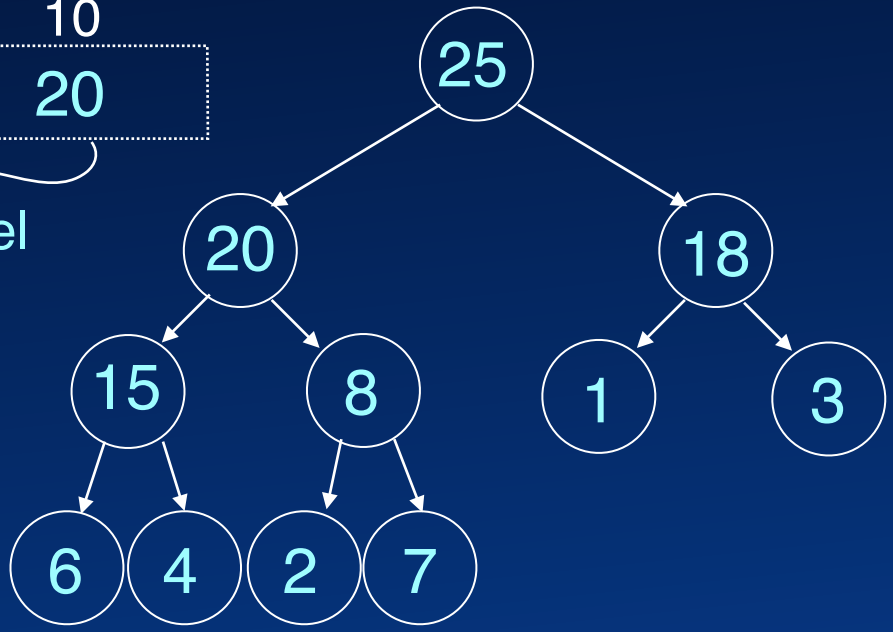




Heap

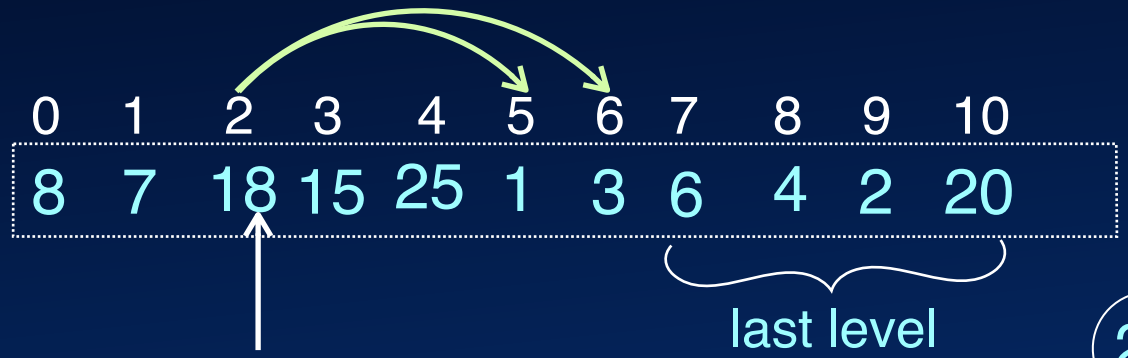
Heap Construction

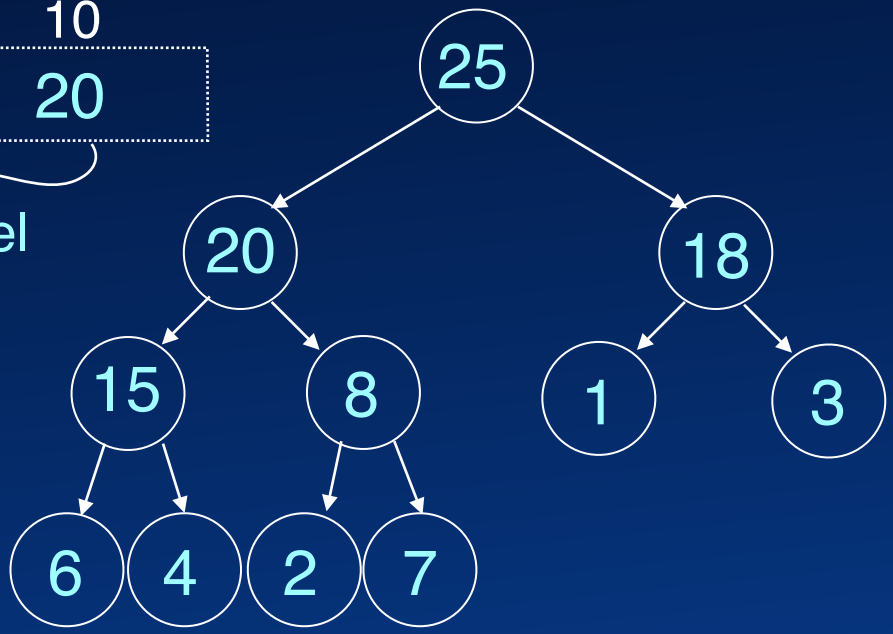




Heap

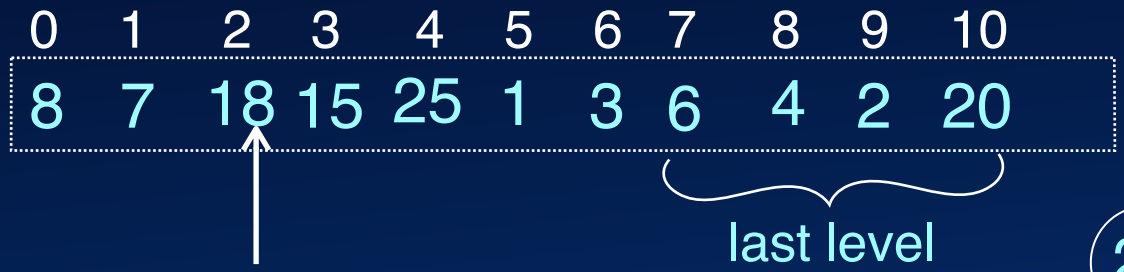
Heap Construction

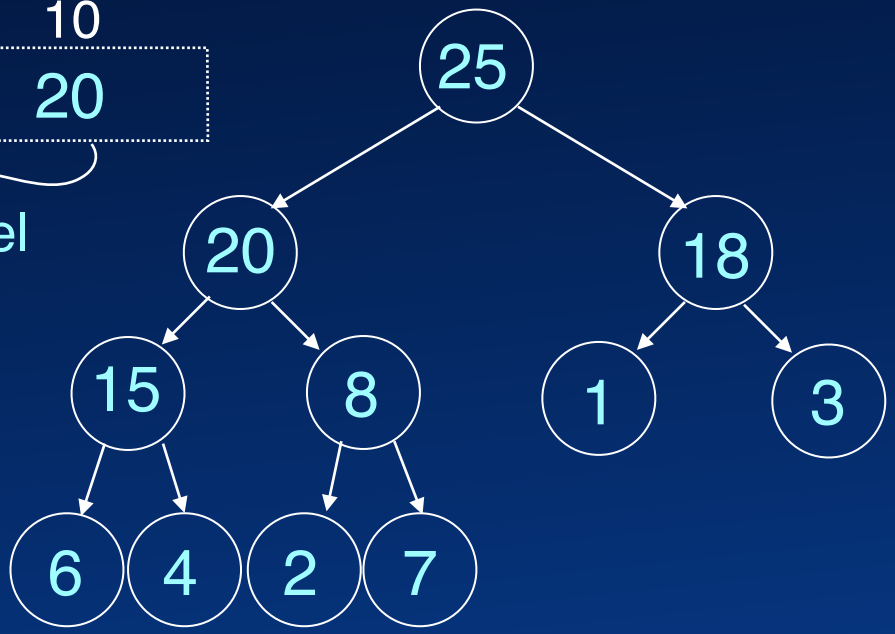




Heap

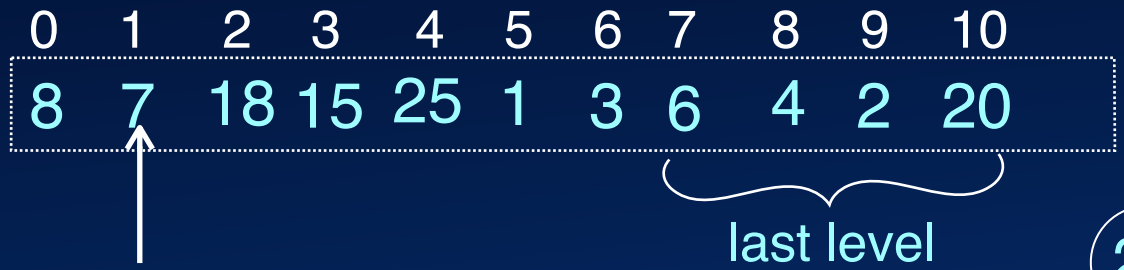
Heap Construction

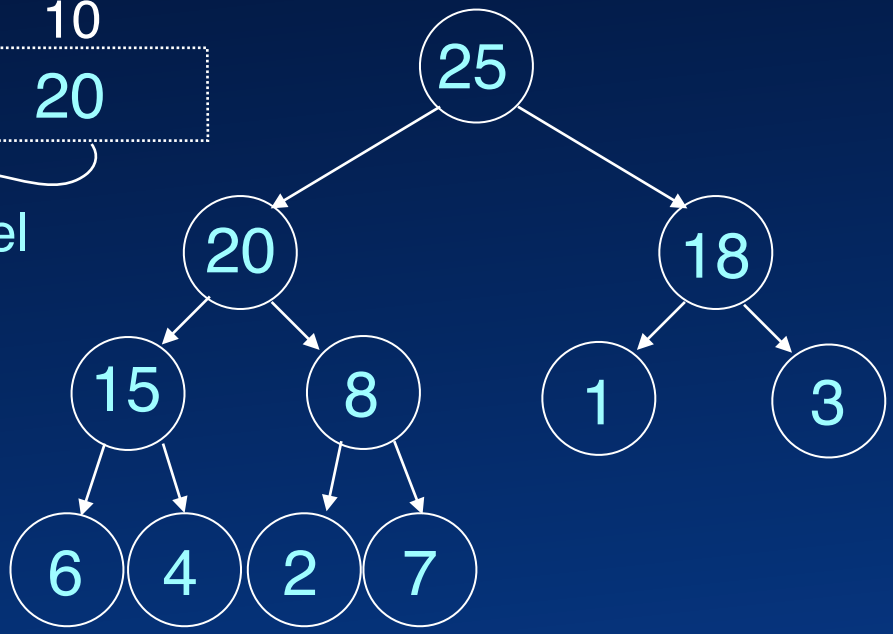




Heap

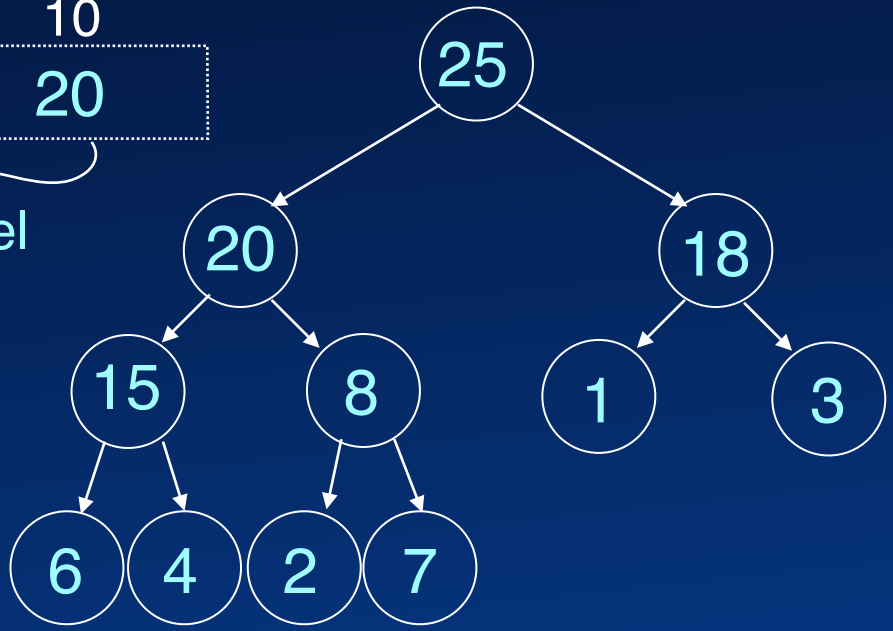
Heap Construction





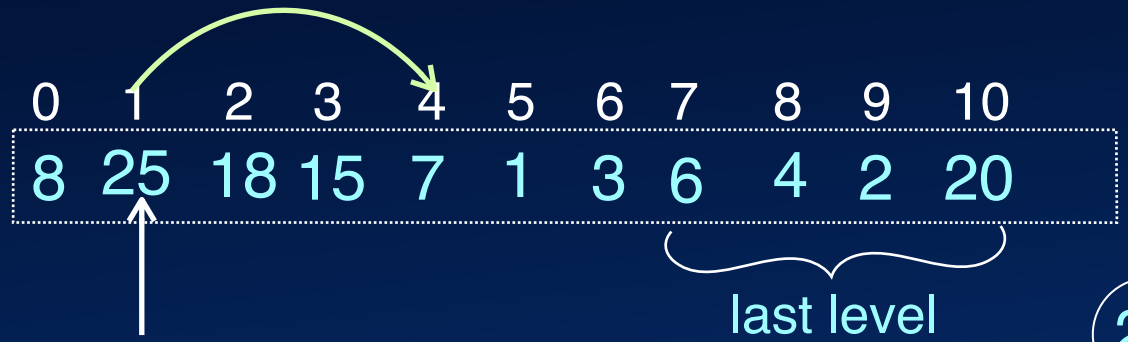
Heap

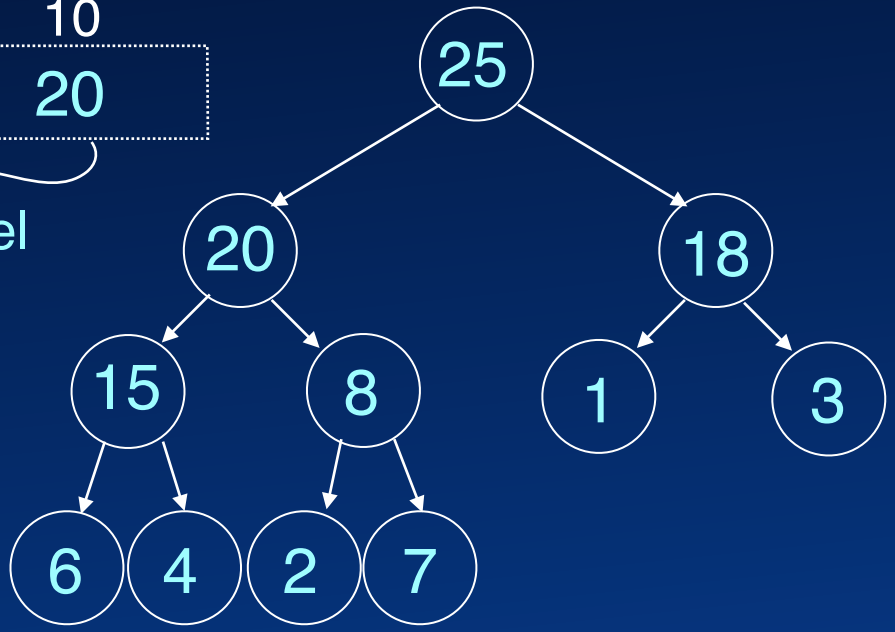
Heap Construction



Heap

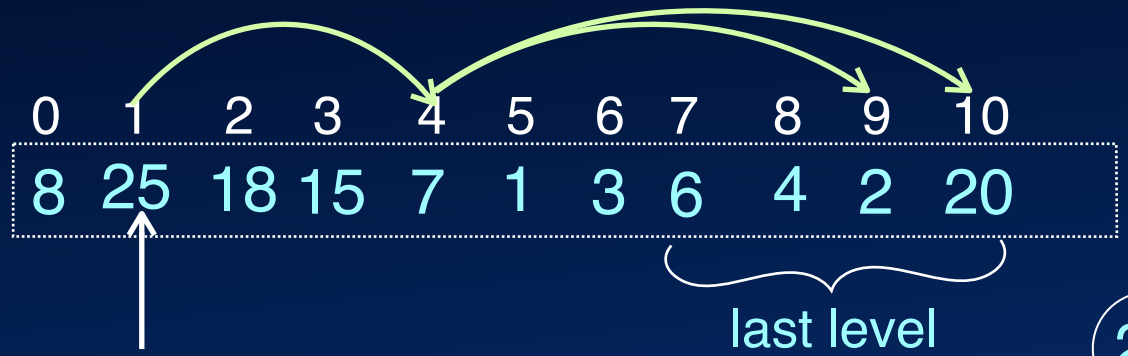
Heap Construction

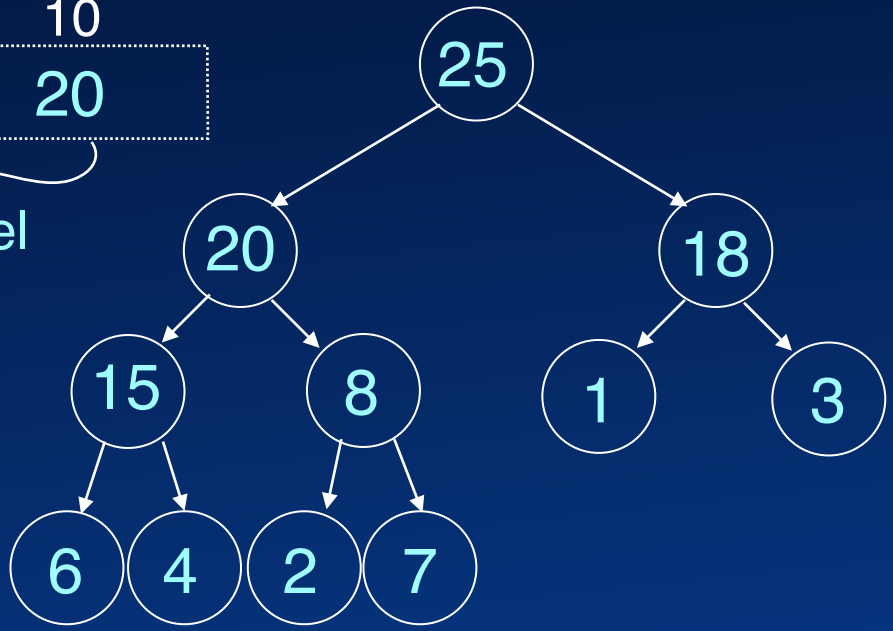




Heap

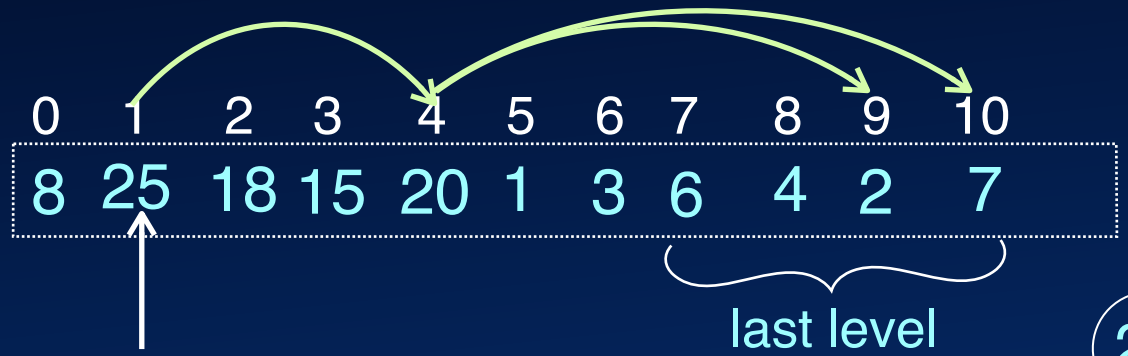
Heap Construction

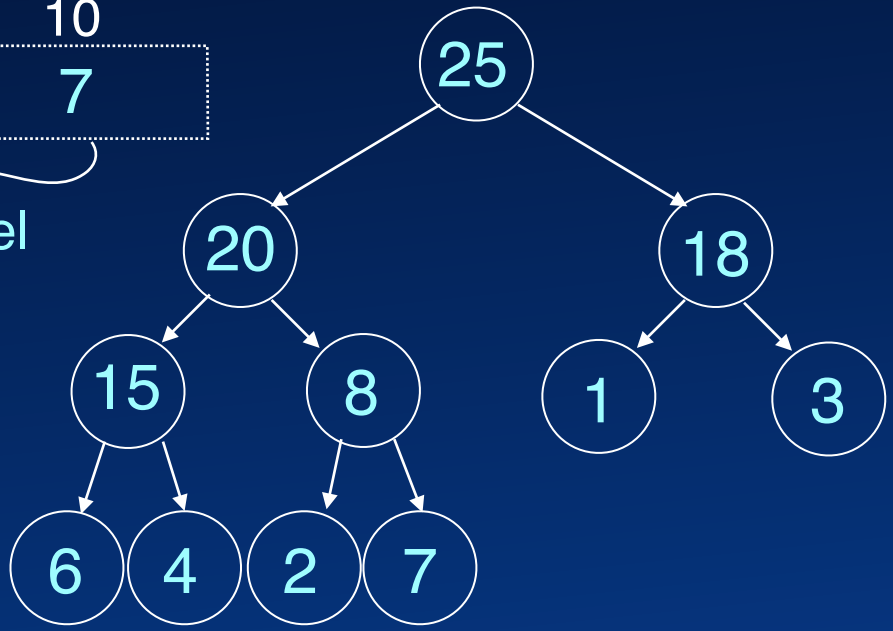




Heap

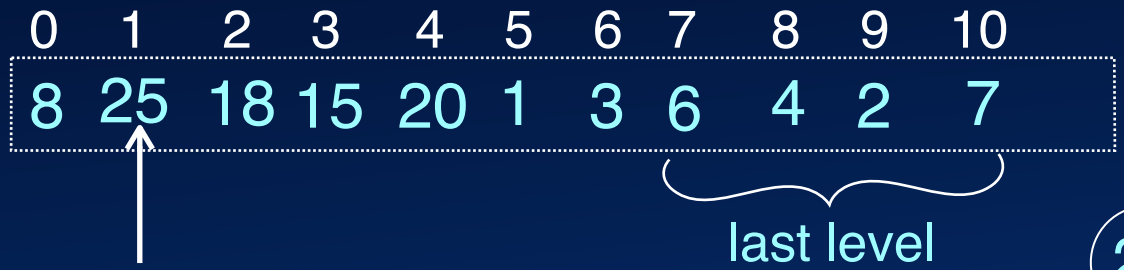
Heap Construction

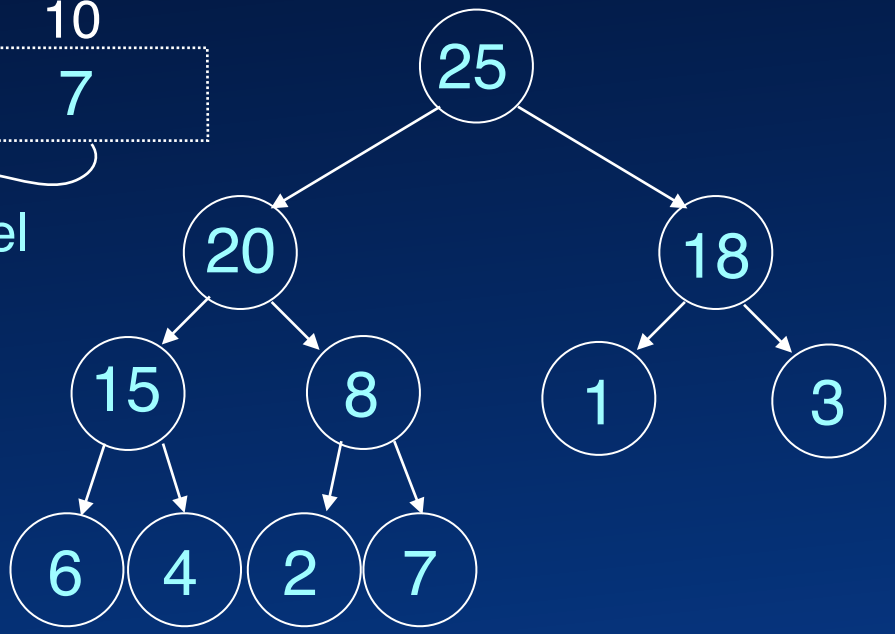




Heap

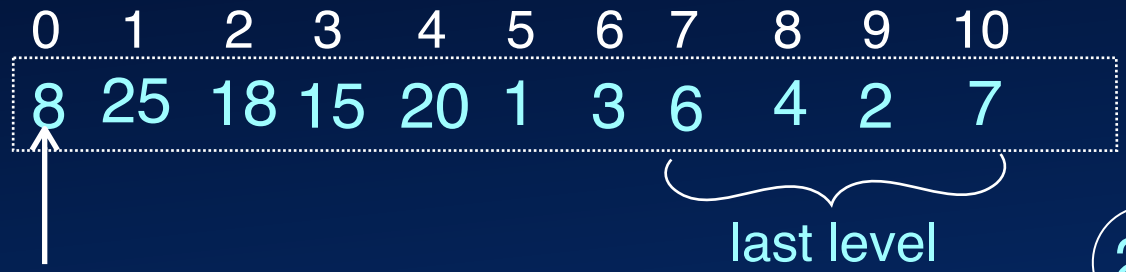
Heap Construction

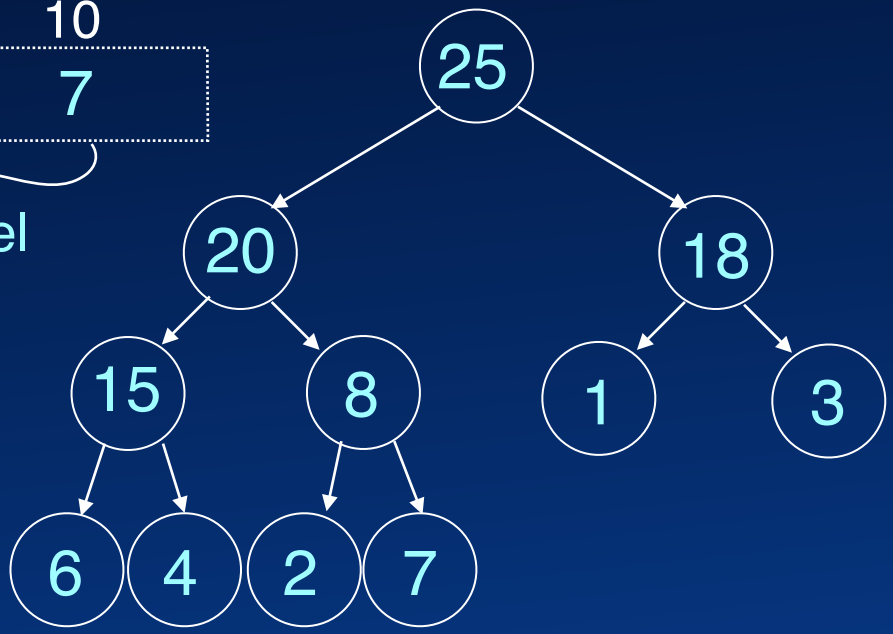




Heap

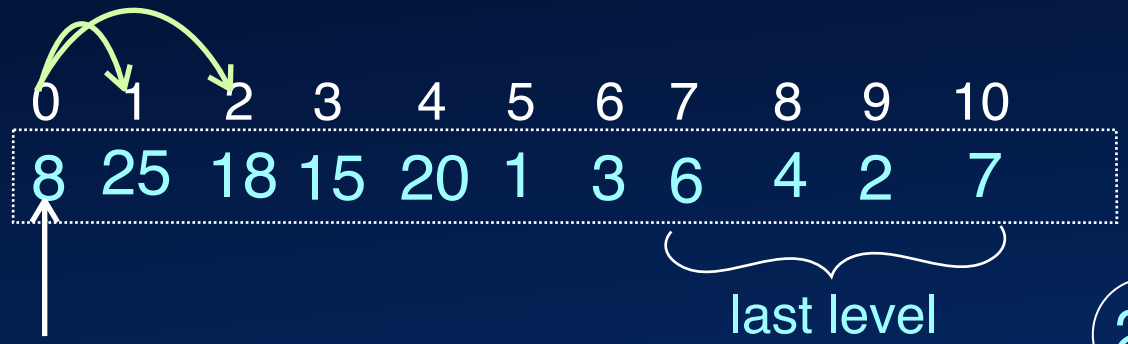
Heap Construction

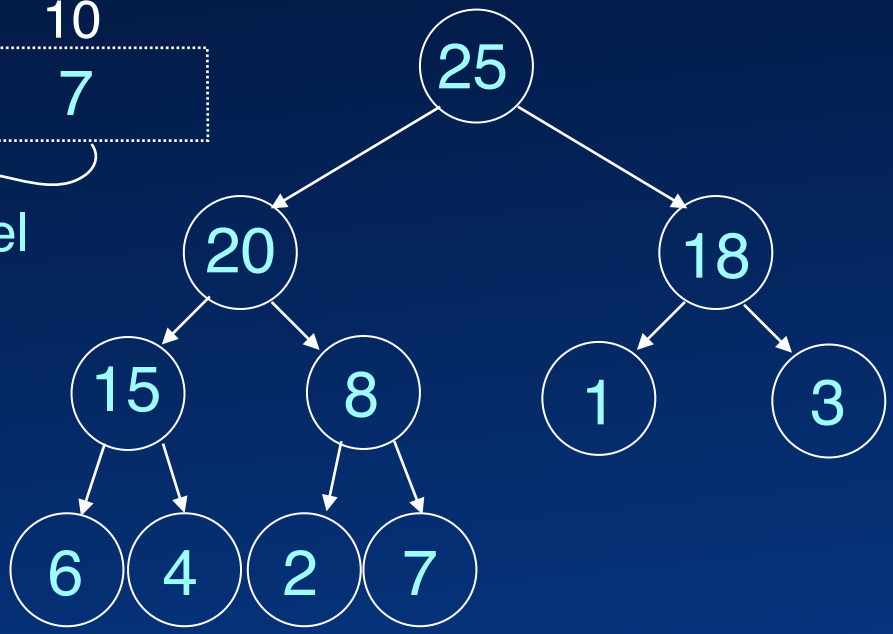




Heap

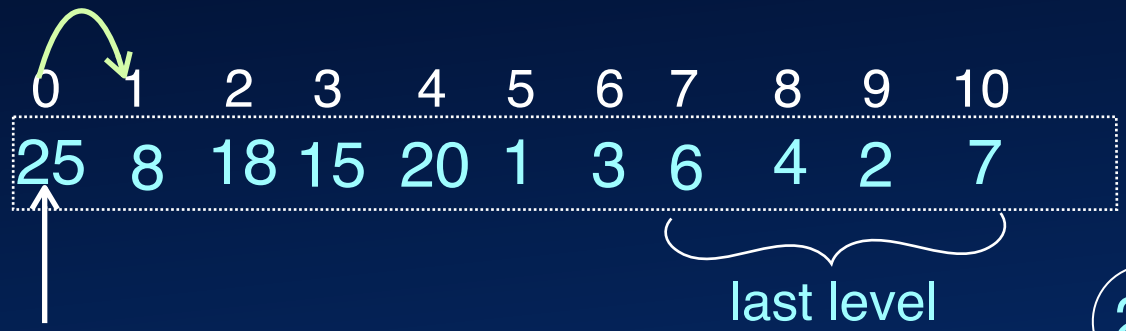
Heap Construction

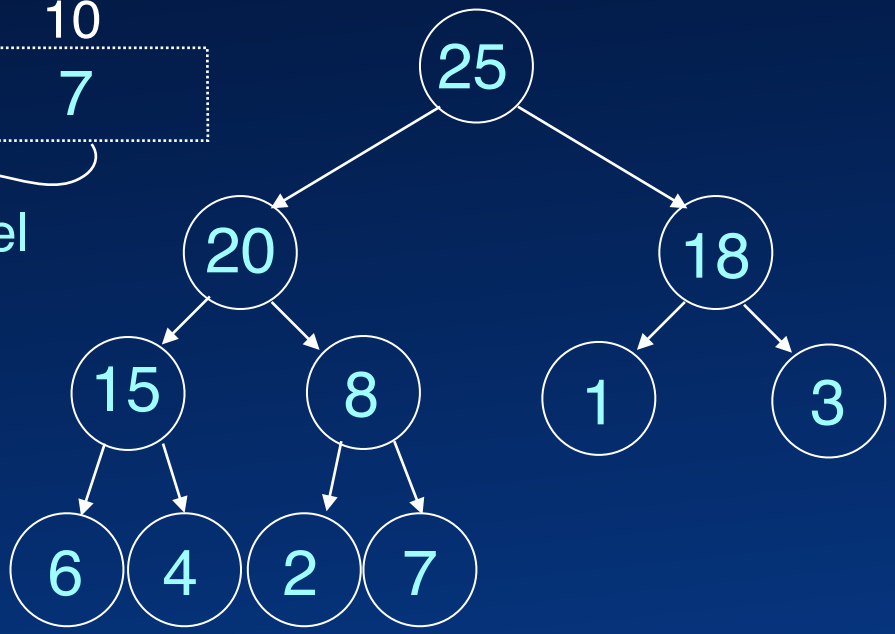




Heap

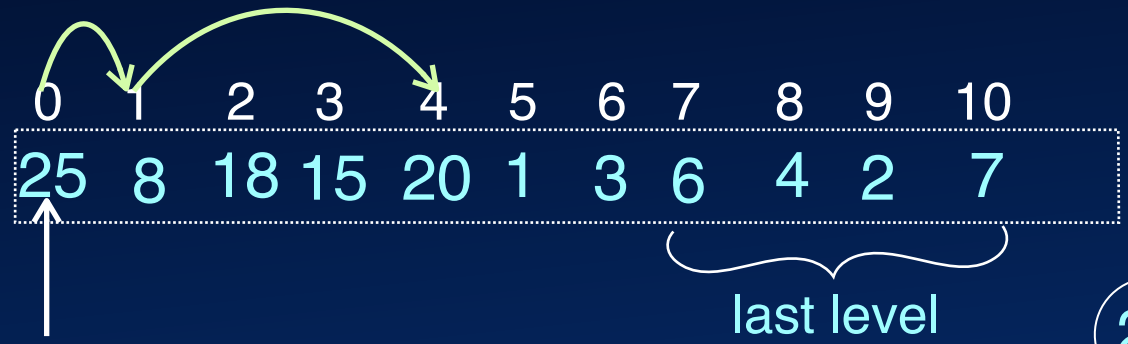
Heap Construction

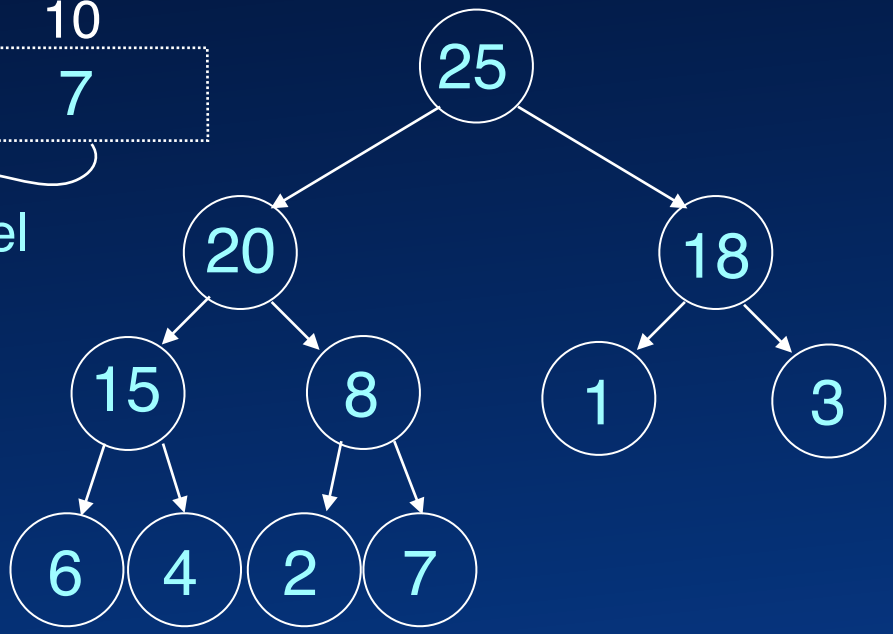




Heap

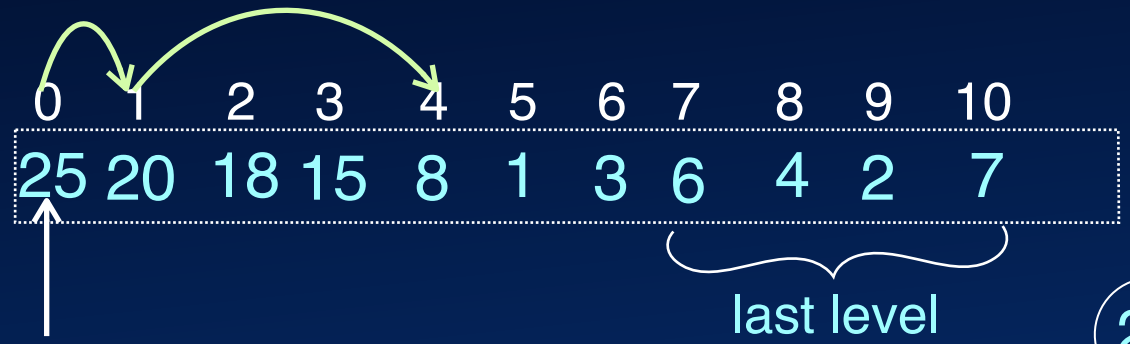
Heap Construction

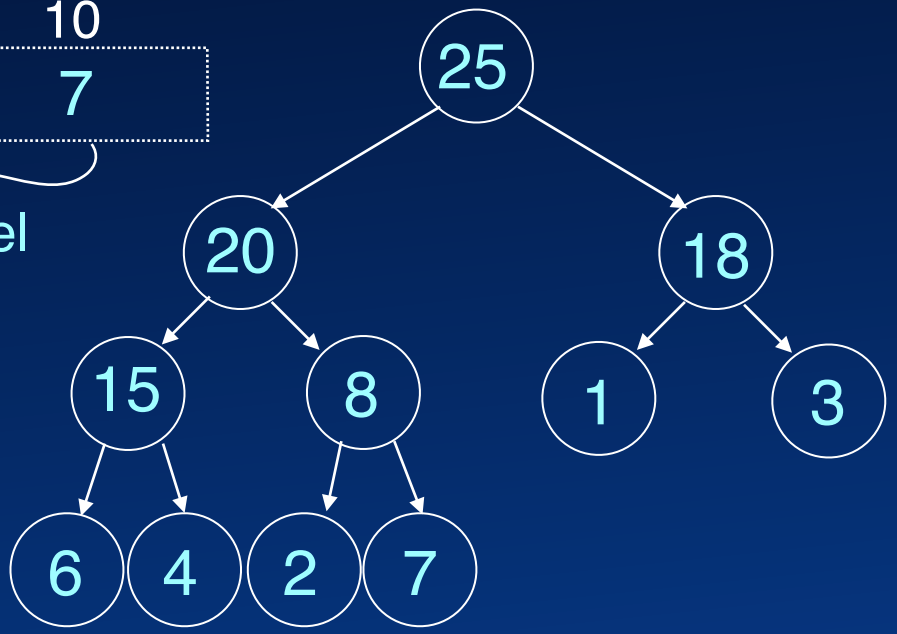




Heap

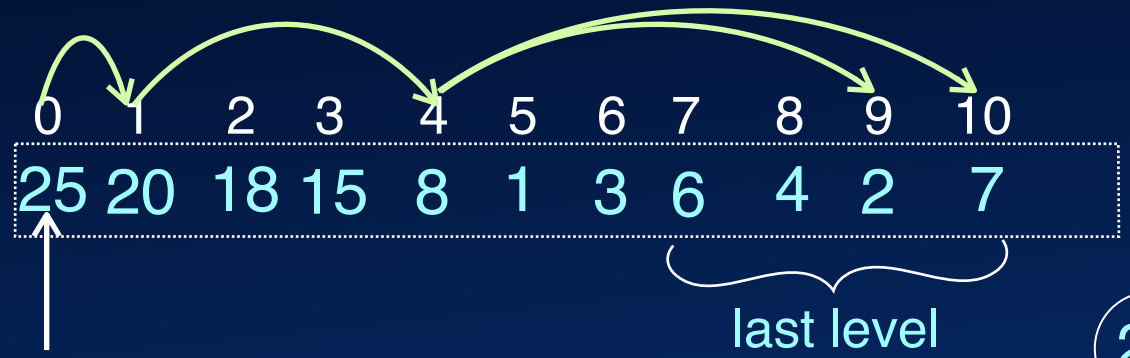
Heap Construction

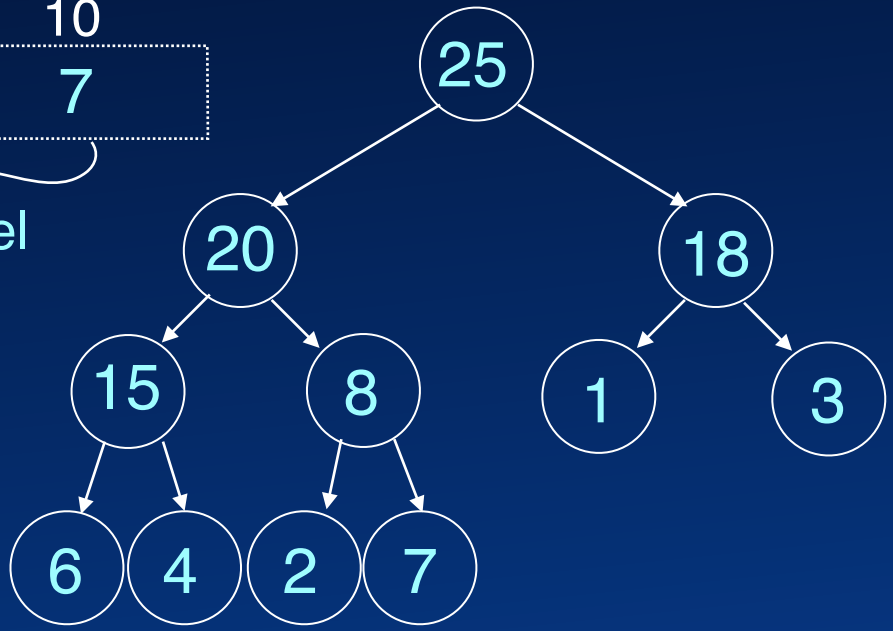




Heap

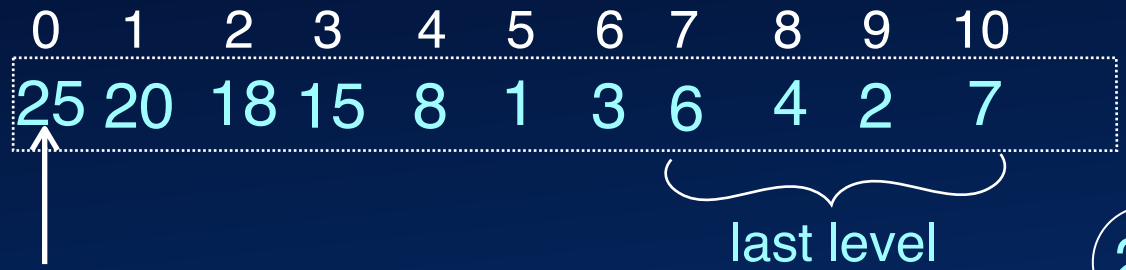
Heap Construction

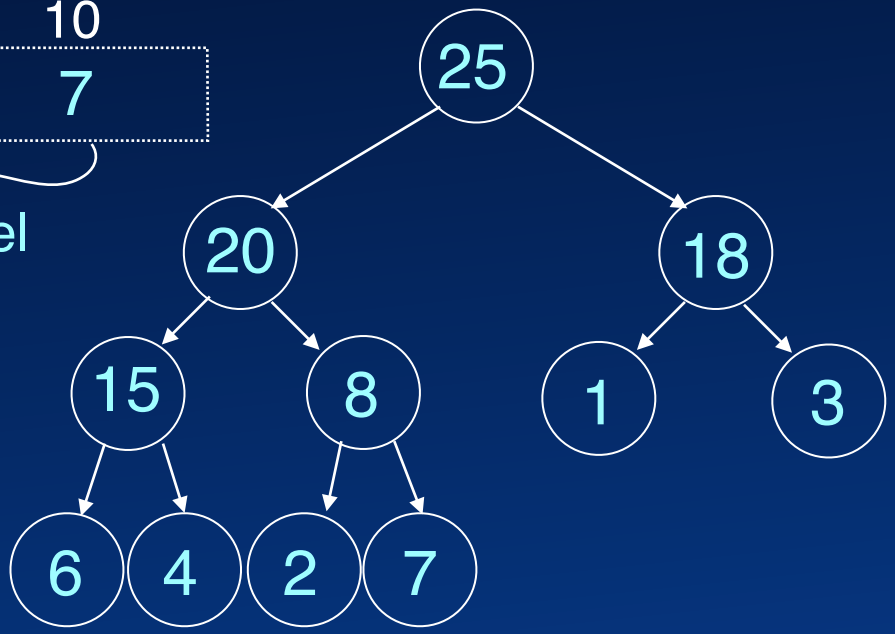




Heap

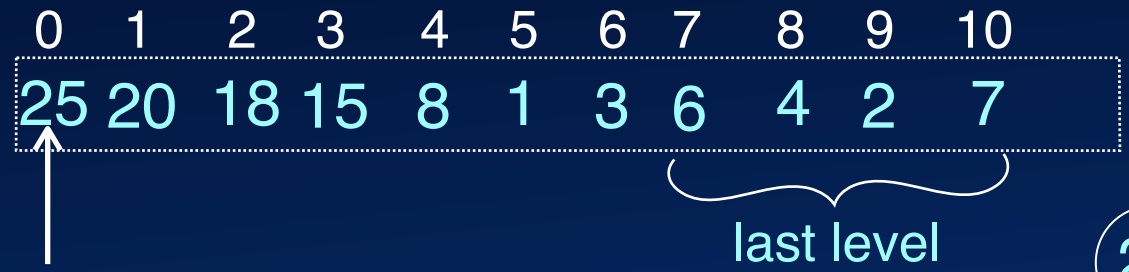
Heap Construction

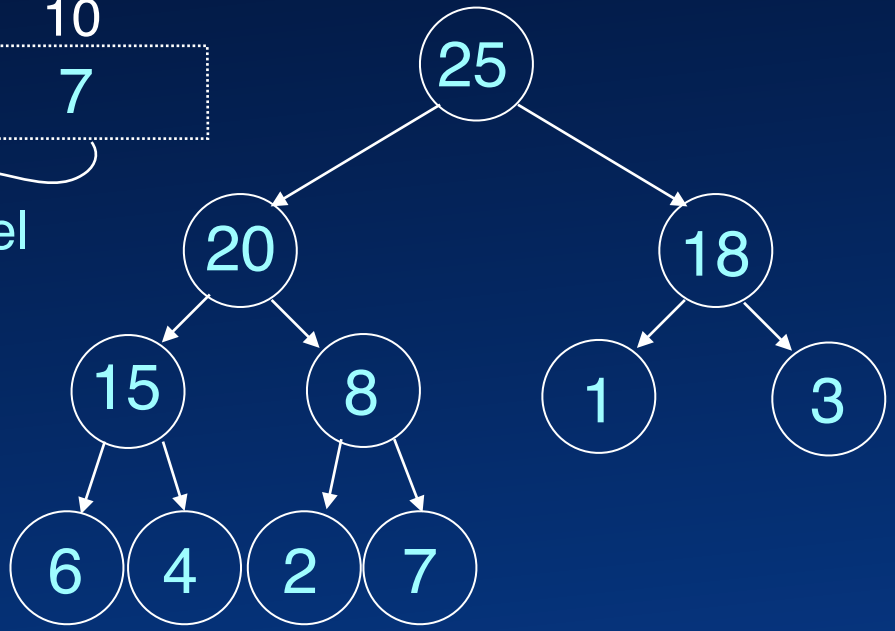




Heap

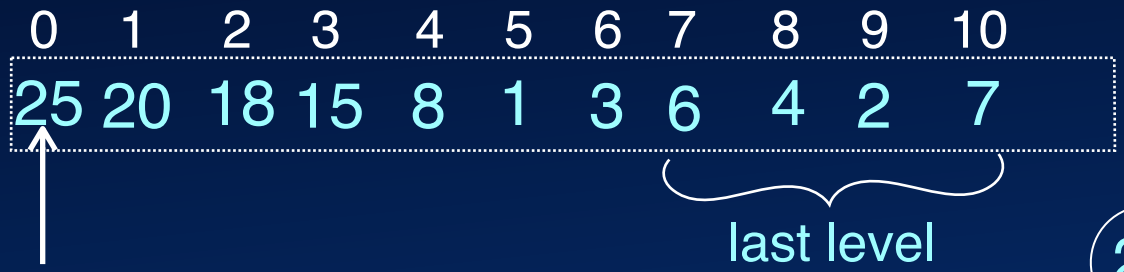
Heap Construction



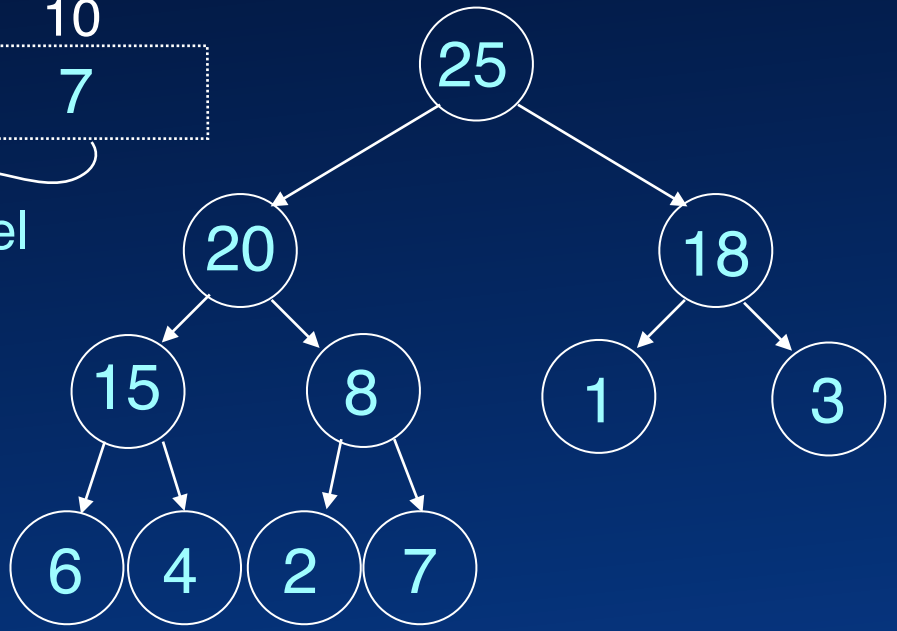


Heap

Heap Construction

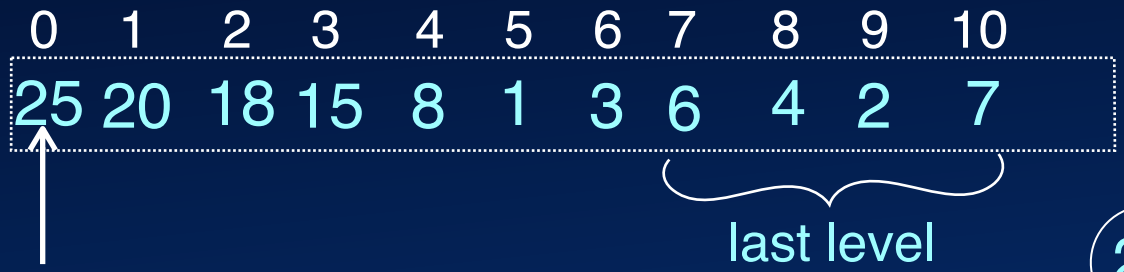


height $h = \lceil \lg n+1 \rceil - 1$

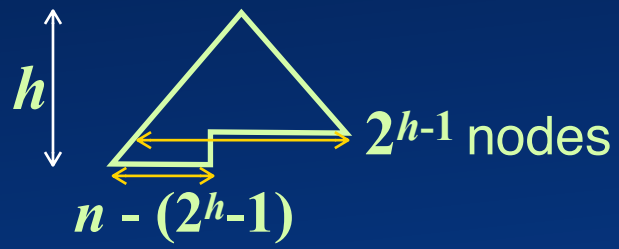


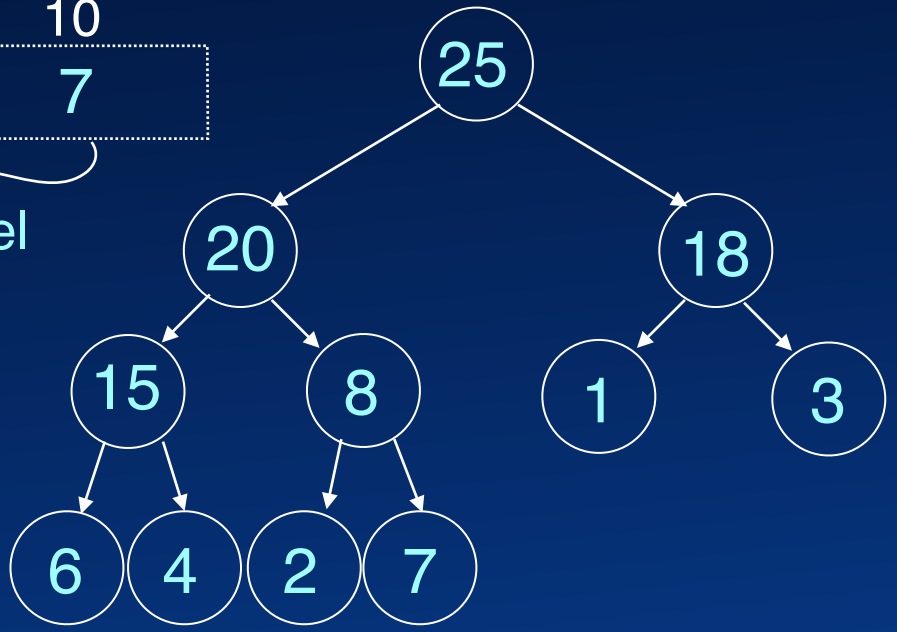
Heap

Heap Construction



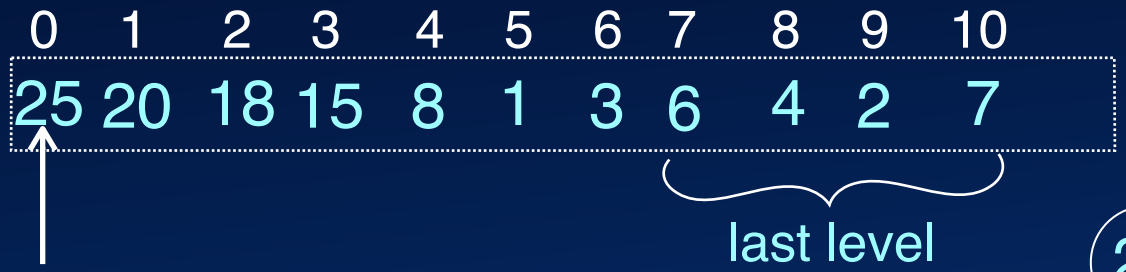
height $h = \lceil \lg n+1 \rceil - 1$



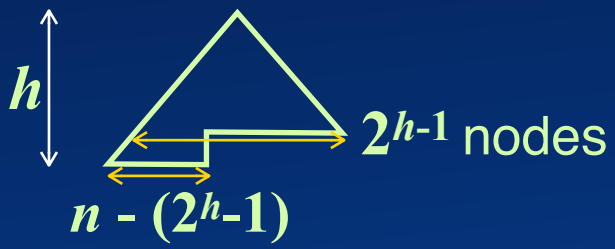


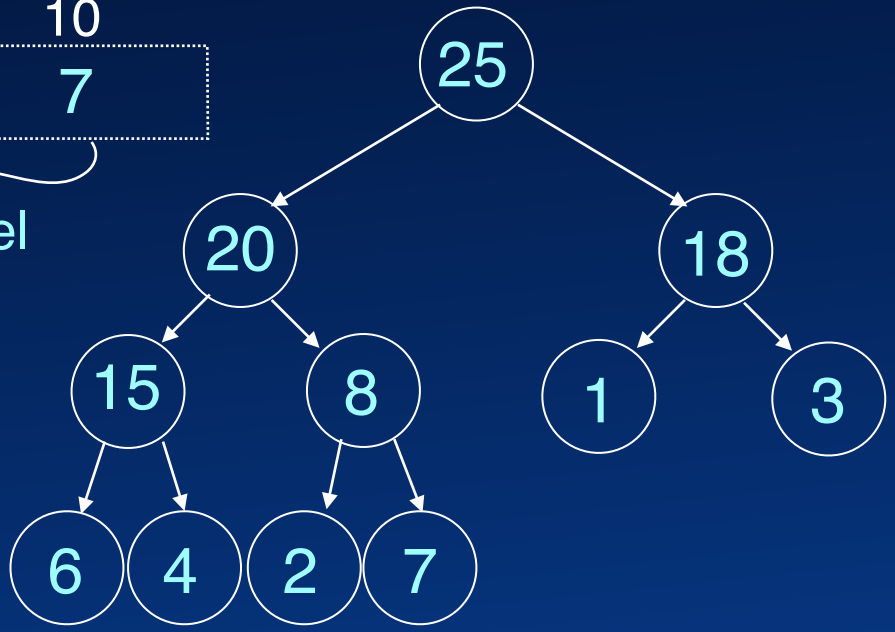
Heap

Heap Construction



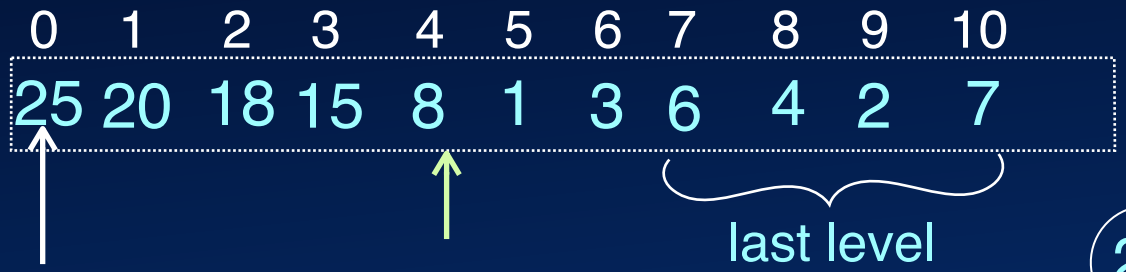
height $h = \lceil \lg n+1 \rceil - 1$



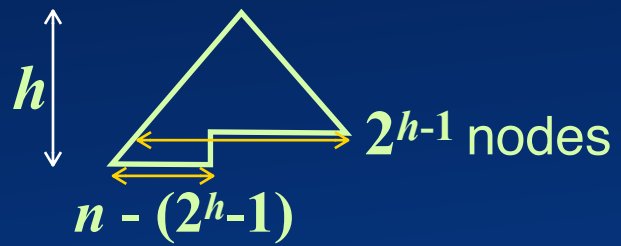


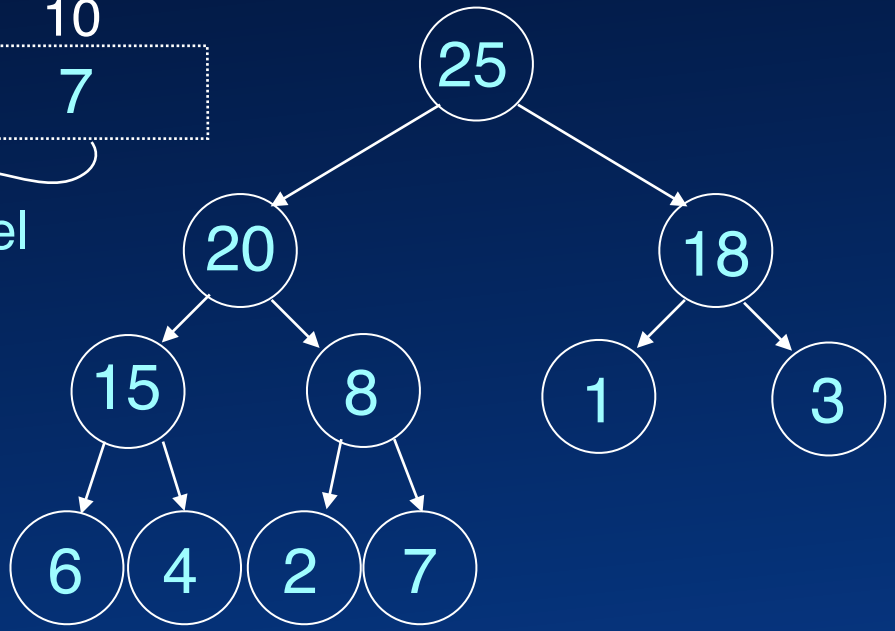
Heap

Heap Construction



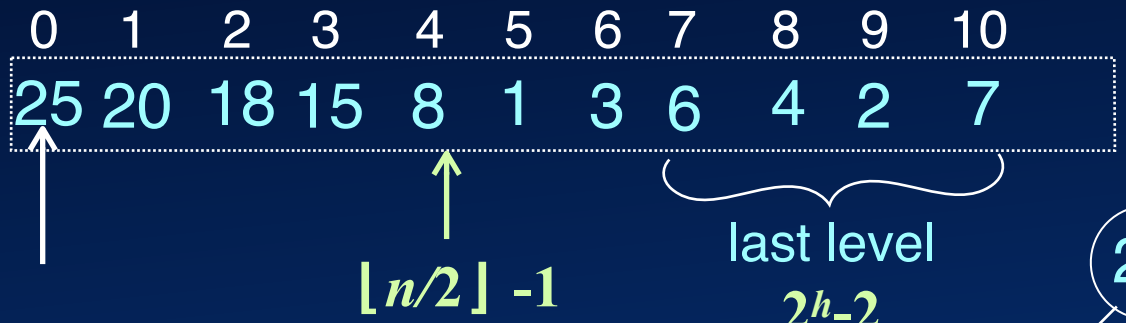
height $h = \lceil \lg n+1 \rceil - 1$



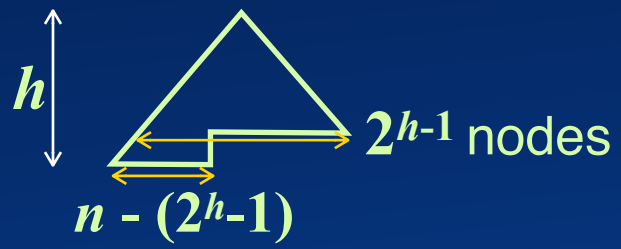


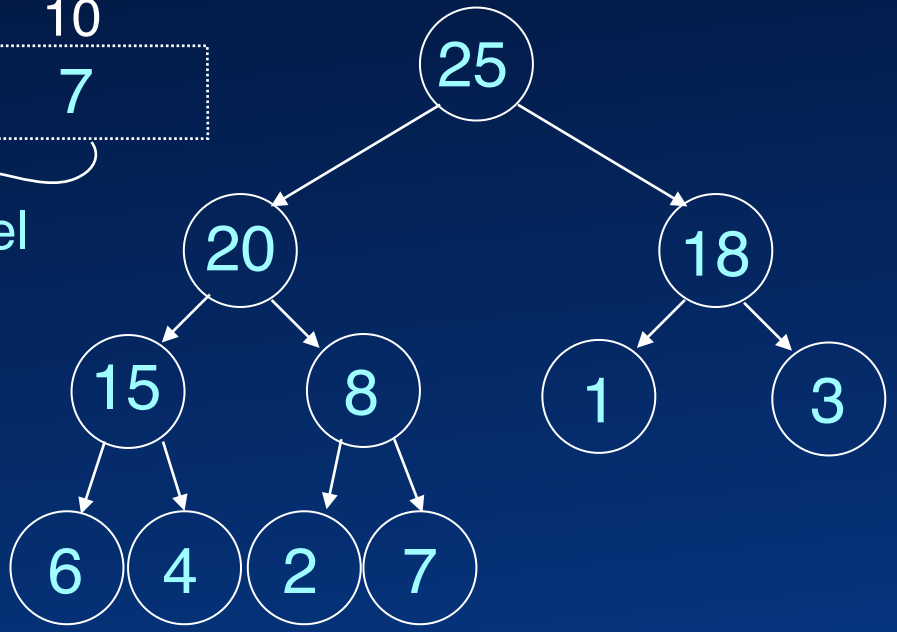
Heap

Heap Construction



height $h = \lceil \lg n+1 \rceil - 1$





Heap

Quiz

Time required to turn an array of n comparable objects into a heap is:

a. $\Theta(1)$

b. $\Theta(n)$

c. $O(\log n)$

d. $\Theta(n \log n)$

e. None of the above

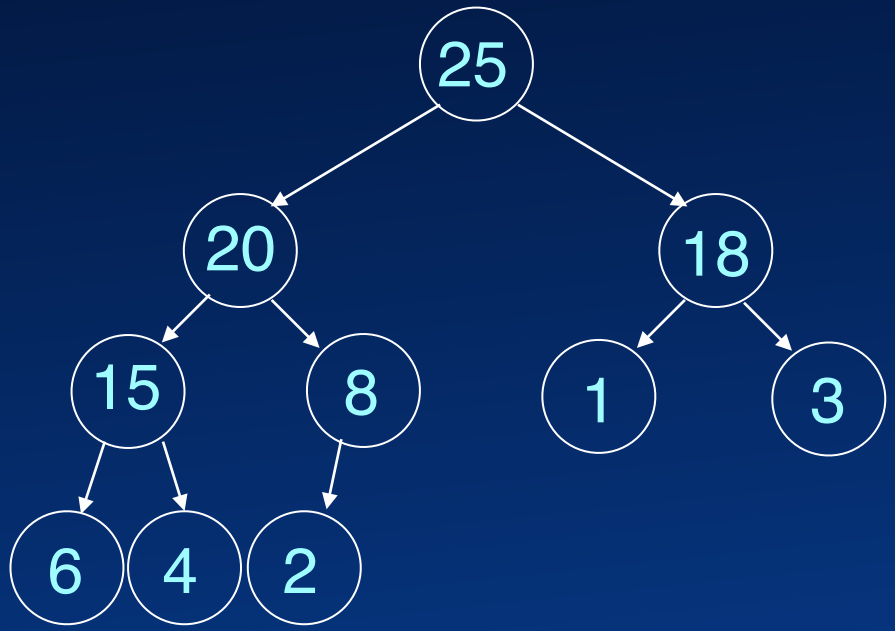
mailto: col106quiz@cse.iitd.ac.in
format: b

Priority Queue

- Heap
- Balanced tree
- Skip list

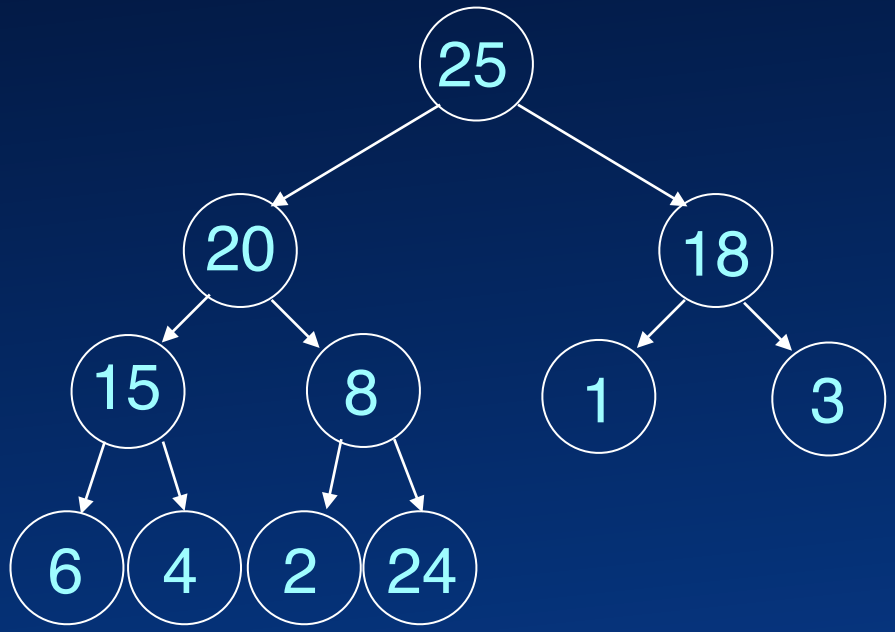
Priority Queue

- Heap
- Balanced tree
- Skip list



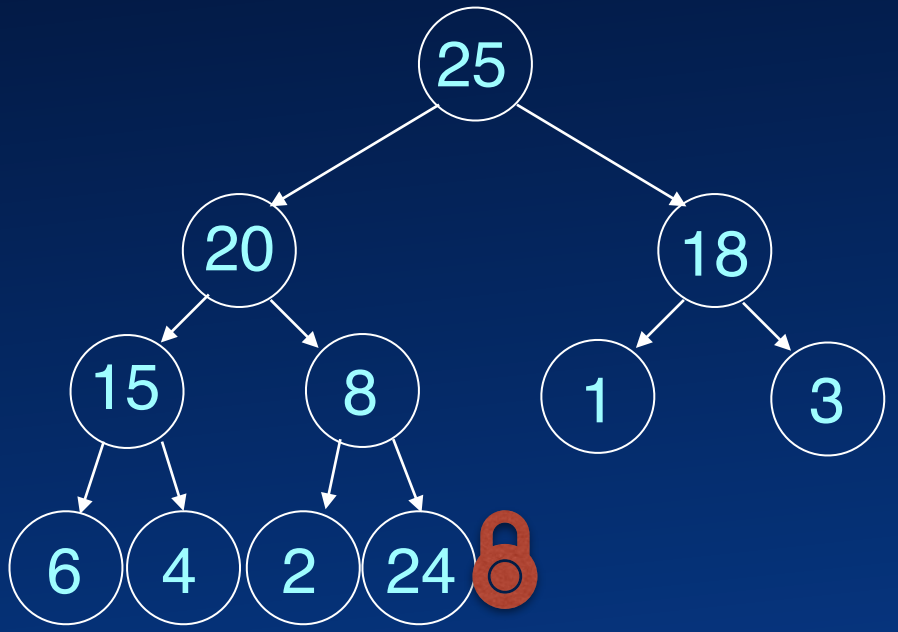
Priority Queue

- Heap
- Balanced tree
- Skip list



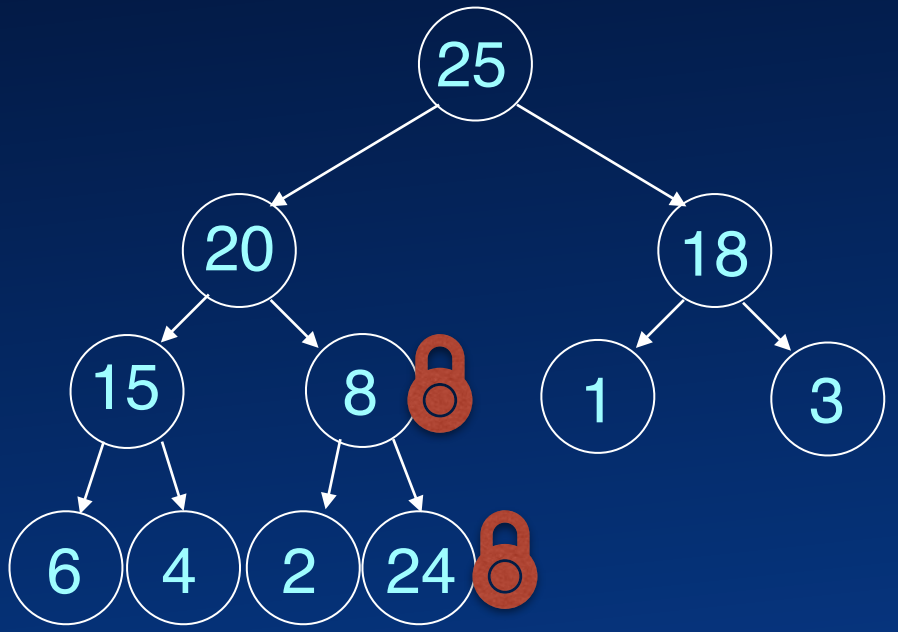
Priority Queue

- Heap
- Balanced tree
- Skip list



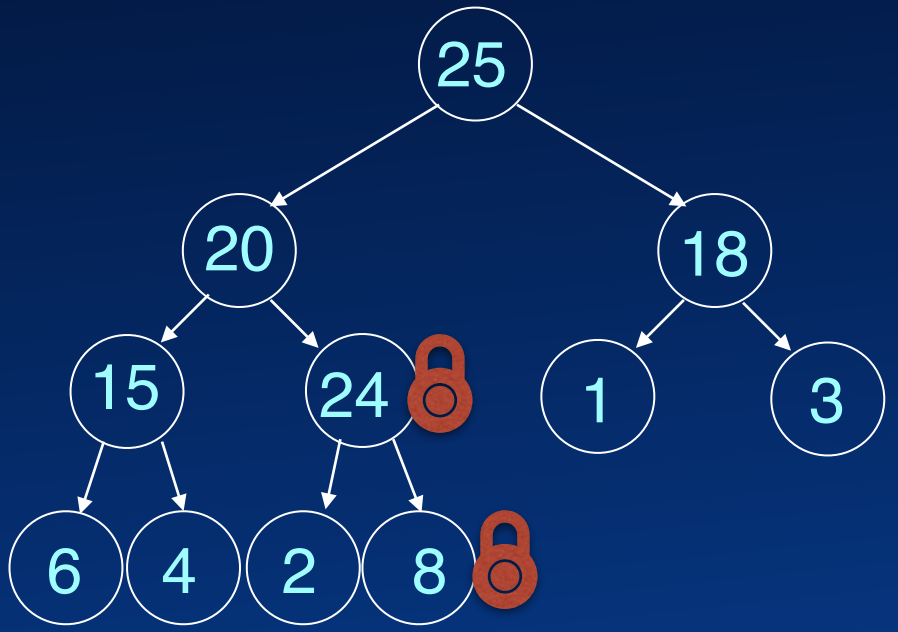
Priority Queue

- Heap
- Balanced tree
- Skip list



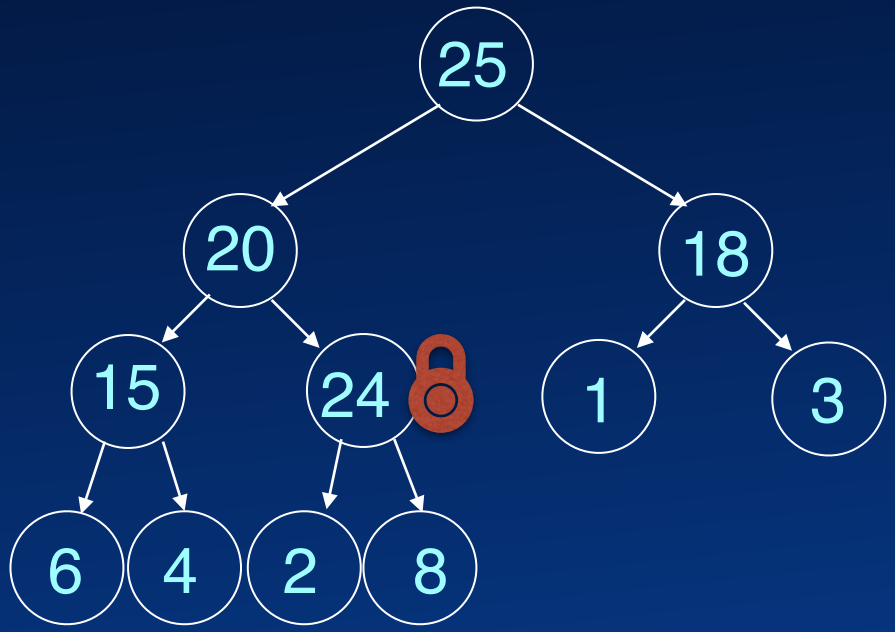
Priority Queue

- Heap
- Balanced tree
- Skip list



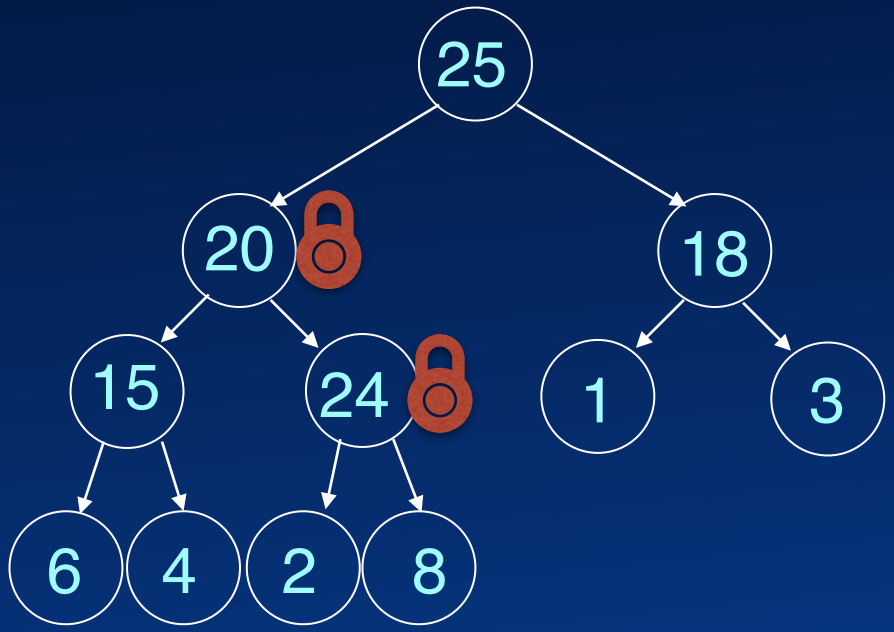
Priority Queue

- Heap
- Balanced tree
- Skip list



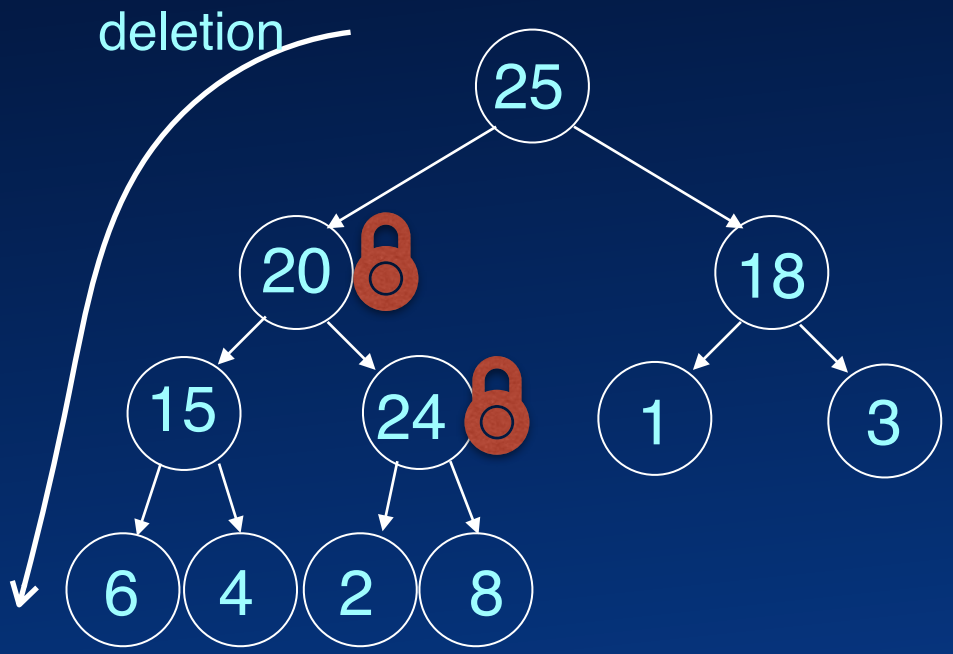
Priority Queue

- Heap
- Balanced tree
- Skip list



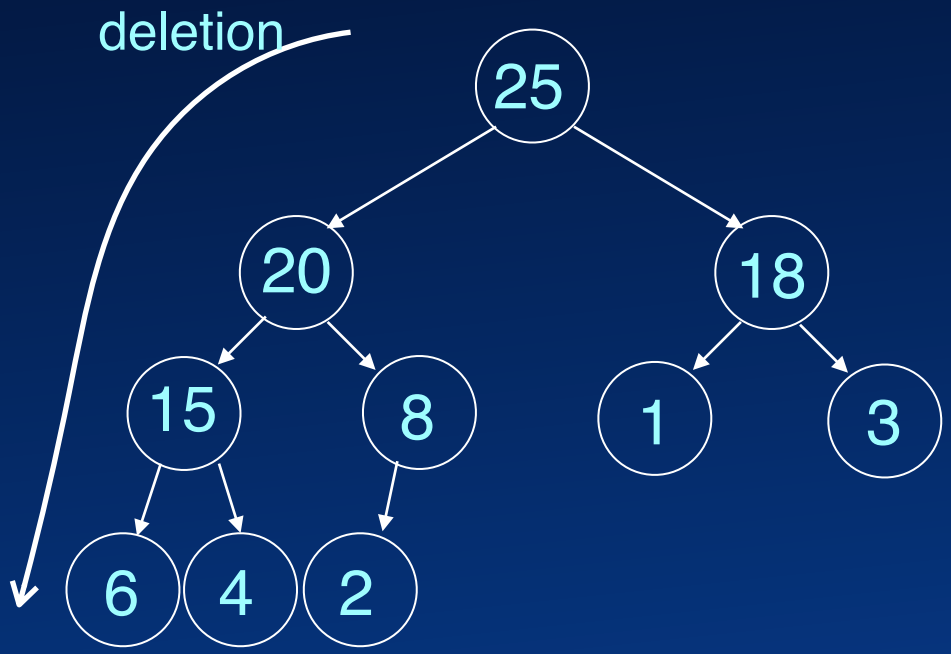
Priority Queue

- Heap
- Balanced tree
- Skip list



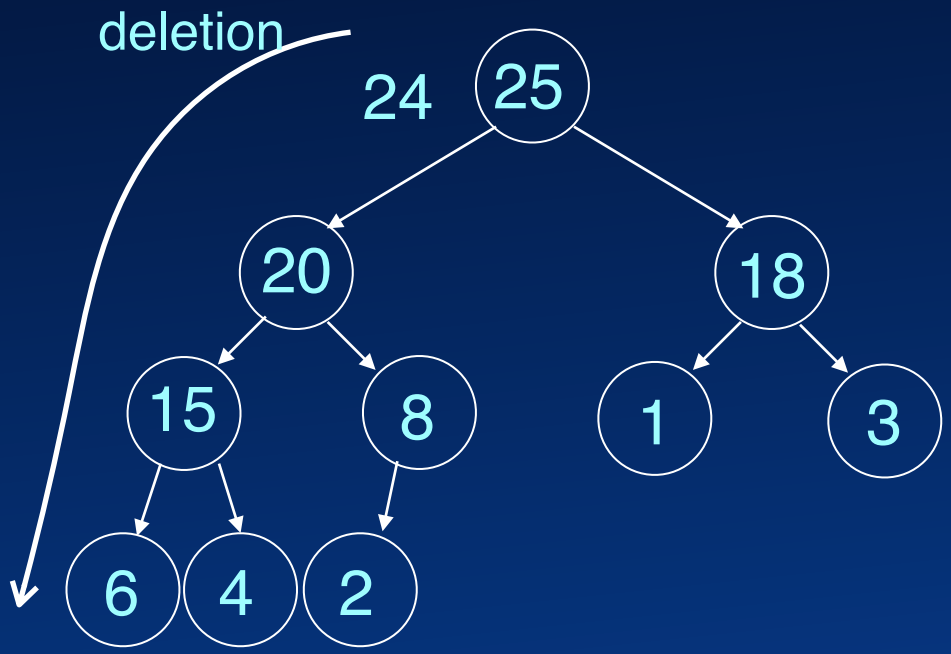
Priority Queue

- Heap
- Balanced tree
- Skip list



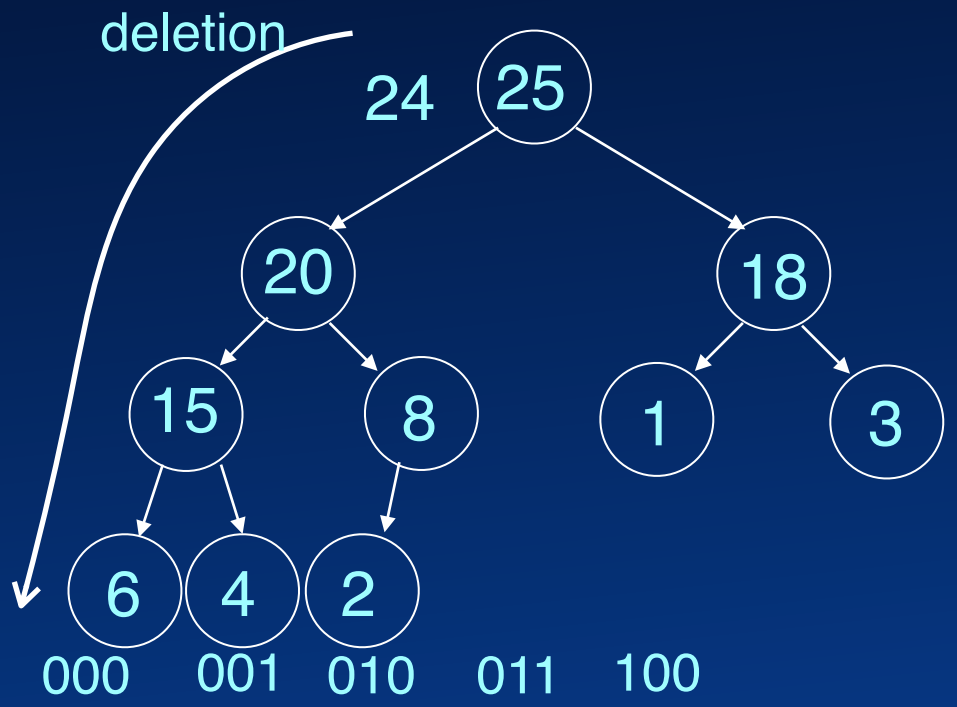
Priority Queue

- Heap
- Balanced tree
- Skip list



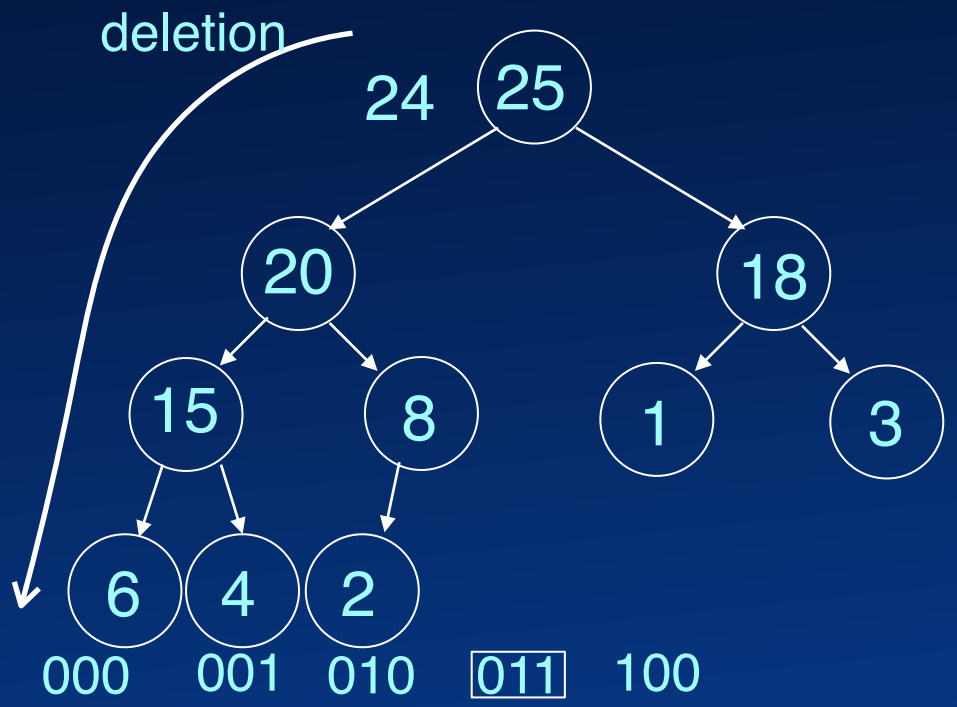
Priority Queue

- Heap
- Balanced tree
- Skip list



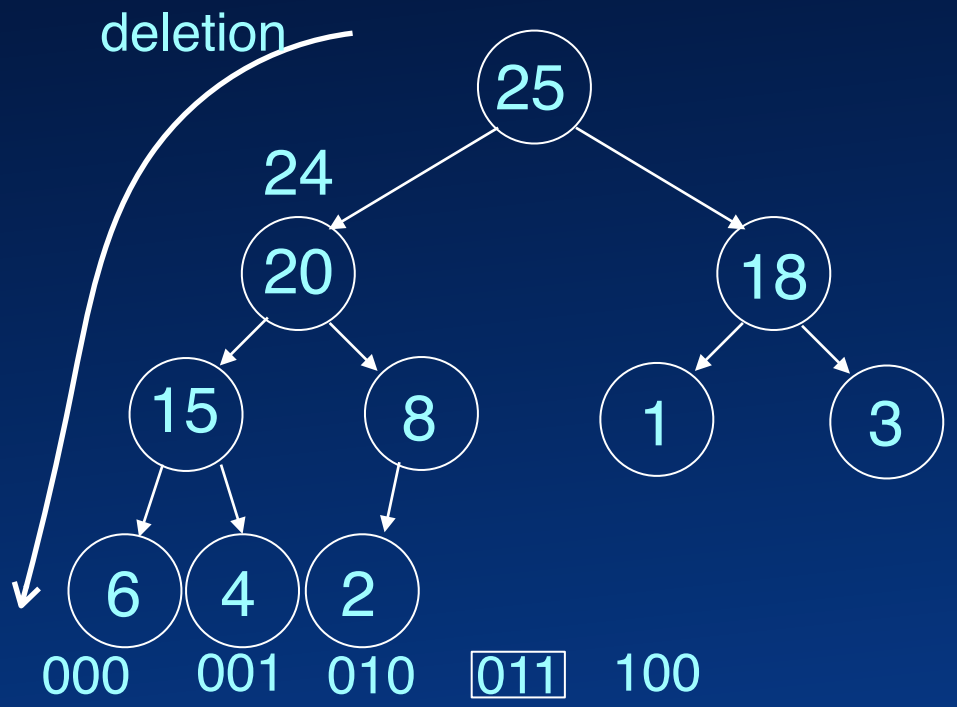
Priority Queue

- Heap
- Balanced tree
- Skip list



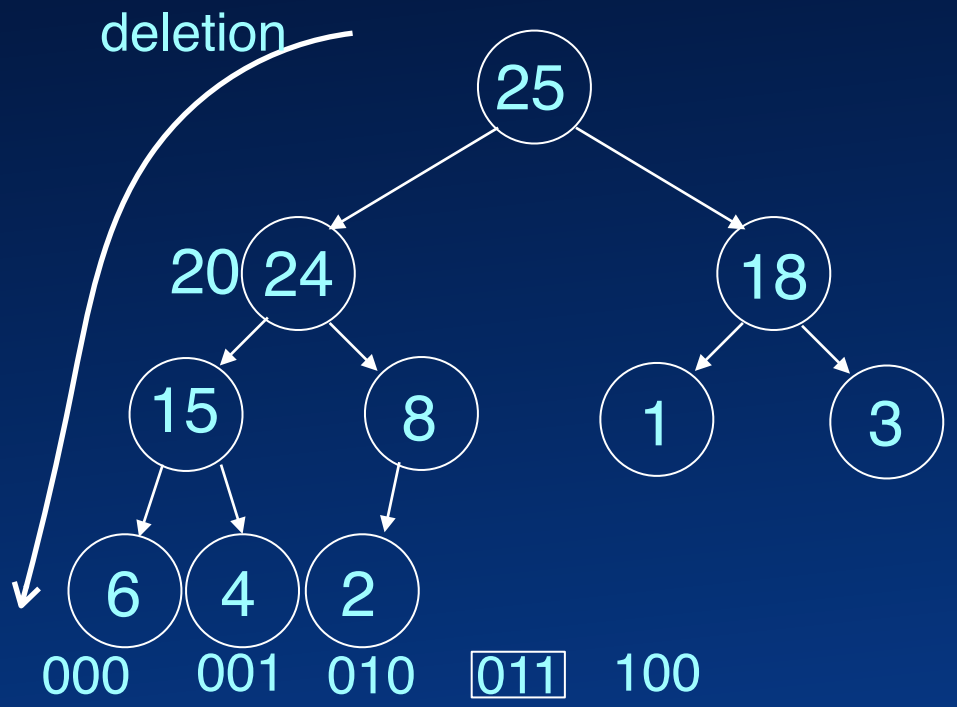
Priority Queue

- Heap
- Balanced tree
- Skip list



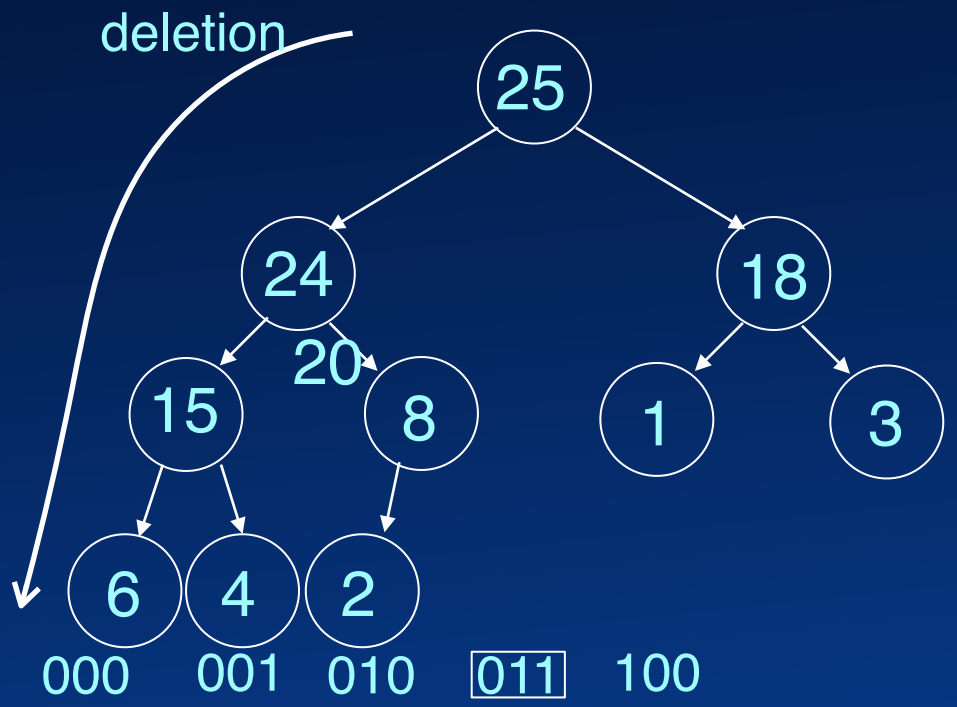
Priority Queue

- Heap
- Balanced tree
- Skip list



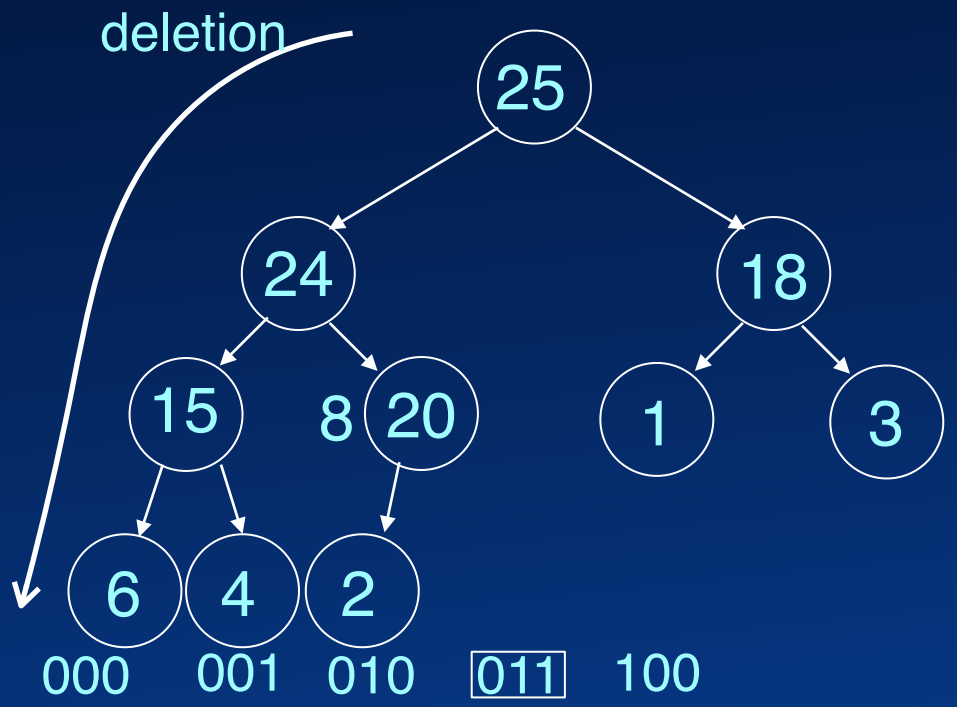
Priority Queue

- Heap
- Balanced tree
- Skip list



Priority Queue

- Heap
- Balanced tree
- Skip list



Priority Queue

- Heap
- Balanced tree
- Skip list