Data Structures in Python

- **Heap and Priority Queue**
- Heap Coding
- Heap Sort & Min/MaxHeap

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204, Handong Global University

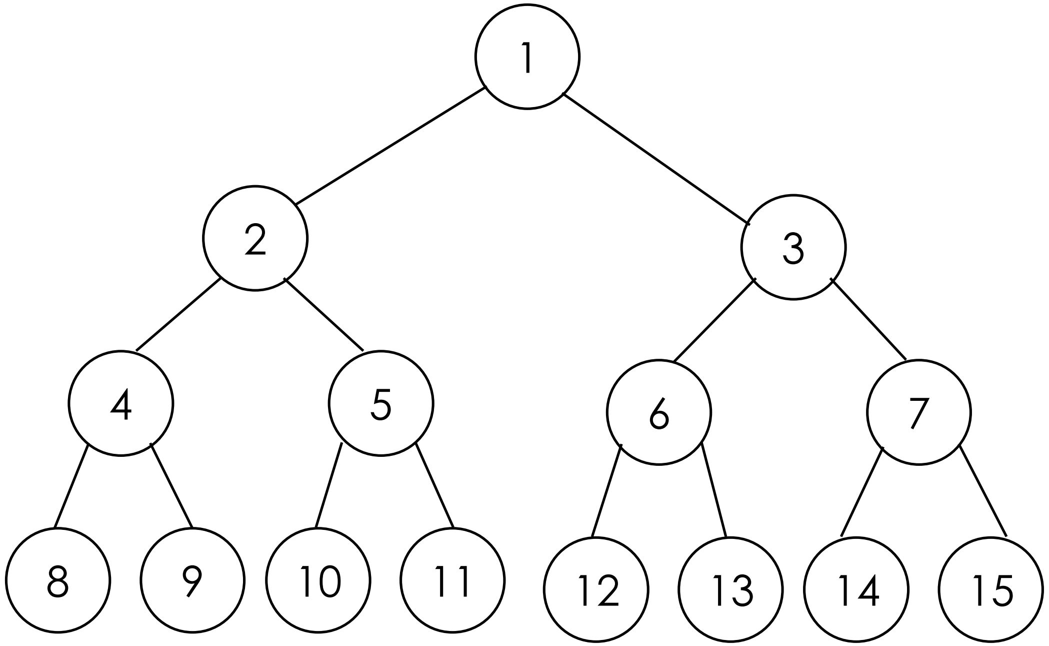
Agenda & Readings

- Heap and Priority Queue
  - Complete Binary Tee (Review)
  - Heap and Priority Queue
  - Heap ADT
  - Time Complexity

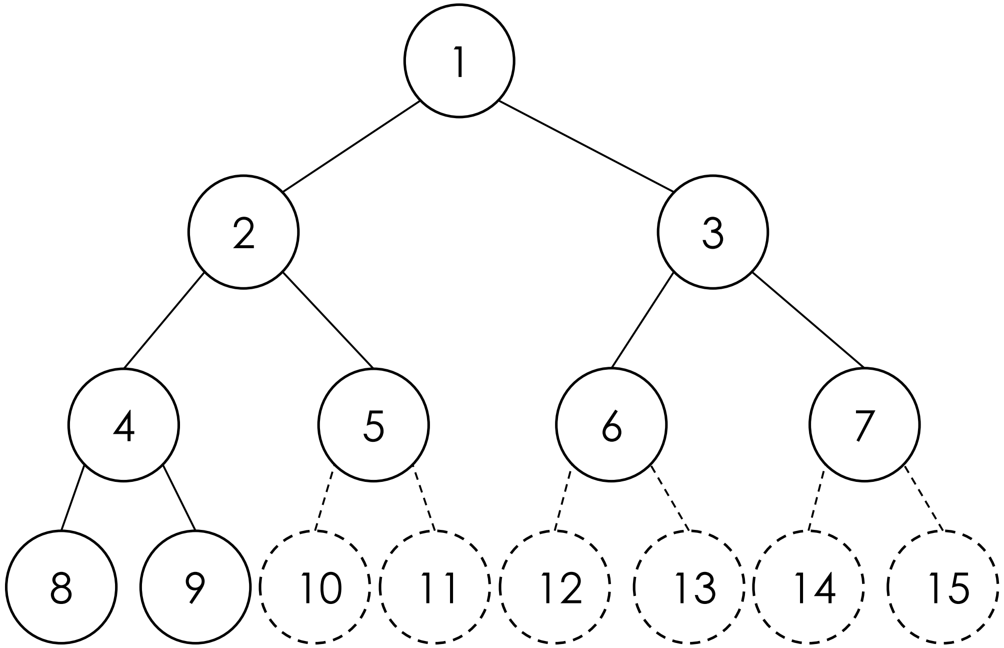
- Reference:
  - Problem Solving with Algorithms and Data Structures

Binary trees - Properties

- **Definition:** *A* **full** binary tree of level k is a binary tree having **2^k - 1 nodes**, $k \geq 0$.
- **Definition:** A binary tree with n nodes and level k is **complete** if and only if its nodes correspond to the nodes numbered from 1 to n in the full binary tree of level k .



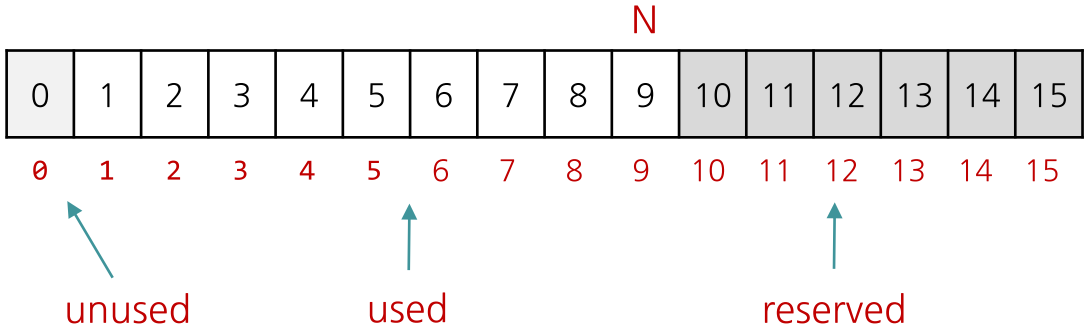
*A **full** binary tree*

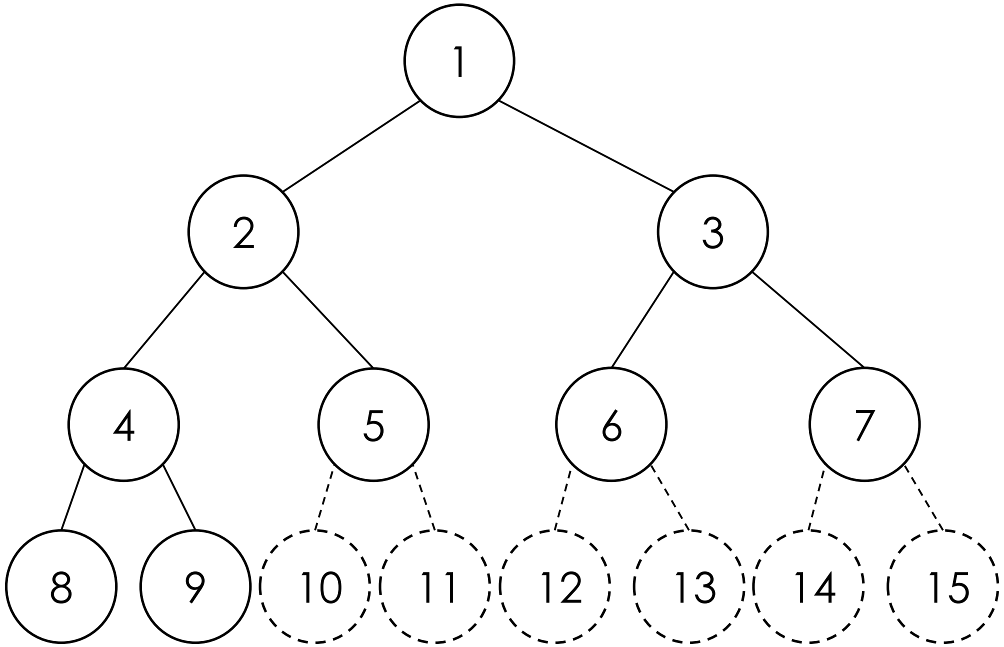


*A **complete** binary tree*

Binary trees - Array representation

- A **complete** binary tree with n nodes, any node index i, $1 \leq i \leq n$, we have
  - `parent(i) is at` $\lfloor i/2 \rfloor$ `If i = 1, i is at the root and has no parent`
  - `leftChild(i) is at 2i  if 2i <= n. If 2i > n, then i has no left child.`
  - `rightChild(i) is at 2i+1 if 2i+1 <= n. If 2i+1 > n, then i has no right child.`





*A **complete** binary tree*

Heaps & Priority Queues

- **Heaps** are frequently used to implement **priority queues.**
  - Because it provides an efficient implementation for **priority queues.**

Heaps & Priority Queues

- **Heaps** are frequently used to implement **priority queues.**
  - Because it provides an efficient implementation for **priority queues.**

- **Priority queues.**
  - Queues with priorities associated to.
  - **Example:** A line waiting to be served at a bank and served FIFO except if a senior or a disabled person arrives in the line. They are served first. Seniors and disabled persons have higher priority than others.

Heaps & Priority Queues

- **Heaps** are frequently used to implement **priority queues.**
  - Because it provides an efficient implementation for **priority queues.**

- **Priority queues.**
  - Queues with priorities associated to.
  - **Example:** A line waiting to be served at a bank and served FIFO except if a senior or a disabled person arrives in the line. They are served first. Seniors and disabled persons have higher priority than others.

- **A typical ADT for Priority Queue**
  - Get the top priority element (min or max)
  - Insert an element
  - Delete the top priority element
  - Decrease the priority of an element

- O(1)
- O(log n)
- O(log n)
- O(log n)

Heaps & Priority Queues

- Challenge: Find the largest M items in a stream of N items.
- Constraints: Not enough memory to store N items.

N huge,
M large

Order of insert of finding the largest M **in a stream of N items**

implementation	insert	delete	min/max
unordered array	1	N	N
ordered array	N	1	1
goal	**log N**	**log N**	log N

N huge

Mission Impossible?

priority queue

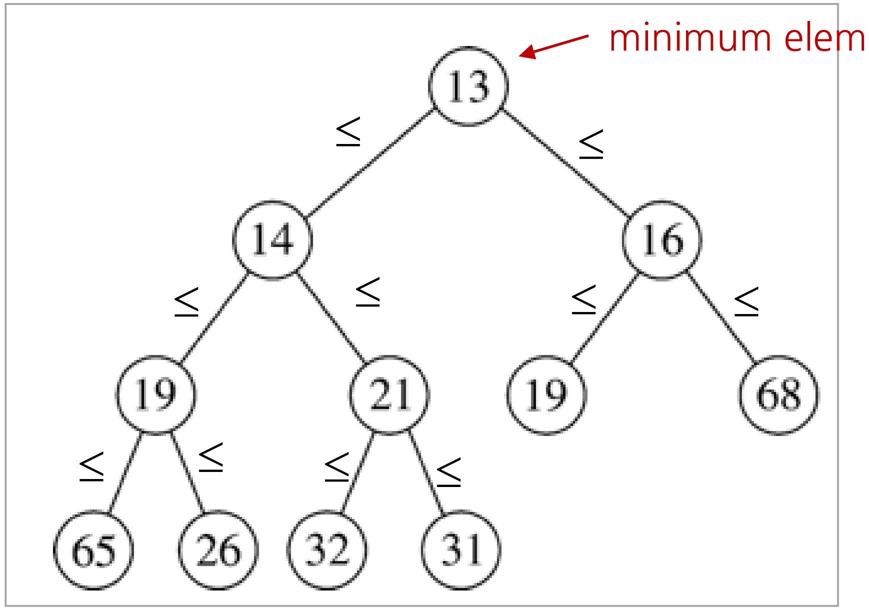
Binary heap

- **Binary heap**: array representation of a **heap-ordered** complete binary tree
- **Properties:**
  - **Heap-ordered:**
    Parent's key no smaller than children's keys. [max-heap]
  - **Heap-structure**:
    A complete binary tree

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204, Handong Global University

Binary heap

- **Binary heap**: array representation of a **heap-ordered** complete binary tree
- **Properties:**
  - **Heap-ordered:** Parent's key no smaller than children's keys. [max-heap]
  - **Heap-structure**: A complete binary tree


minimum element

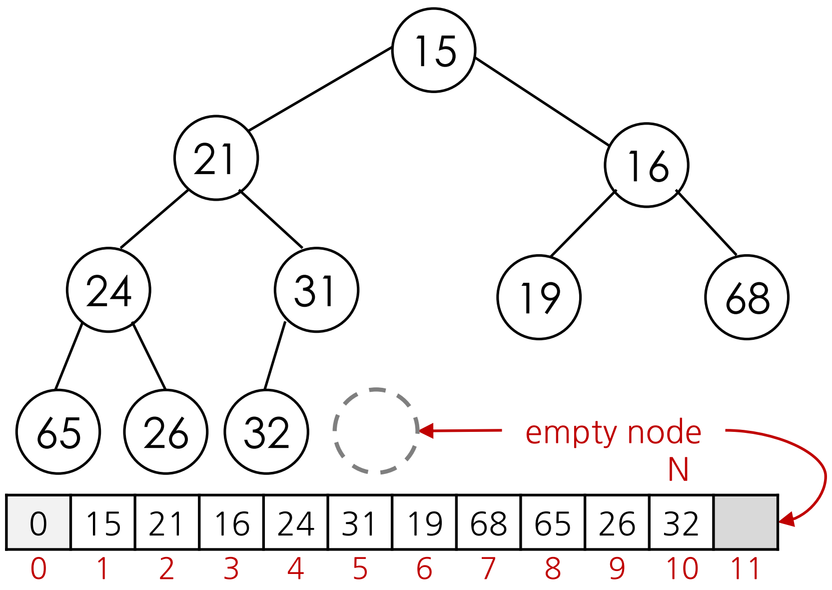
- Duplicates are allowed
- No order implied for elements which do not share ancestor-descendant relationship

min-heap: insert(heap, 14)

Algorithm:

- Insert a new element **while maintaining a heap-structure**
- Move the element up the heap **while not satisfying heap-ordered**

initial heap

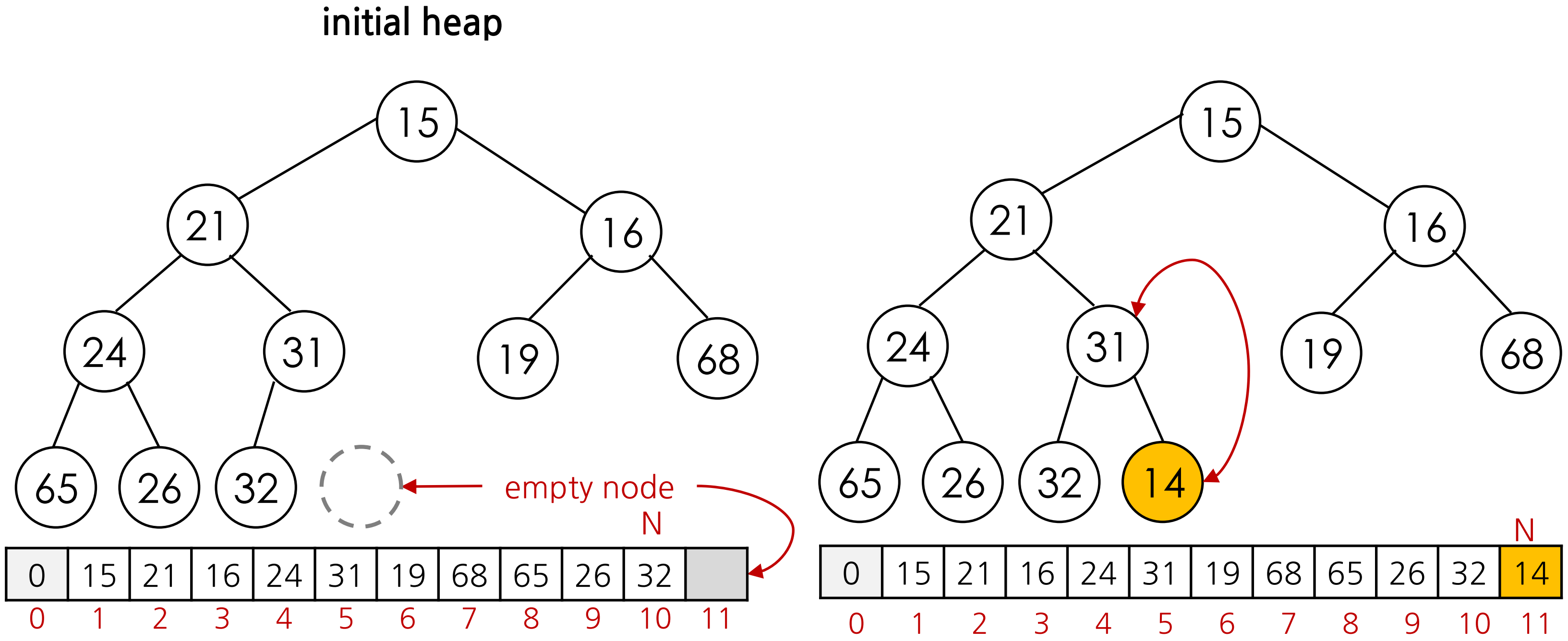


0	15	21	16	24	31	19	68	65	26	32	
0	1	2	3	4	5	6	7	8	9	10	11

min-heap: insert(heap, 14)

Algorithm:

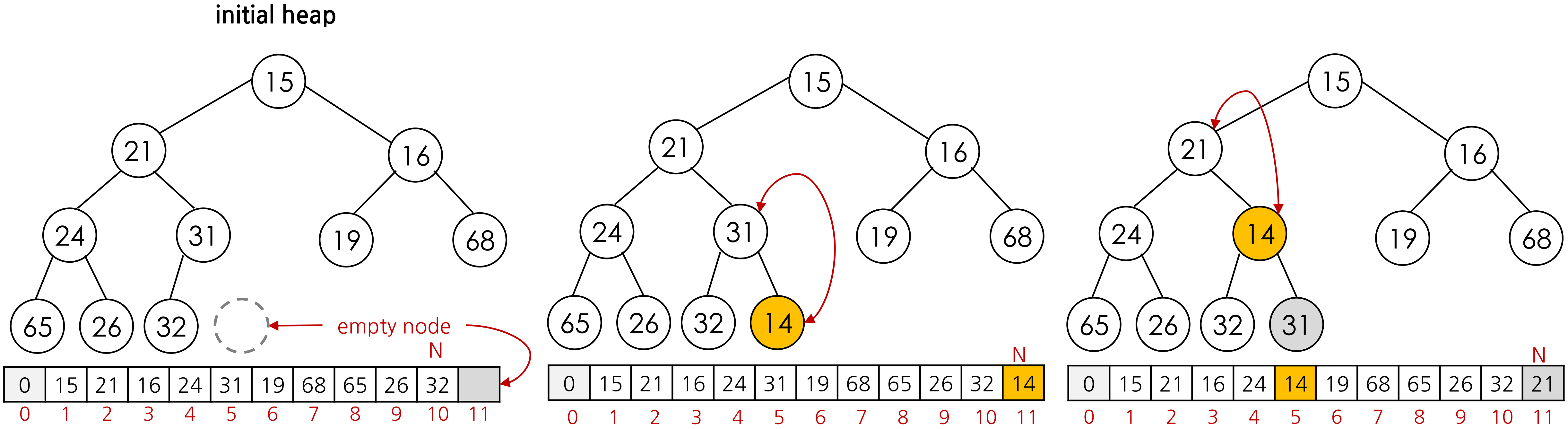
- Insert a new element **while maintaining a heap-structure**
- Move the element up the heap **while not satisfying heap-ordered**



min-heap: insert(heap, 14)

Algorithm:

- Insert a new element **while maintaining a heap-structure**
- Move the element up the heap **while not satisfying heap-ordered**



0	15	21	16	24	31	19	68	65	26	32	
0	1	2	3	4	5	6	7	8	9	10	11

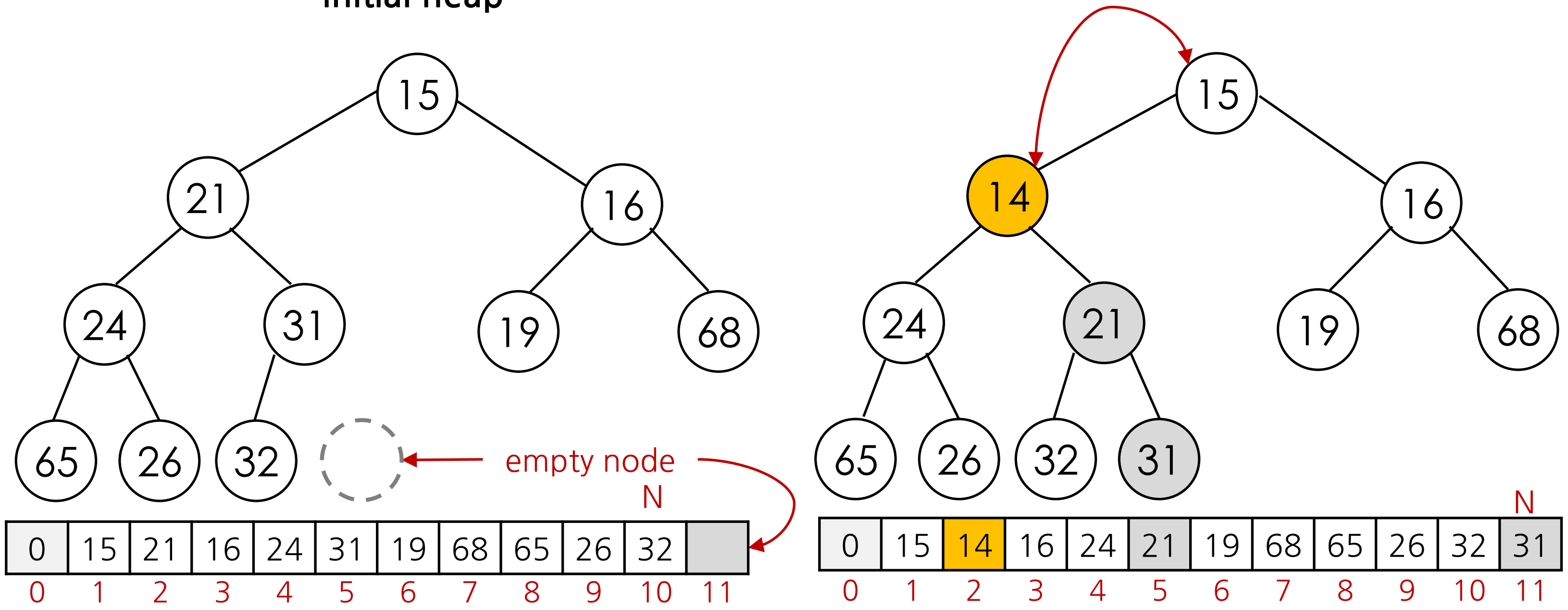
0	15	21	16	24	31	19	68	65	26	32	14
0	1	2	3	4	5	6	7	8	9	10	11

0	15	21	16	24	14	19	68	65	26	32	21
0	1	2	3	4	5	6	7	8	9	10	11

min-heap: insert(heap, 14)

Algorithm:

- Insert a new element **while maintaining a heap-structure**
- Move the element up the heap **while not satisfying heap-ordered**



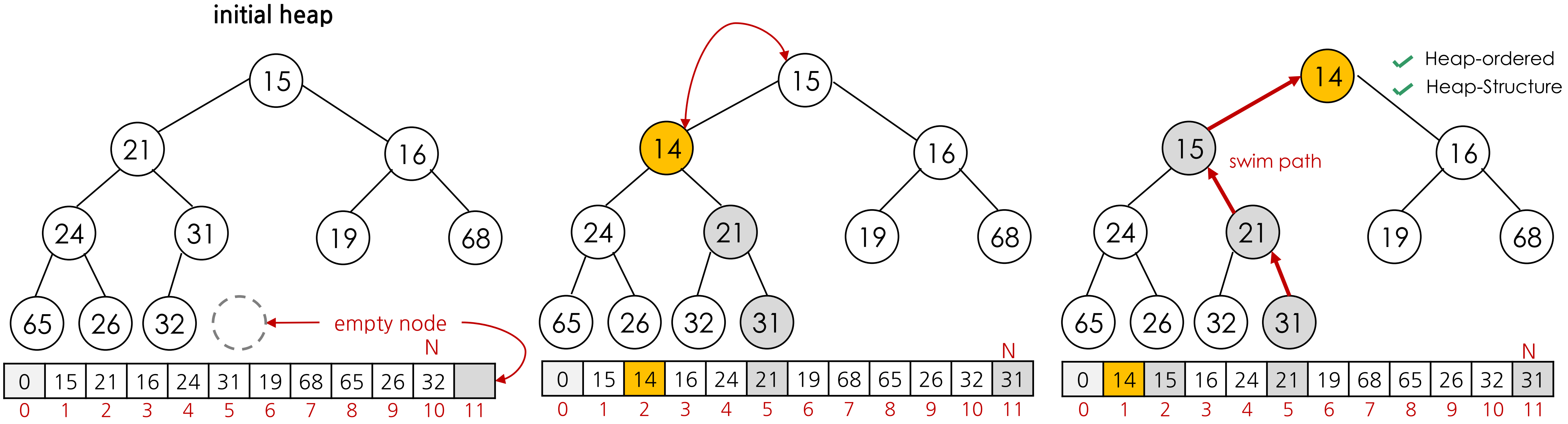
0	15	21	16	24	31	19	68	65	26	32	
0	1	2	3	4	5	6	7	8	9	10	11

0	15	14	16	24	21	19	68	65	26	32	31
0	1	2	3	4	5	6	7	8	9	10	11

min-heap: insert(heap, 14)

Algorithm:

- Insert a new element **while maintaining a** **heap-structure**
- Move the element up the heap **while not satisfying** **heap-ordered**



0	15	21	16	24	31	19	68	65	26	32	
0	1	2	3	4	5	6	7	8	9	10	11

0	15	14	16	24	21	19	68	65	26	32	31
0	1	2	3	4	5	6	7	8	9	10	11

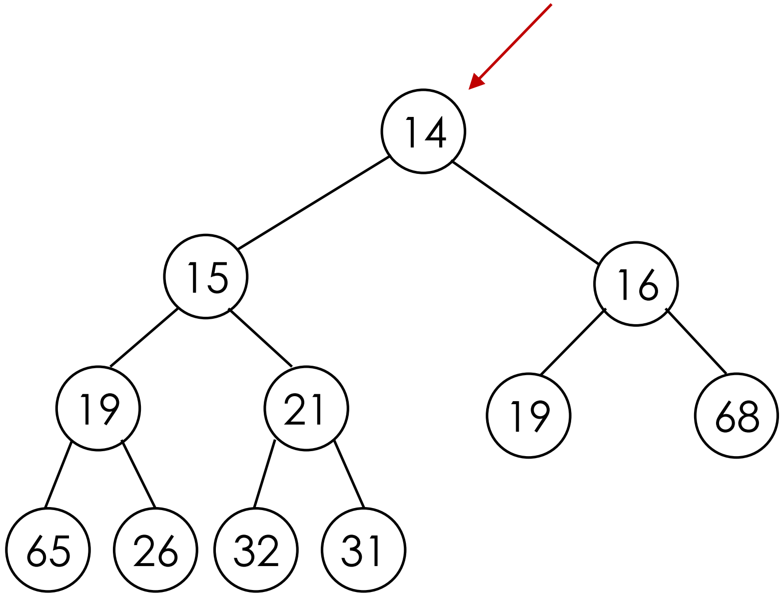
0	14	15	16	24	21	19	68	65	26	32	31
0	1	2	3	4	5	6	7	8	9	10	11

min-heap: delete() or dequeue()

- Swap the root and the last element.
- Heap decreases by one in size.
- **Move down (sink) the root** while not satisfying heap-ordered.
  - Minimum element is always at the root (by min-heap definition).

min-heap: delete() or dequeue()

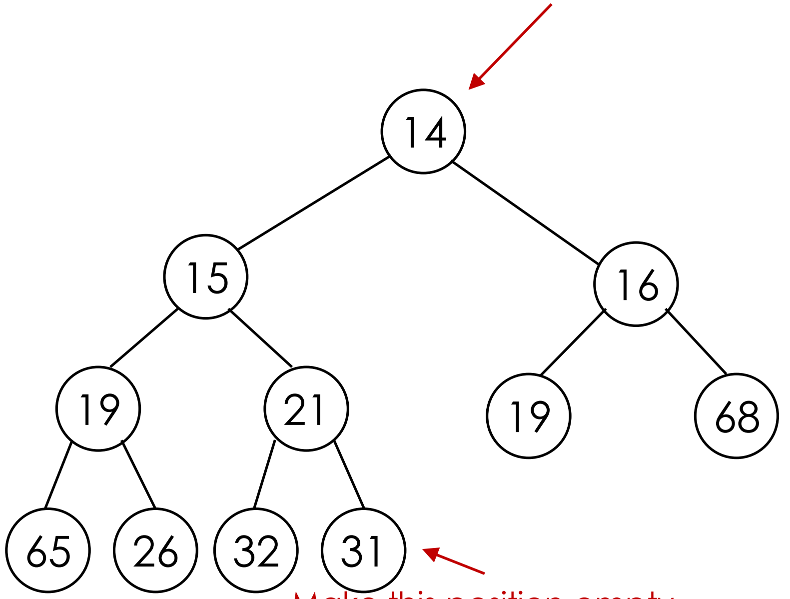
- Which position of the node will be empty?



											N
0	14	15	16	24	21	19	68	65	26	32	31
0	1	2	3	4	5	6	7	8	9	10	11

min-heap: delete() or dequeue()

- Which position of the node will be empty?

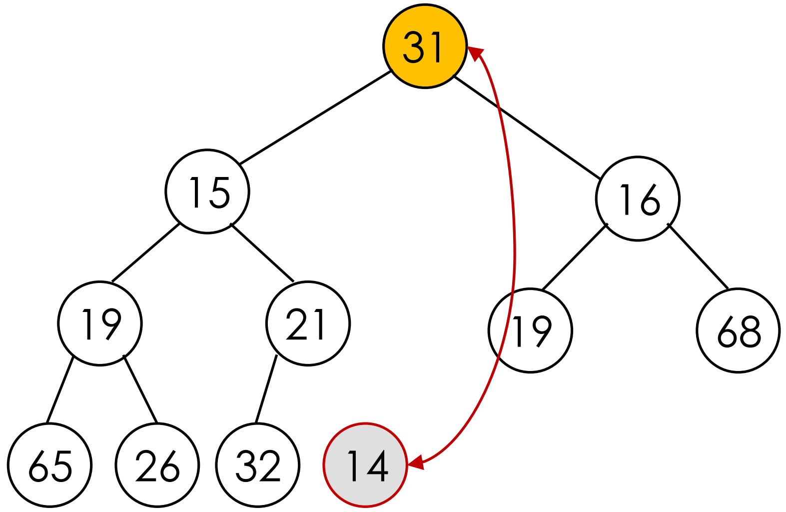


Make this position empty

N

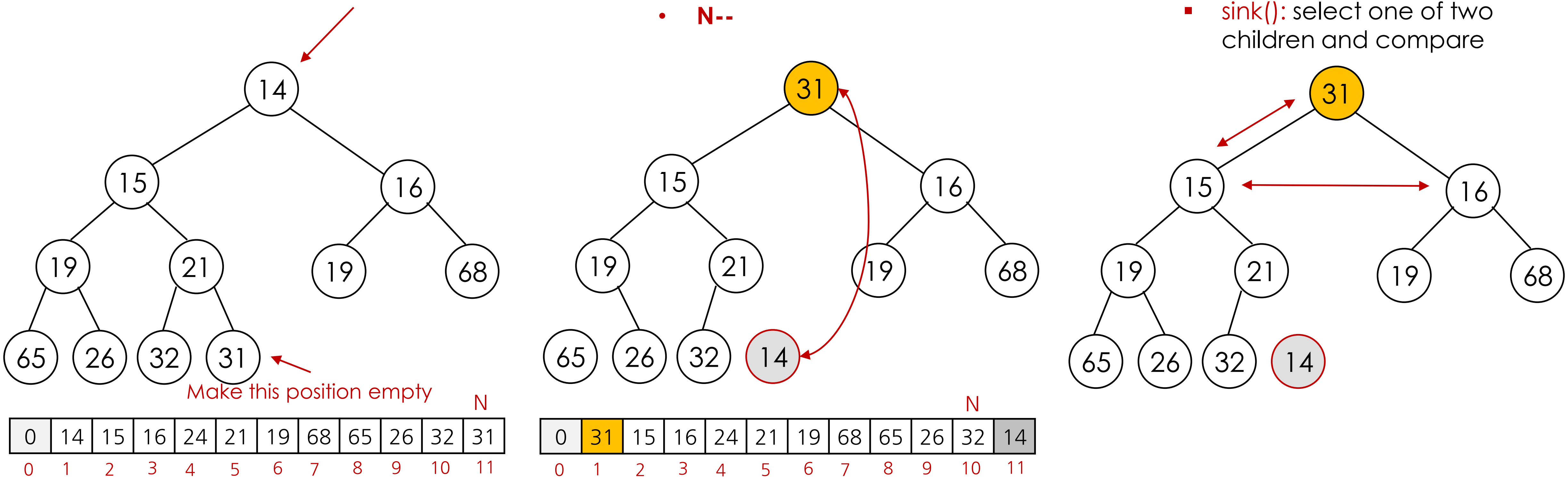
0	14	15	16	24	21	19	68	65	26	32	31
0	1	2	3	4	5	6	7	8	9	10	11

- swap the root & last
- **N--**



min-heap: delete() or dequeue()

- Which position of the node will be empty?



Make this position empty

0	14	15	16	24	21	19	68	65	26	32	31
0	1	2	3	4	5	6	7	8	9	10	11

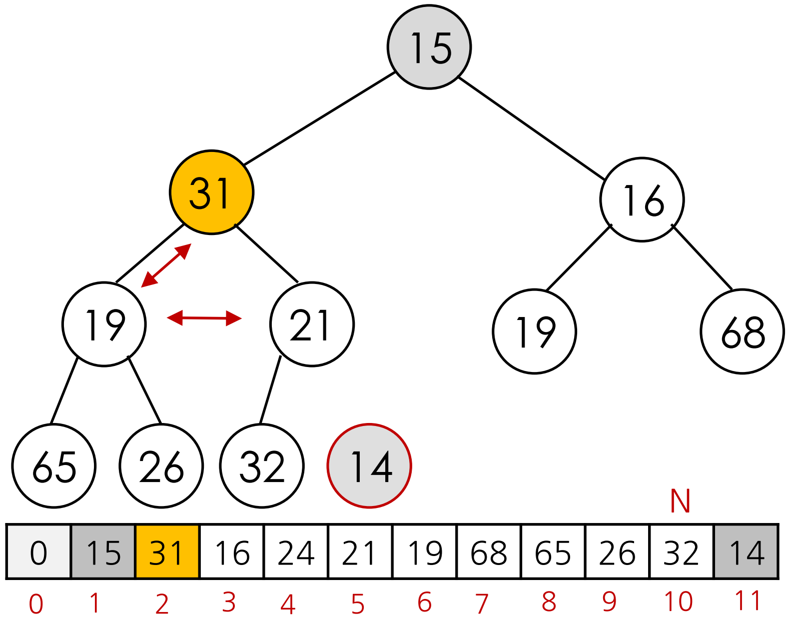
- swap the root & last
- **N--**

0	31	15	16	24	21	19	68	65	26	32	14
0	1	2	3	4	5	6	7	8	9	10	11

- heap-ordered?
- sink(): select one of two children and compare

min-heap: delete() or dequeue()

- heap-ordered?
- sink(): select one of two children and compare

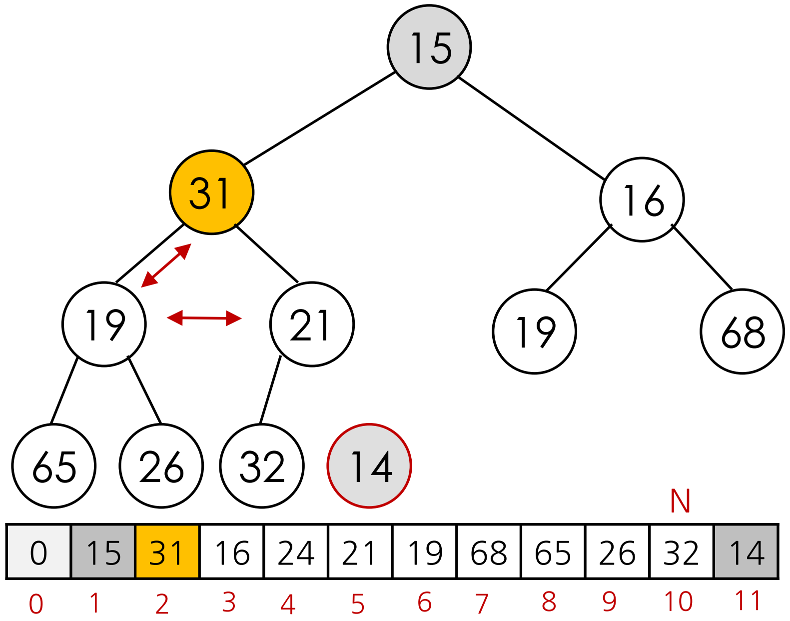


0	15	31	16	24	21	19	68	65	26	32	14
0	1	2	3	4	5	6	7	8	9	10	11

- Is 31 > min(14,16)?
- Yes - swap 31 with min(14,16)

min-heap: delete() or dequeue()

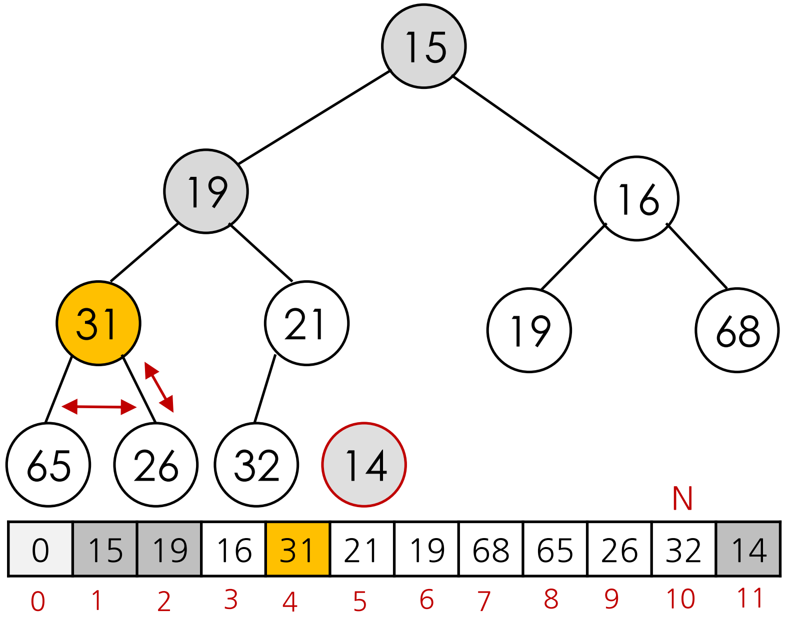
- heap-ordered?
- sink(): select one of two children and compare



0	15	31	16	24	21	19	68	65	26	32	14
0	1	2	3	4	5	6	7	8	9	10	11

- Is 31 > min(14,16)?
- Yes - swap 31 with min(14,16)

- heap-ordered?
- sink(): select one of two children and compare



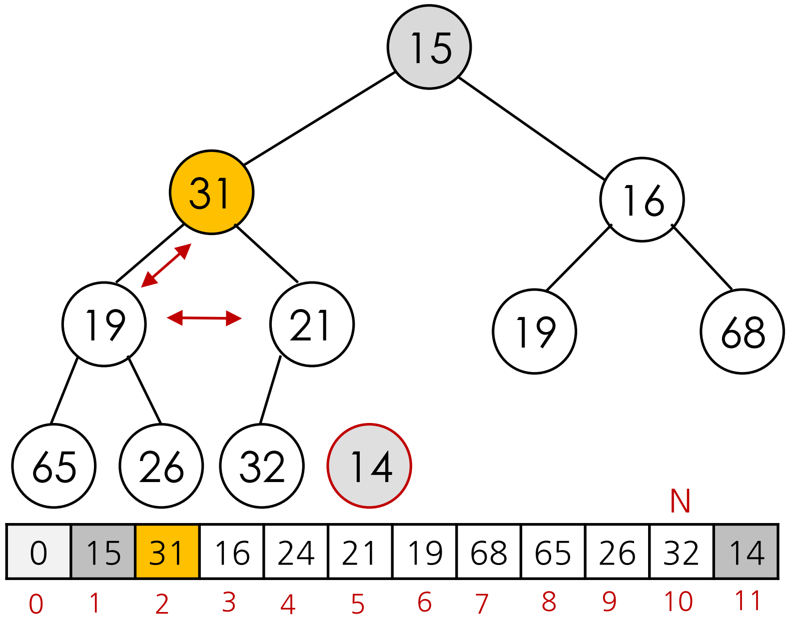
0	15	19	16	31	21	19	68	65	26	32	14
0	1	2	3	4	5	6	7	8	9	10	11

- Is 31 > min(19,21)?
- Yes - swap 31 with min(19,21)

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204, Handong Global University

min-heap: delete() or dequeue()

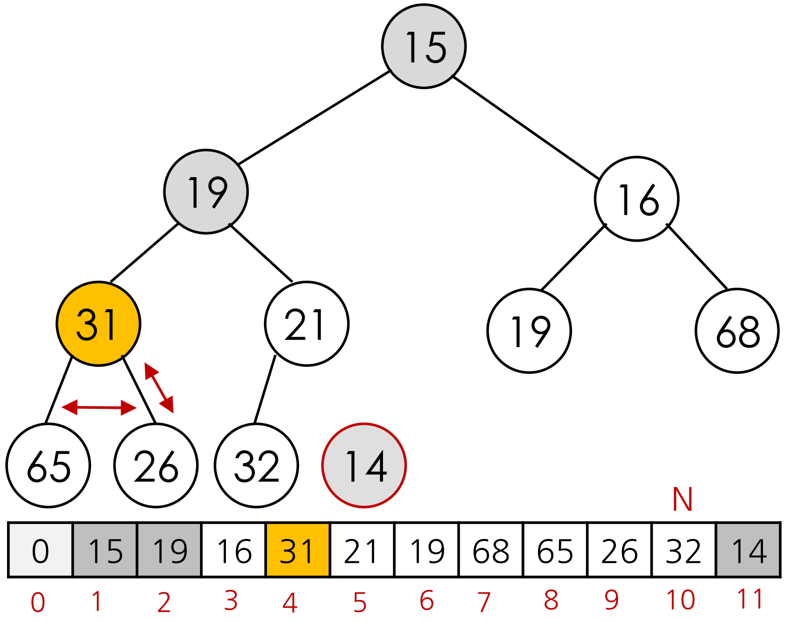
- heap-ordered?
- sink(): select one of two children and compare



0	15	31	16	24	21	19	68	65	26	32	14
0	1	2	3	4	5	6	7	8	9	10	11

- Is 31 > min(14,16)?
- Yes - swap 31 with min(14,16)

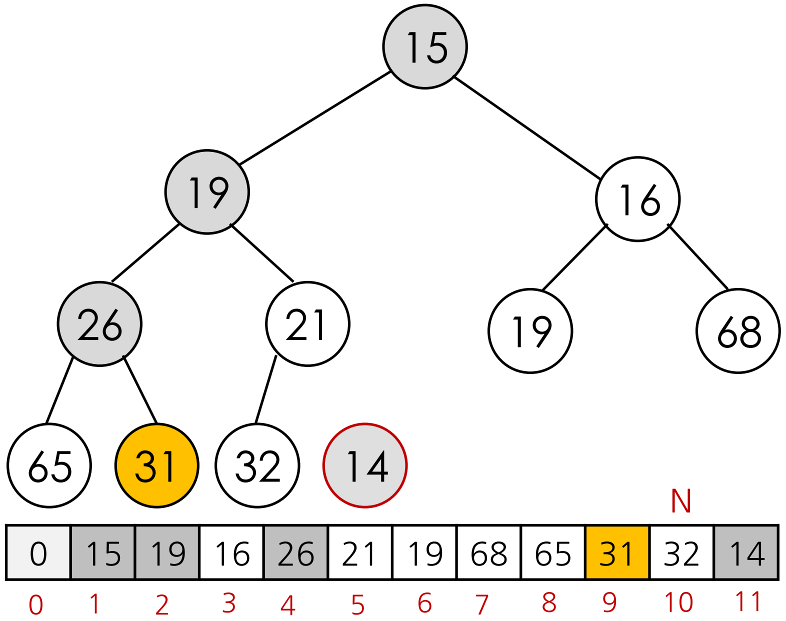
- heap-ordered?
- sink(): select one of two children and compare



0	15	19	16	31	21	19	68	65	26	32	14
0	1	2	3	4	5	6	7	8	9	10	11

- Is 31 > min(19,21)?
- Yes - swap 31 with min(19,21)

- heap-ordered?
- sink(): select one of two children and compare



0	15	19	16	26	21	19	68	65	31	32	14
0	1	2	3	4	5	6	7	8	9	10	11

- Is 31 > min(65,26)?
- Yes - swap 31 with min(65,26)

✓ Heap-ordered

✓ Heap-Structure

Binary heap: Time complexity:

- Level of heap is $\lfloor \log_2 N \rfloor$
- insert: O(log N) for each insert
  - In practice, expect less
- delete: O(log N) // deleting root node or any node
- increase/decrease key: O(log N)

Implementation	Insert	Delete	max
Unordered array	1	N	N
Ordered array	N	1	1
Binary heap	**log N**	**log N**	1

Mission Completed

Data Structures in Python

- Heap and Priority Queue
- **Heap Coding**
- Min/MaxHeap and Heap sort

Proof:

https://stackoverflow.com/questions/9755721/how-can-building-a-heap-be-on-time-complexity
https://www.insertingwiththeweb.com/data-structures/binary-heap/build-heap-proof/
https://www.quora.com/How-is-the-time-complexity-of-building-a-heap-is-o-n

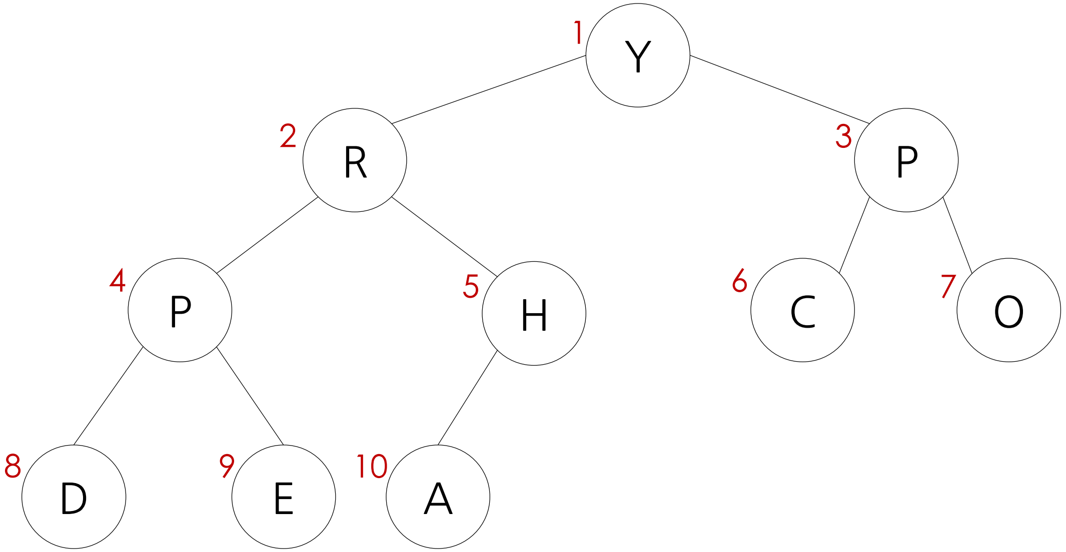
References in Korean:

https://ratsgo.github.io/data%20structure&algorithm/2017/09/27/heapsort/
https://zeddios.tistory.com/56

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204, Handong Global University

max-heap example

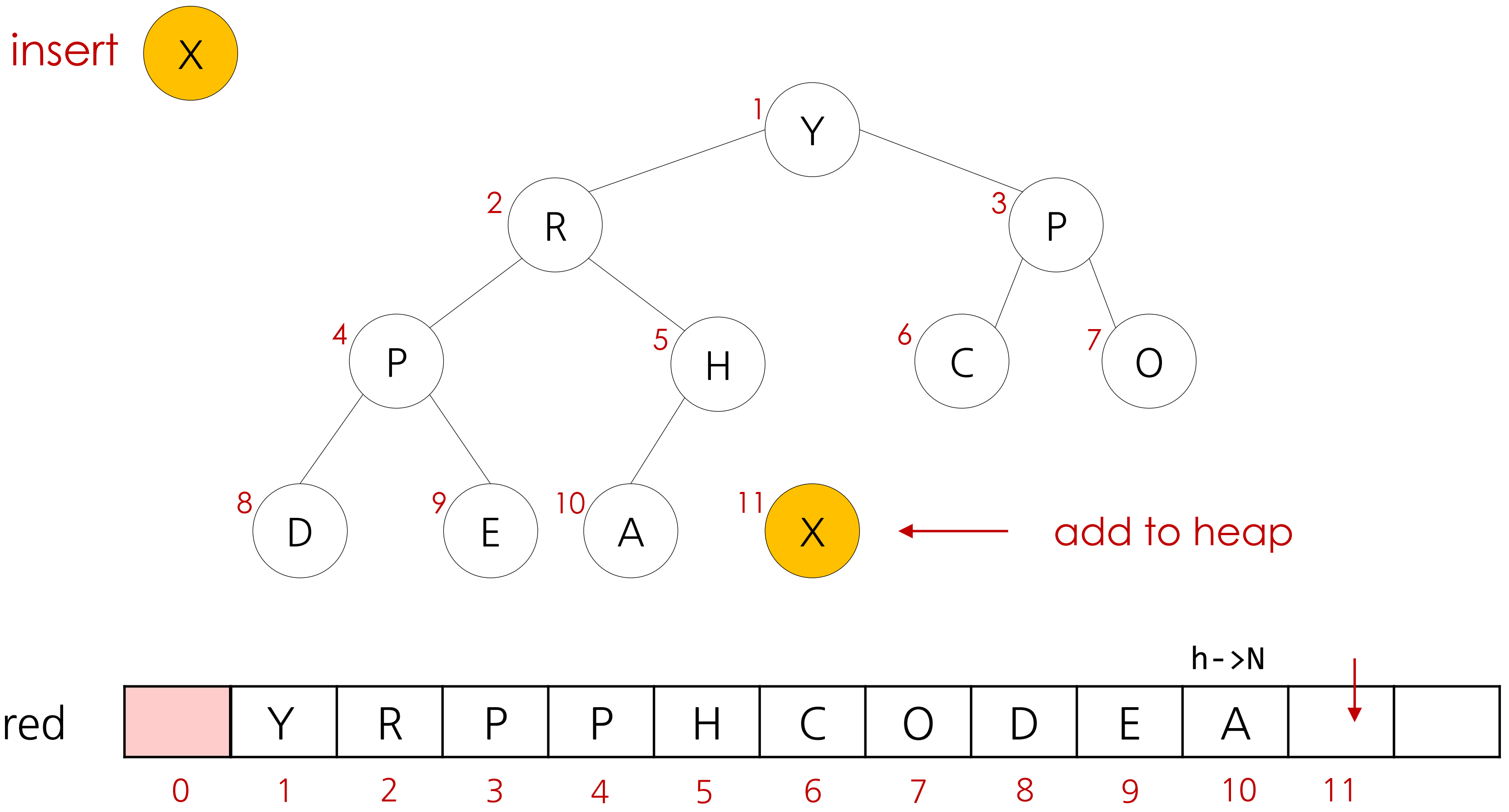
- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



											N		
heap-ordered		Y	R	P	P	H	C	O	D	E	A		
	0	1	2	3	4	5	6	7	8	9	10	11	

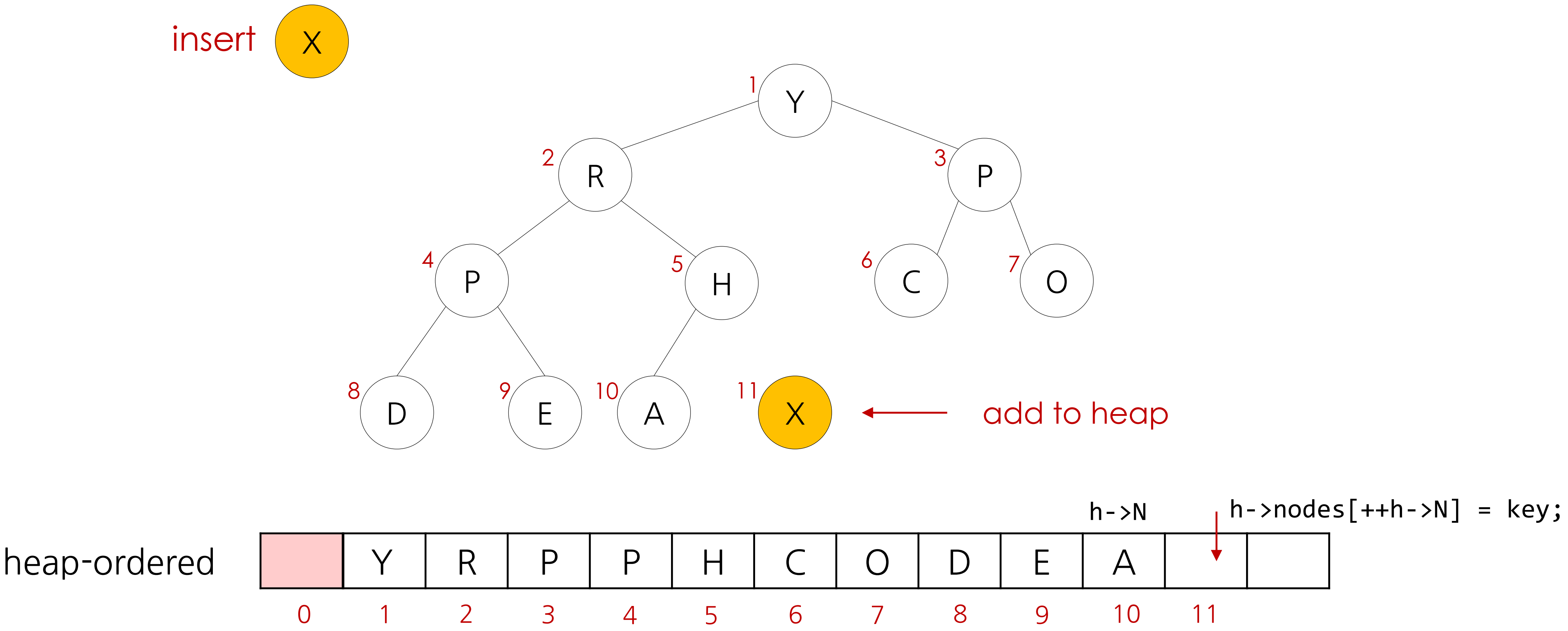
max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

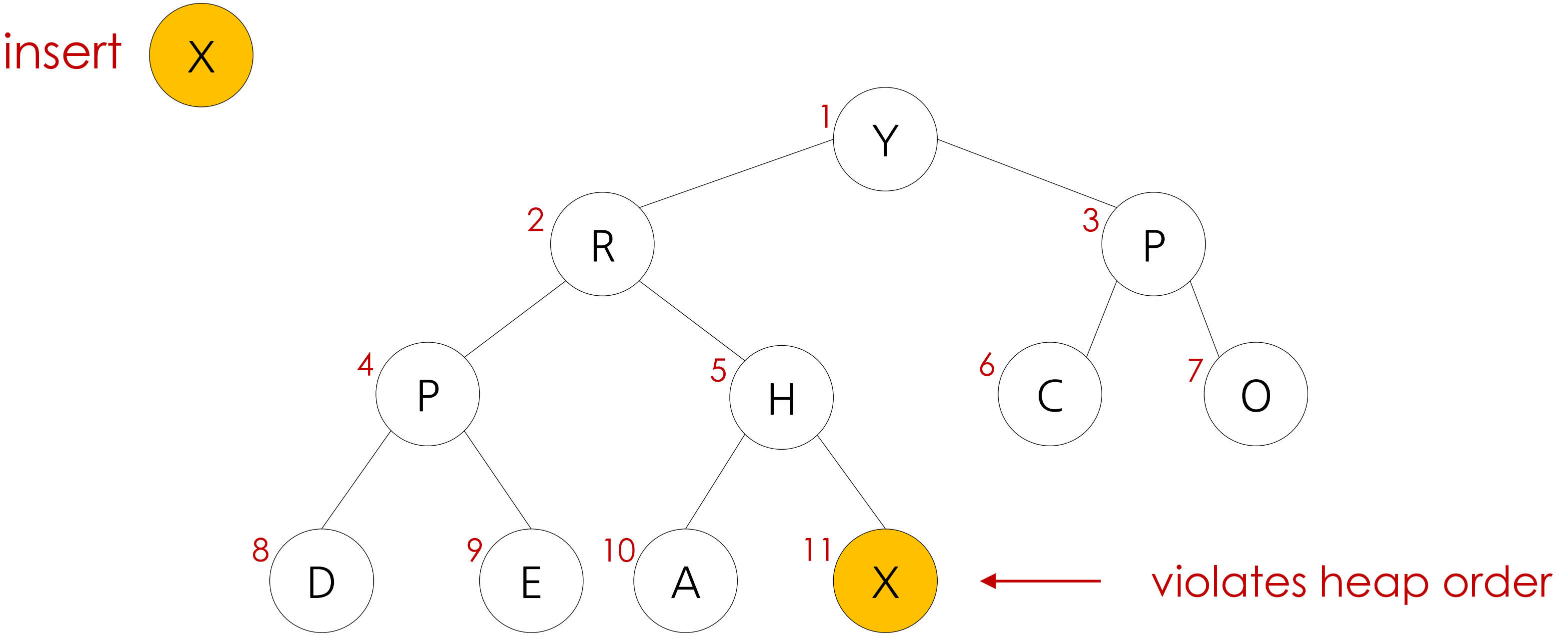


```
h->nodes[++h->N] = key;
```

	Y	R	P	P	H	C	O	D	E	A		
0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



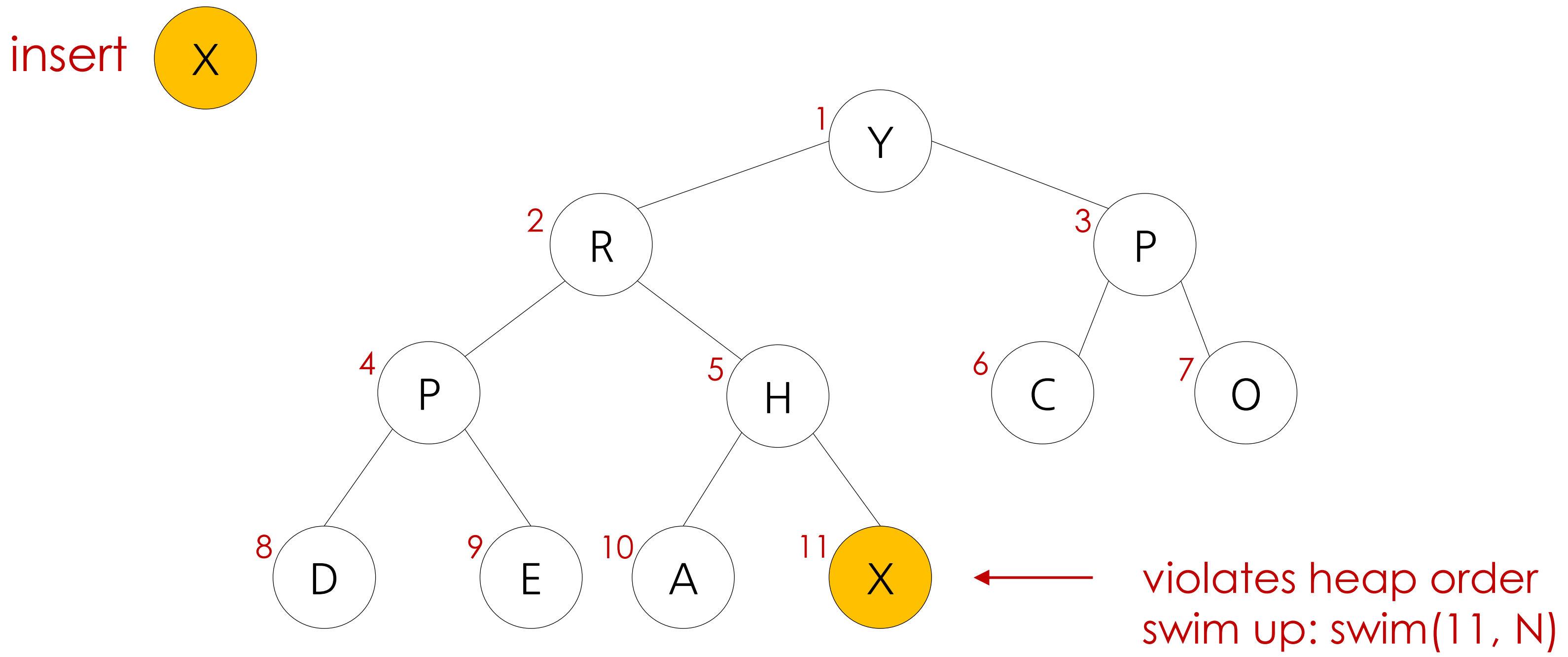
heap-ordered

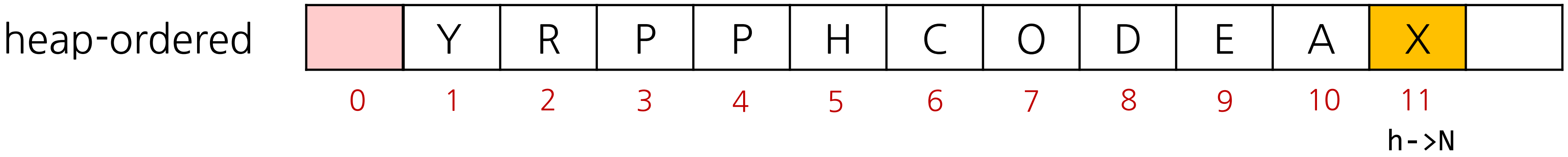
0	1	2	3	4	5	6	7	8	9	10	11	
	Y	R	P	P	H	C	O	D	E	A	X	

`h->N`

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

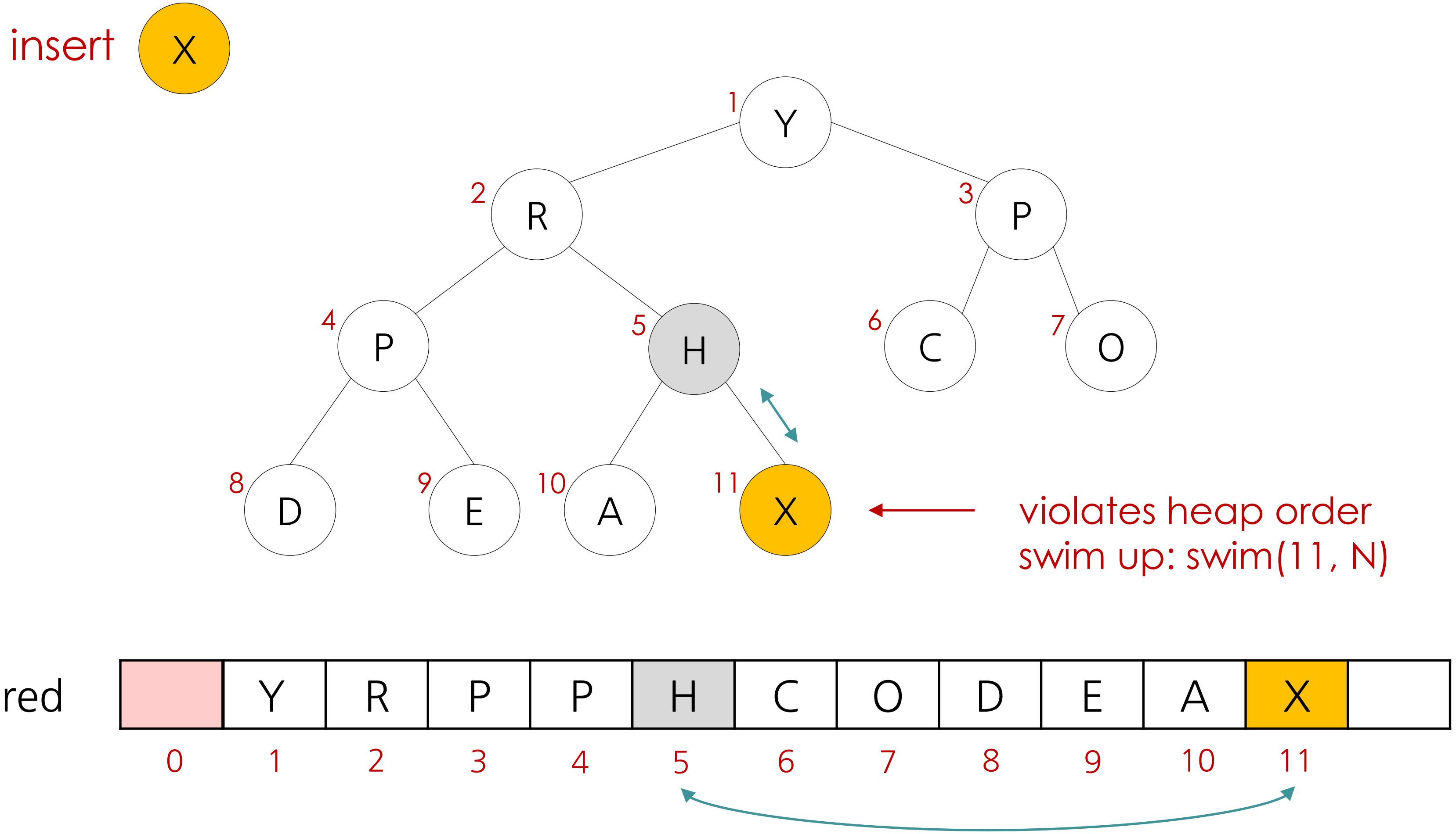



heap-ordered

	Y	R	P	P	H	C	O	D	E	A	X	
0	1	2	3	4	5	6	7	8	9	10	11	
											h->N	

max-heap example

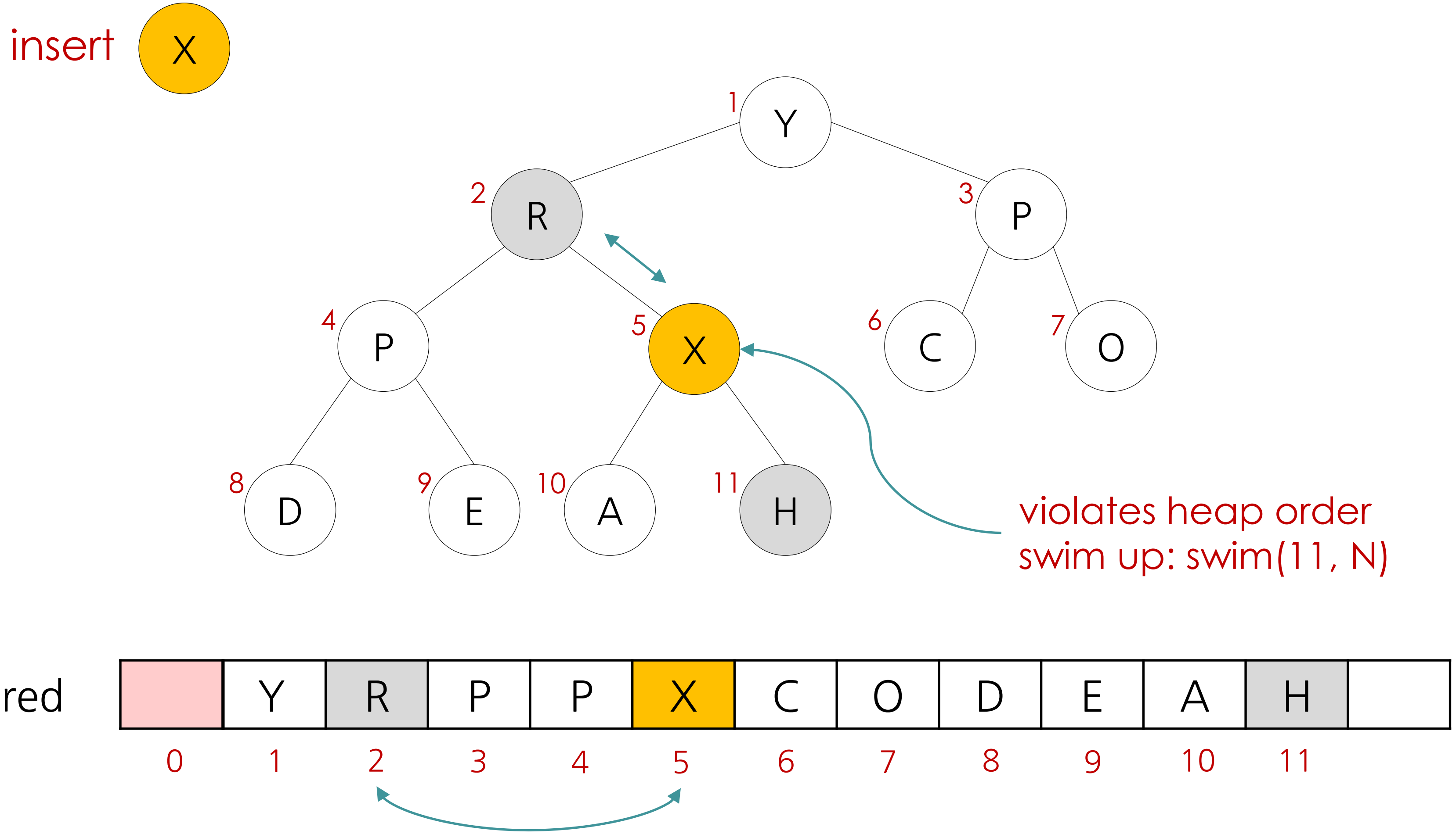
- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



	Y	R	P	P	H	C	O	D	E	A	X
0	1	2	3	4	5	6	7	8	9	10	11

max-heap example

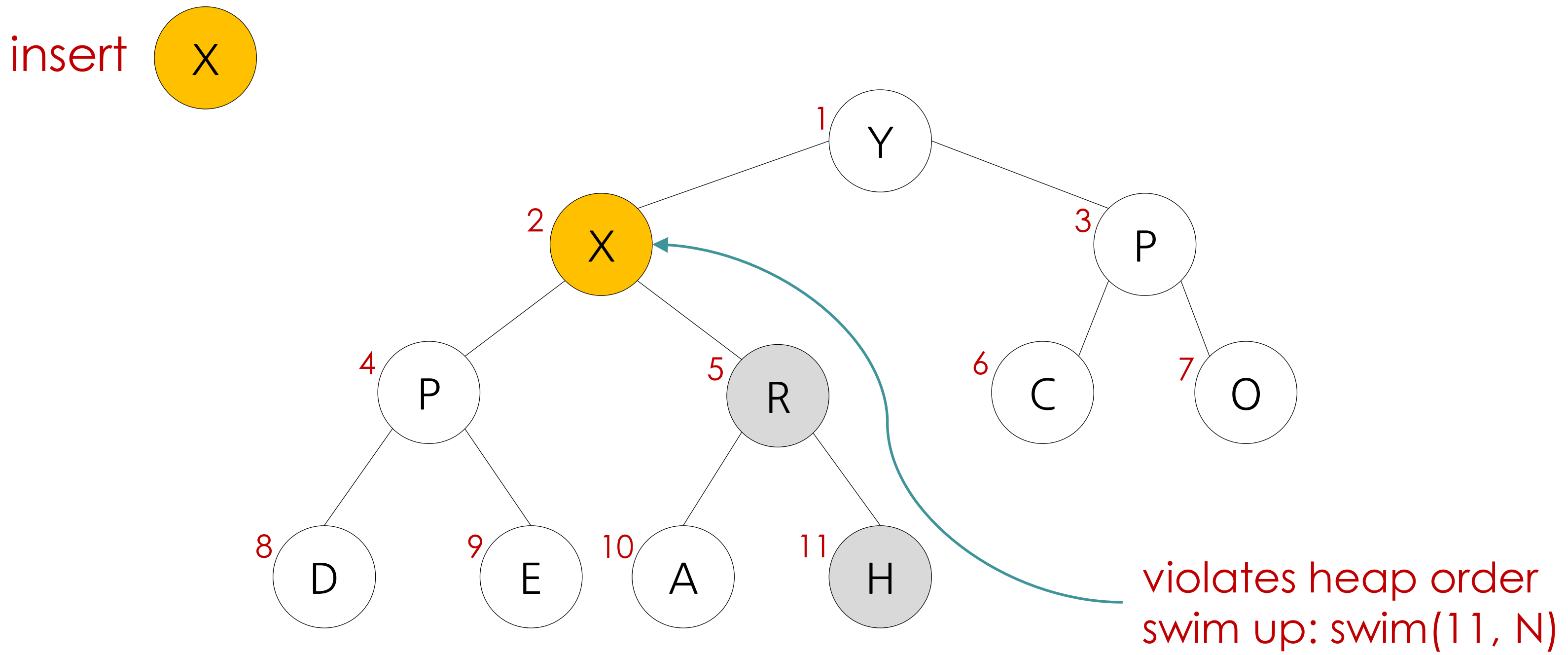
- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



0	1	2	3	4	5	6	7	8	9	10	11	
	Y	R	P	P	X	C	O	D	E	A	H	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



violates heap order
swim up: swim(11, N)

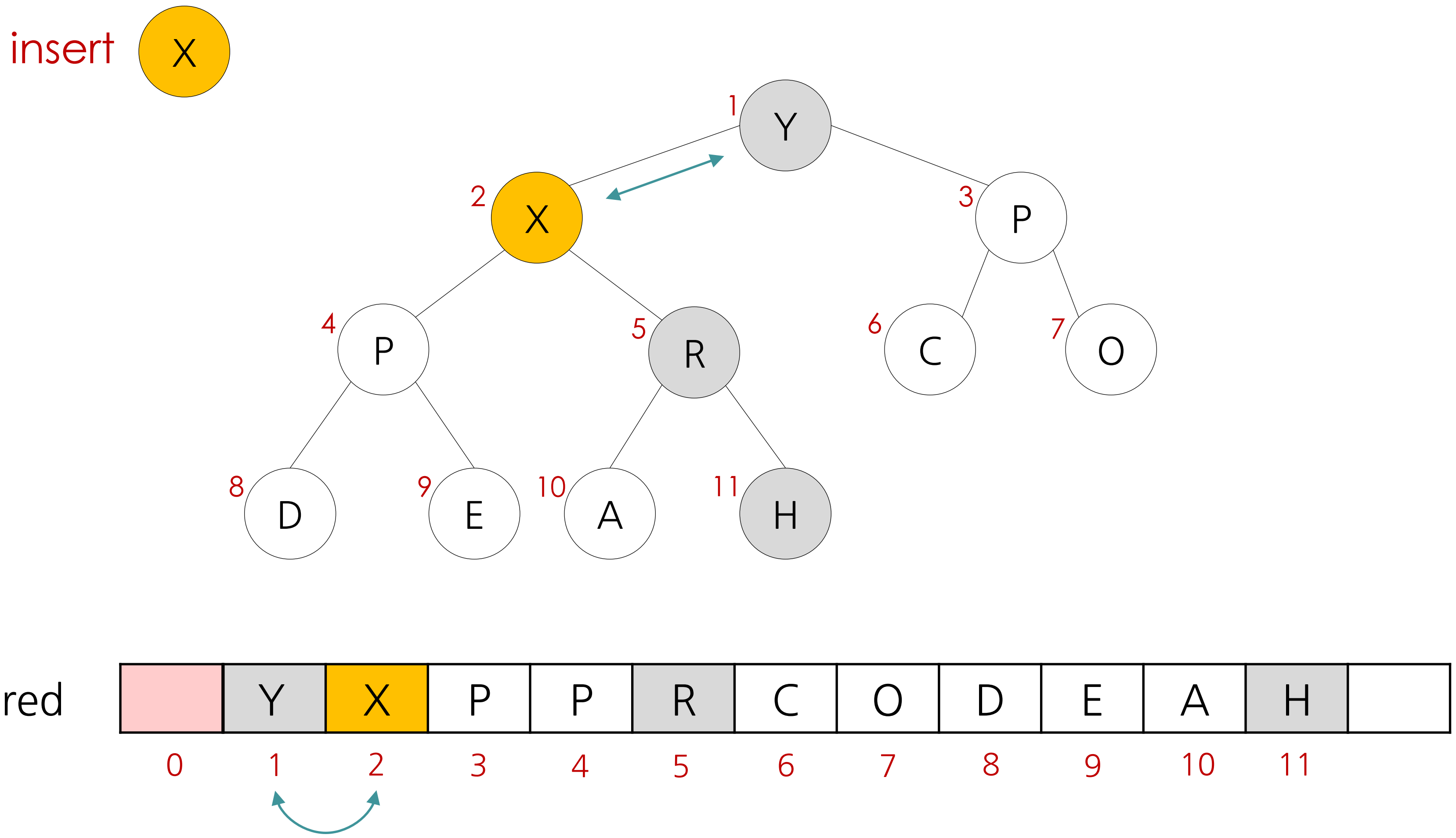
heap-ordered

	Y	X	P	P	R	C	O	D	E	A	H	
0	1	2	3	4	5	6	7	8	9	10	11	

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204, Handong Global University

max-heap example

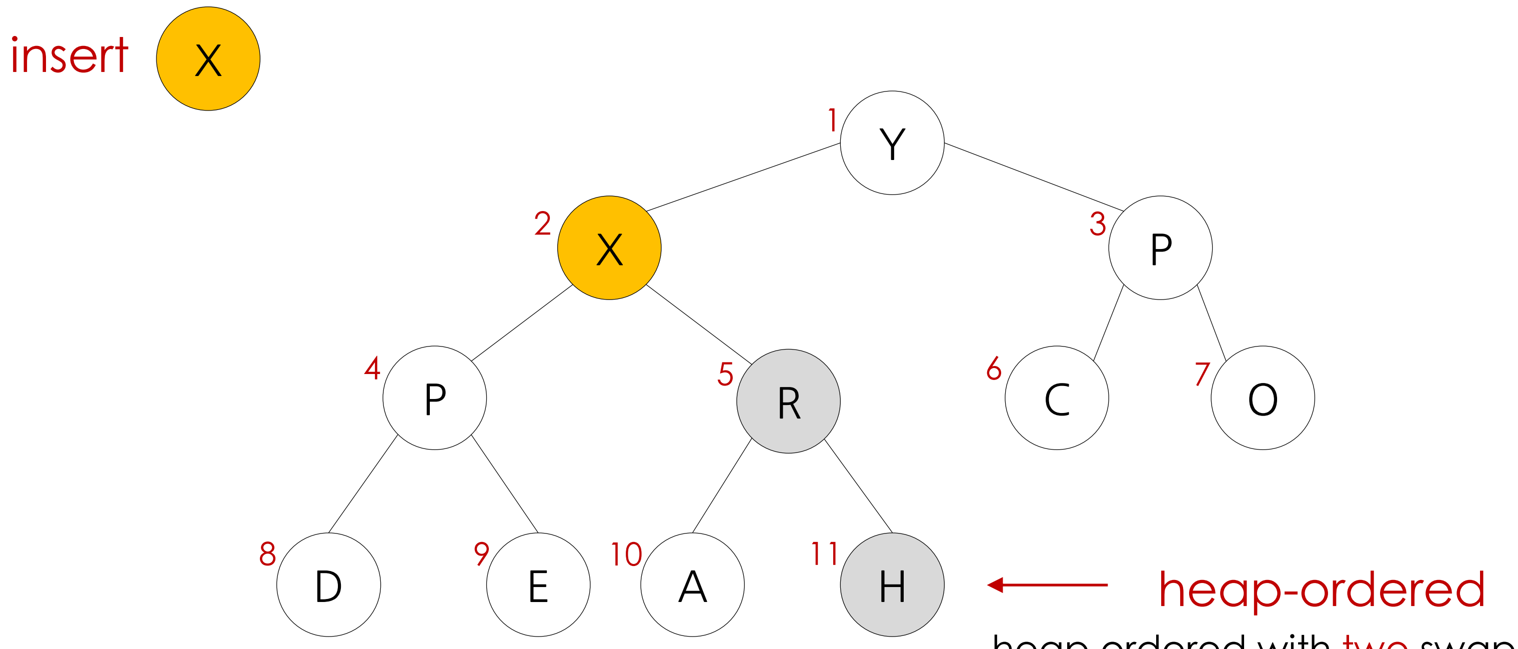
- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



	Y	X	P	P	R	C	O	D	E	A	H	
0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



heap ordered with two swaps only!

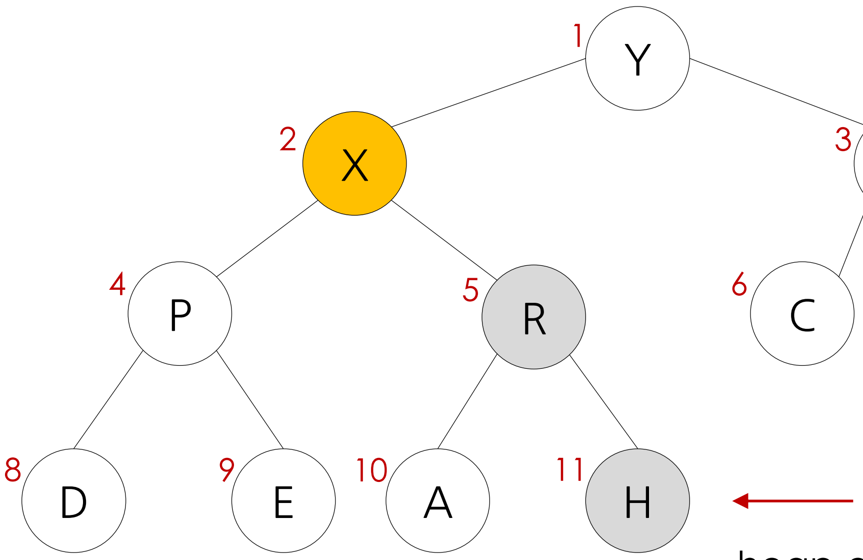
heap-ordered

0	1	2	3	4	5	6	7	8	9	10	11	
	Y	X	P	P	R	C	O	D	E	A	H	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

How to code insert X



heap-ordered

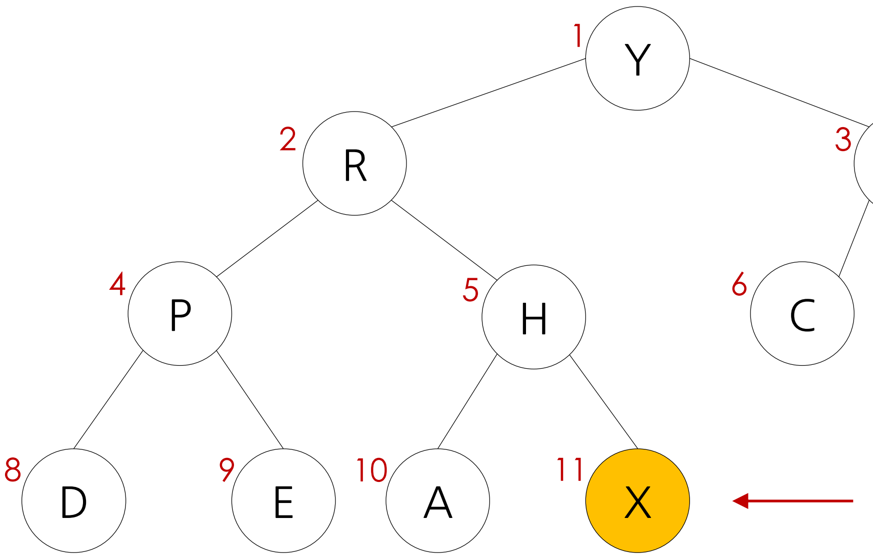
heap ordered with two swaps only!

heap-ordered		Y	X	P	P	R	C	O	D	E	A	H	
	0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

How to code insert X



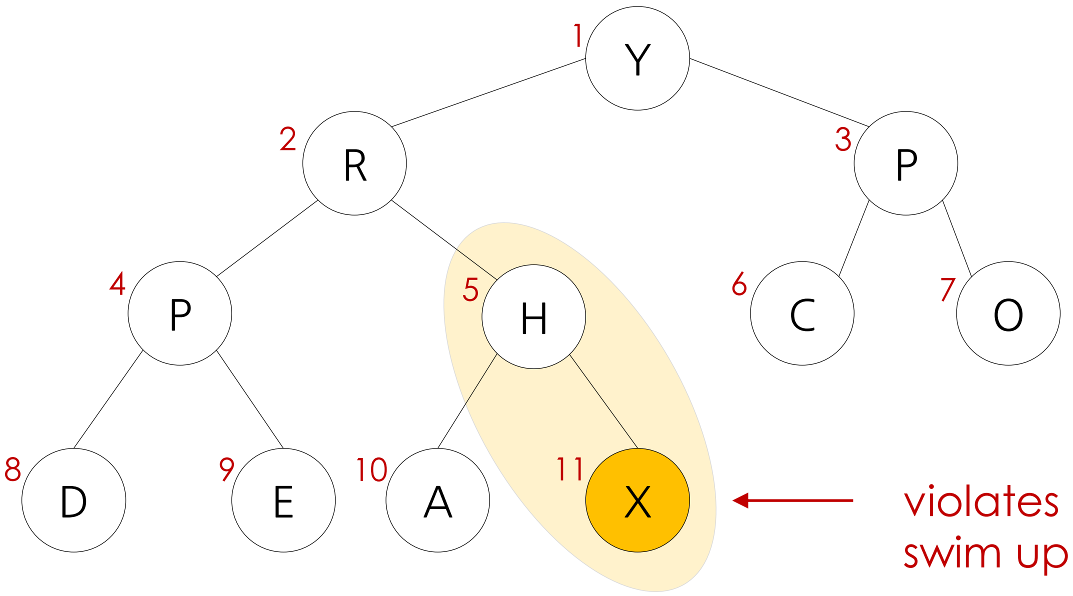
violates heap order
swim up: swim(11, N)

heap-ordered		Y	R	P	P	H	C	O	D	E	A	X	
	0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

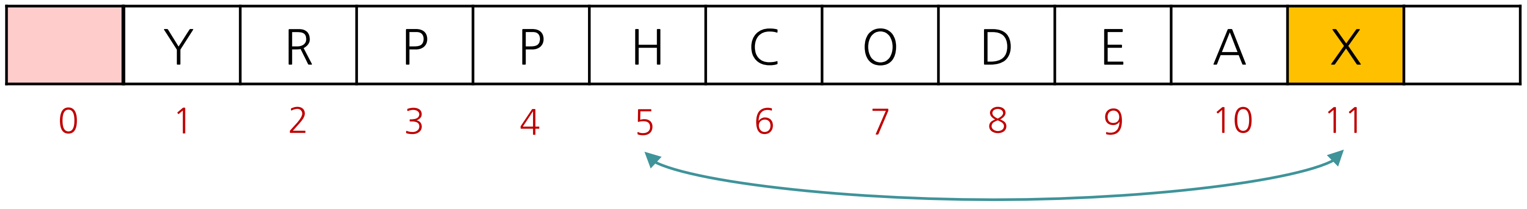
- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

How to code insert X



violates heap order
swim up: swim(11, N)

heap-ordered

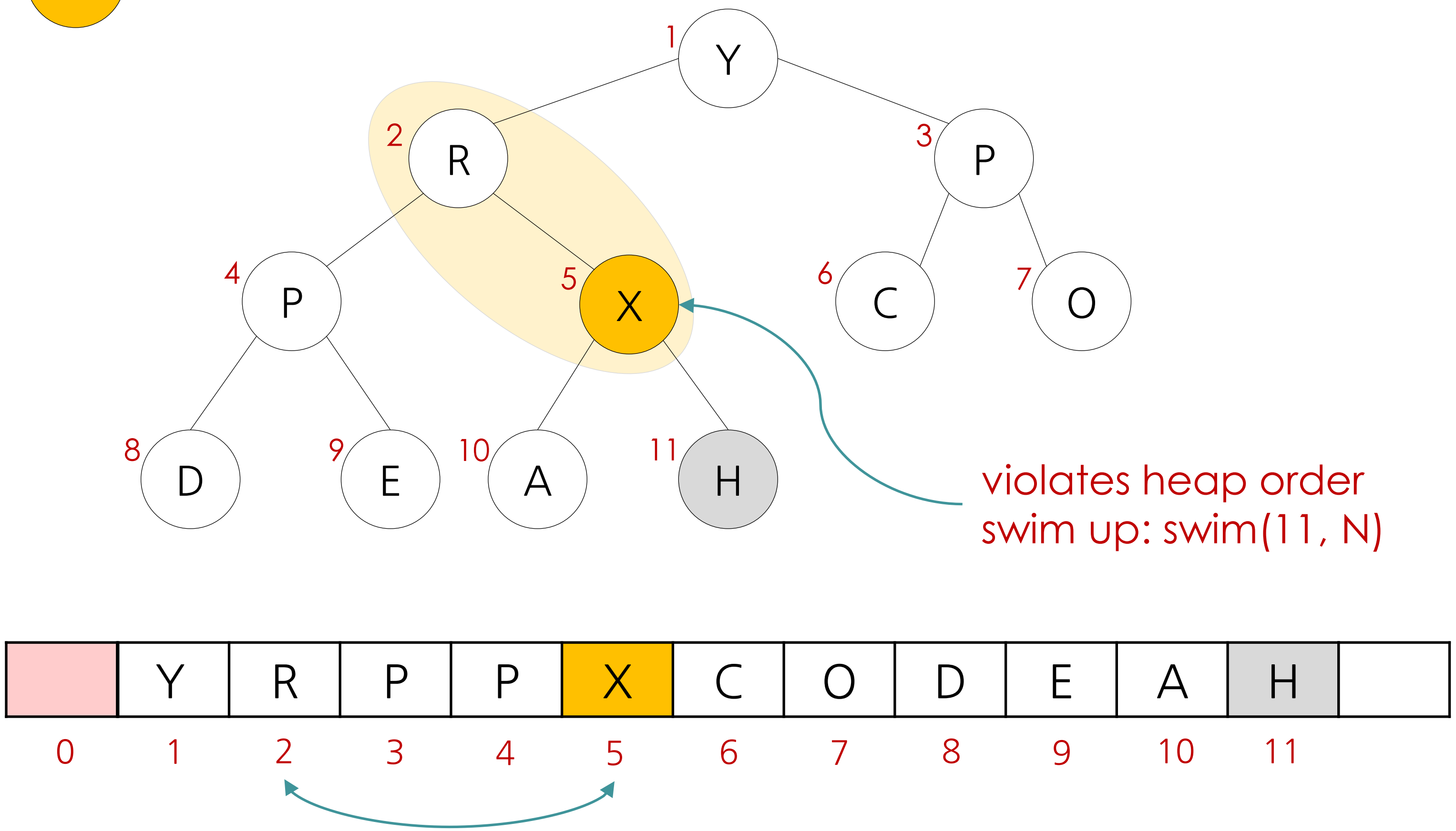


	Y	R	P	P	H	C	O	D	E	A	X	
0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

How to code insert X



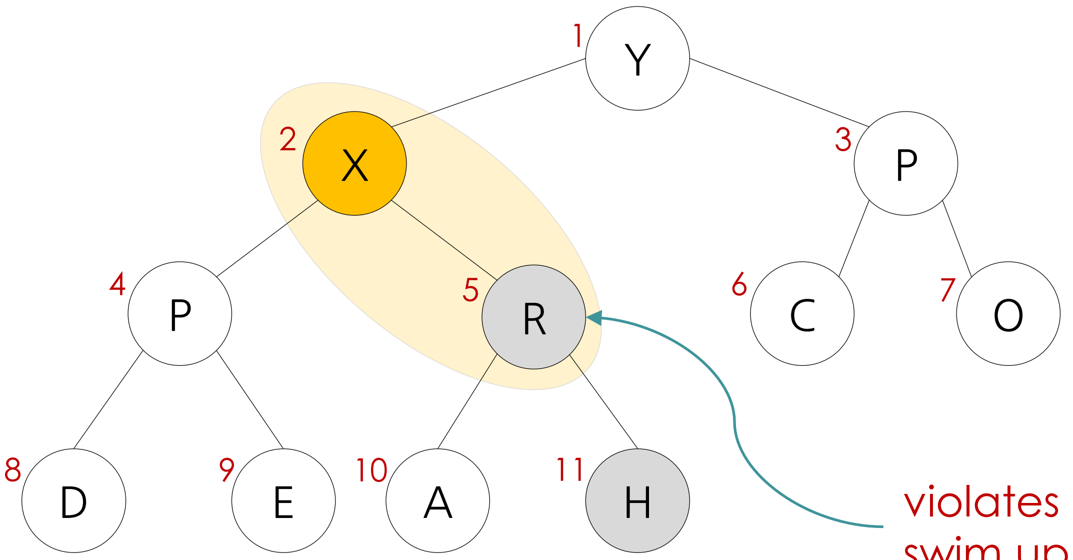
violates heap order
swim up: swim(11, N)

heap-ordered		Y	R	P	P	X	C	O	D	E	A	H	
	0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

How to code insert X



violates heap order
swim up: swim(11, N)

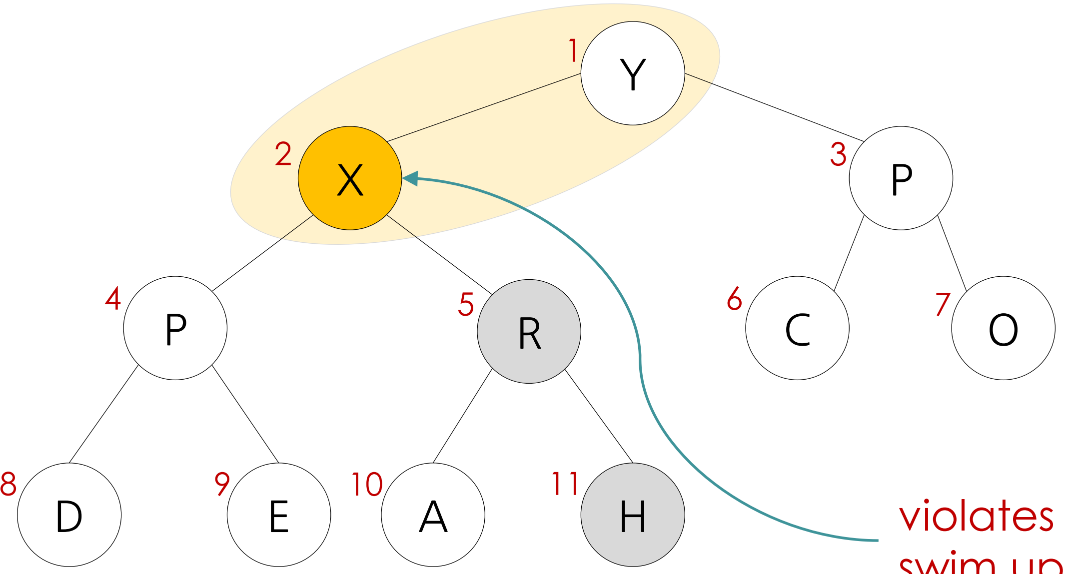
heap-ordered

	Y	X	P	P	R	C	O	D	E	A	H	
0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

How to code insert X



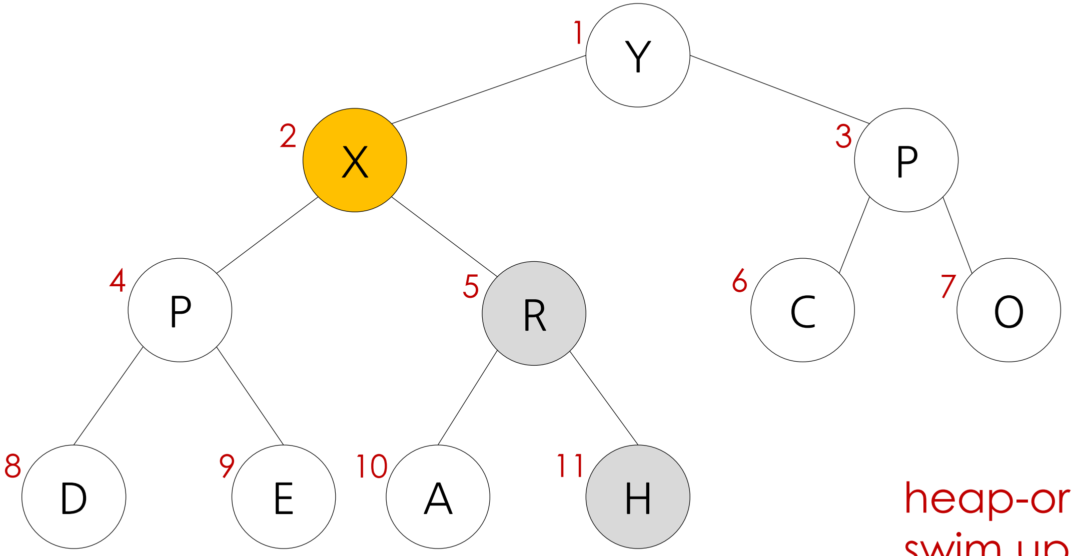
violates heap order
swim up: swim(11, N)

heap-ordered		Y	X	P	P	R	C	O	D	E	A	H	
	0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

How to code insert X



heap-ordered
swim up: swim(11, N)

heap-ordered

	Y	X	P	P	R	C	O	D	E	A	H	
0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

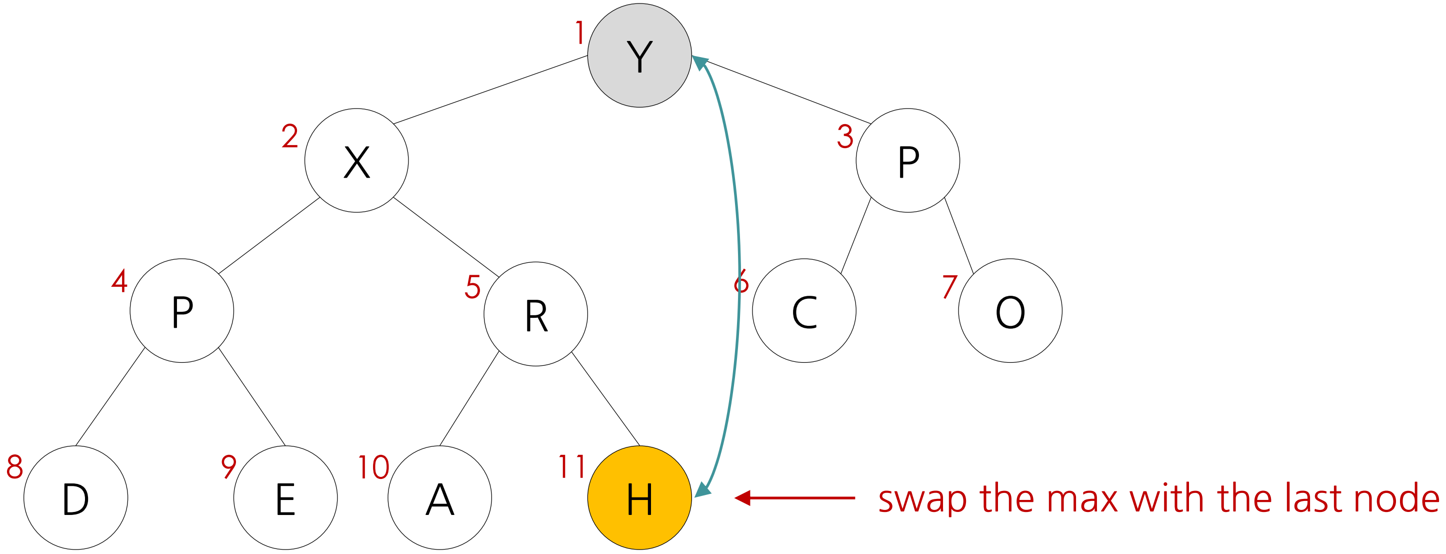
remove the max (root) – pop in priority queue

heap-ordered		Y	X	P	P	R	C	O	D	E	A	H	
	0	1	2	3	4	5	6	7	8	9	10	11	
												h->N	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)



swap the max with the last node

heap-ordered

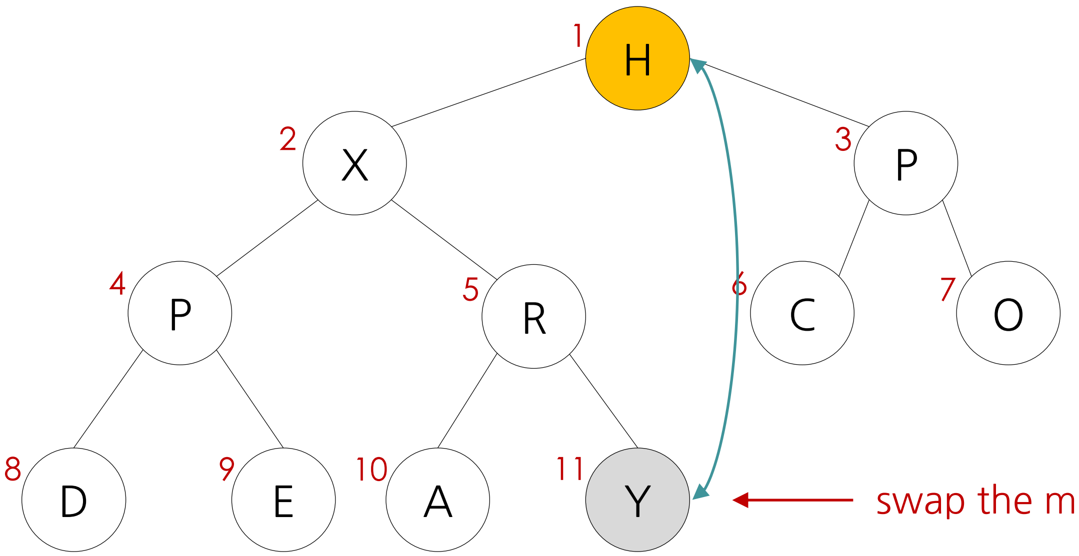
0	1	2	3	4	5	6	7	8	9	10	11	
	Y	X	P	P	R	C	O	D	E	A	H	

h->N

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)



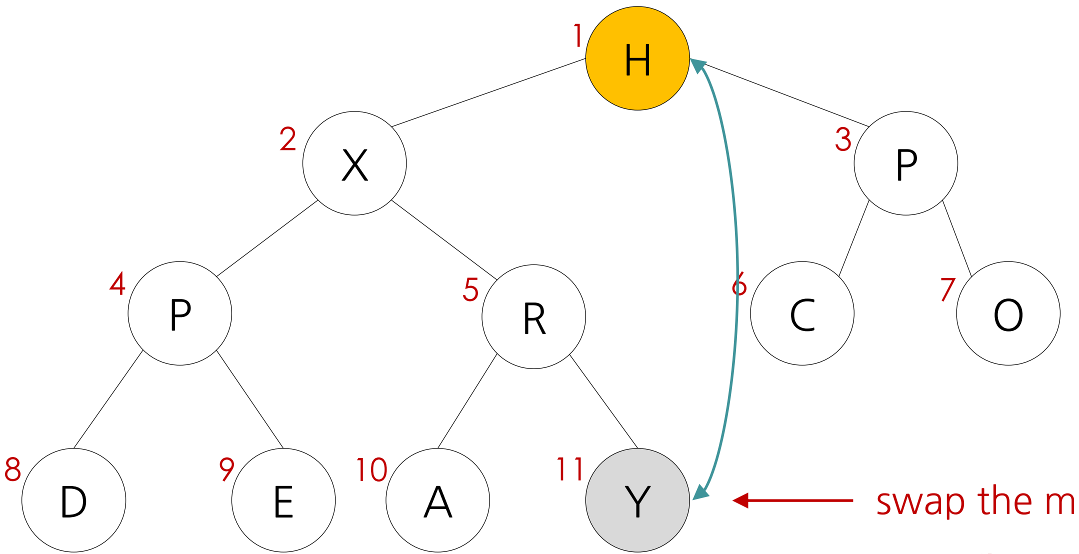
swap the max with the last node

heap-ordered		H	X	P	P	R	C	O	D	E	A	Y	
	0	1	2	3	4	5	6	7	8	9	10	11	
												h->N	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)



swap the max with the last node

`swap(h, 1, h->N)`

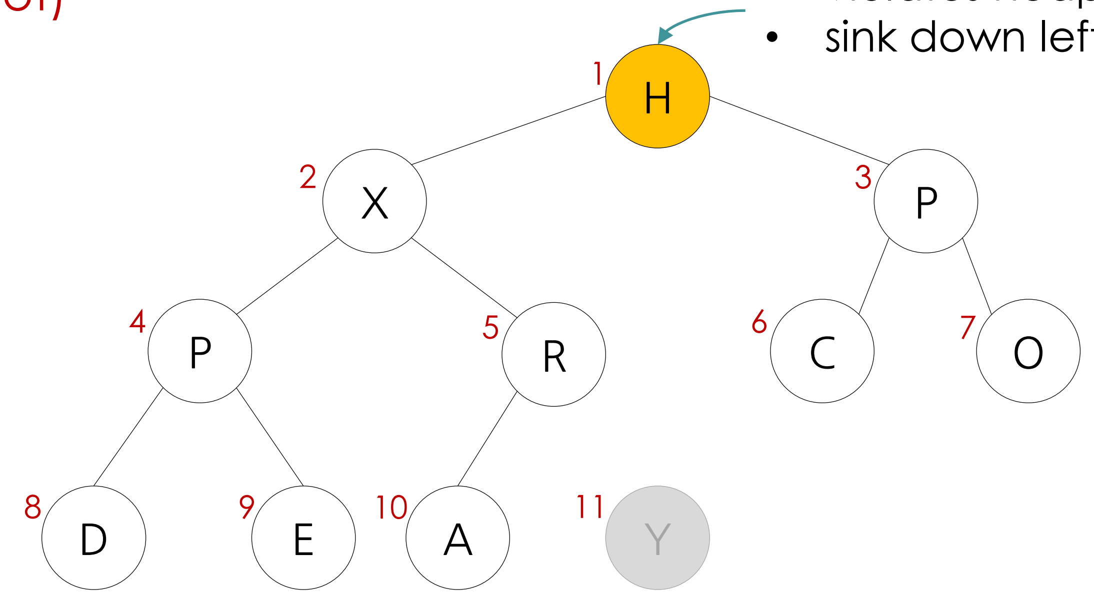
heap-ordered		H	X	P	P	R	C	O	D	E	A	Y	
	0	1	2	3	4	5	6	7	8	9	10	11	
												h->N	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)

- violates heap order: sink(1, N)
- sink down left or right?



`swap(h, 1, h->N--)`

heap-ordered		H	X	P	P	R	C	O	D	E	A	Y
	0	1	2	3	4	5	6	7	8	9	10	11

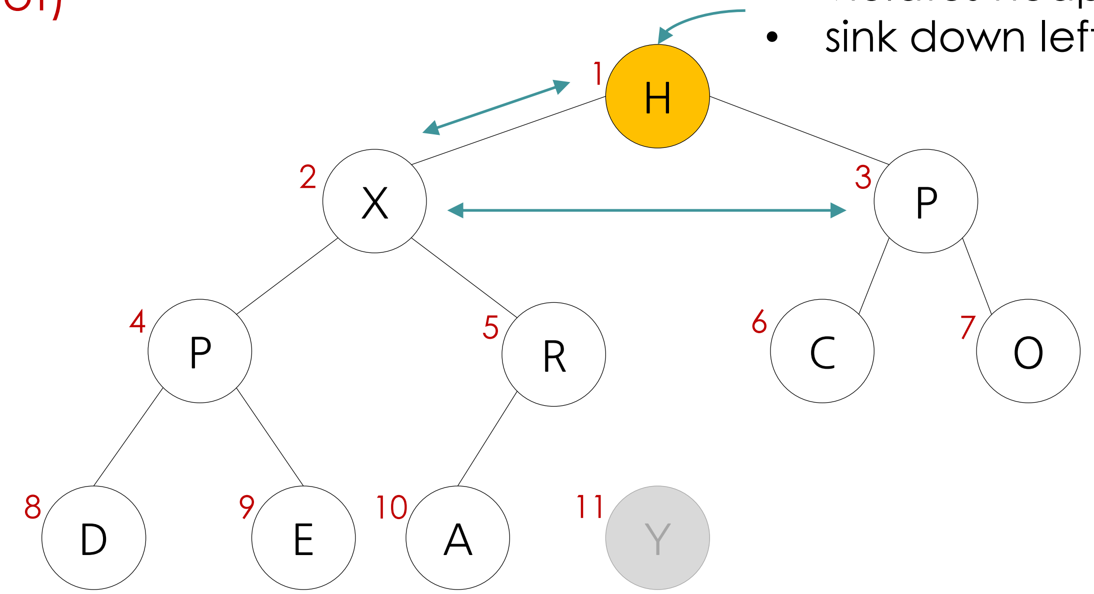
`h->N` heap size decreased by one

max-heap example

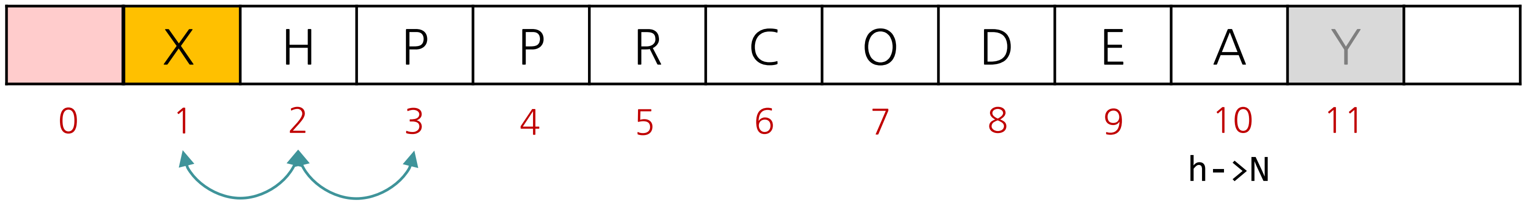
- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)

- violates heap order: sink(1, N)
- sink down left or right?



heap-ordered



0	1	2	3	4	5	6	7	8	9	10	11	
	X	H	P	P	R	C	O	D	E	A	Y	

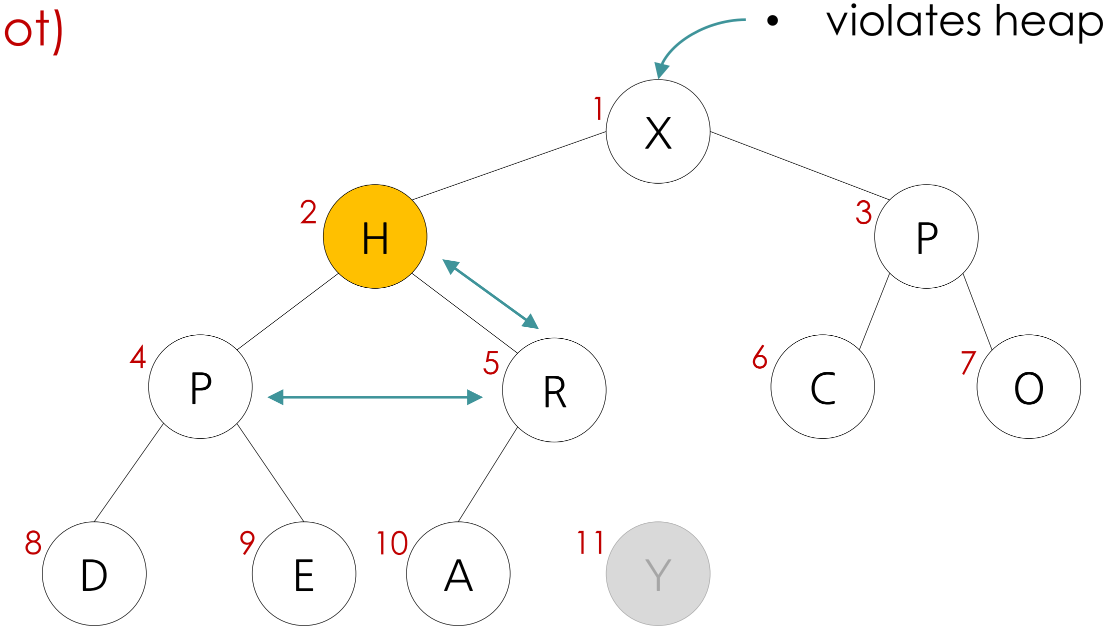
h->N

max-heap example

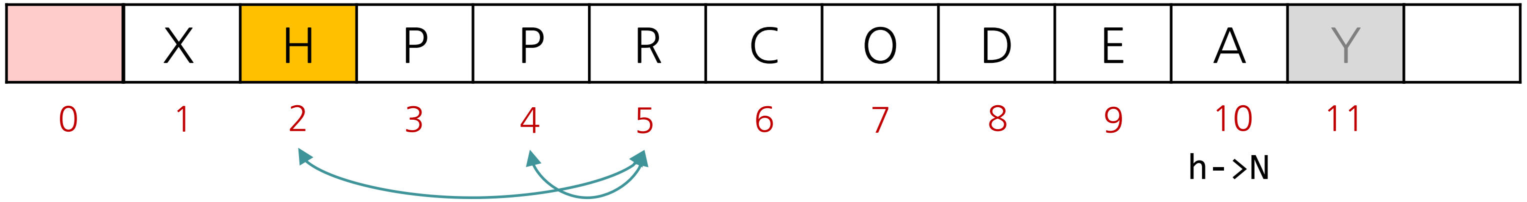
- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)

- violates heap order



heap-ordered

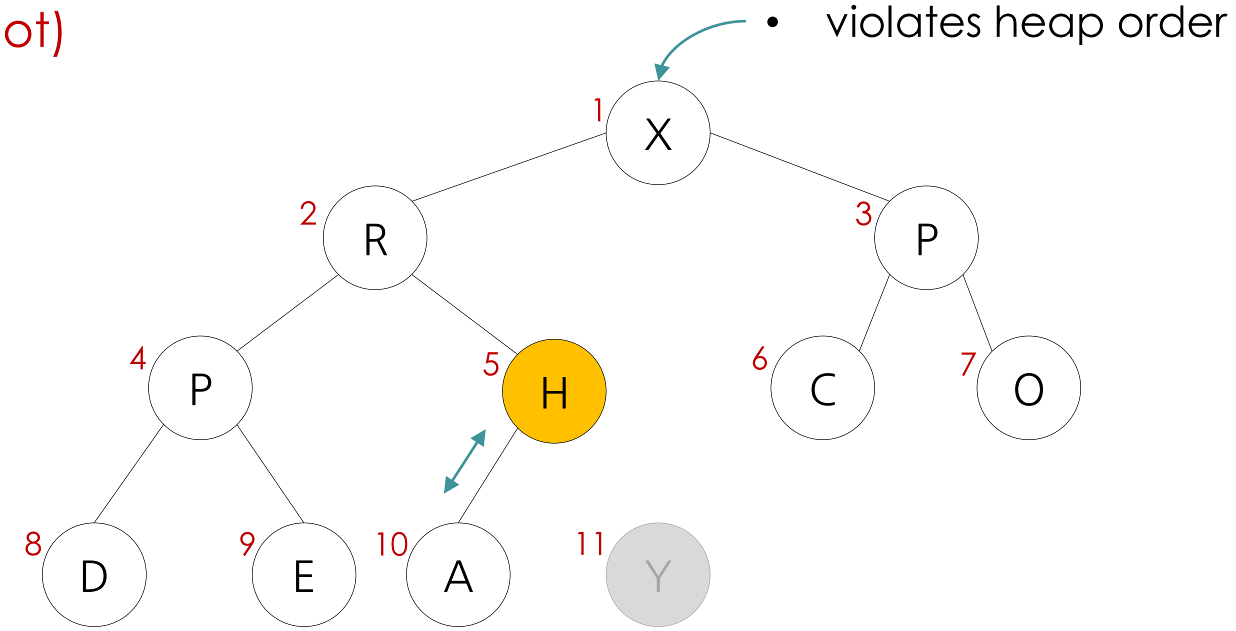


	X	H	P	P	R	C	O	D	E	A	Y	
0	1	2	3	4	5	6	7	8	9	10	11	
										h->N		

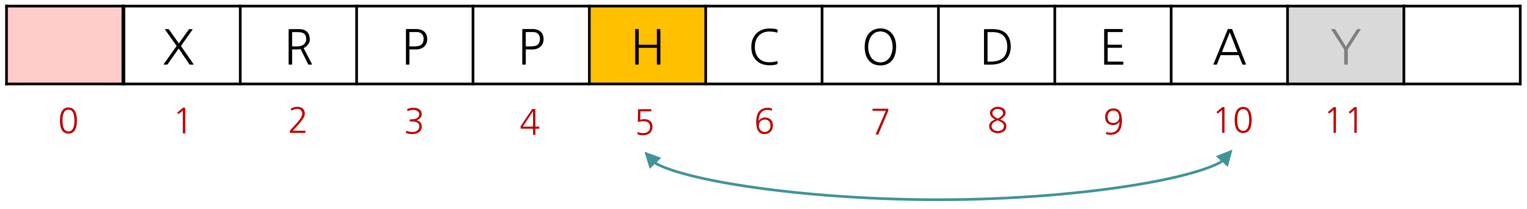
max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)



heap-ordered

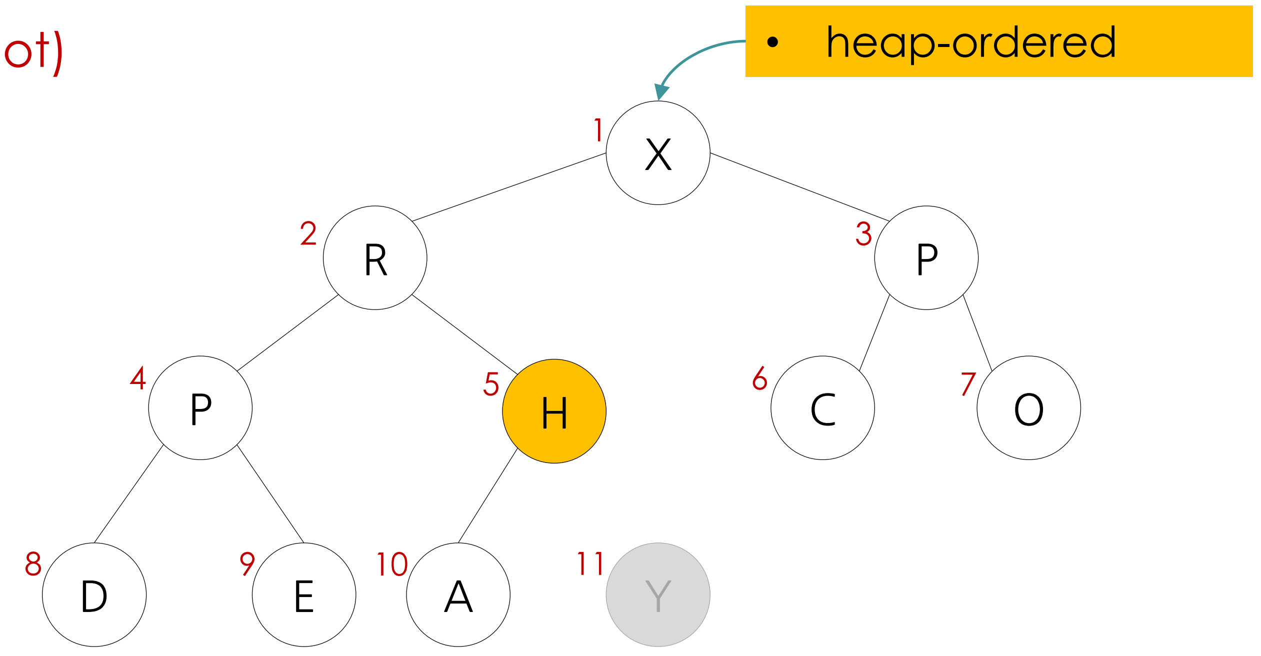


	X	R	P	P	H	C	O	D	E	A	Y	
0	1	2	3	4	5	6	7	8	9	10	11	

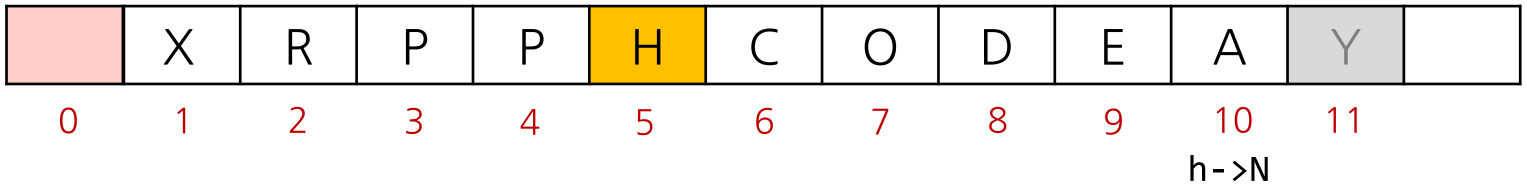
max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)



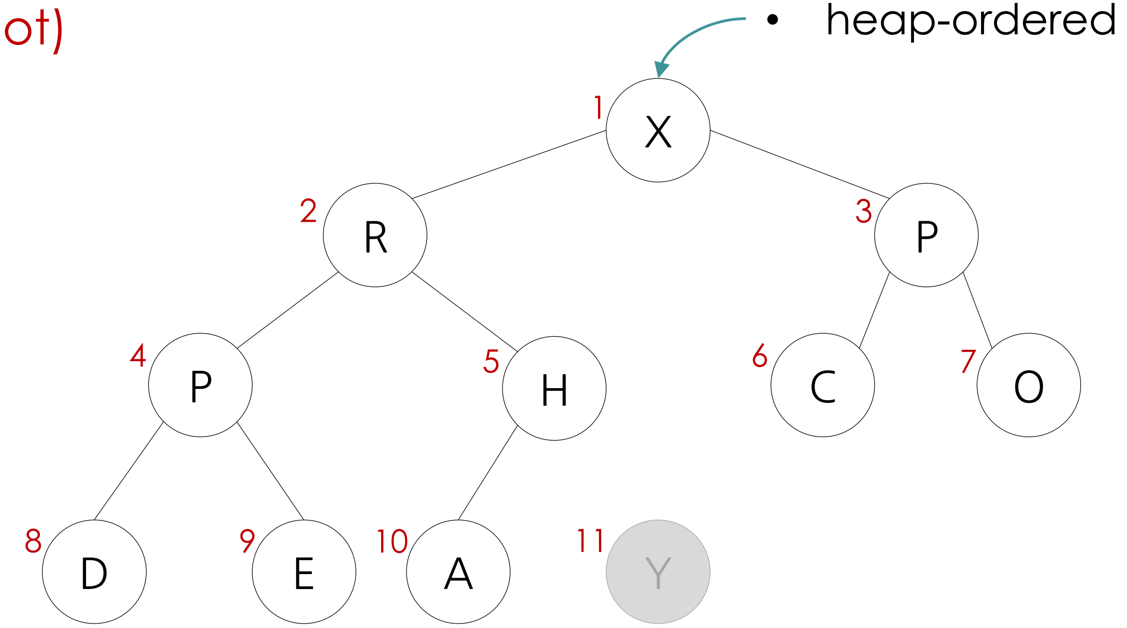
heap-ordered



max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)



- heap-ordered

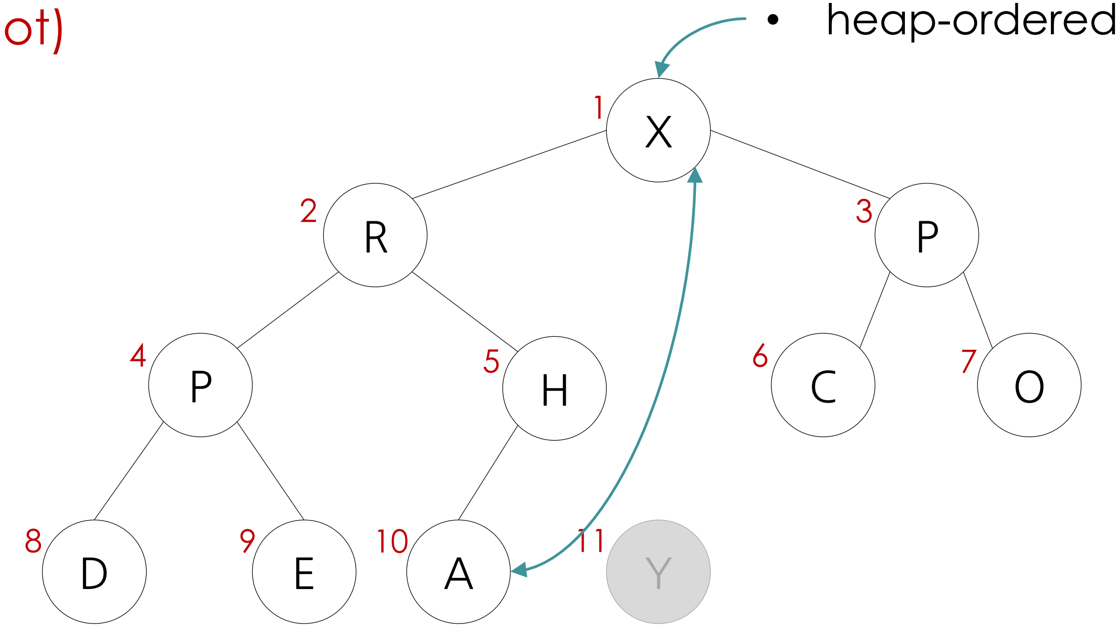
heap-ordered

	X	R	P	P	H	C	O	D	E	A	Y	
0	1	2	3	4	5	6	7	8	9	10	11	
										h->N		

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)



- heap-ordered

`swap(h, 1, h->N--)`

heap-ordered

	X	R	P	P	H	C	O	D	E	A	Y	
0	1	2	3	4	5	6	7	8	9	10	11	

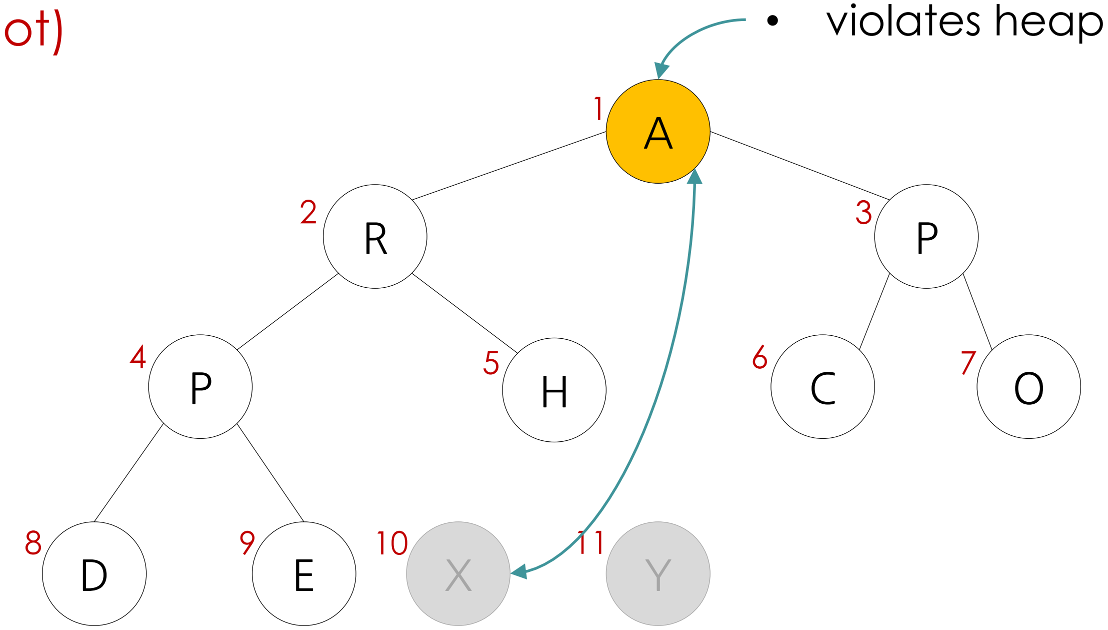
`h->N` heap size decreased by one

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)

- violates heap order: sink(1, N)



`swap(h, 1, h->N--)`

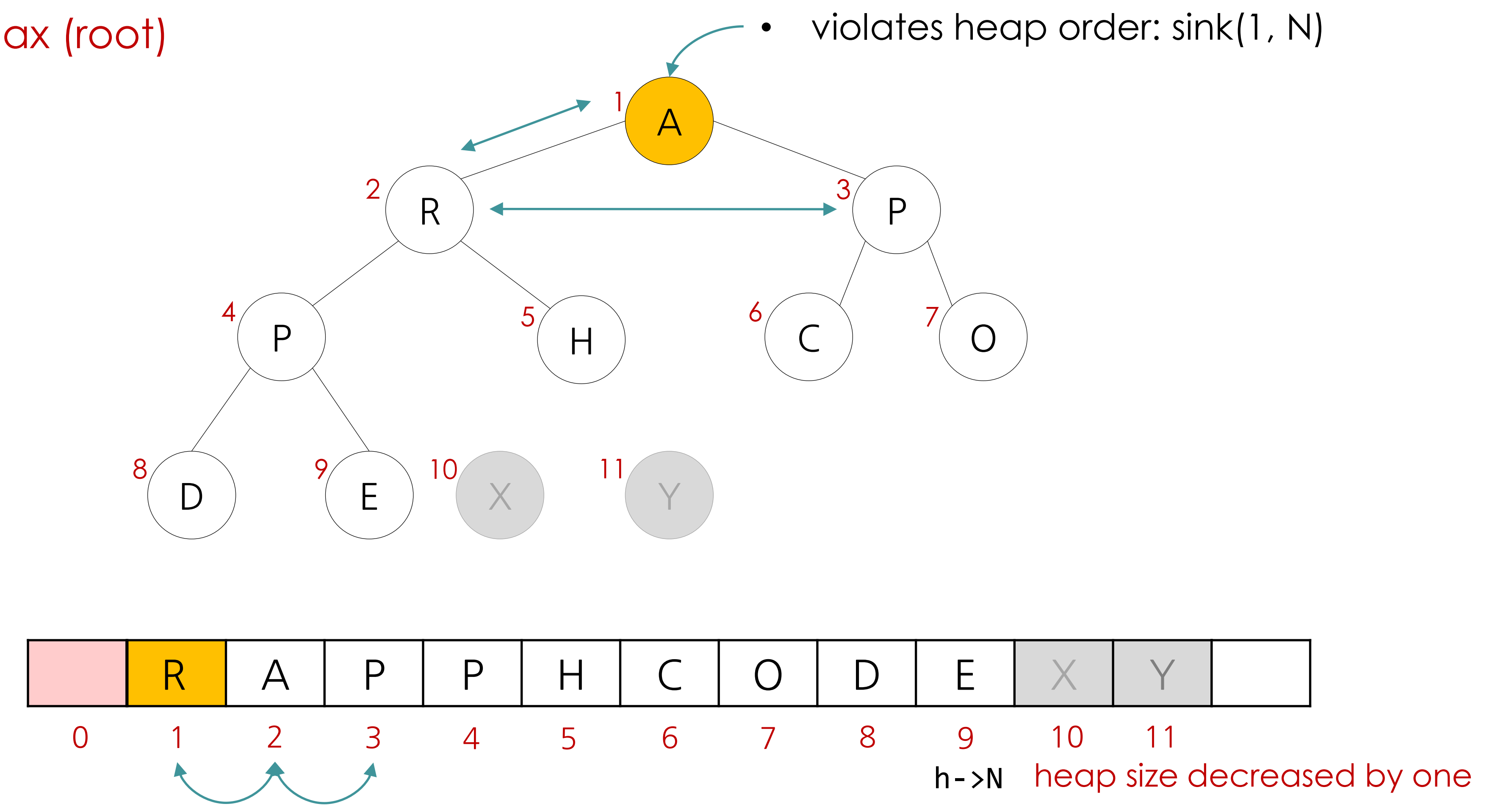
heap-ordered		A	R	P	P	H	C	O	D	E	X	Y	
	0	1	2	3	4	5	6	7	8	9	10	11	

`h->N` heap size decreased by one

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)



- violates heap order: sink(1, N)

heap-ordered

	R	A	P	P	H	C	O	D	E	X	Y	
0	1	2	3	4	5	6	7	8	9	10	11	

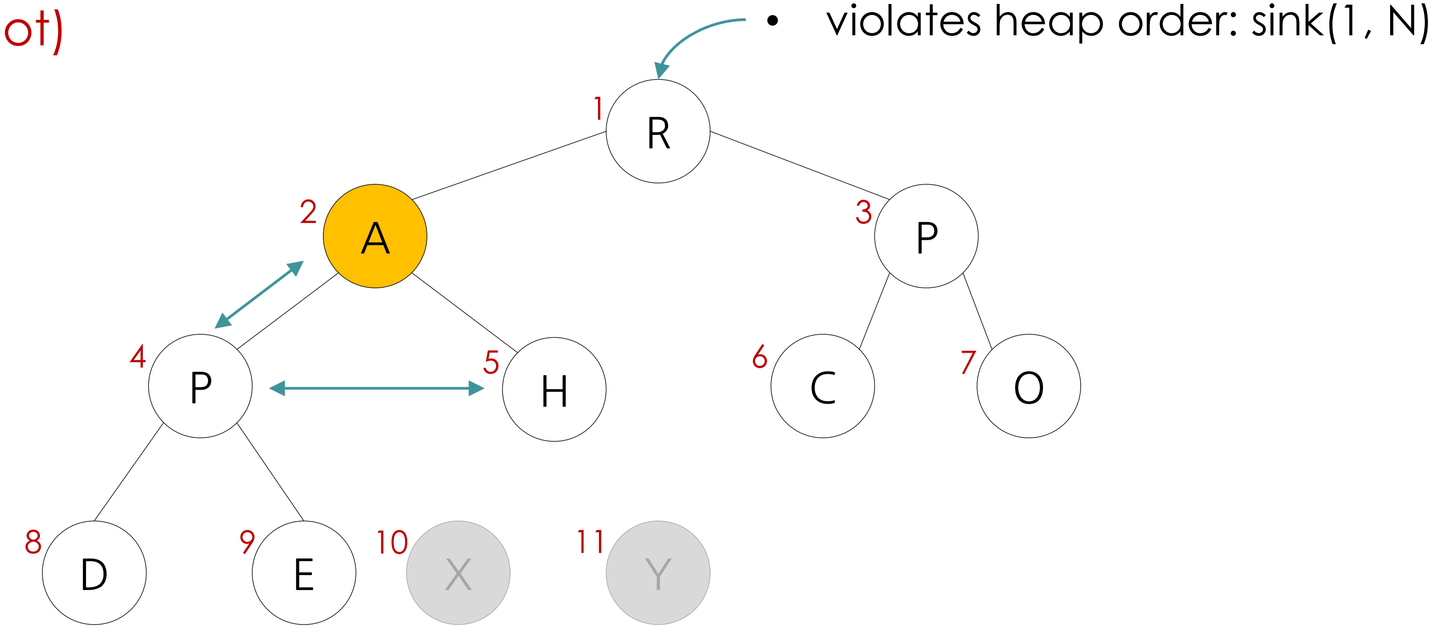
`h->N` heap size decreased by one

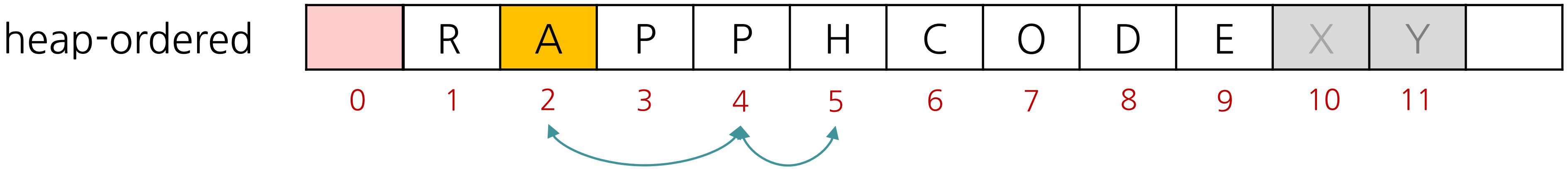
Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204, Handong Global University

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)



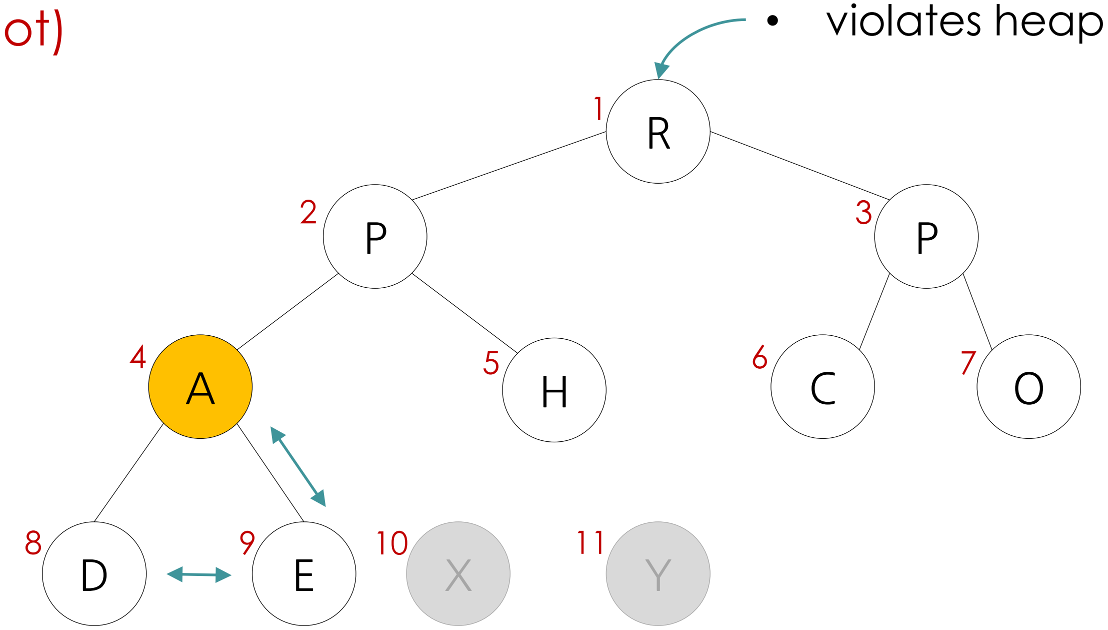


max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)

- violates heap order: sink(1, N)



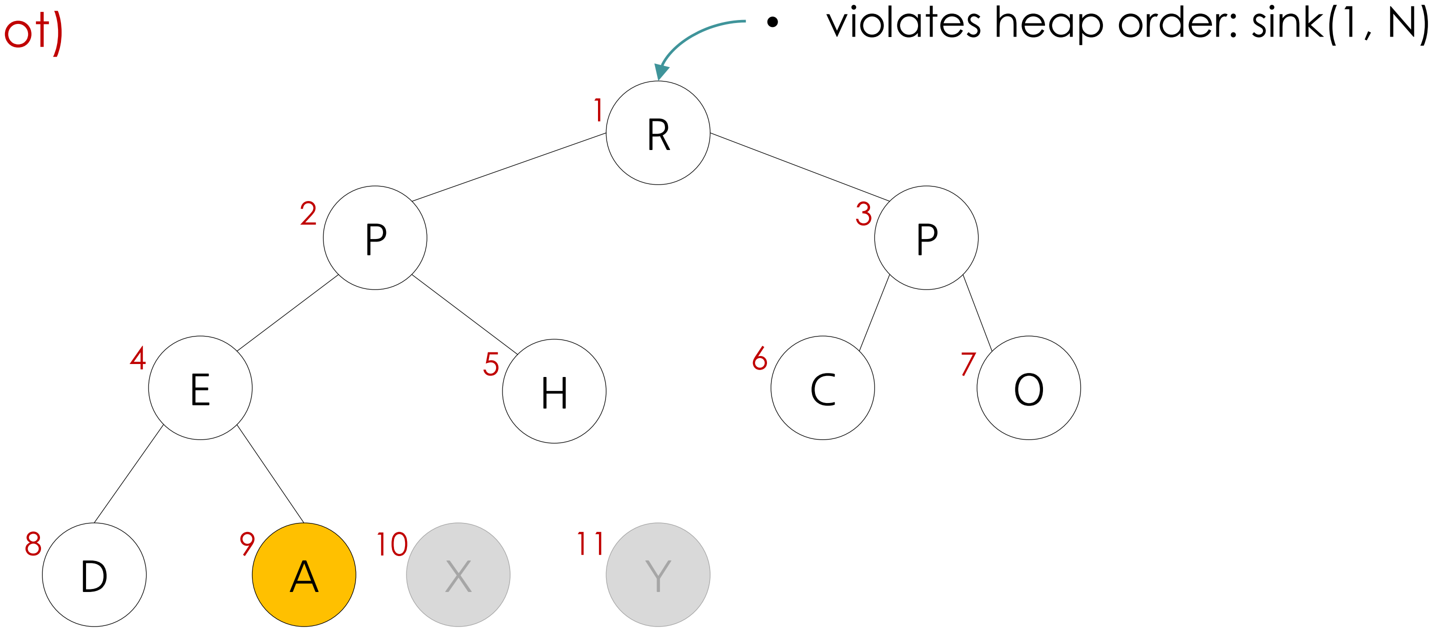
heap-ordered

	R	P	P	A	H	C	O	D	E	X	Y
0	1	2	3	4	5	6	7	8	9	10	11

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)



- violates heap order: sink(1, N)

heap-ordered

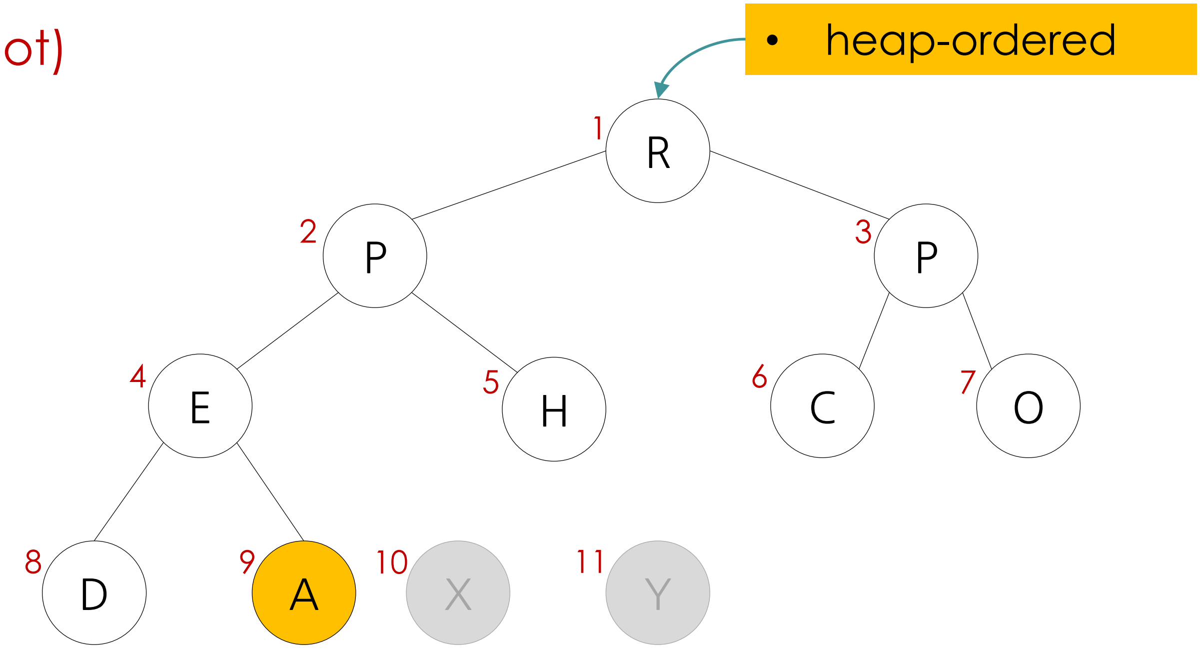
0	1	2	3	4	5	6	7	8	9	10	11	
	R	P	P	E	H	C	O	D	A	X	Y	

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204, Handong Global University

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

remove the max (root)

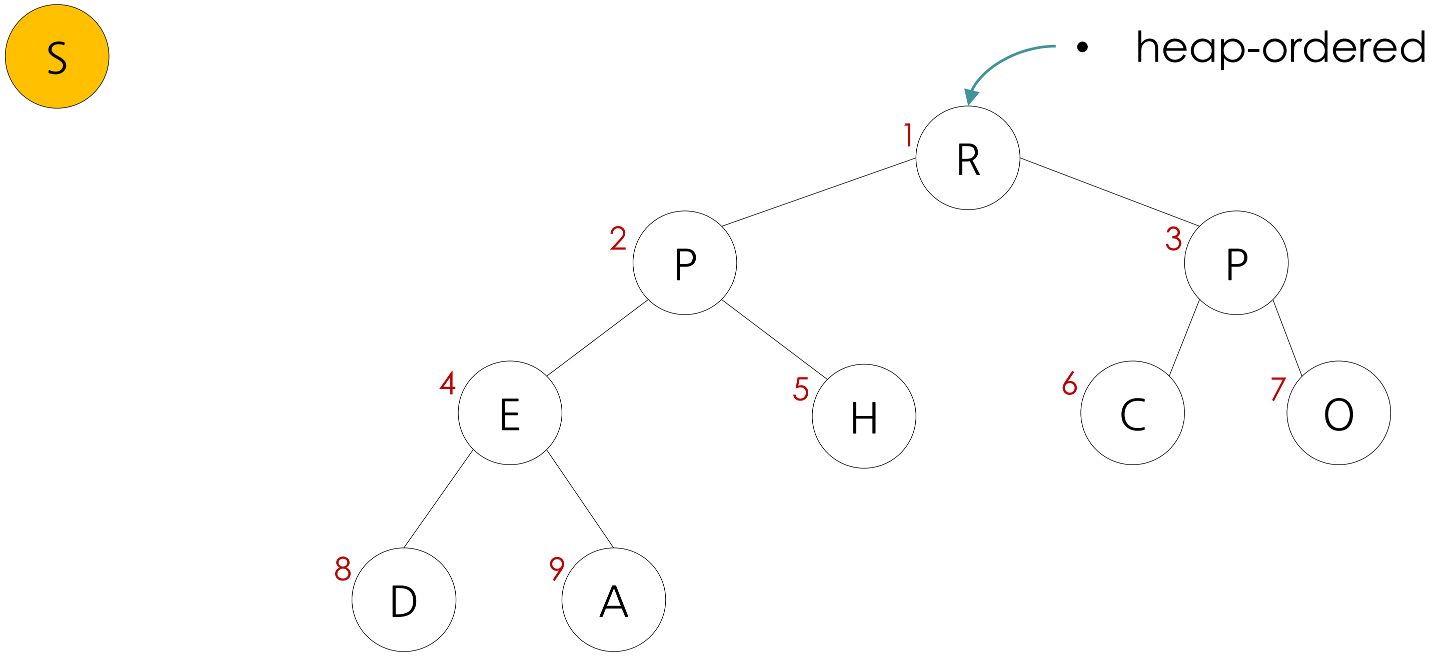

- heap-ordered

heap-ordered		R	P	P	E	H	C	O	D	A	X	Y	
	0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

insert S



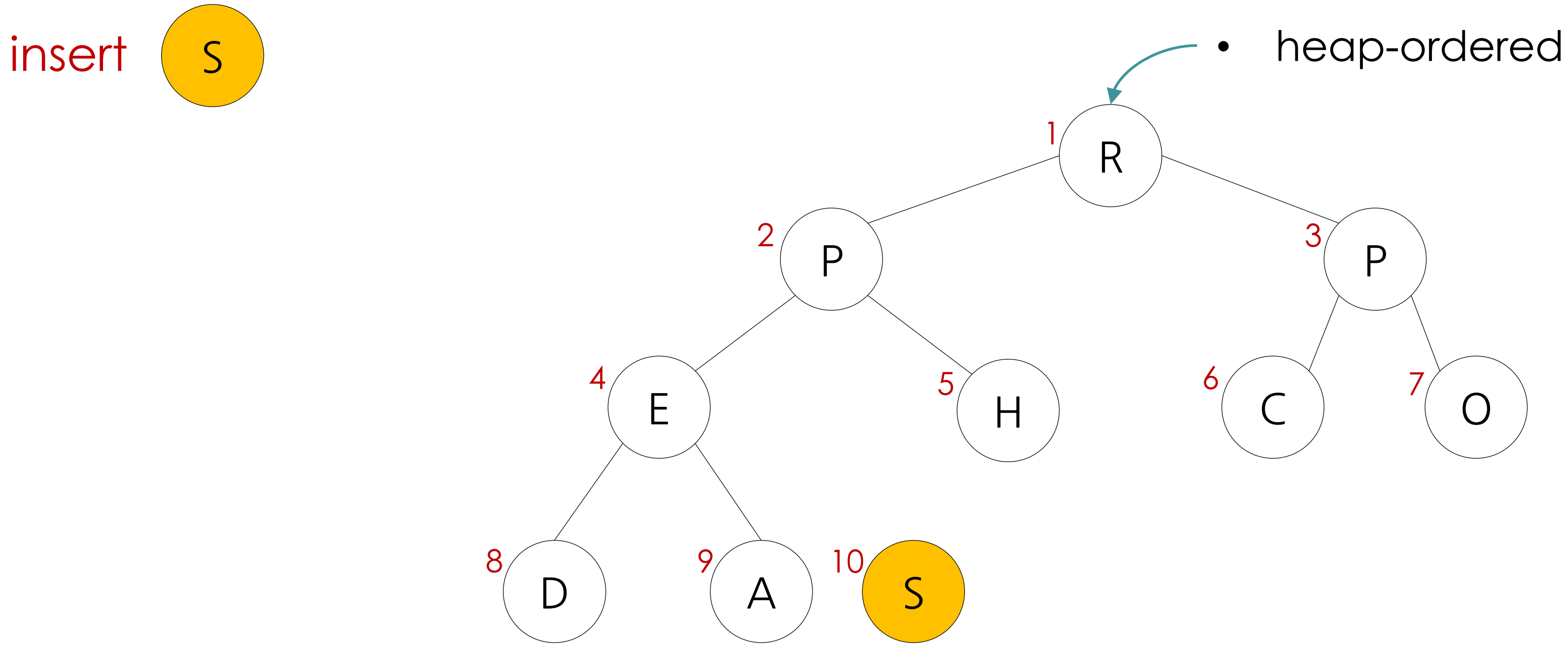
heap-ordered

0	1	2	3	4	5	6	7	8	9	10	11
	R	P	P	E	H	C	O	D	A		

`h->N`

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

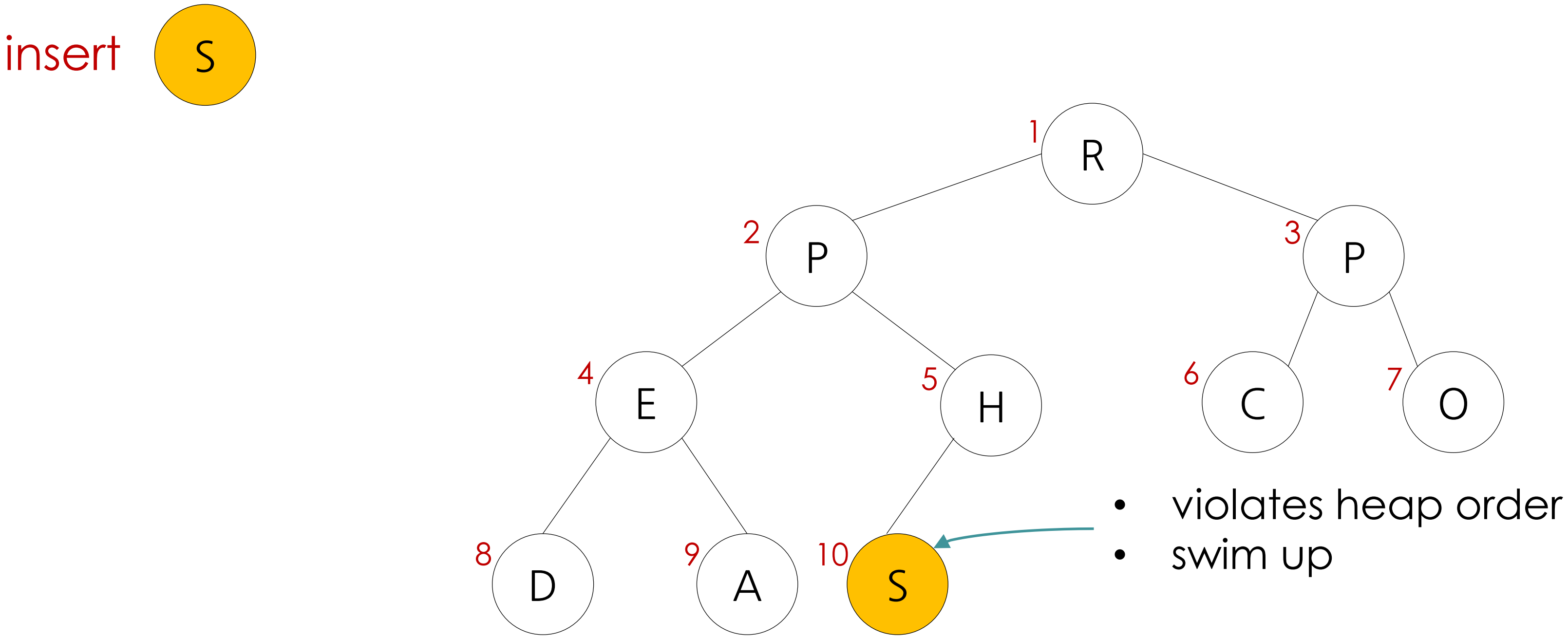


`h->N` `h->nodes[++h->N] = key;`

heap-ordered		R	P	P	E	H	C	O	D	A	S		
	0	1	2	3	4	5	6	7	8	9	10	11

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



- violates heap order
- swim up

h->N

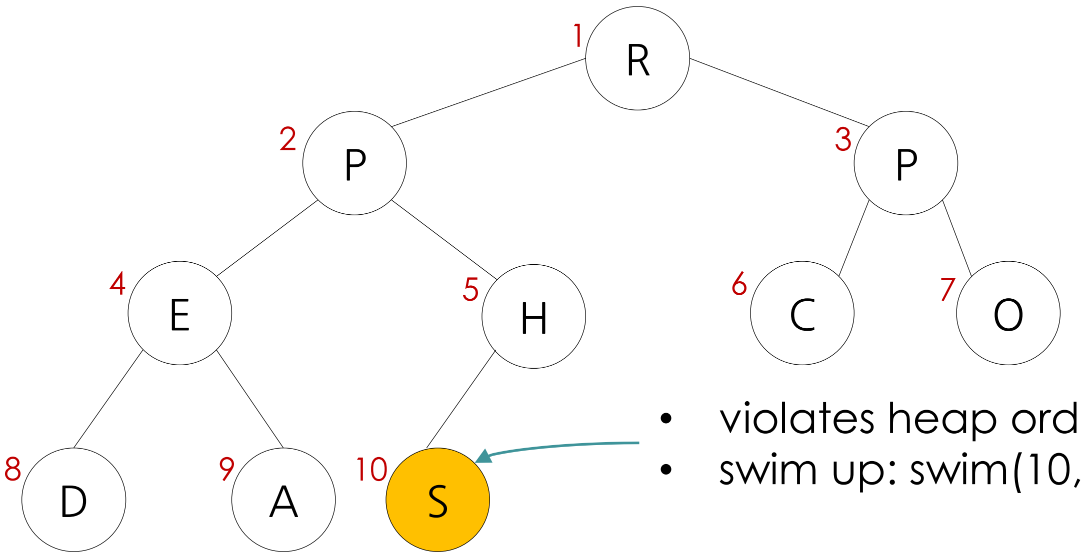
heap-ordered		R	P	P	E	H	C	O	D	A	S		
	0	1	2	3	4	5	6	7	8	9	10	11	

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204, Handong Global University

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

insert S



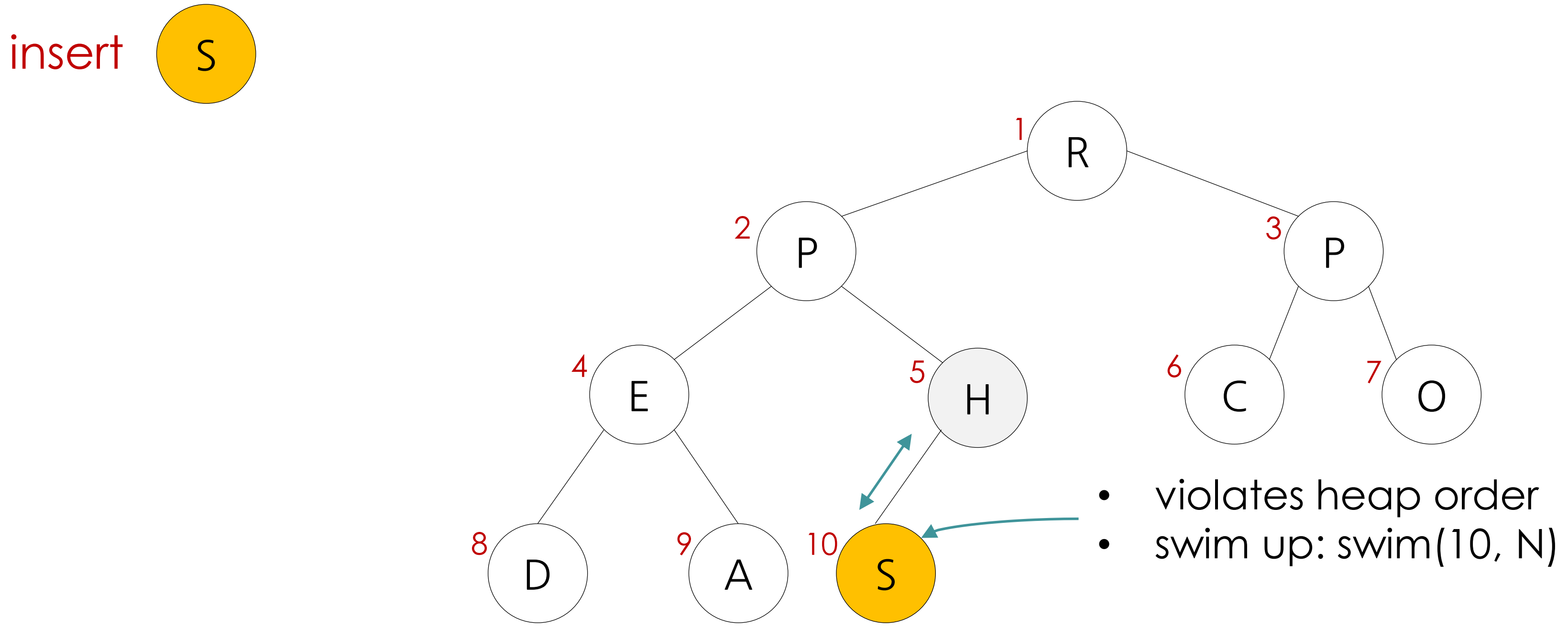
- violates heap order
- swim up: swim(10, N)

heap-ordered

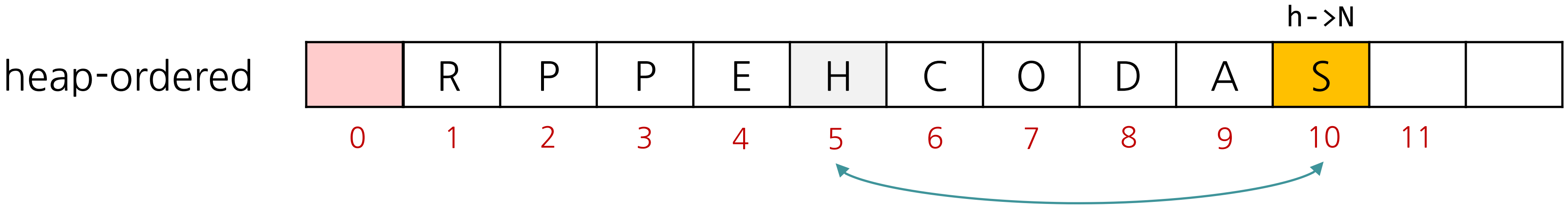
										h->N	
	R	P	P	E	H	C	O	D	A	S	
0	1	2	3	4	5	6	7	8	9	10	11

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



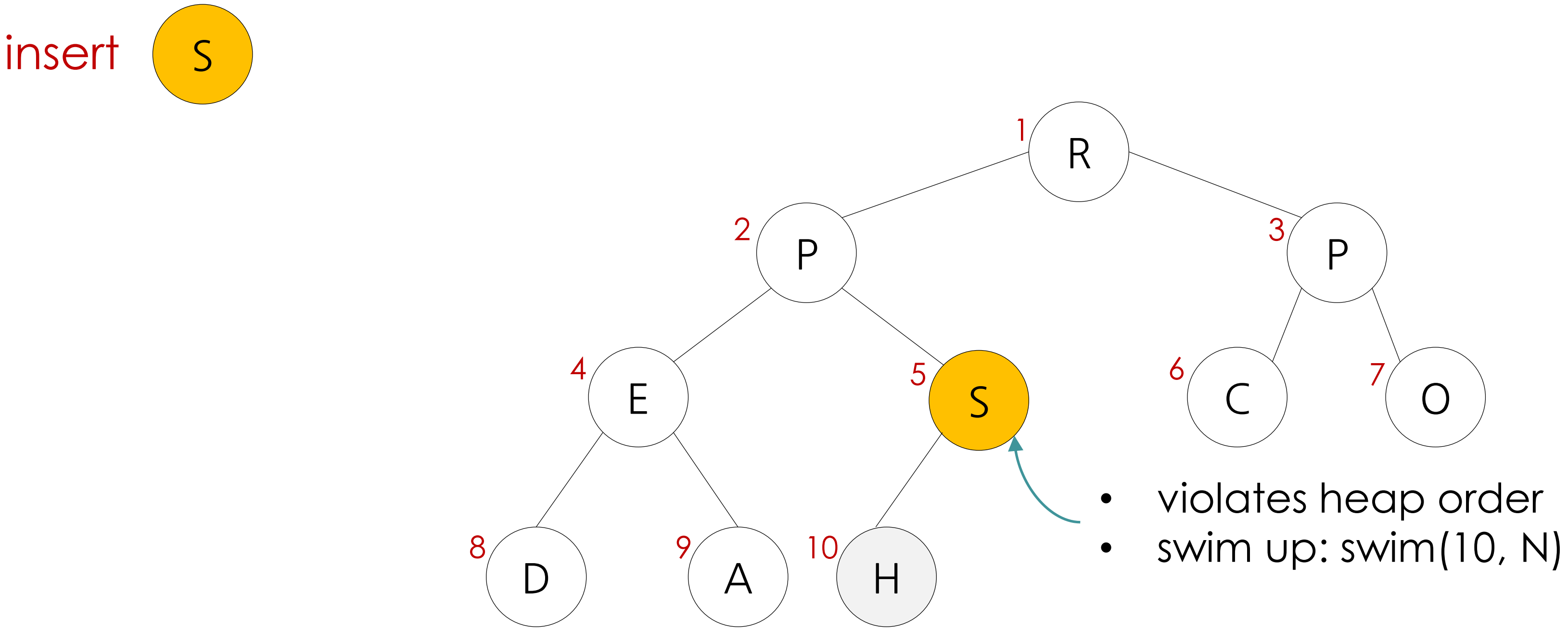
- violates heap order
- swim up: swim(10, N)



	R	P	P	E	H	C	O	D	A	S	
0	1	2	3	4	5	6	7	8	9	10	11

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



- violates heap order
- swim up: swim(10, N)

heap-ordered

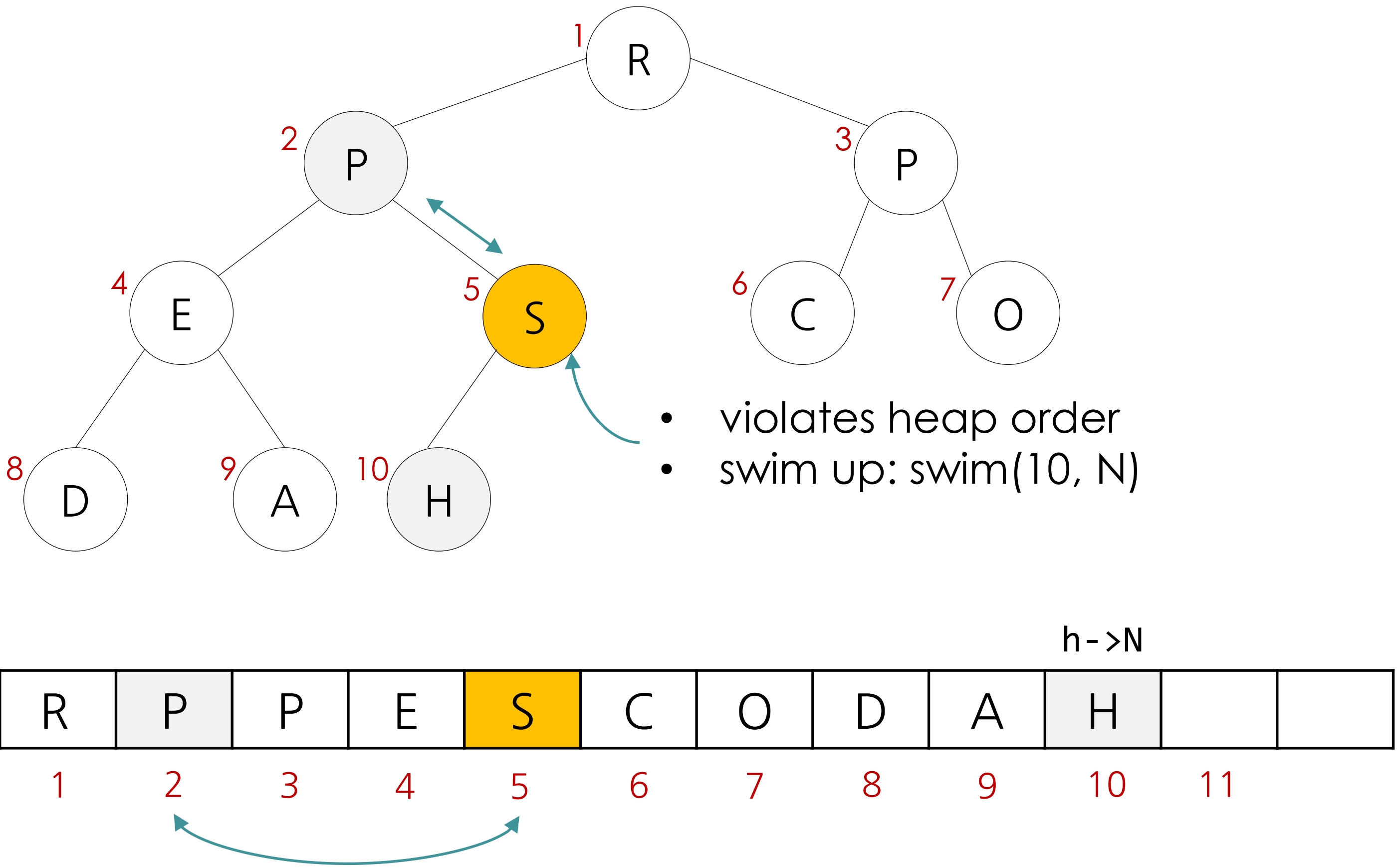
`h->N`

0	1	2	3	4	5	6	7	8	9	10	11	
	R	P	P	E	S	C	O	D	A	H		

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

insert S



- violates heap order
- swim up: swim(10, N)

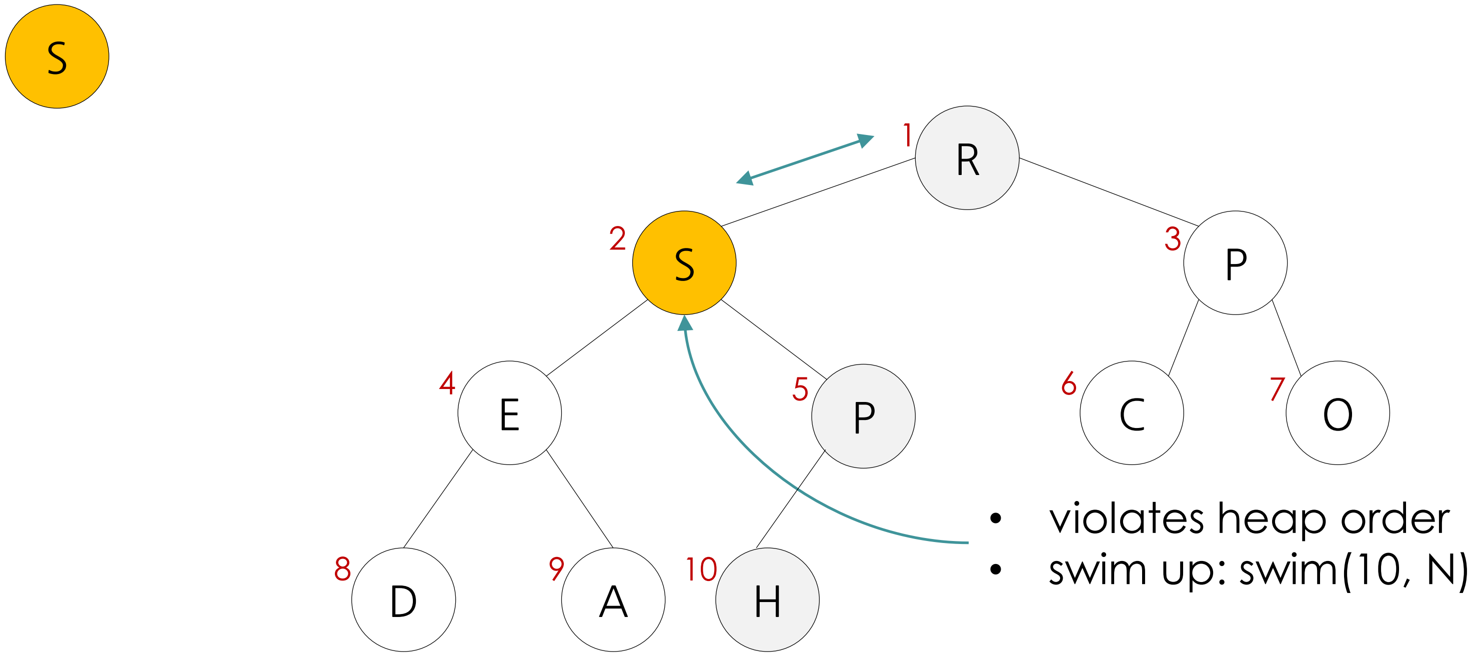
heap-ordered

										h->N		
	R	P	P	E	S	C	O	D	A	H		
0	1	2	3	4	5	6	7	8	9	10	11	

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**

insert



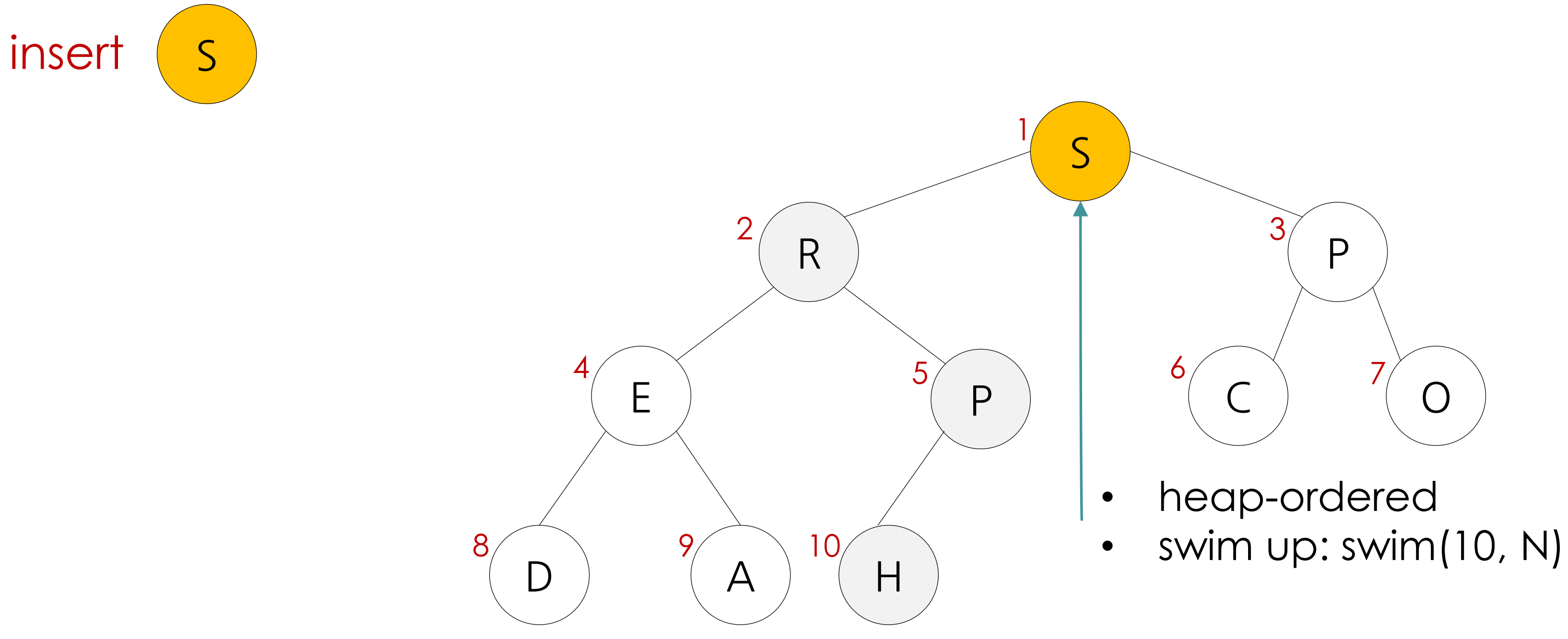
- violates heap order
- swim up: swim(10, N)

heap-ordered

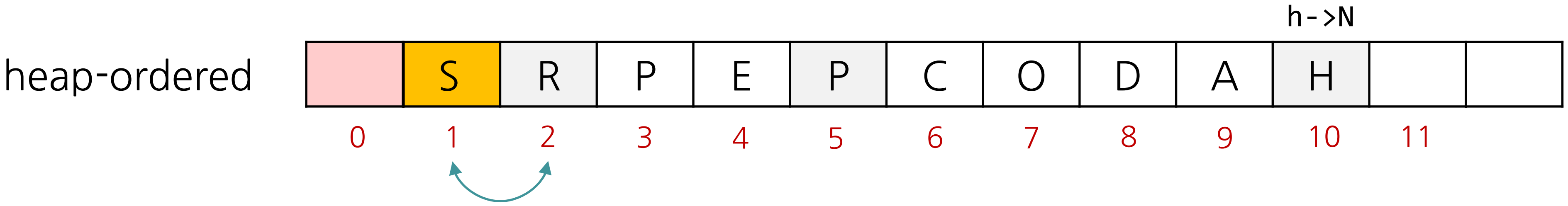
										h->N		
		R	S	P	E	P	C	O	D	A	H		
	0	1	2	3	4	5	6	7	8	9	10	11

max-heap example

- **Insert**: Add node at end, then swim it up.
- **Remove: Swap root with last node, then sink down.**



- heap-ordered
- swim up: swim(10, N)



0	1	2	3	4	5	6	7	8	9	10	11
	S	R	P	E	P	C	O	D	A	H	

Binary heap operations time complexity with N items:

- Level of heap is $\lfloor \log_2 N \rfloor$
- insert: O(log N) for each insert
  - In practice, expect less
- delete: O(log N) // deleting root node or any node
- increase/decrease key: O(log N)
- **Heapify():** O(N)
- **Heapsort():** O(N log N)
- Because O(N) heapify + O(log N) delete = O(N log N)

- **Proof:**
  - https://stackoverflow.com/questions/9755721/how-can-building-a-heap-be-on-time-complexity
  - https://www.insertingwiththeweb.com/data-structures/binary-heap/build-heap-proof/
  - https://www.quora.com/How-is-the-time-complexity-of-building-a-heap-is-o-n
- **References in Korean:**
  - https://ratsgo.github.io/data%20structure&algorithm/2017/09/27/heapsort/
  - https://zeddios.tistory.com/56

Data Structures in Python

- **Heap and Priority Queue**
- Heap Coding
- Heap Sort & Min/MaxHeap