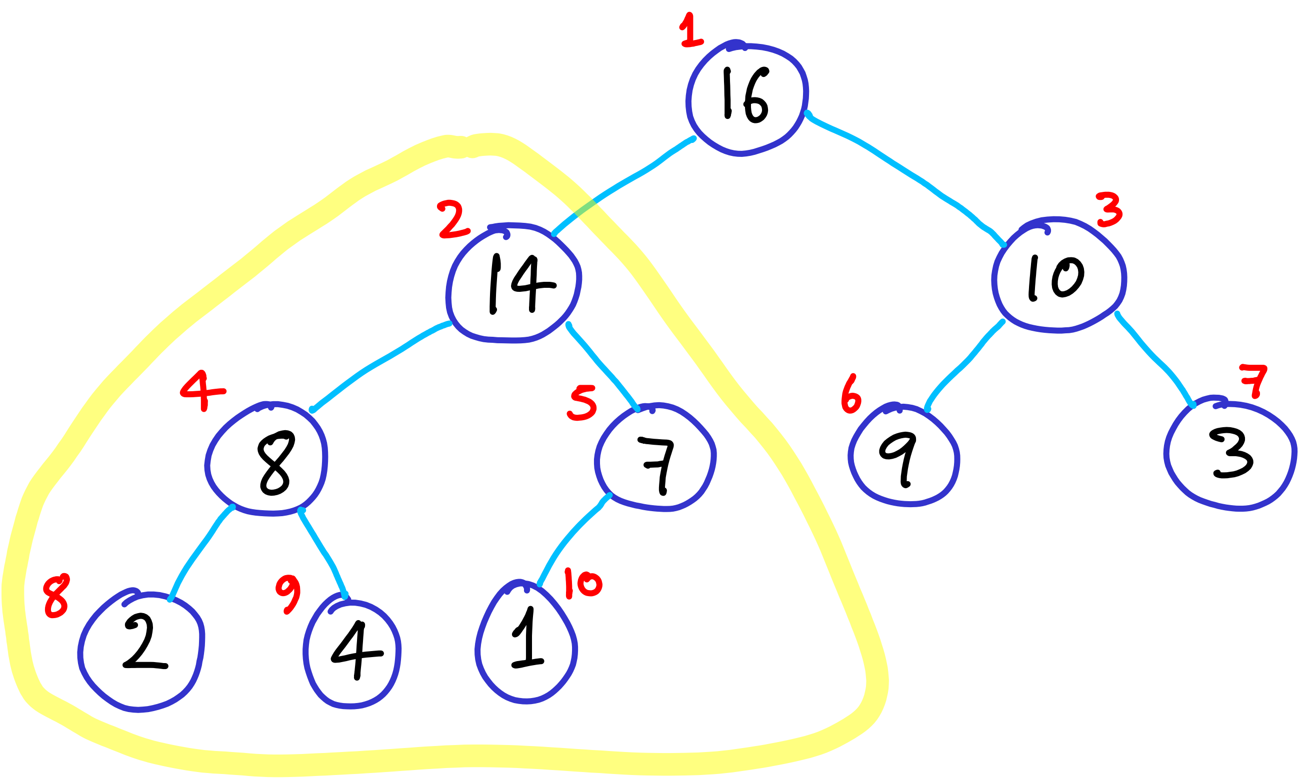
HEAPS and HEAP-SORT

specifically binary MAX-heaps

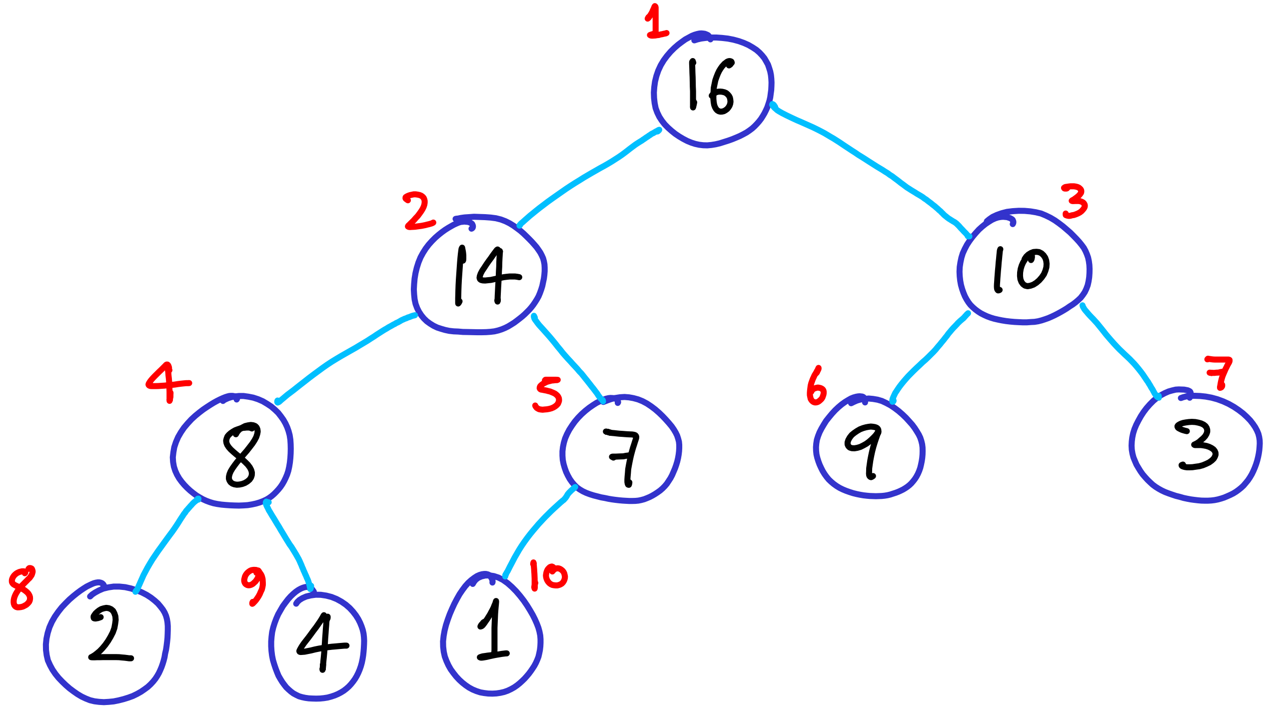


[Notice every subtree is also a heap]

Rules:

- **binary**: internal nodes have 1 or 2 children
- **max**: parent ≥ child
- **complete**: all levels filled (lowest can be partial, left to right)

some applications don't need this but we will enforce it



How can we identify the indices of the children of a given node?

Use array to store heap

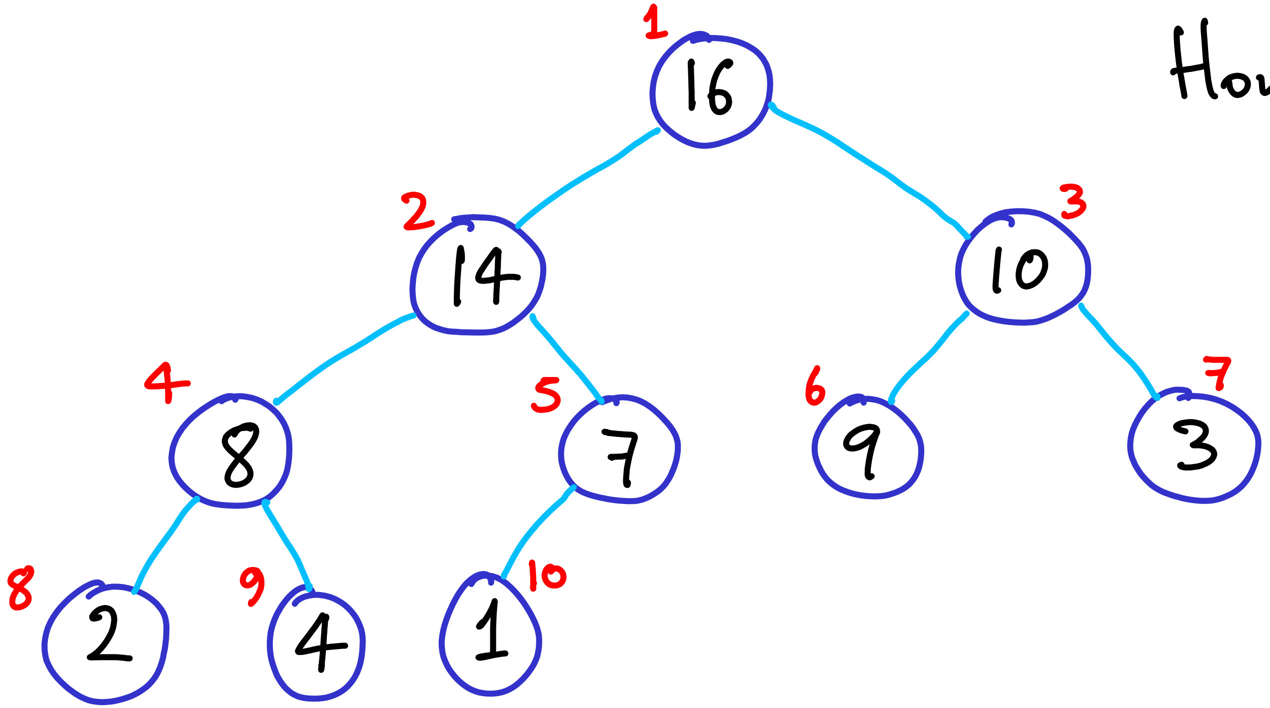
(avoid wasting space with pointers)

1	2	3	4	5	6	7	8	9	10	
16	14	10	8	7	9	3	2	4	1	. . . . .

$\text{left-child}(i) = 2i$

$\text{right-child}(i) = 2i+1$

$\text{parent}(i) = \lfloor i/2 \rfloor$



How does this relate to sorting?

Largest element is on top.

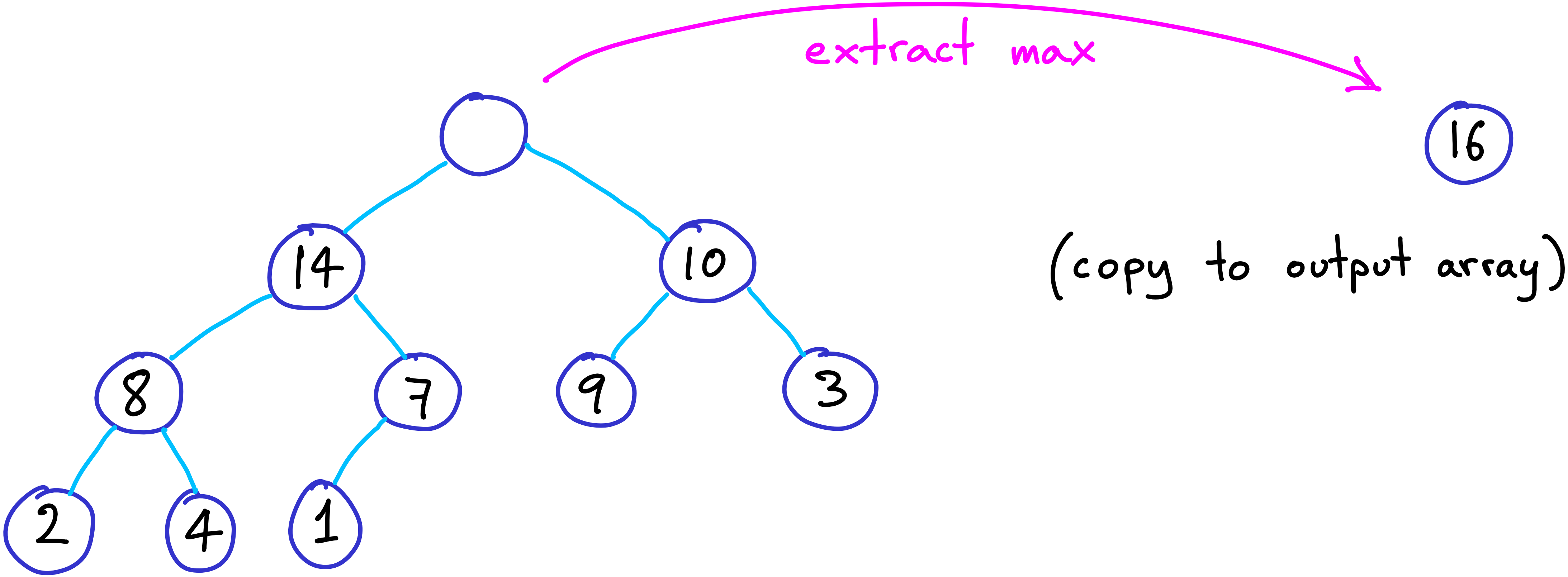
2nd largest is in level 2.

3rd largest is
↳ in level 2
OR
↳ in level 3
& child of 2nd

⋮
getting messy

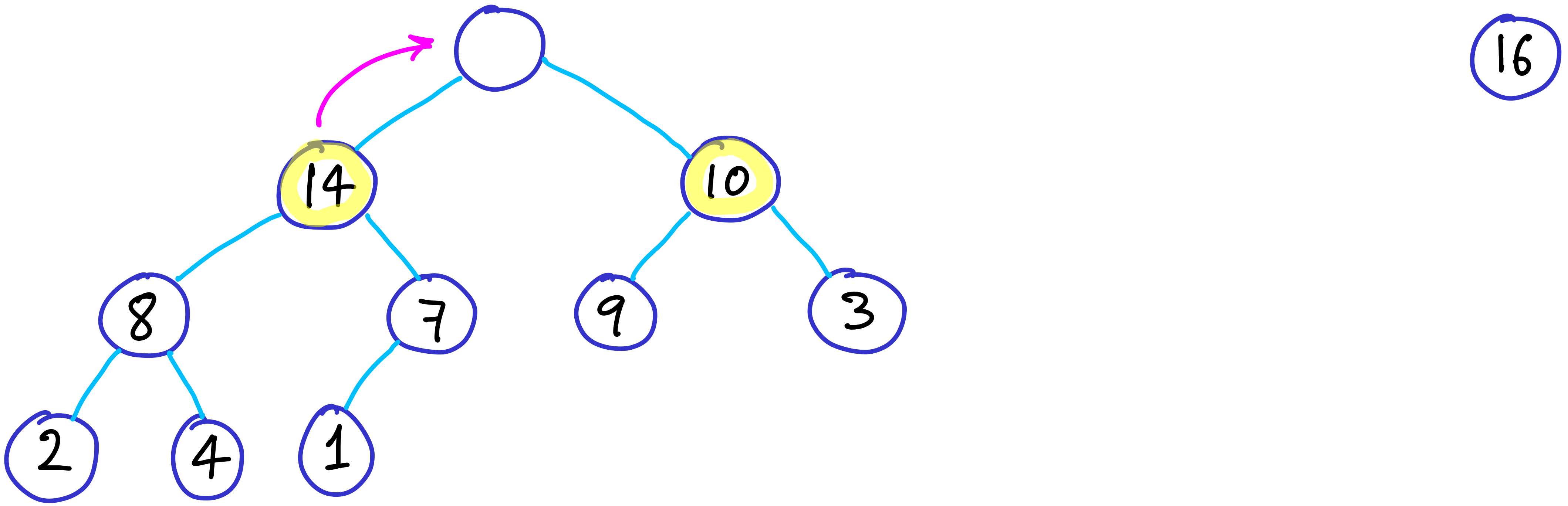
Heaps are not "sorted"

How to sort data in a heap



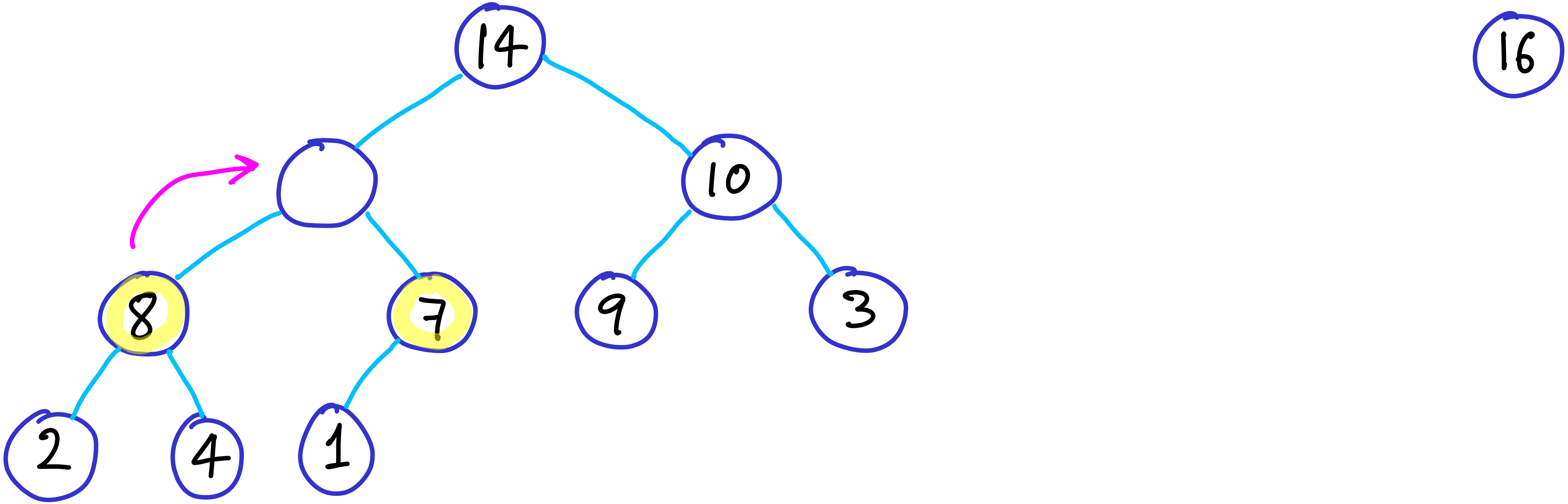
How to sort data in a heap

Update max : larger of 2 children

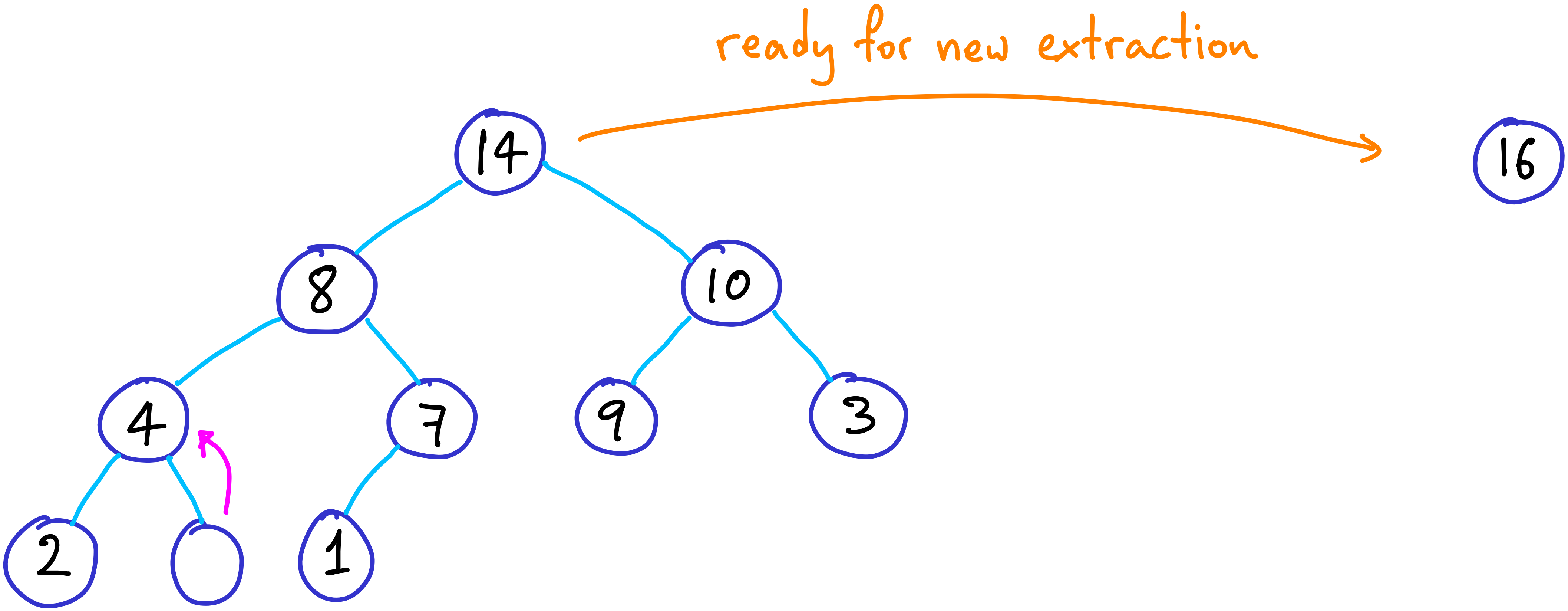


How to sort data in a heap

Update max recursively

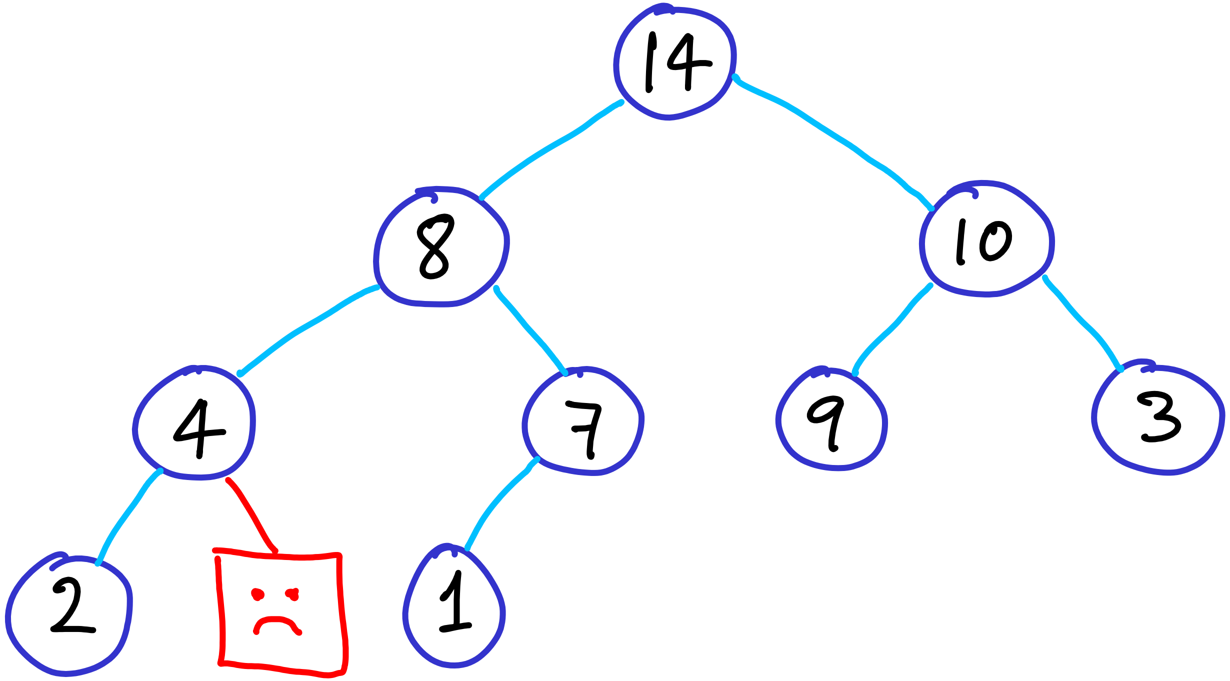


How to sort data in a heap



How to sort data in a heap

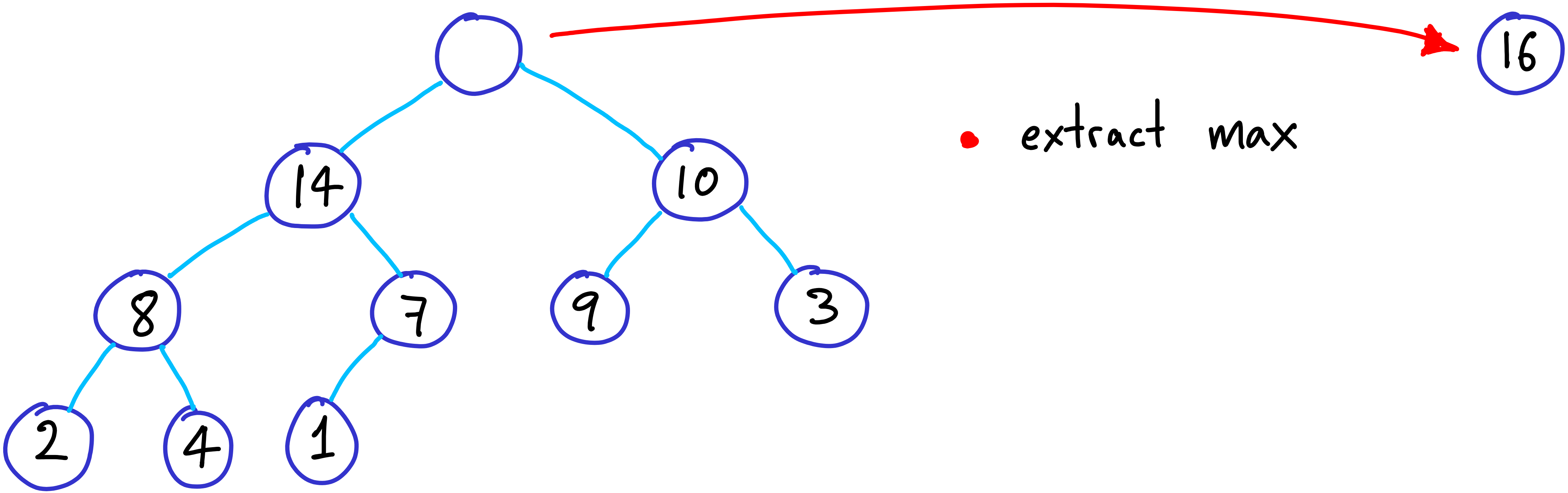
↳ if we don't care about
- keeping the heap complete
- using extra space



(output array)

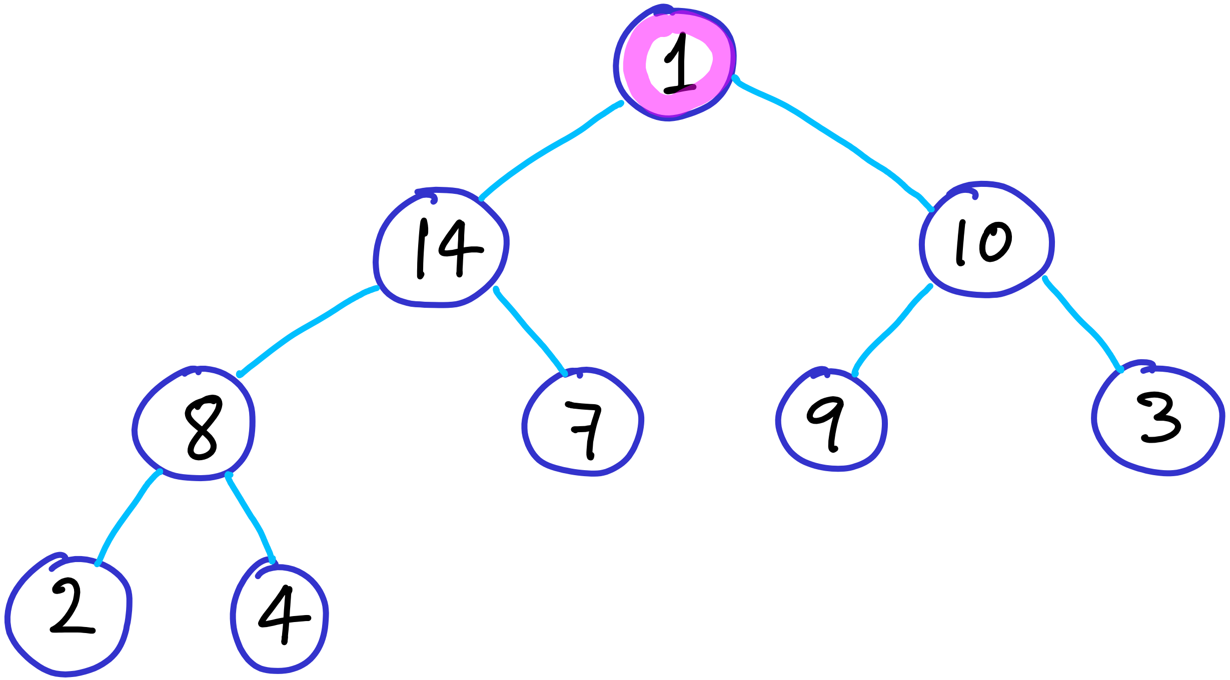
How to sort data in a complete heap ... using extra space

How to sort data in a complete heap ... using extra space



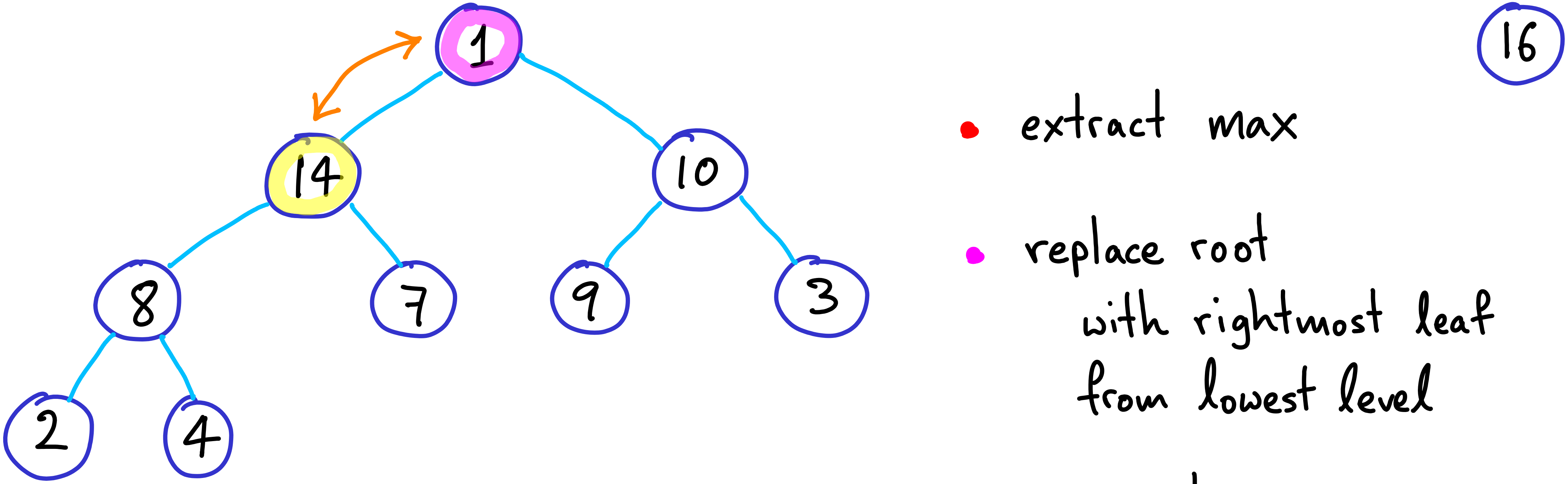
- extract max

How to sort data in a complete heap ... using extra space



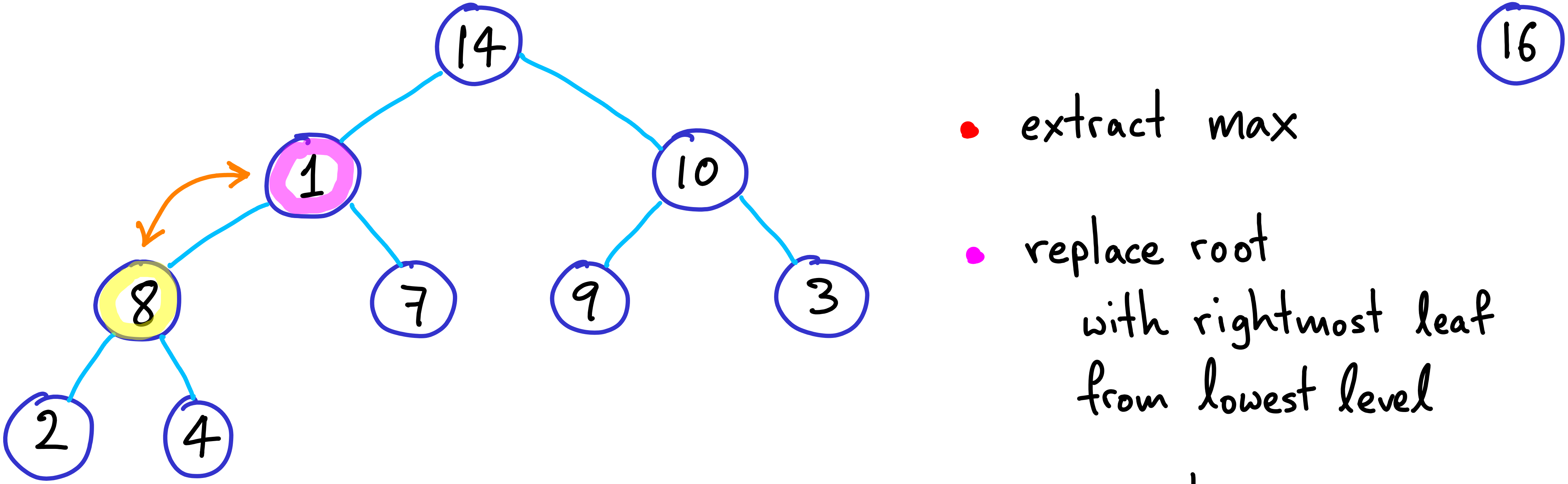
- extract max
- replace root with rightmost leaf from lowest level

How to sort data in a complete heap ... using extra space



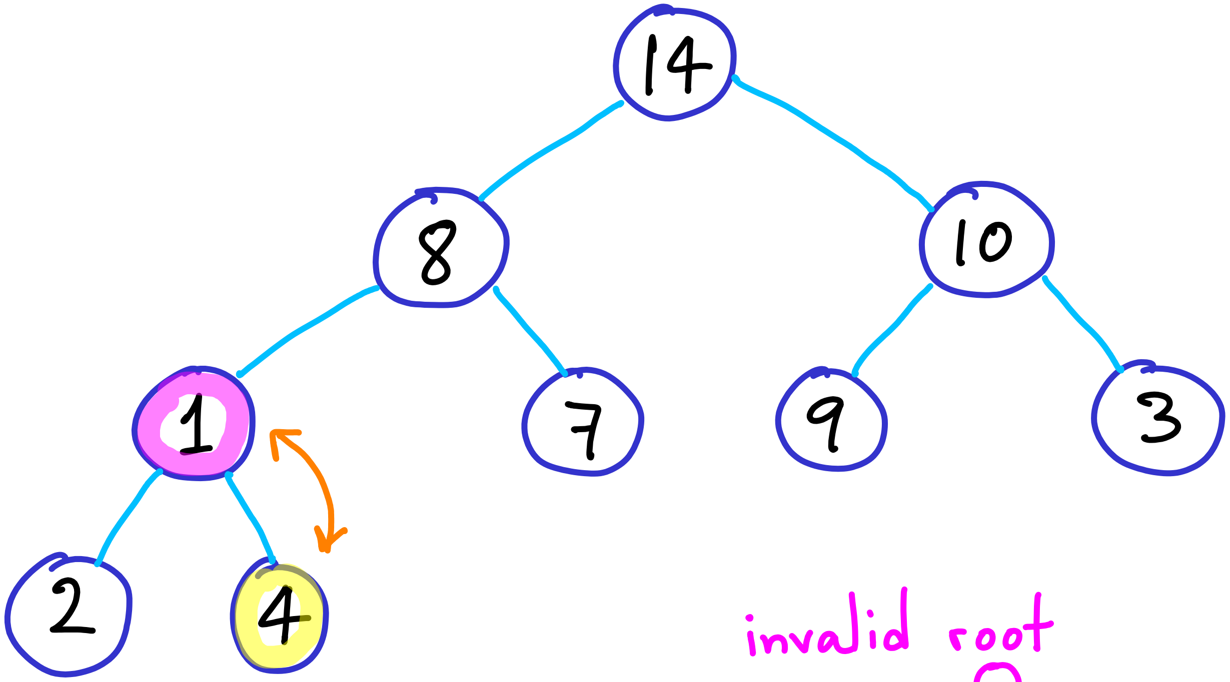
- extract max
- replace root with rightmost leaf from lowest level
- recursively swap with largest child while heap not restored

How to sort data in a complete heap ... using extra space



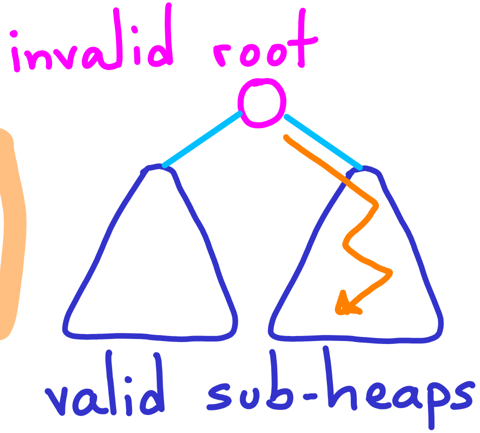
- extract max
- replace root with rightmost leaf from lowest level
- recursively swap with largest child while heap not restored

How to sort data in a complete heap ... using extra space

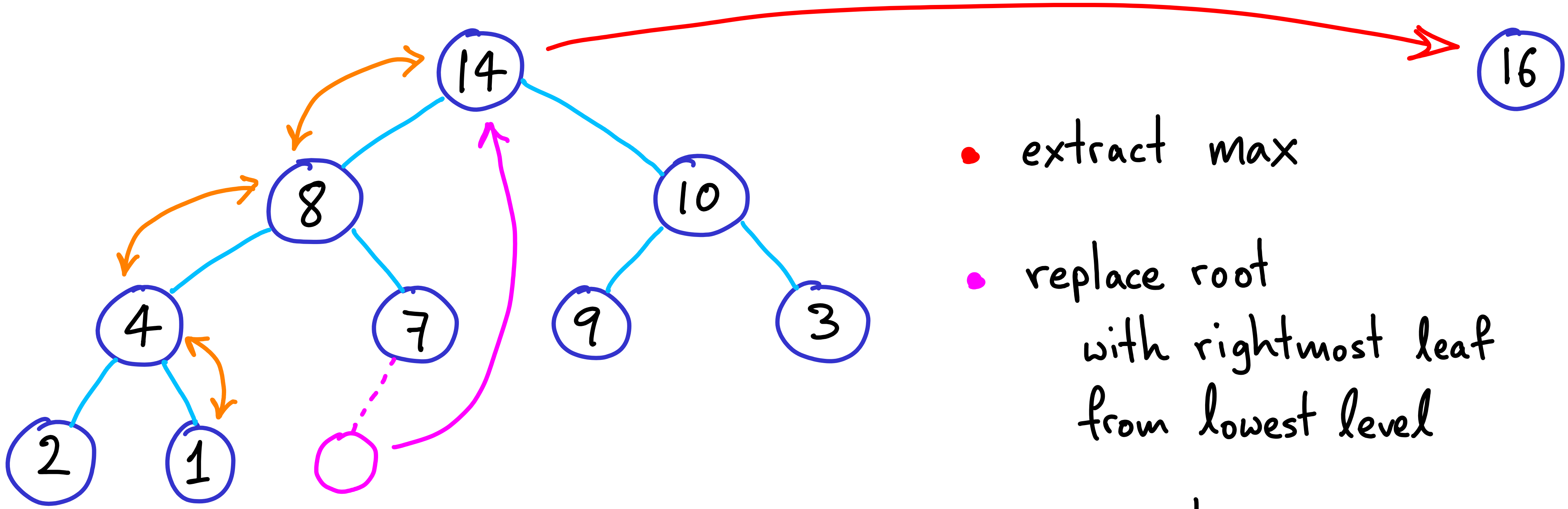


- extract max
- replace root with rightmost leaf from lowest level
- recursively swap with largest child while heap not restored

"heapify"



How to sort data in a complete heap ... using extra space



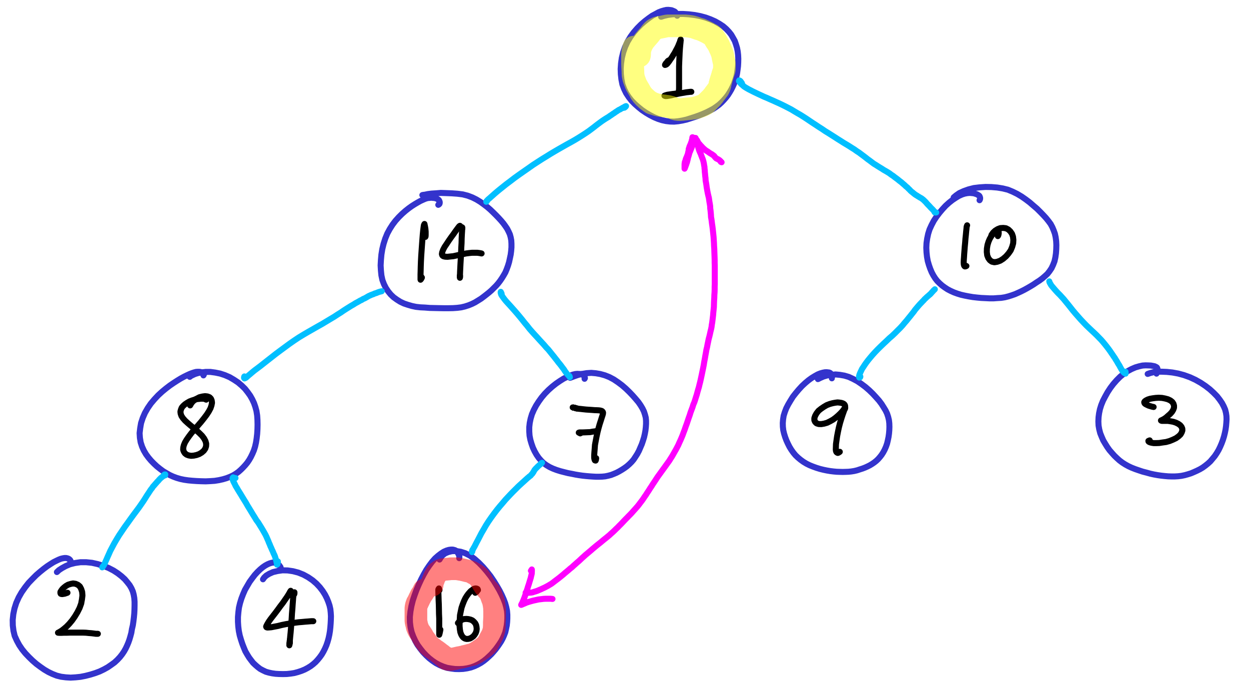
- extract max
- replace root with rightmost leaf from lowest level
- recursively swap with largest child while heap not restored

time = $O(n\log n)$

$O(\log n)$ per extraction

How to sort data in a complete heap in place

(without an output array)



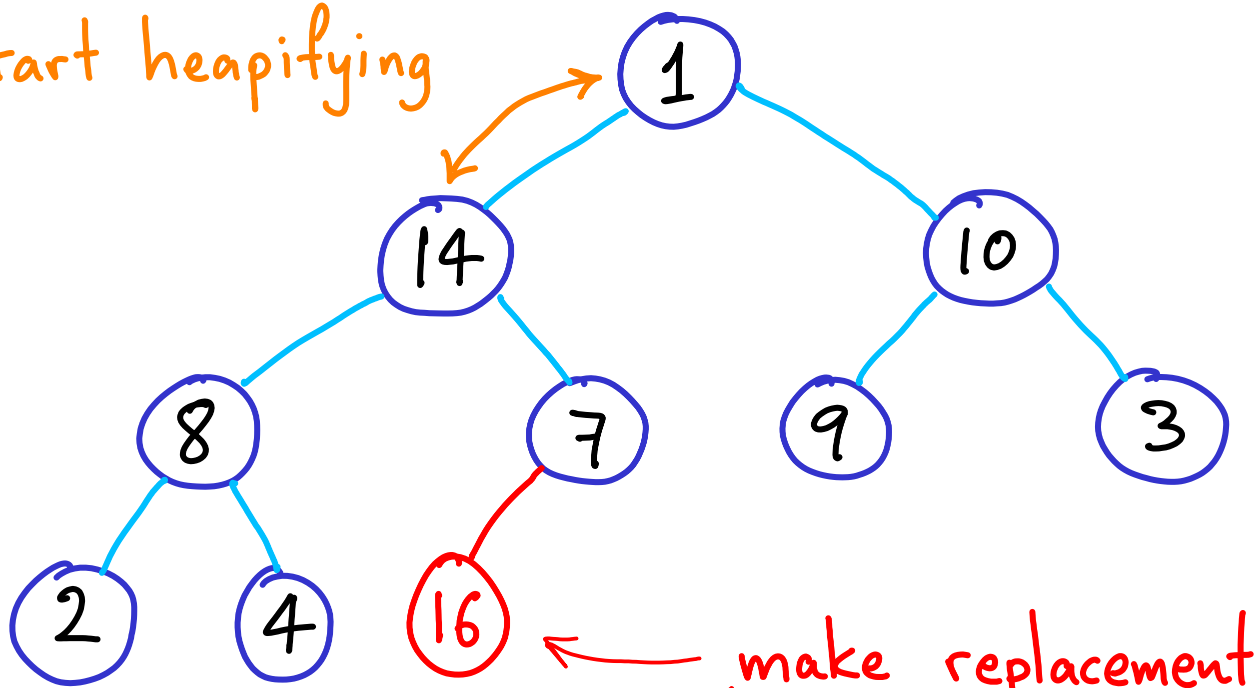
Same as before
but we swap
max with replacement

1	2	3	4	5	6	7	8	9	10
1	14	10	8	7	9	3	2	4	16

How to sort data in a complete heap in place

(without an output array)

start heapifying



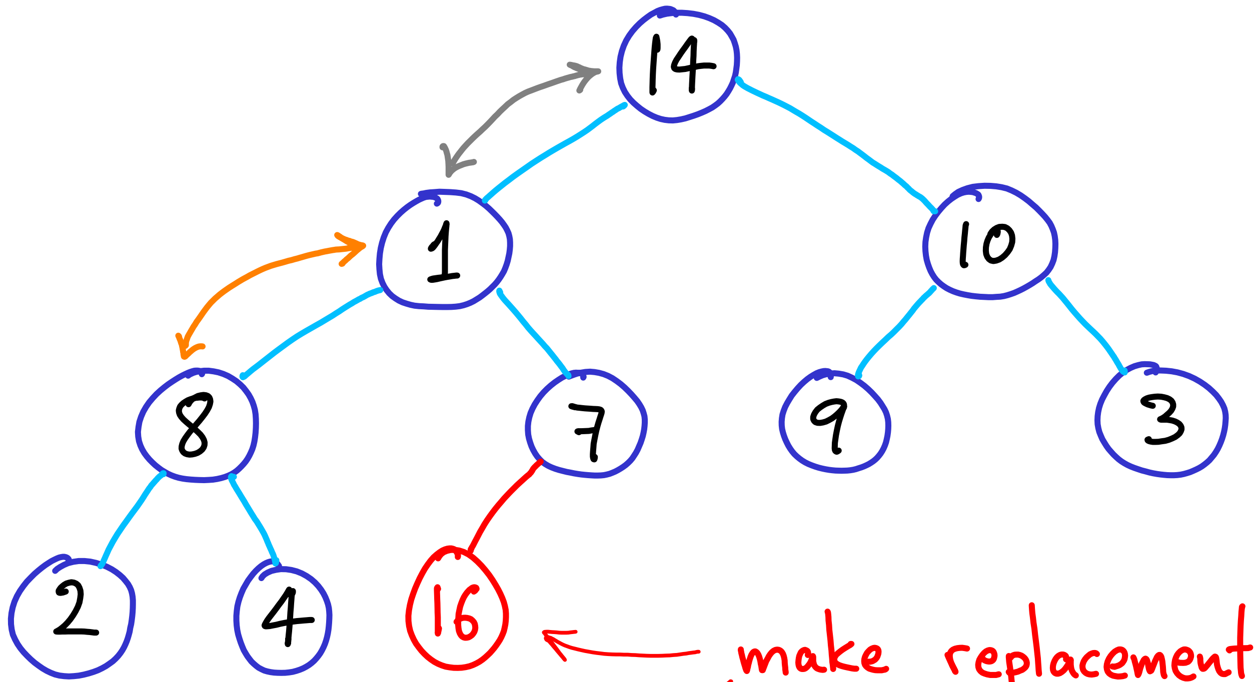
Same as before
but we swap
max with replacement

make replacement position inactive
as though extracted

1	2	3	4	5	6	7	8	9	~~10~~
1	14	10	8	7	9	3	2	4	16

How to sort data in a complete heap in place

(without an output array)



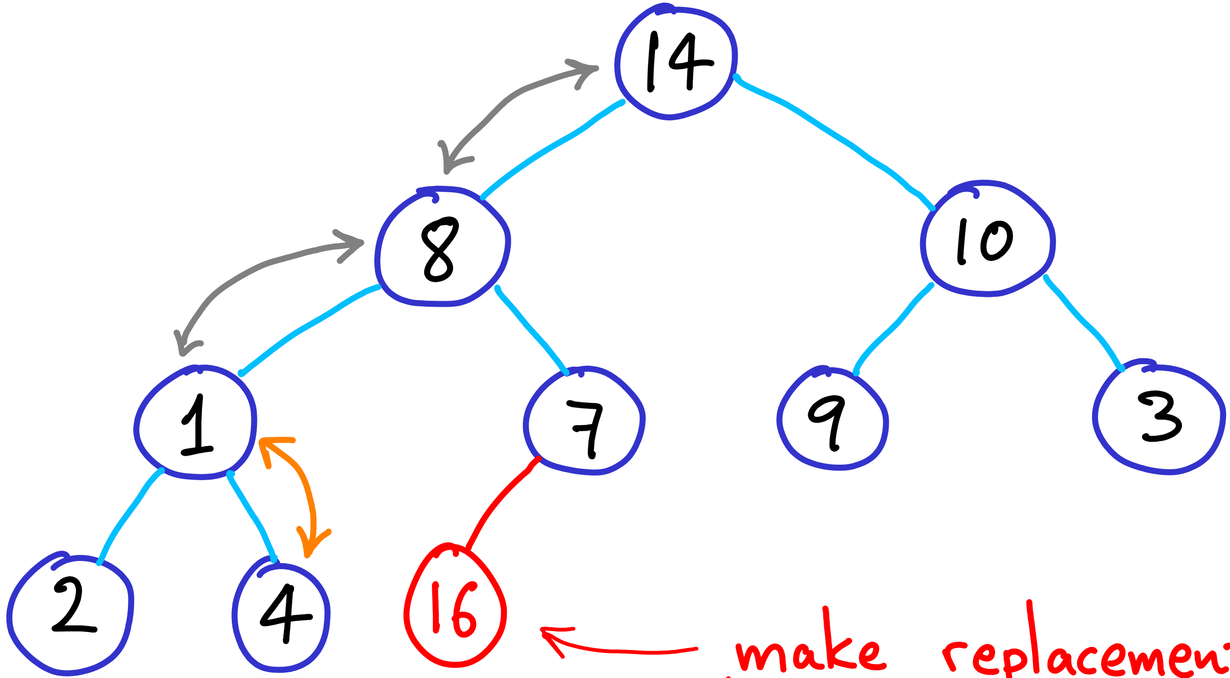
Same as before but we swap max with replacement

make replacement position inactive as though extracted

1	2	3	4	5	6	7	8	9	~~10~~
14	1	10	8	7	9	3	2	4	16

How to sort data in a complete heap in place

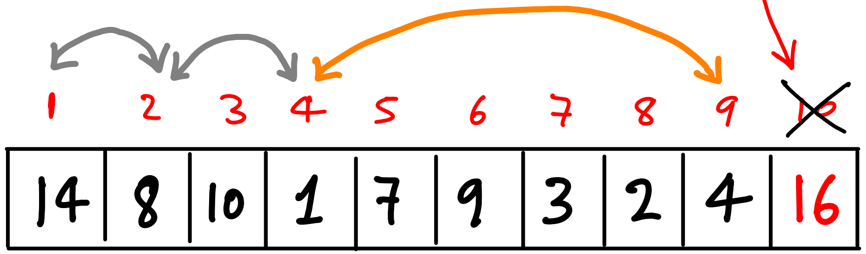
(without an output array)



Same as before but we swap max with replacement

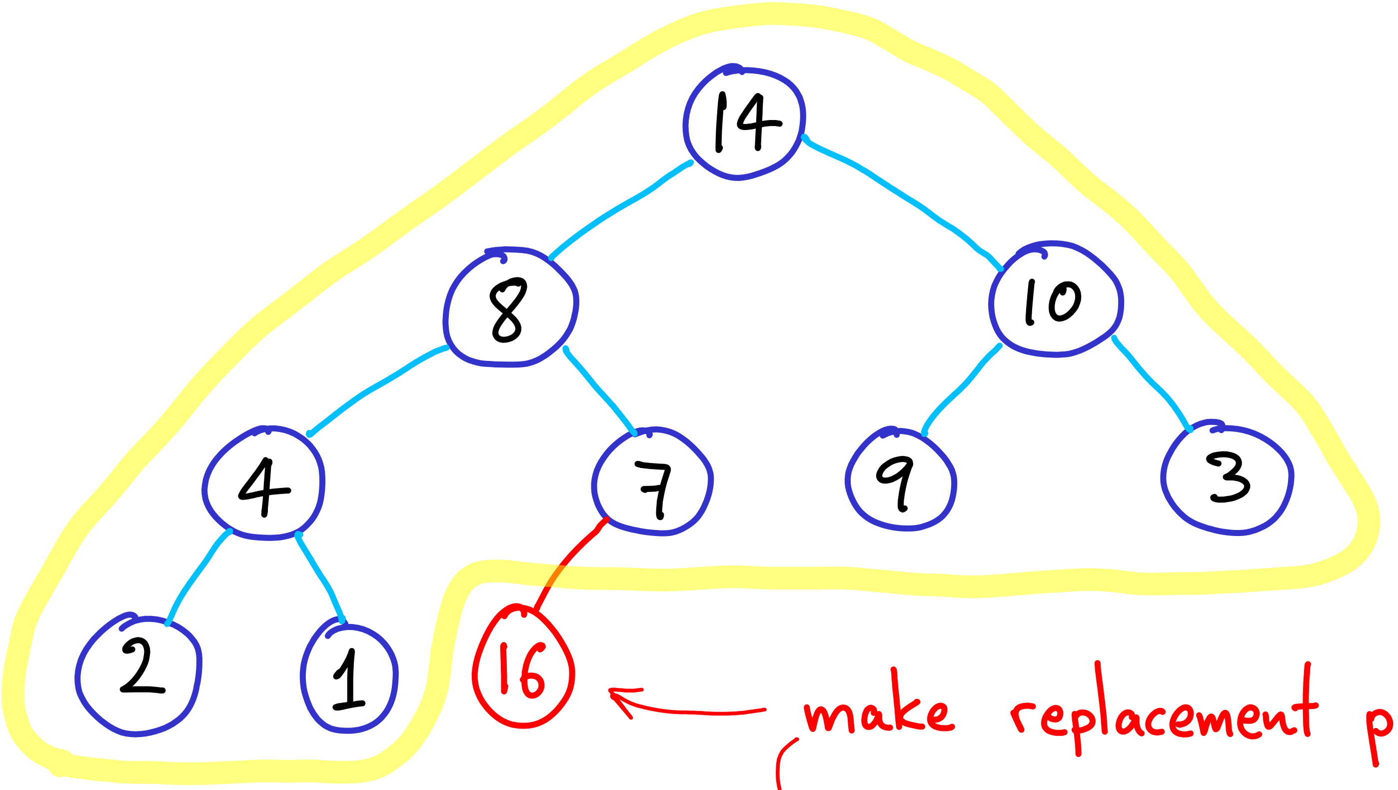
make replacement position inactive

as though extracted



How to sort data in a complete heap in place

(without an output array)



Same as before but we swap max with replacement

make replacement position inactive

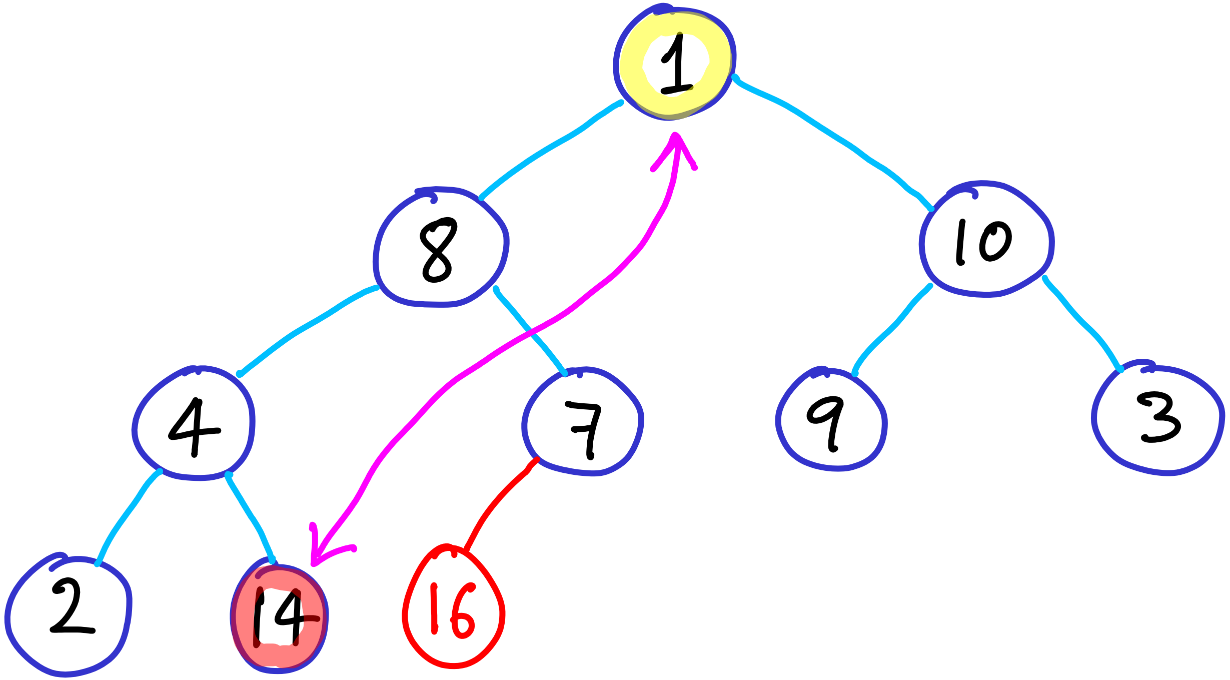
as though extracted

1	2	3	4	5	6	7	8	9	~~10~~
14	8	10	4	7	9	3	2	1	16

valid heap

How to sort data in a complete heap in place

(without an output array)



Same as before

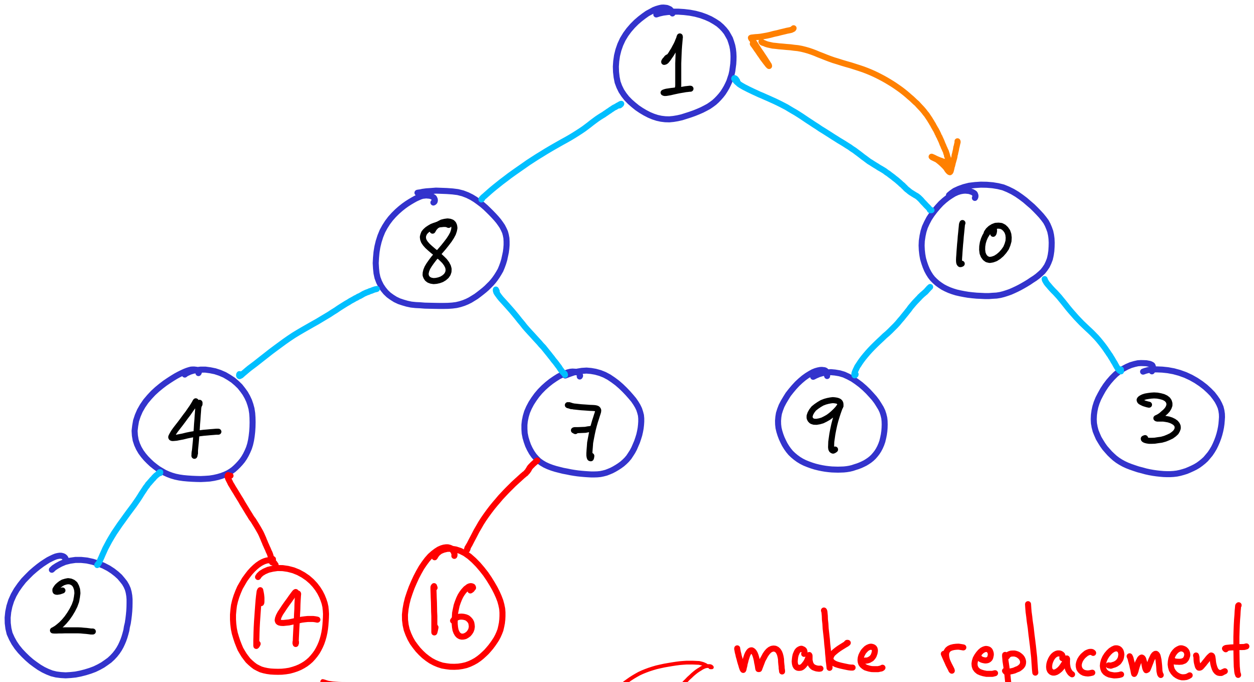
but we swap

max with replacement

1	2	3	4	5	6	7	8	9	~~10~~
1	8	10	4	7	9	3	2	14	16

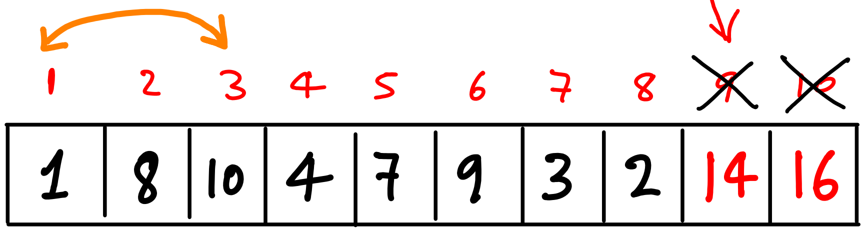
How to sort data in a complete heap in place

(without an output array)



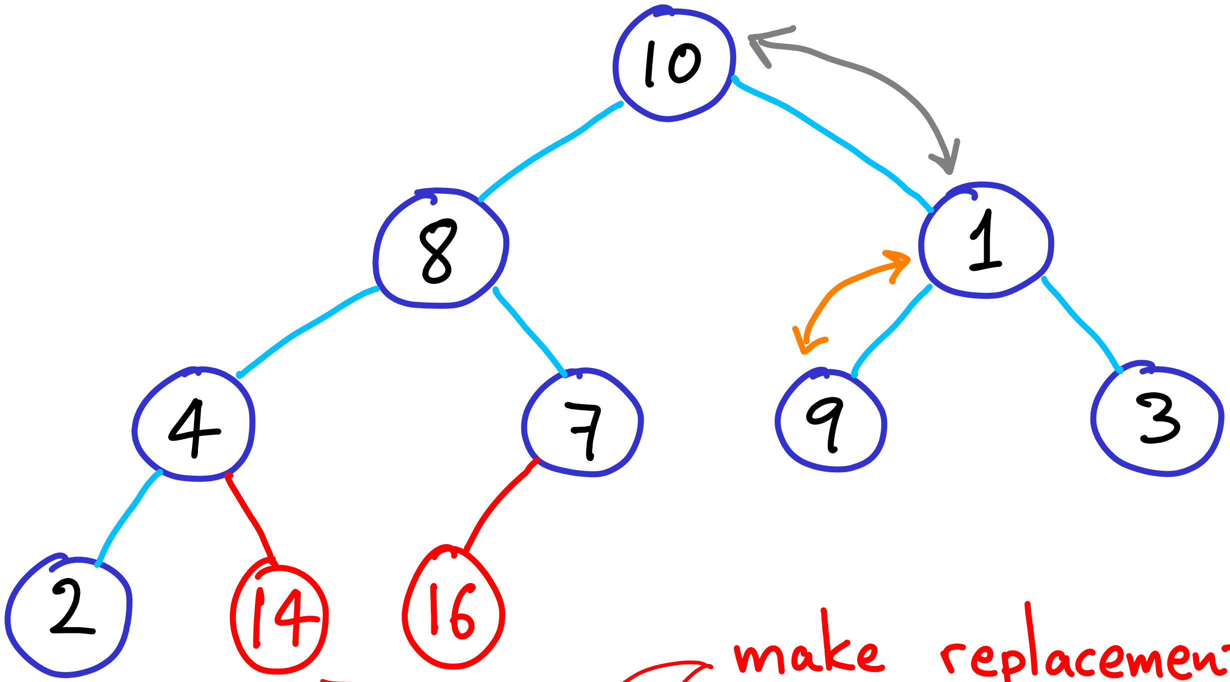
Same as before but we swap max with replacement

make replacement position inactive



How to sort data in a complete heap in place

(without an output array)



Same as before but we swap max with replacement

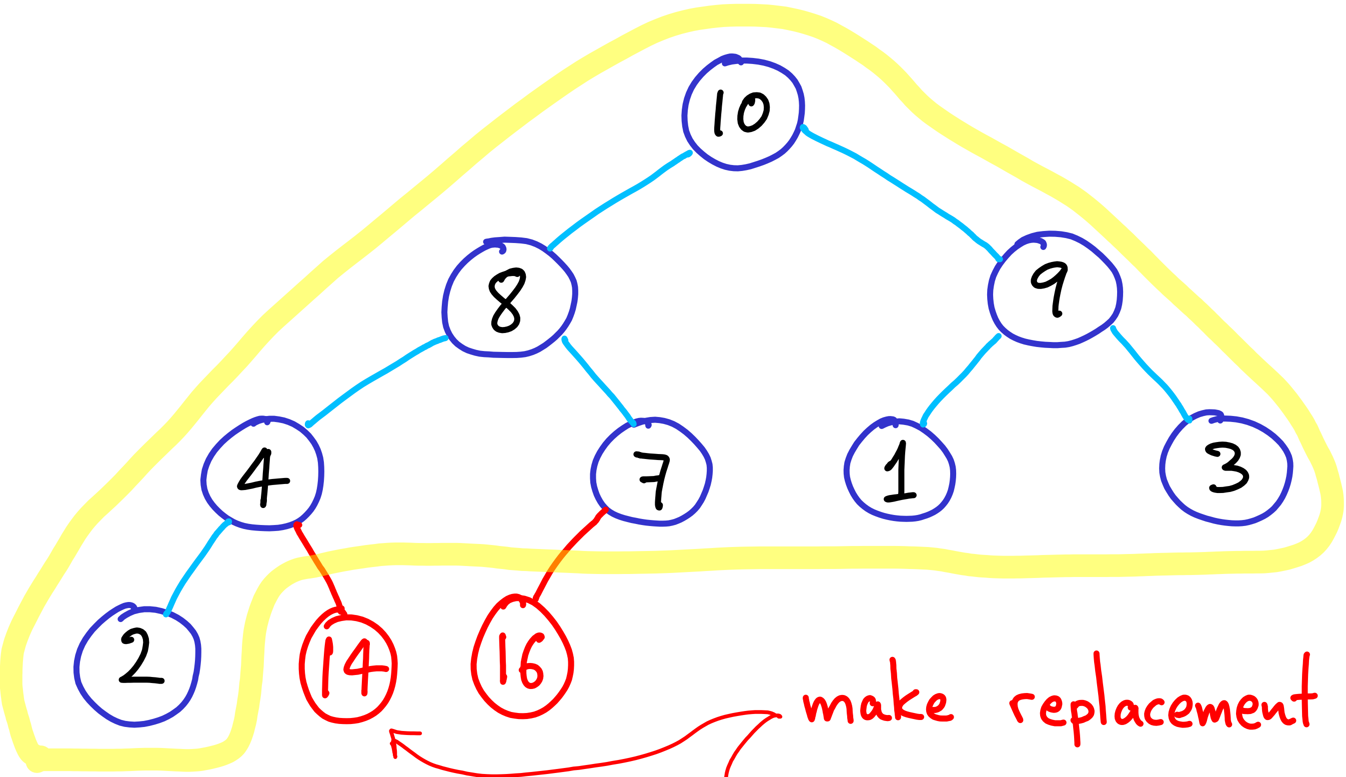
make replacement position inactive

1	2	3	4	5	6	7	8	~~9~~	~~10~~
10	8	1	4	7	9	3	2	14	16

How to sort data in a complete heap in place

(without an output array)

Same as before but we swap max with replacement

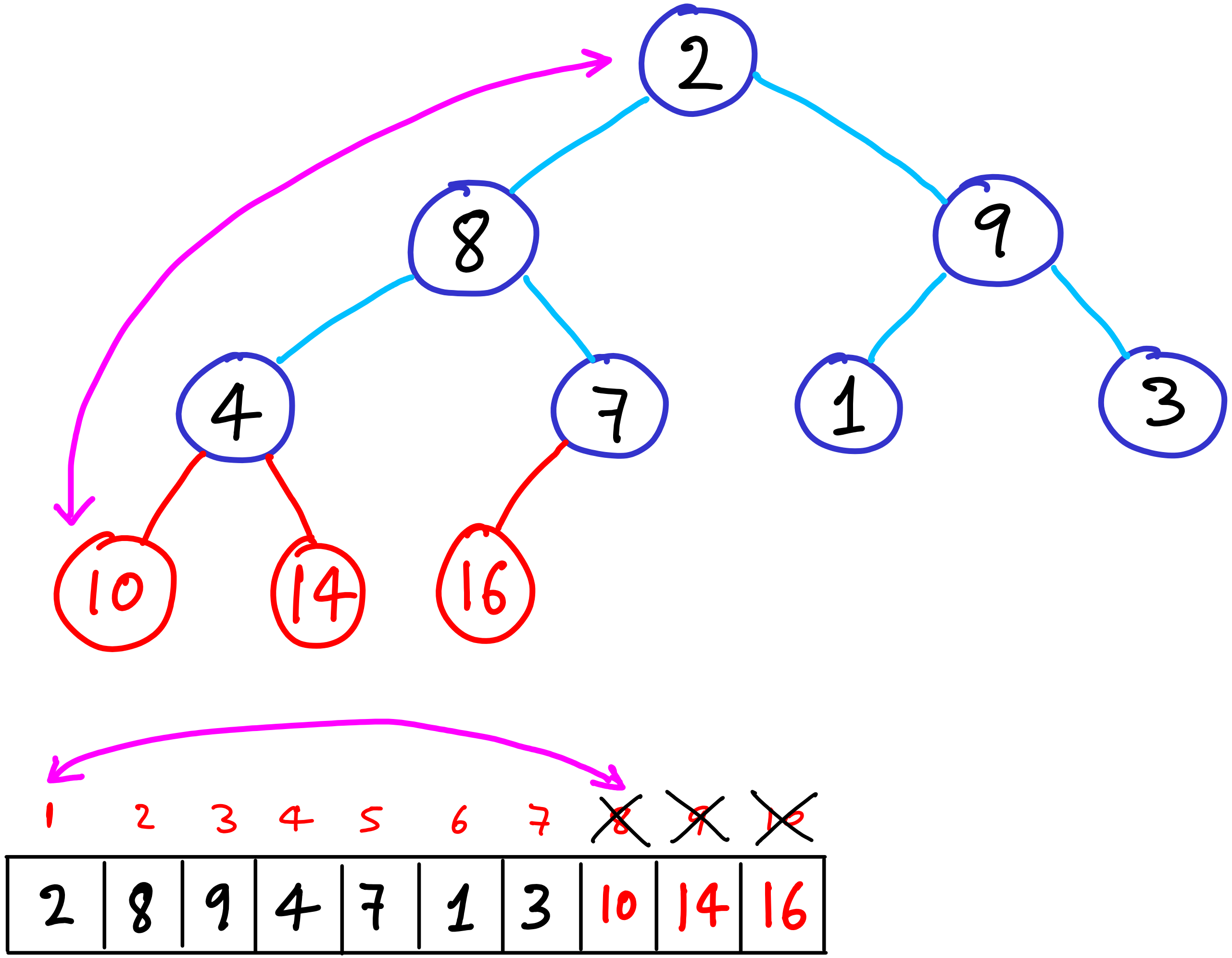


make replacement position inactive

1	2	3	4	5	6	7	8	~~9~~	~~10~~
10	8	9	4	7	1	3	2	14	16

valid heap

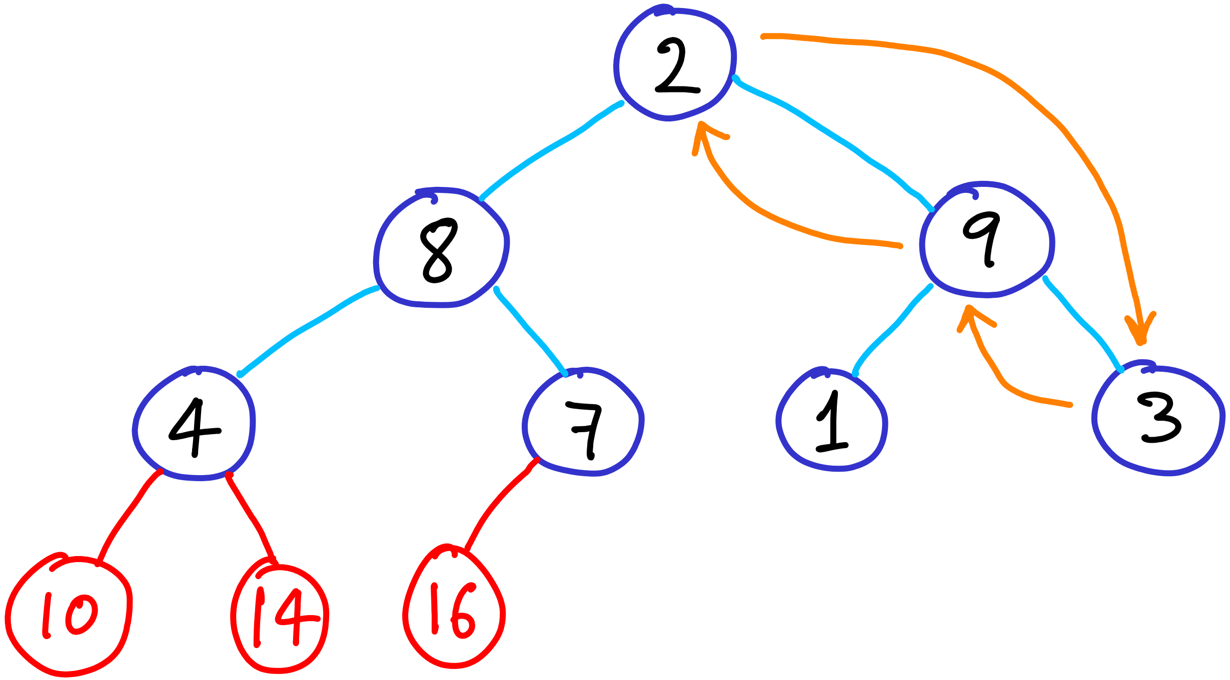
How to sort data in a complete heap in place (without an output array)



Same as before but we swap max with replacement

How to sort data in a complete heap in place

(without an output array)



Same as before

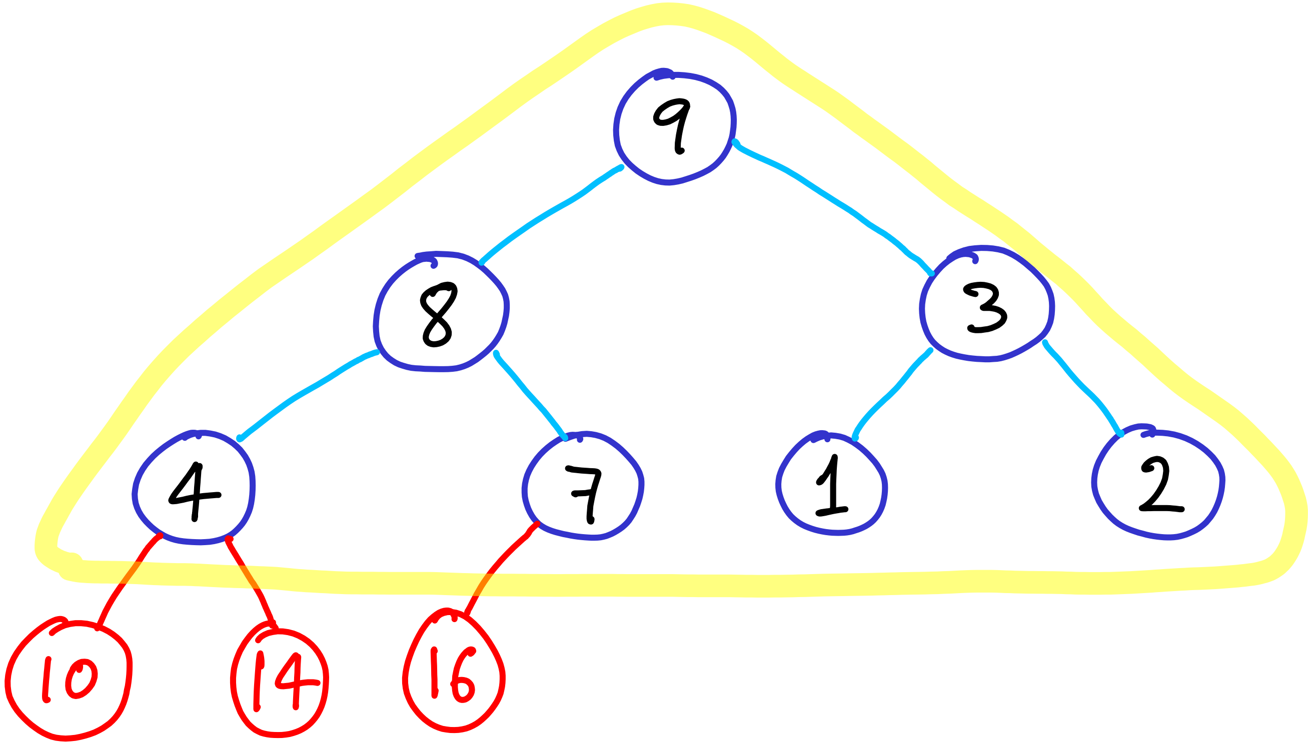
but we swap

max with replacement

1	2	3	4	5	6	7	~~8~~	~~9~~	~~10~~
2	8	9	4	7	1	3	10	14	16

How to sort data in a complete heap in place

(without an output array)

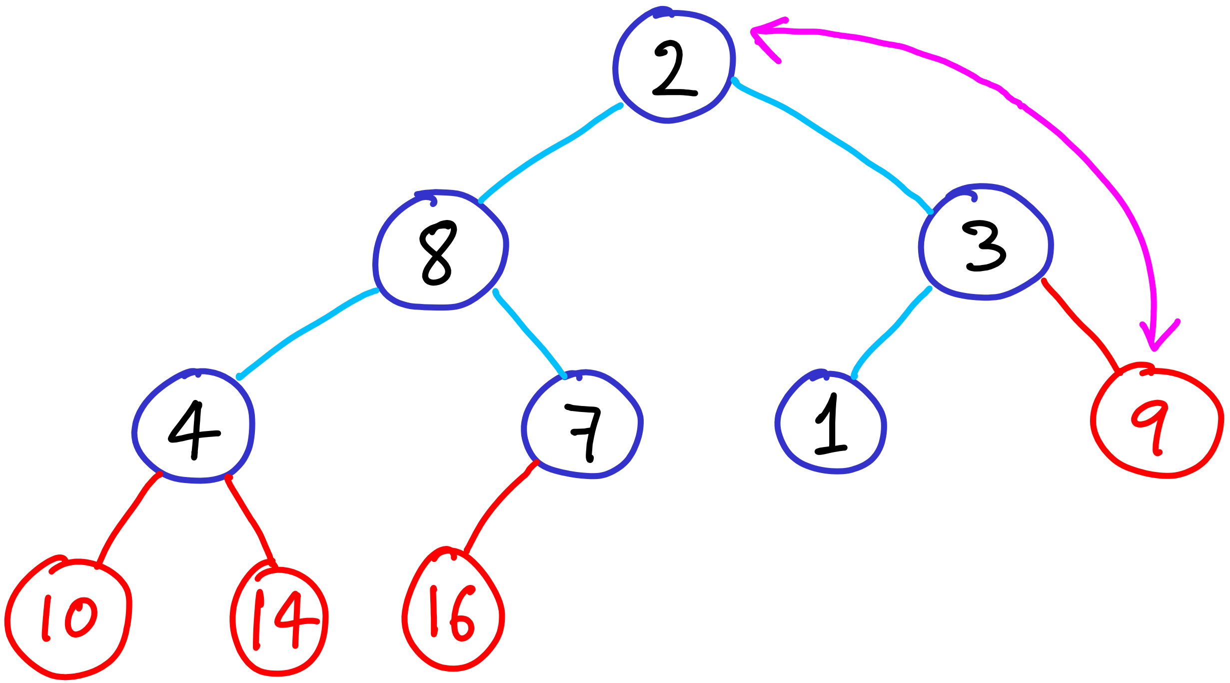


Same as before but we swap max with replacement

1	2	3	4	5	6	7	~~8~~	~~9~~	~~10~~
9	8	3	4	7	1	2	10	14	16

How to sort data in a complete heap in place

(without an output array)

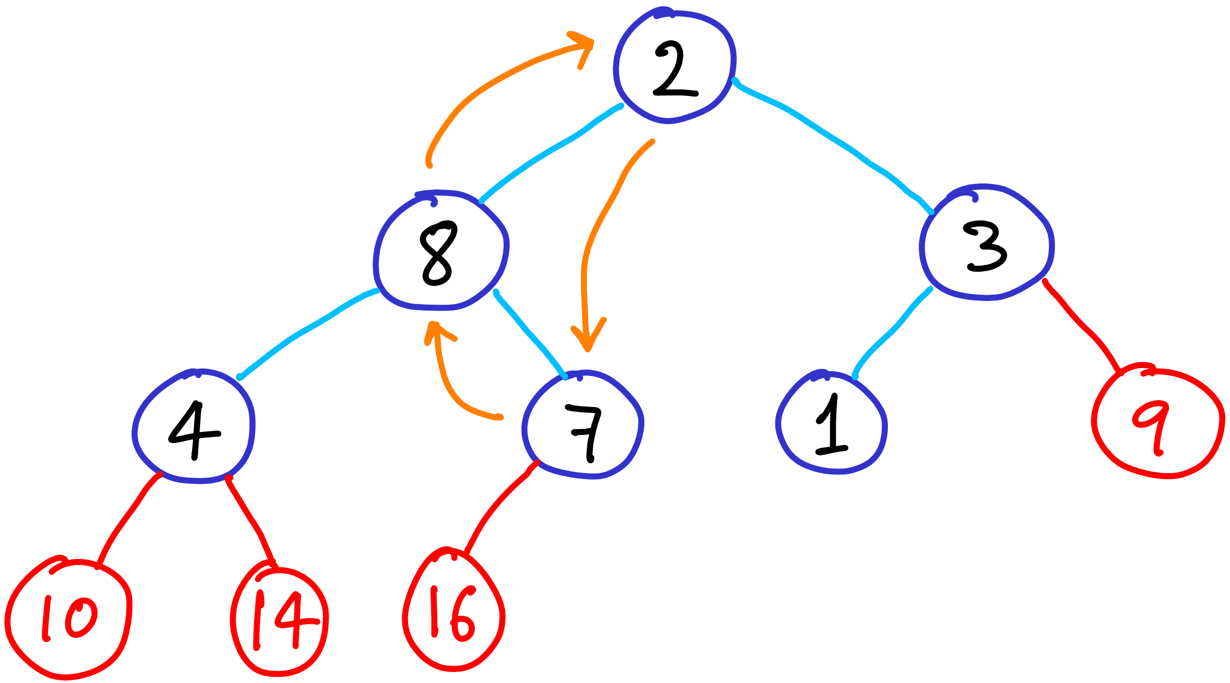


Same as before
but we swap
max with replacement

1	2	3	4	5	6	~~7~~	~~8~~	~~9~~	~~10~~
2	8	3	4	7	1	9	10	14	16

How to sort data in a complete heap in place

(without an output array)

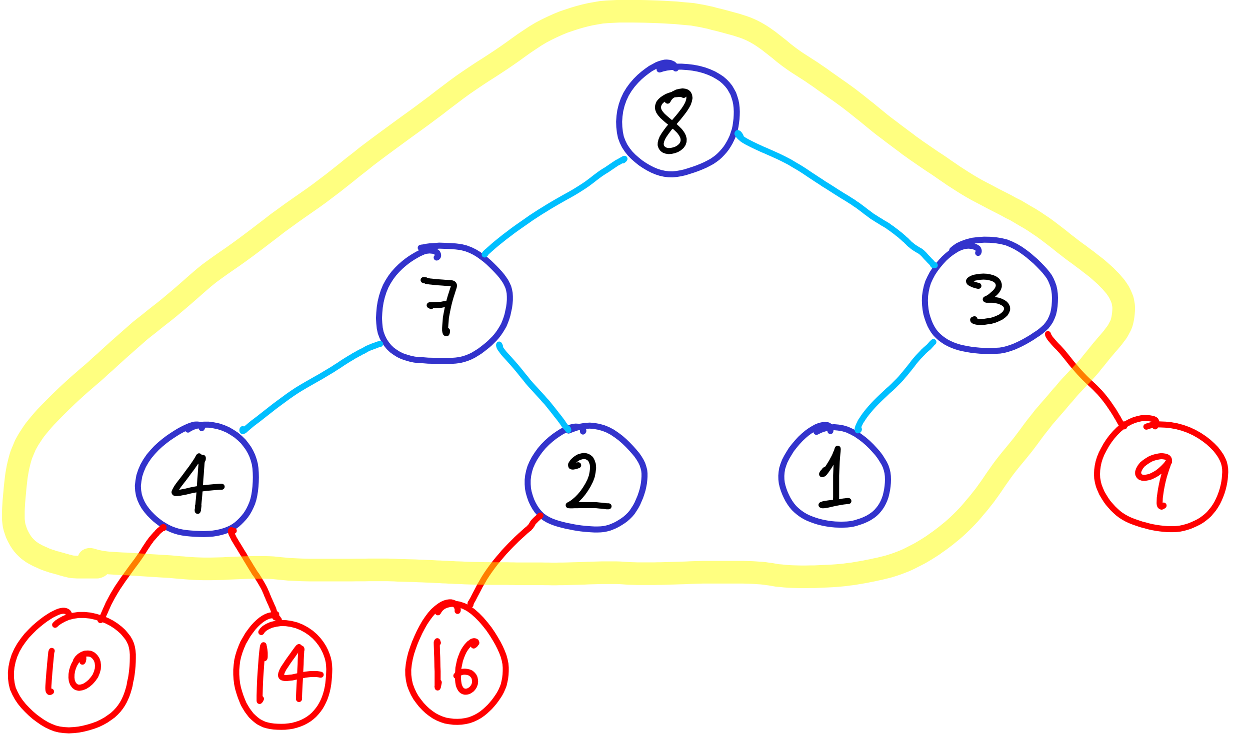


Same as before
but we swap
max with replacement

1	2	3	4	5	6	~~7~~	~~8~~	~~9~~	~~10~~
2	8	3	4	7	1	9	10	14	16

How to sort data in a complete heap in place

(without an output array)

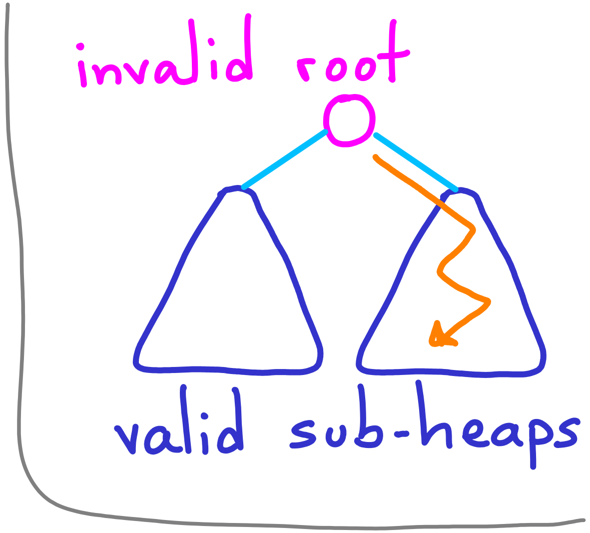


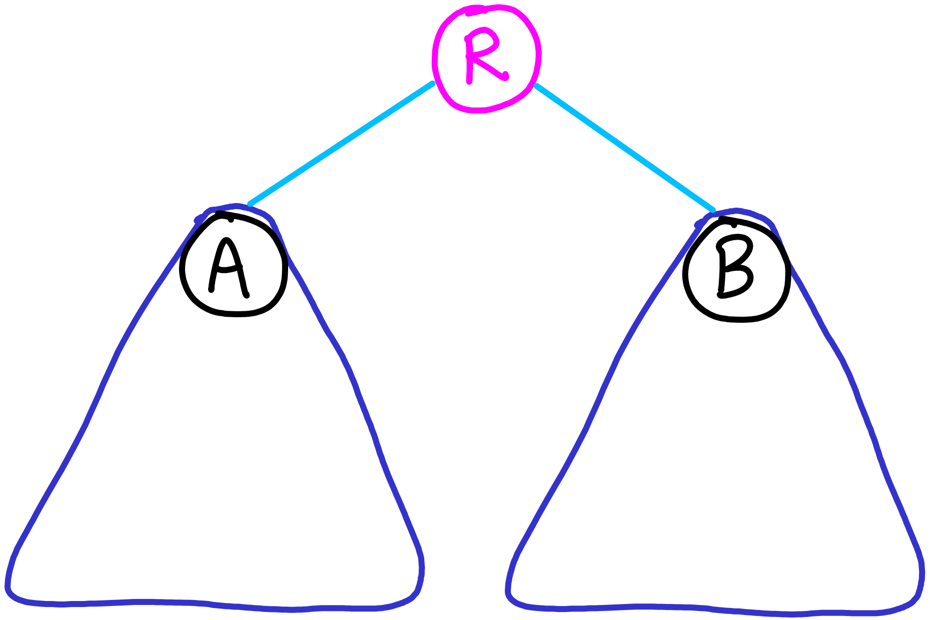
Same as before
but we swap
max with replacement

etc

1	2	3	4	5	6	~~7~~	~~8~~	~~9~~	~~10~~
8	7	3	4	2	1	9	10	14	16

Correctness of "heapify"



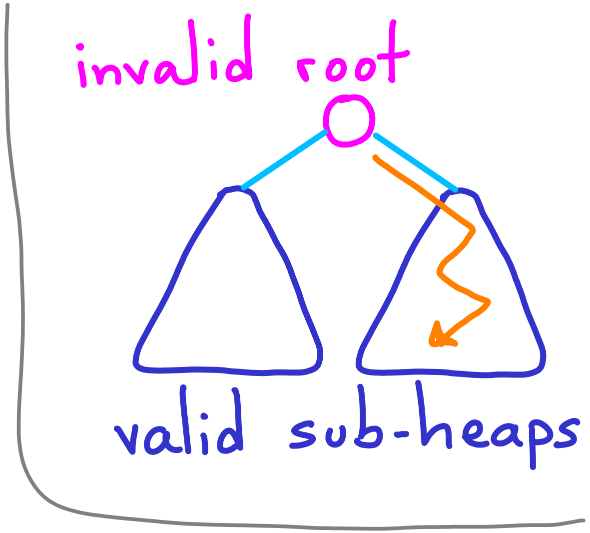


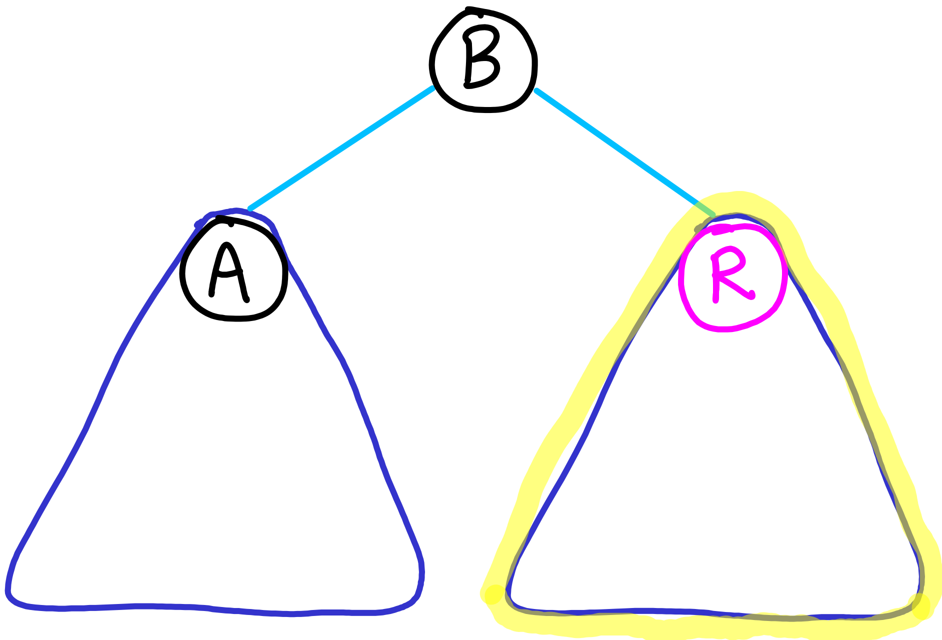
Assume $A < B$

if $R > B$, done.

$(R > B > A)$

Correctness of "heapify"





Assume $A < B$

if $R > B$, done. $(R > B > A)$

else swap B & R $(R < B)$

recurse

Summary

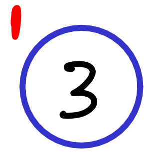
Given a heap we can extract max and heapify in $O(\log n)$ time.

↳ n rounds : $O(n \log n)$ to sort a heap

How do we construct a heap in the first place?

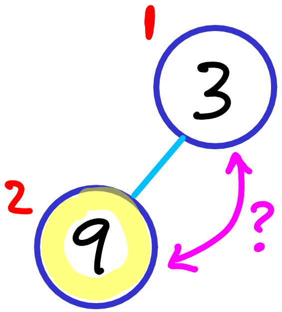
Heap building: the FORWARD METHOD (left to right)

1	2	3	4	5	6	7	8	9	10
3	9	7	10	8	4	14	2	16	1



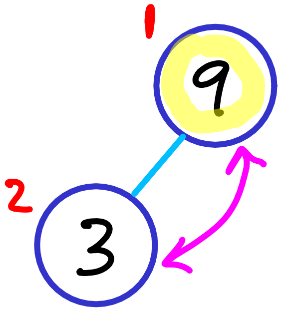
Heap building: the FORWARD METHOD (left to right)

1	2	3	4	5	6	7	8	9	10
3	9	7	10	8	4	14	2	16	1

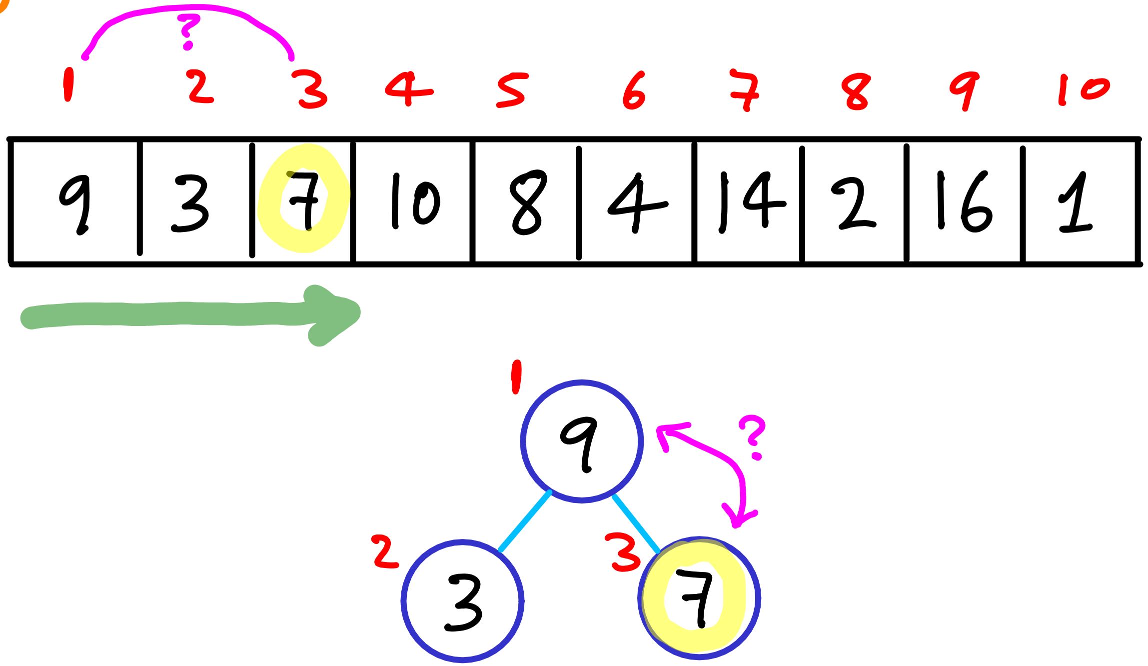


Heap building: the FORWARD METHOD (left to right)

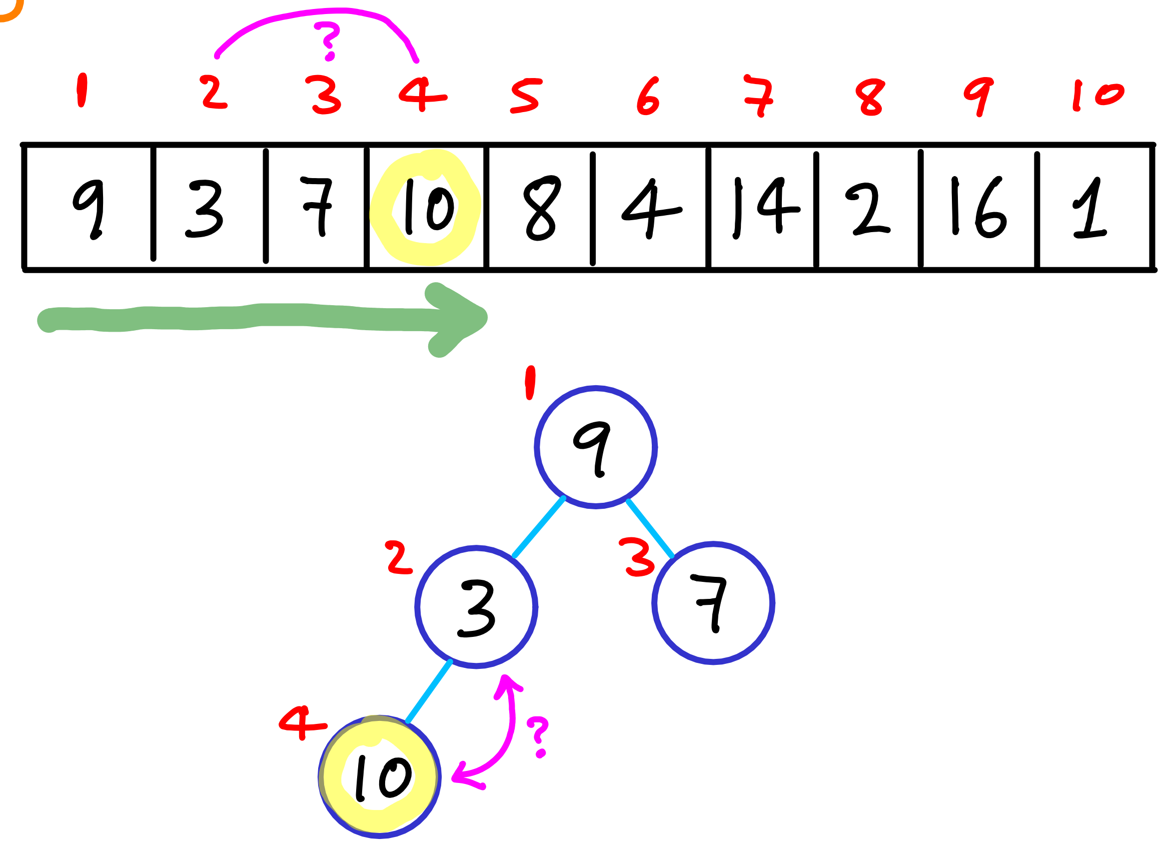
1	2	3	4	5	6	7	8	9	10
9	3	7	10	8	4	14	2	16	1



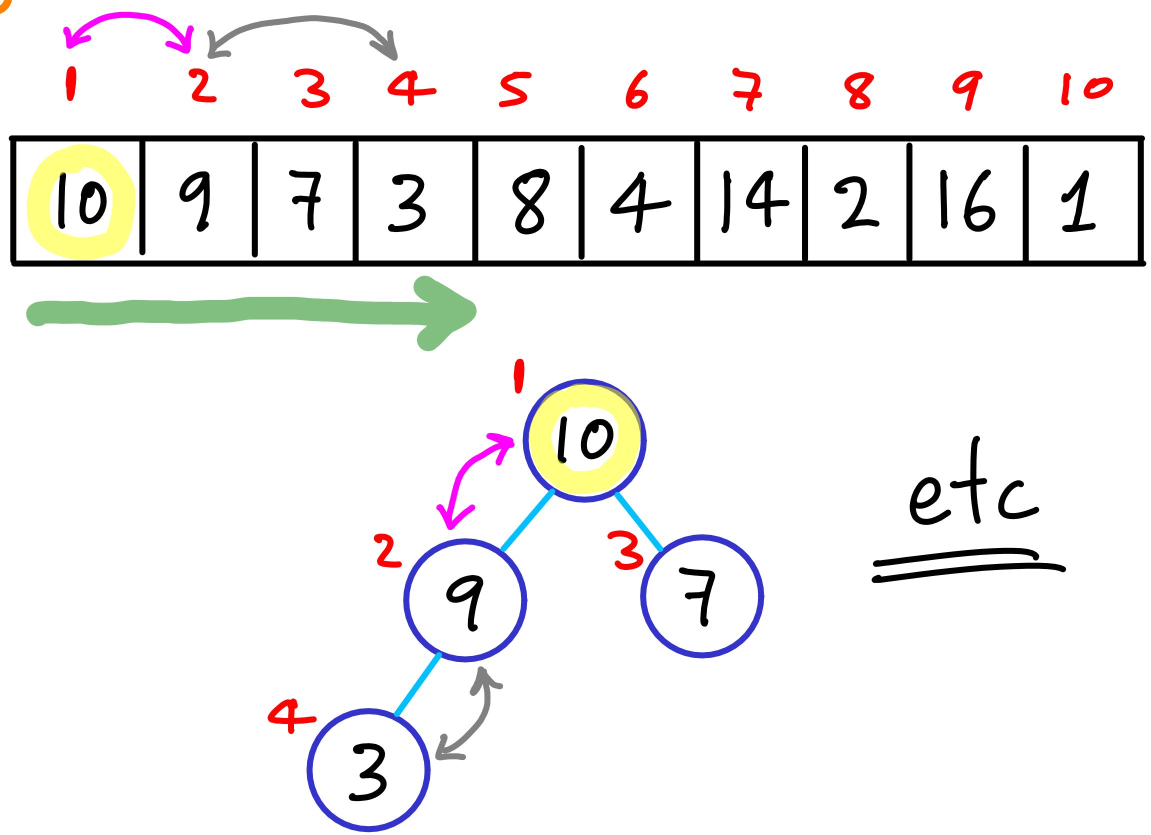
Heap building: the FORWARD METHOD (left to right)



Heap building: the FORWARD METHOD (left to right)



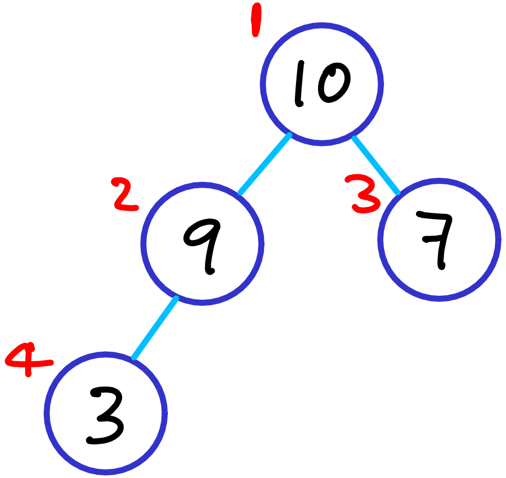
Heap building: the FORWARD METHOD (left to right)



etc

Heap building: the FORWARD METHOD (left to right)

1	2	3	4	5	6	7	8	9	10
10	9	7	3	8	4	14	2	16	1

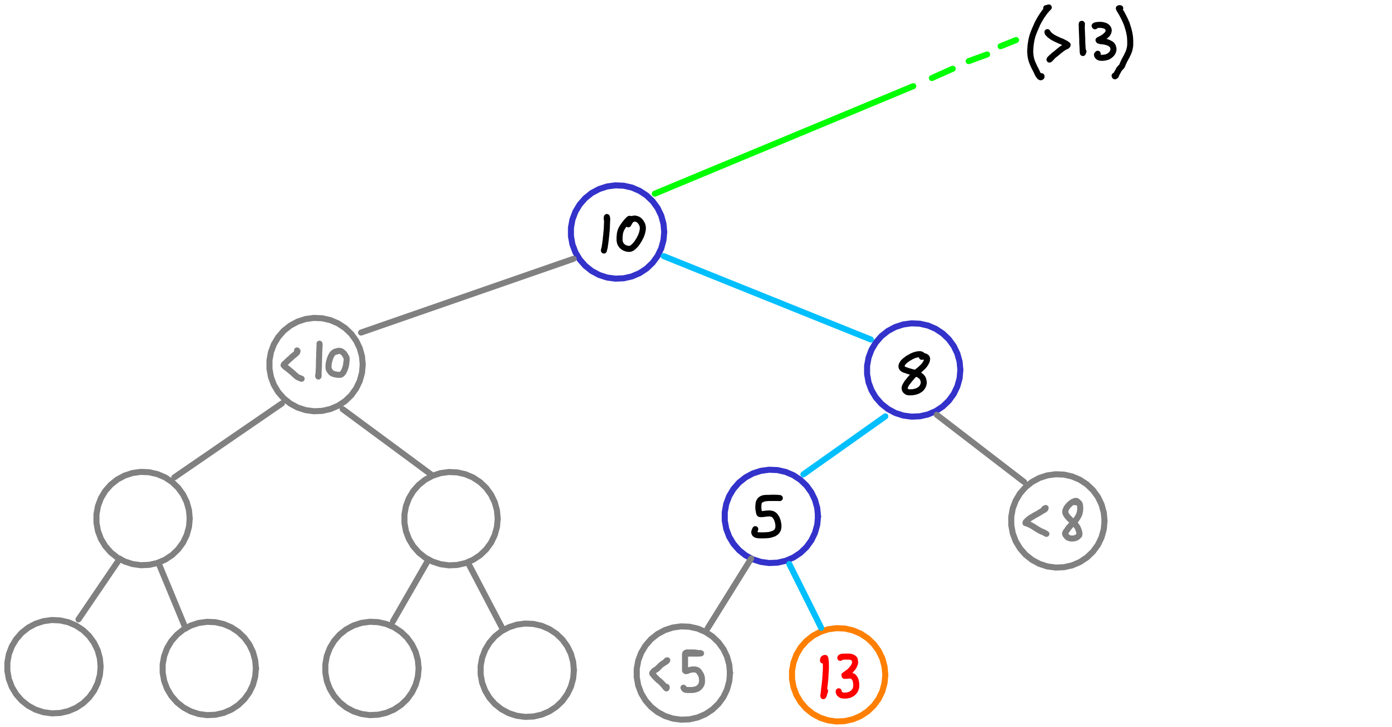


time = $O(n \log n)$

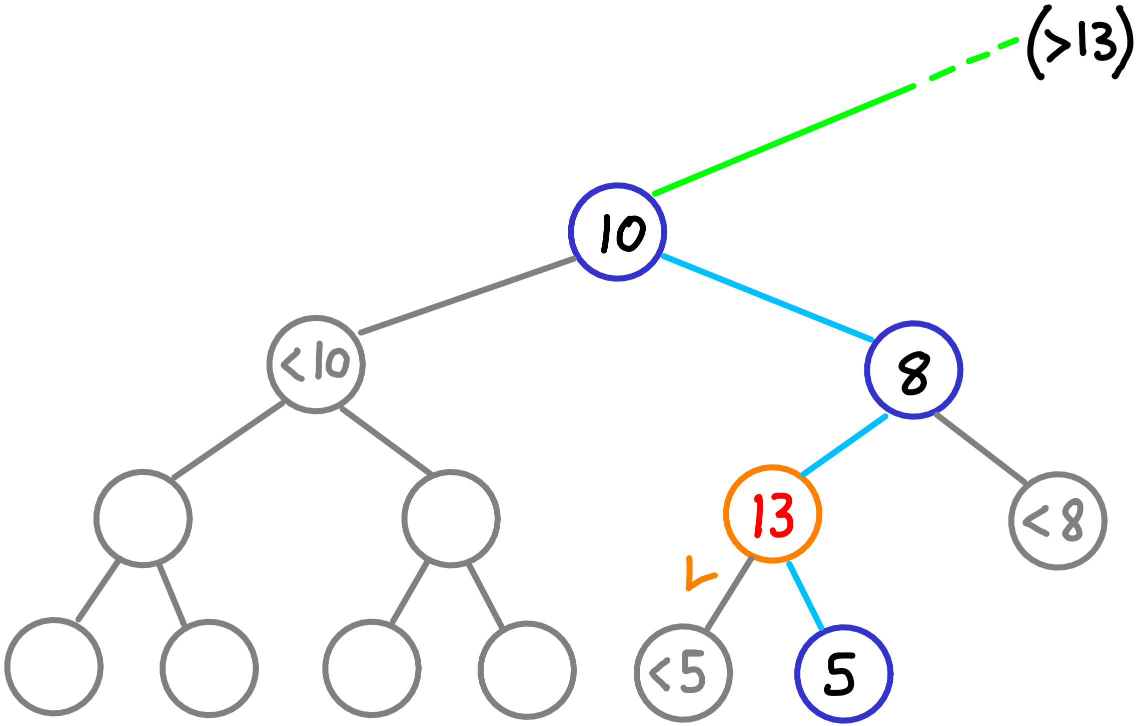
$O(\log n)$ per insertion

Works for streaming data

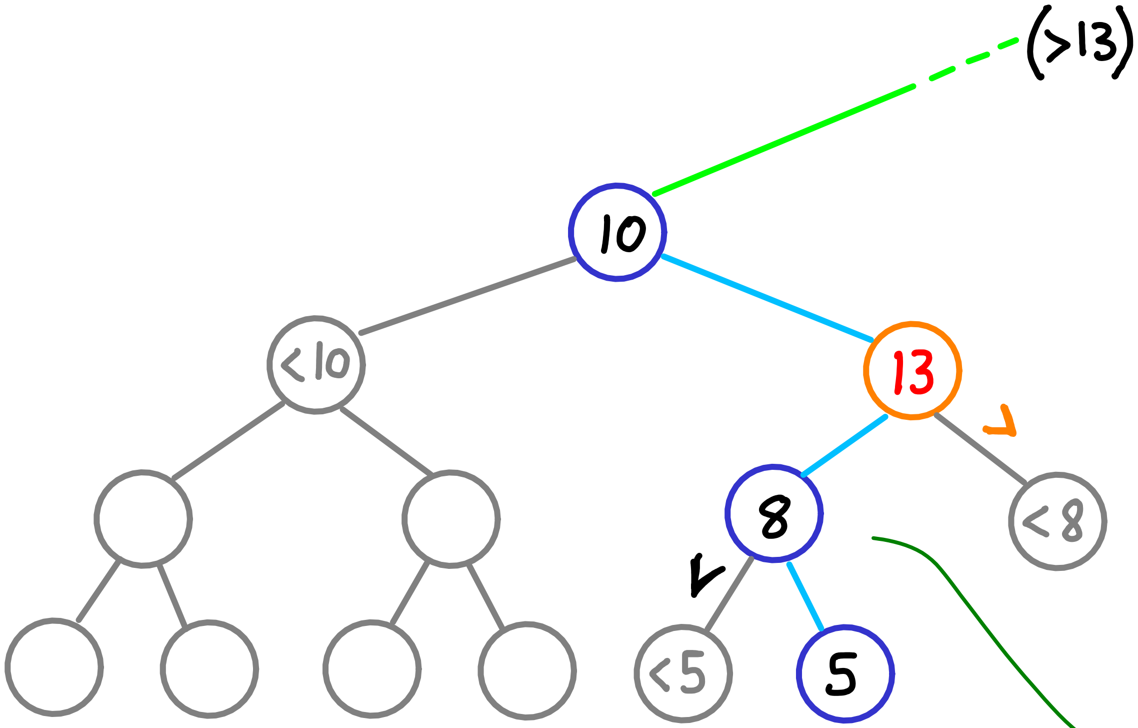
Correctness (sketch)



Correctness (sketch)



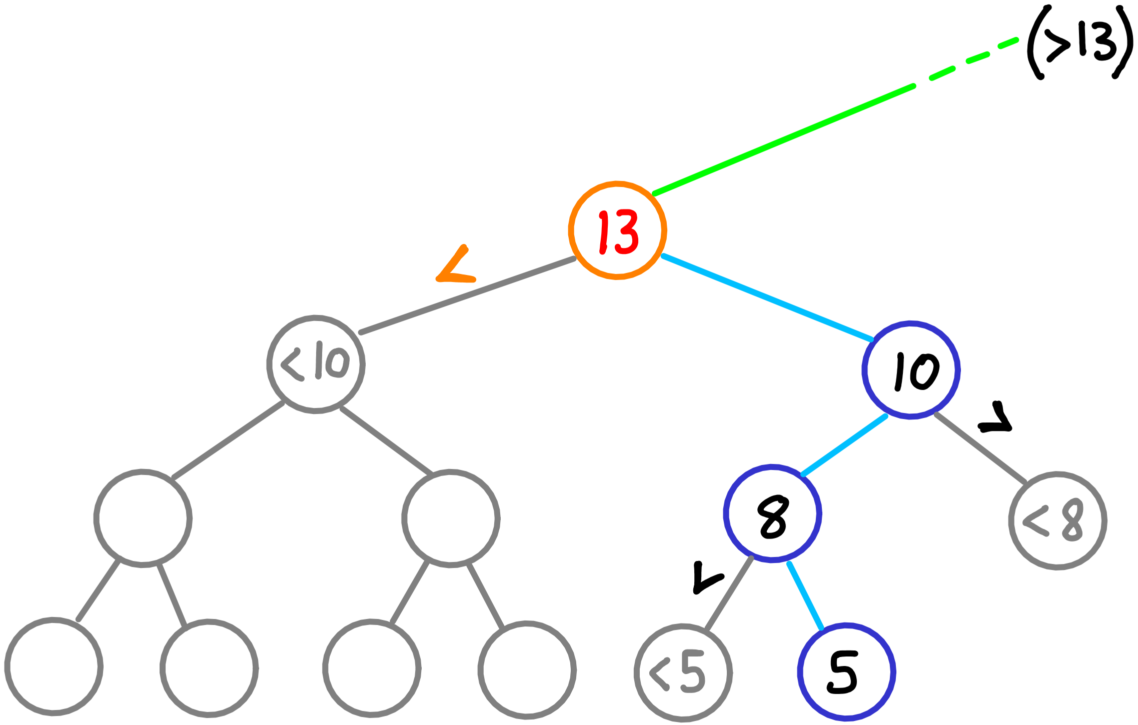
Correctness (sketch)



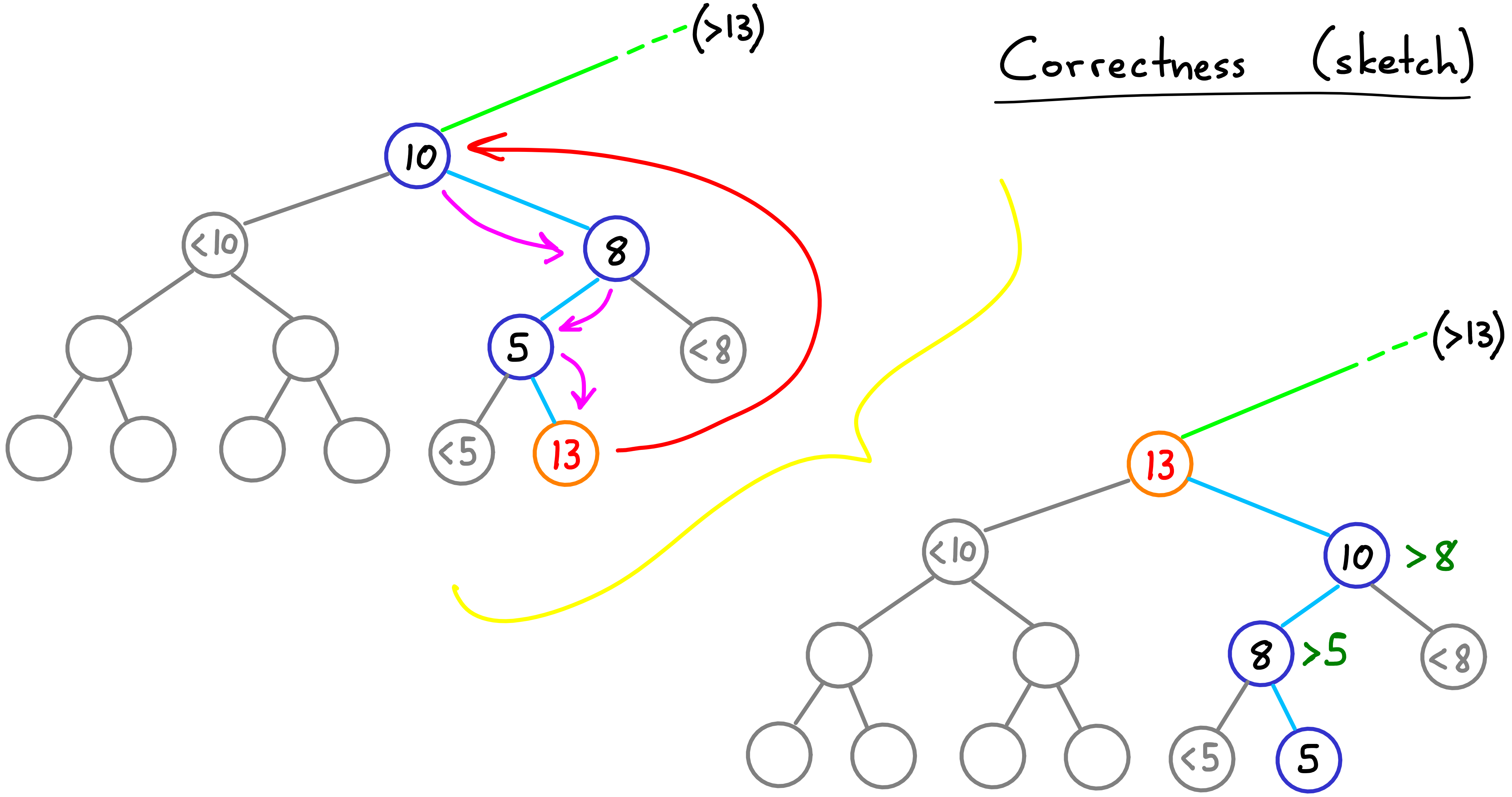
was 5, initially

replaced by parent: no problem

Correctness (sketch)

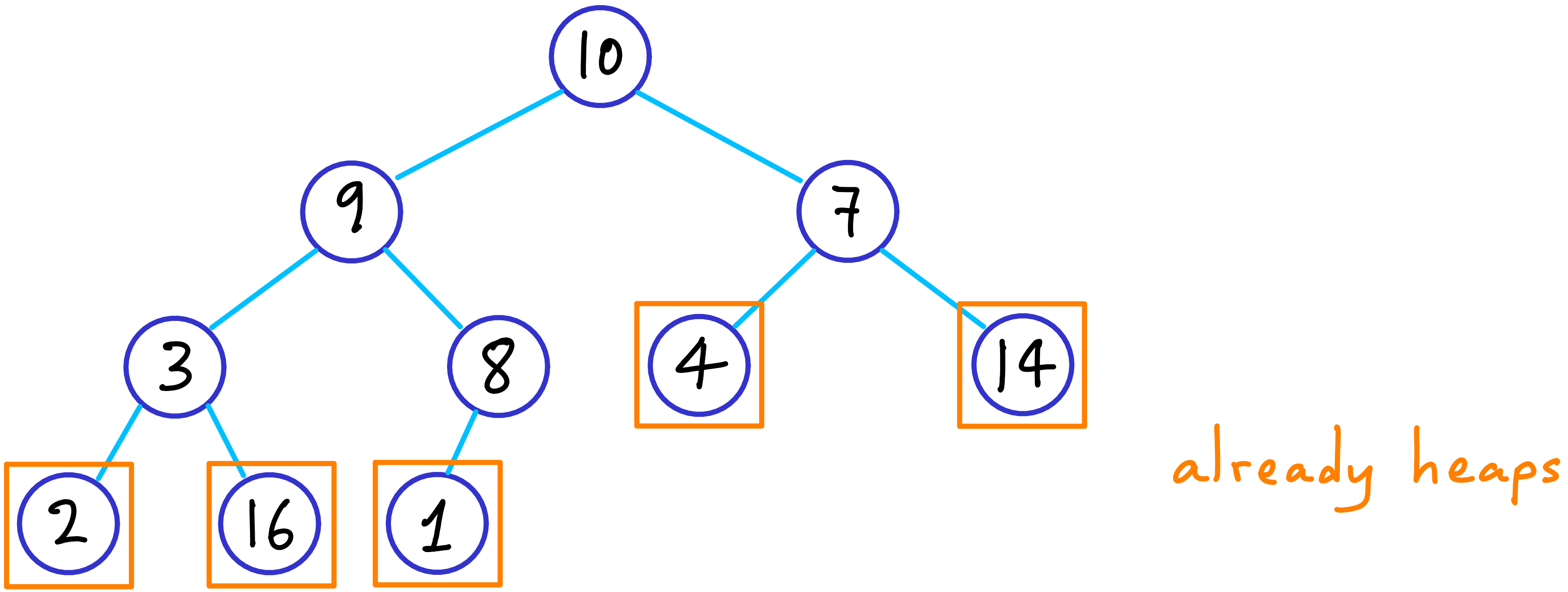


Correctness (sketch)



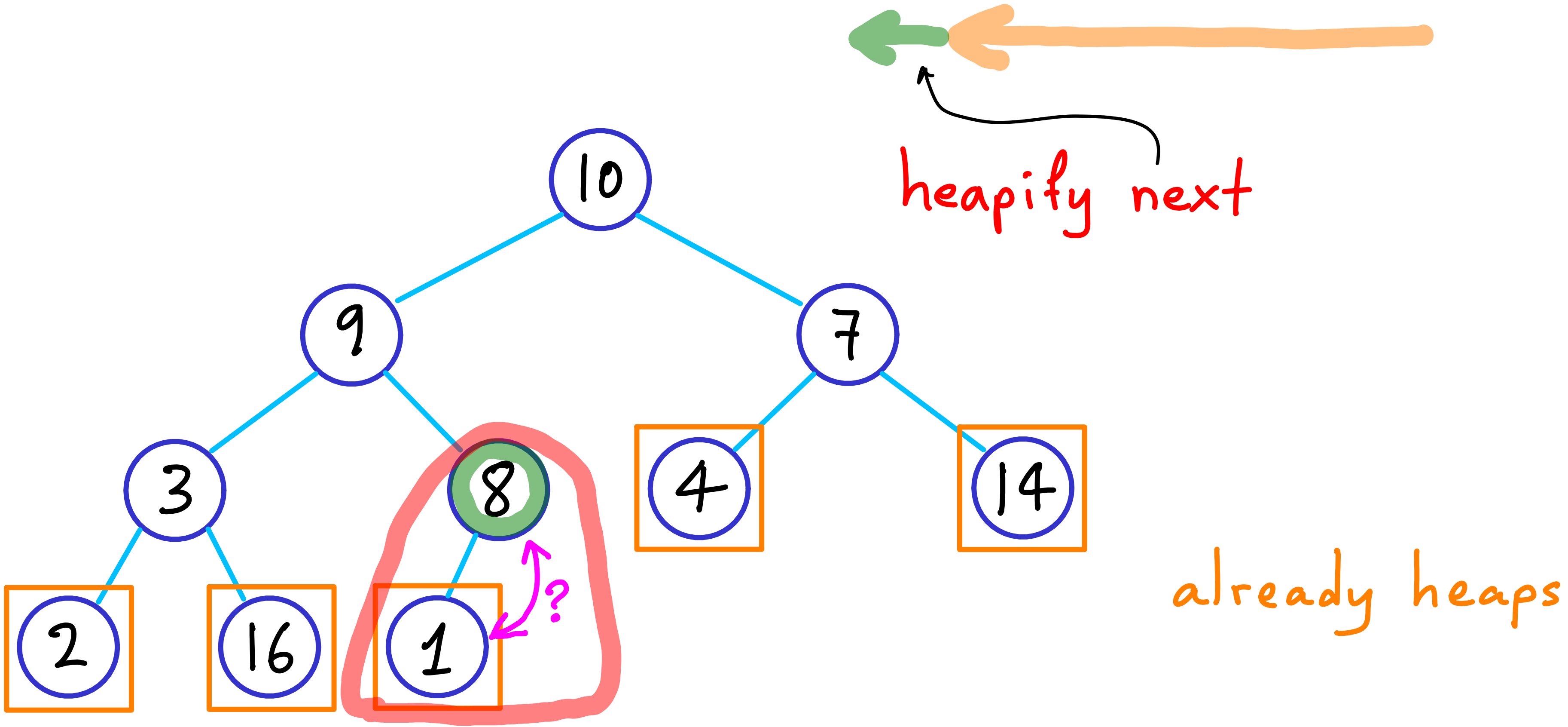
Heap building: the REVERSE METHOD (right to left)

1	2	3	4	5	6	7	8	9	10
10	9	7	3	8	4	14	2	16	1



Heap building: the REVERSE METHOD (right to left)

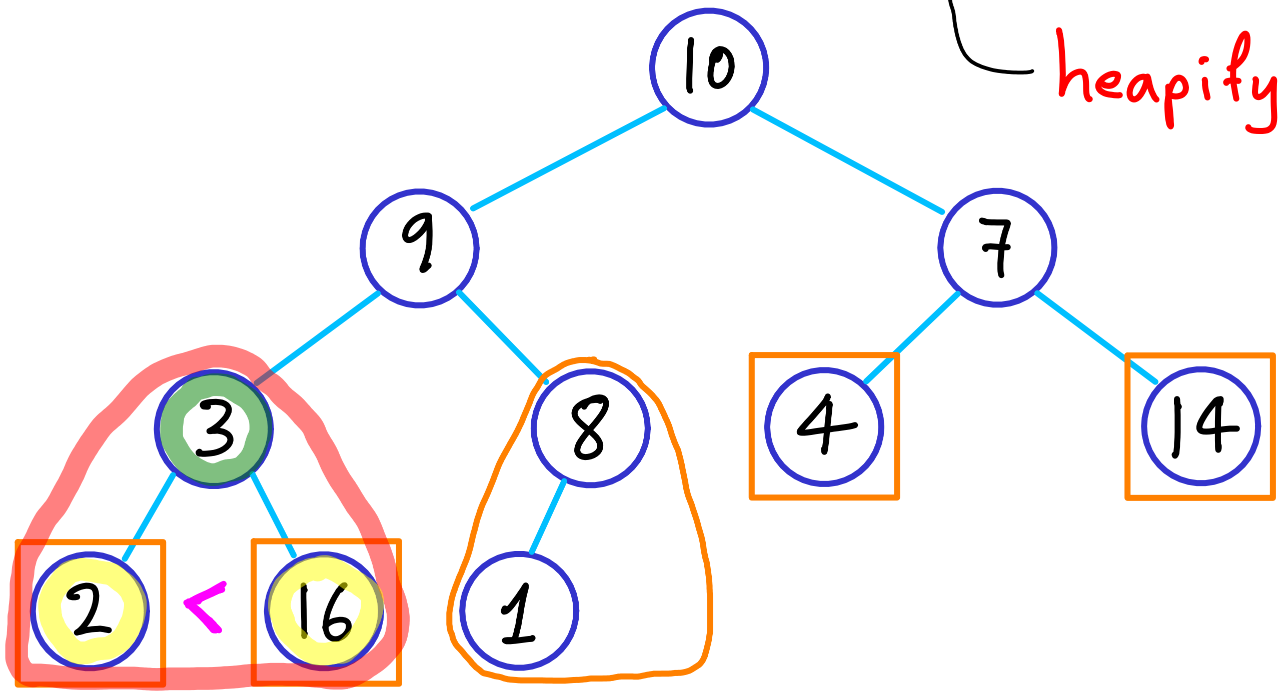
1	2	3	4	5	6	7	8	9	10
10	9	7	3	8	4	14	2	16	1



Heap building: the REVERSE METHOD (right to left)

1	2	3	4	5	6	7	8	9	10
10	9	7	3	8	4	14	2	16	1

heapify next

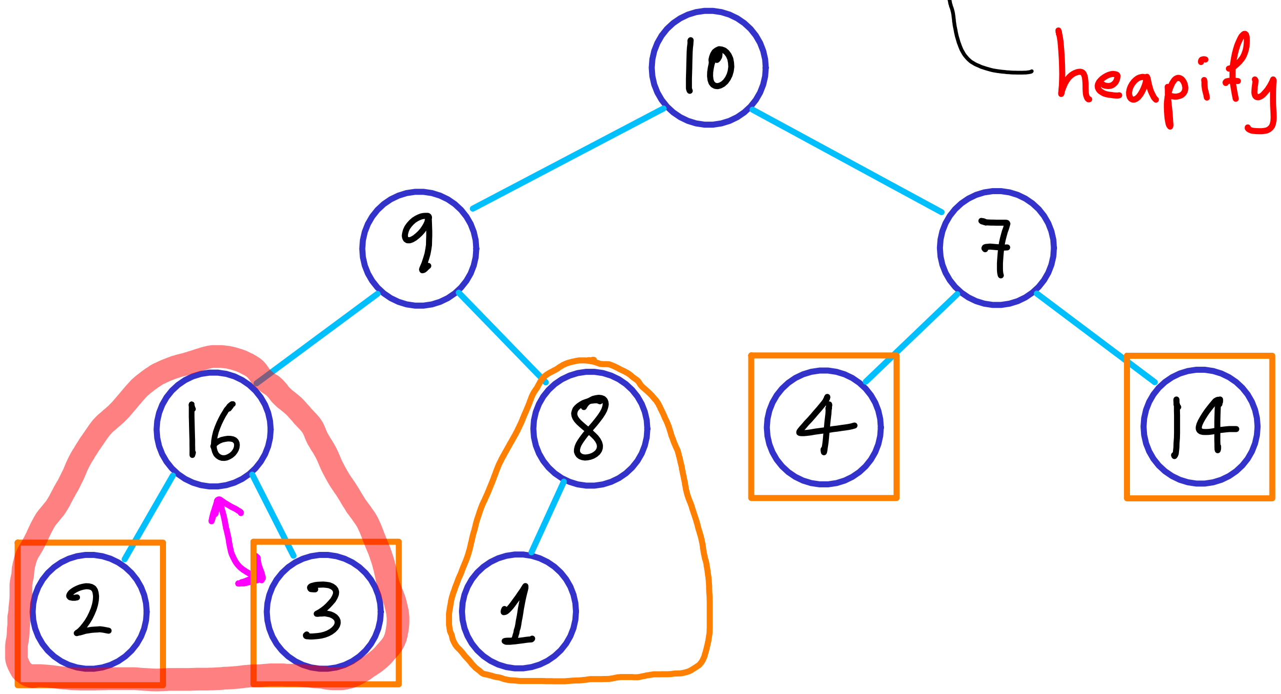


already heaps

Heap building: the REVERSE METHOD (right to left)

1	2	3	4	5	6	7	8	9	10
10	9	7	16	8	4	14	2	3	1

heapify next

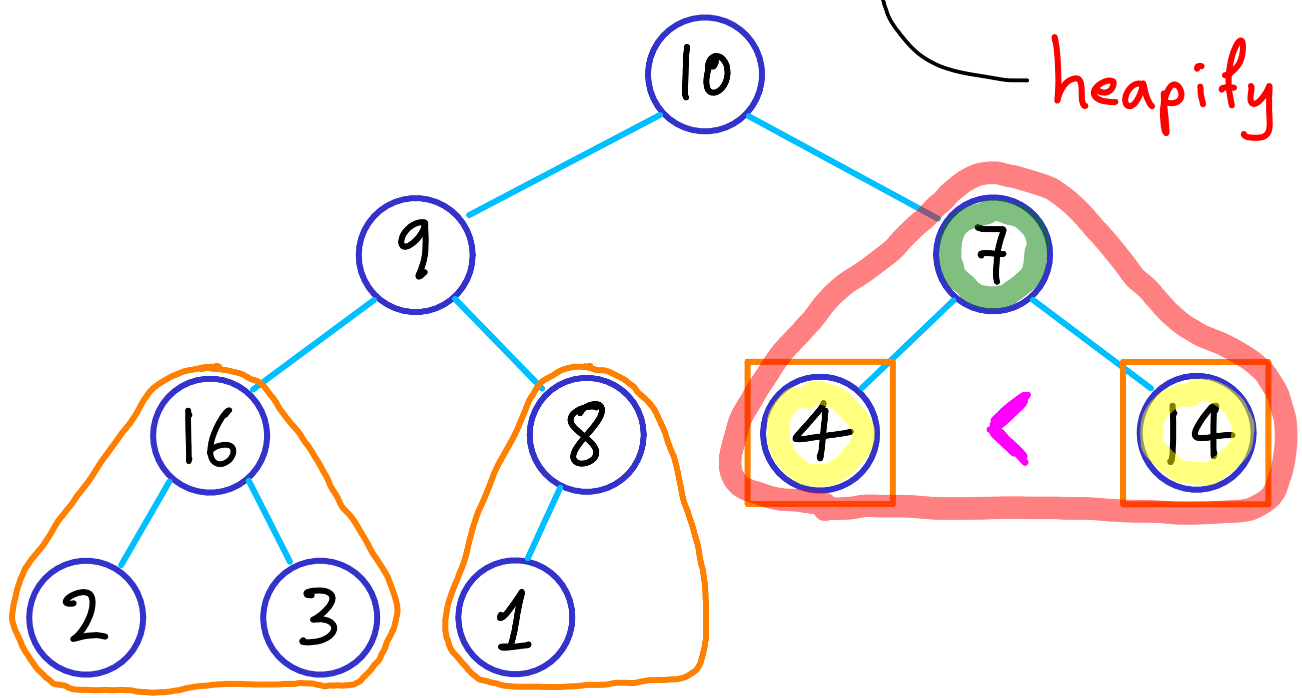


already heaps

Heap building: the REVERSE METHOD (right to left)

1	2	3	4	5	6	7	8	9	10
10	9	7	16	8	4	14	2	3	1

heapify next

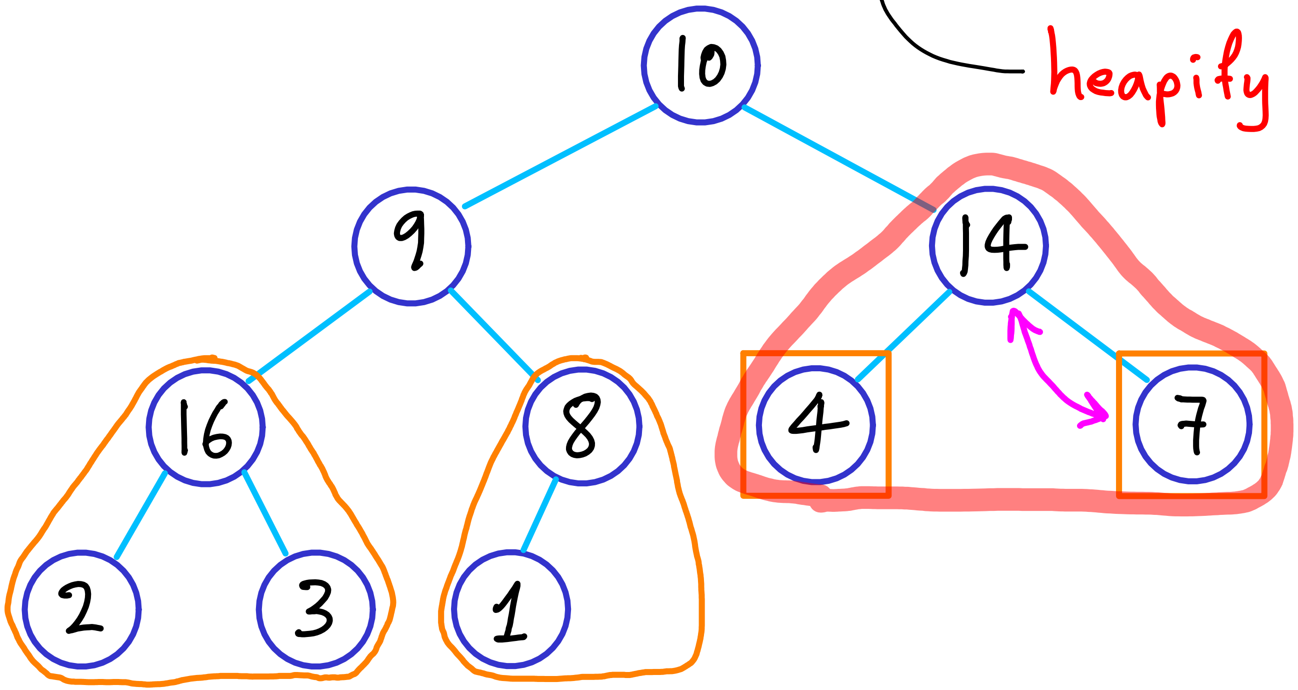


already heaps

Heap building: the REVERSE METHOD (right to left)

1	2	3	4	5	6	7	8	9	10
10	9	14	16	8	4	7	2	3	1

heapify next

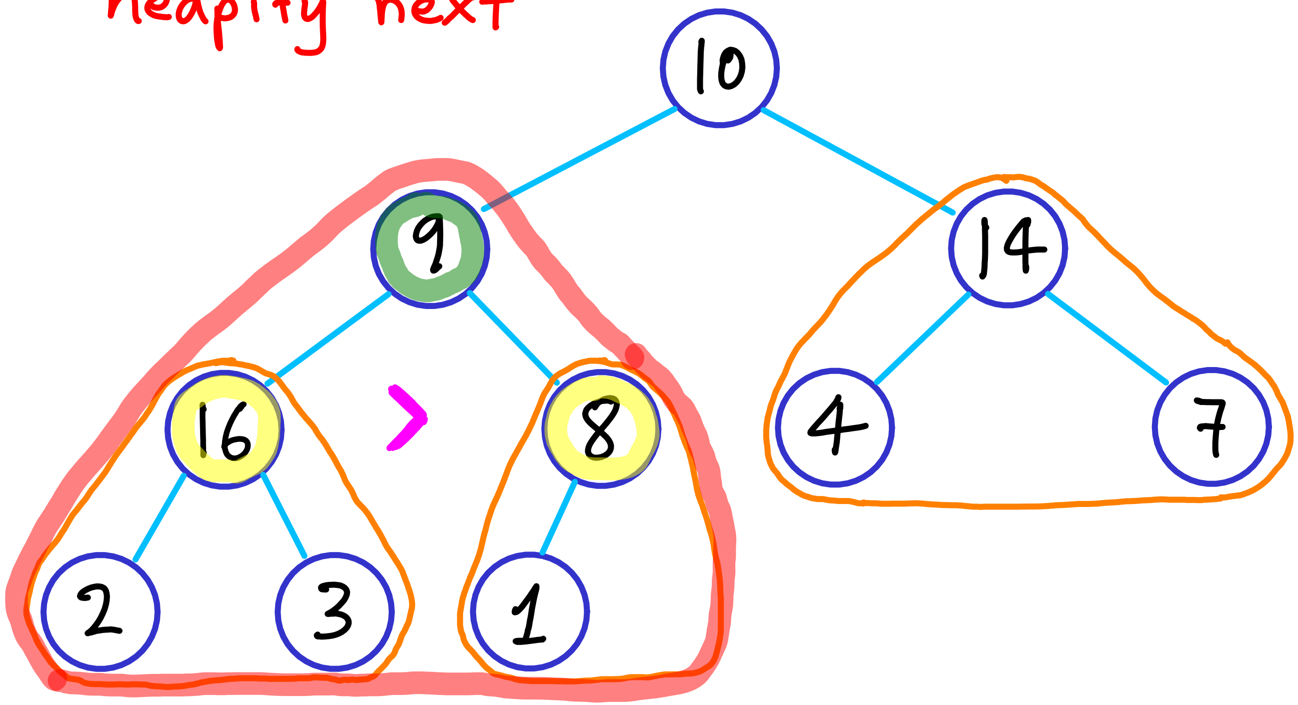


already heaps

Heap building: the REVERSE METHOD (right to left)

1	2	3	4	5	6	7	8	9	10
10	9	14	16	8	4	7	2	3	1

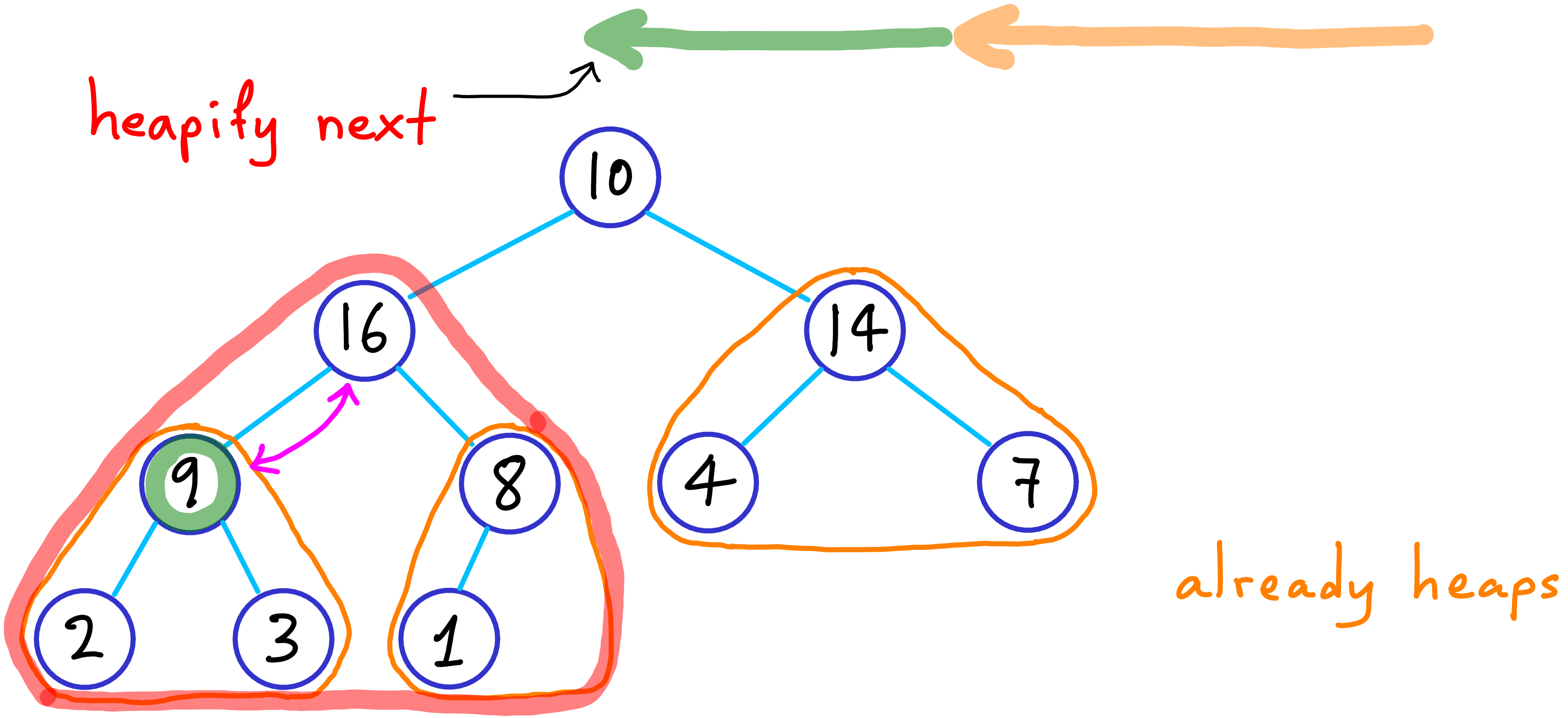
heapify next



already heaps

Heap building: the REVERSE METHOD (right to left)

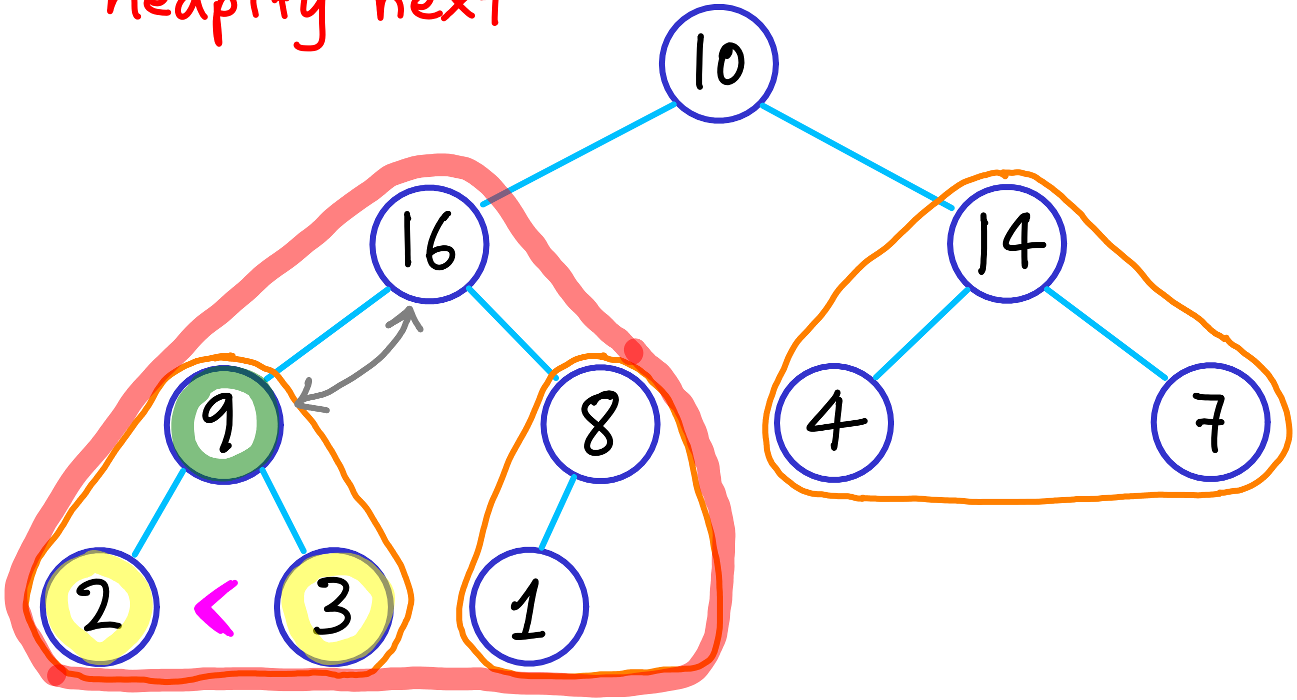
1	2	3	4	5	6	7	8	9	10
10	16	14	9	8	4	7	2	3	1



Heap building: the REVERSE METHOD (right to left)

1	2	3	4	5	6	7	8	9	10
10	16	14	9	8	4	7	2	3	1

heapify next

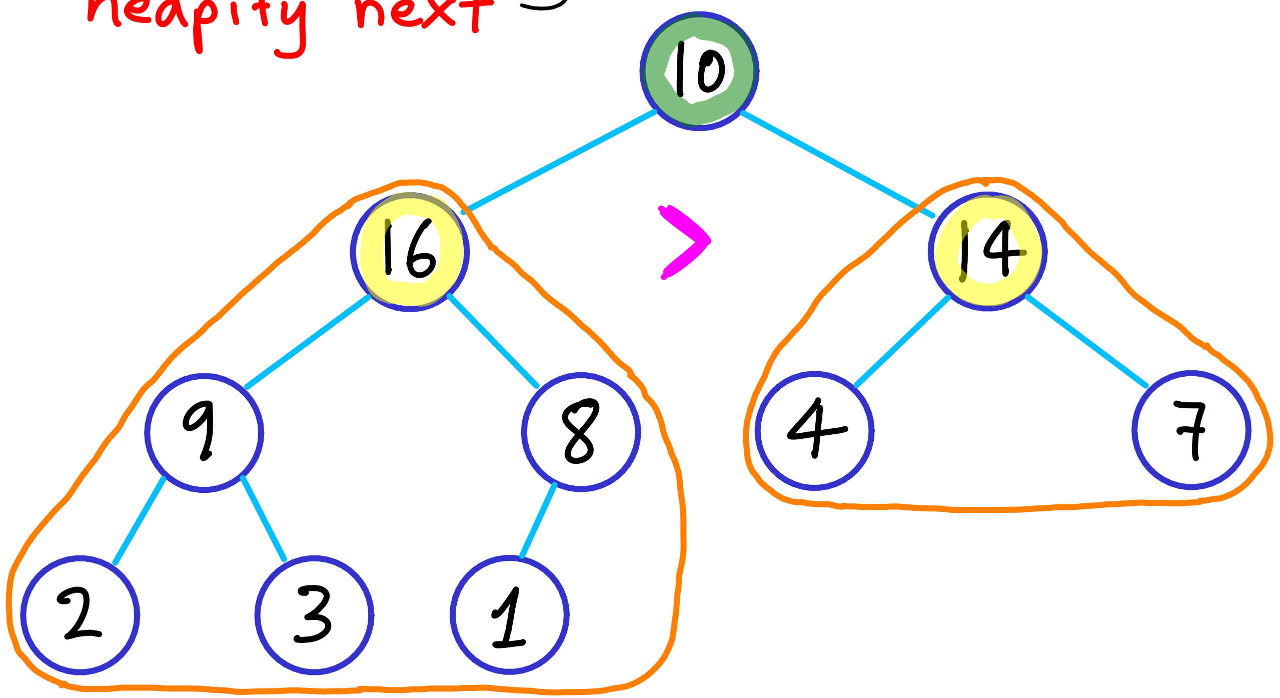


already heaps

Heap building: the REVERSE METHOD (right to left)

1	2	3	4	5	6	7	8	9	10
10	16	14	9	8	4	7	2	3	1

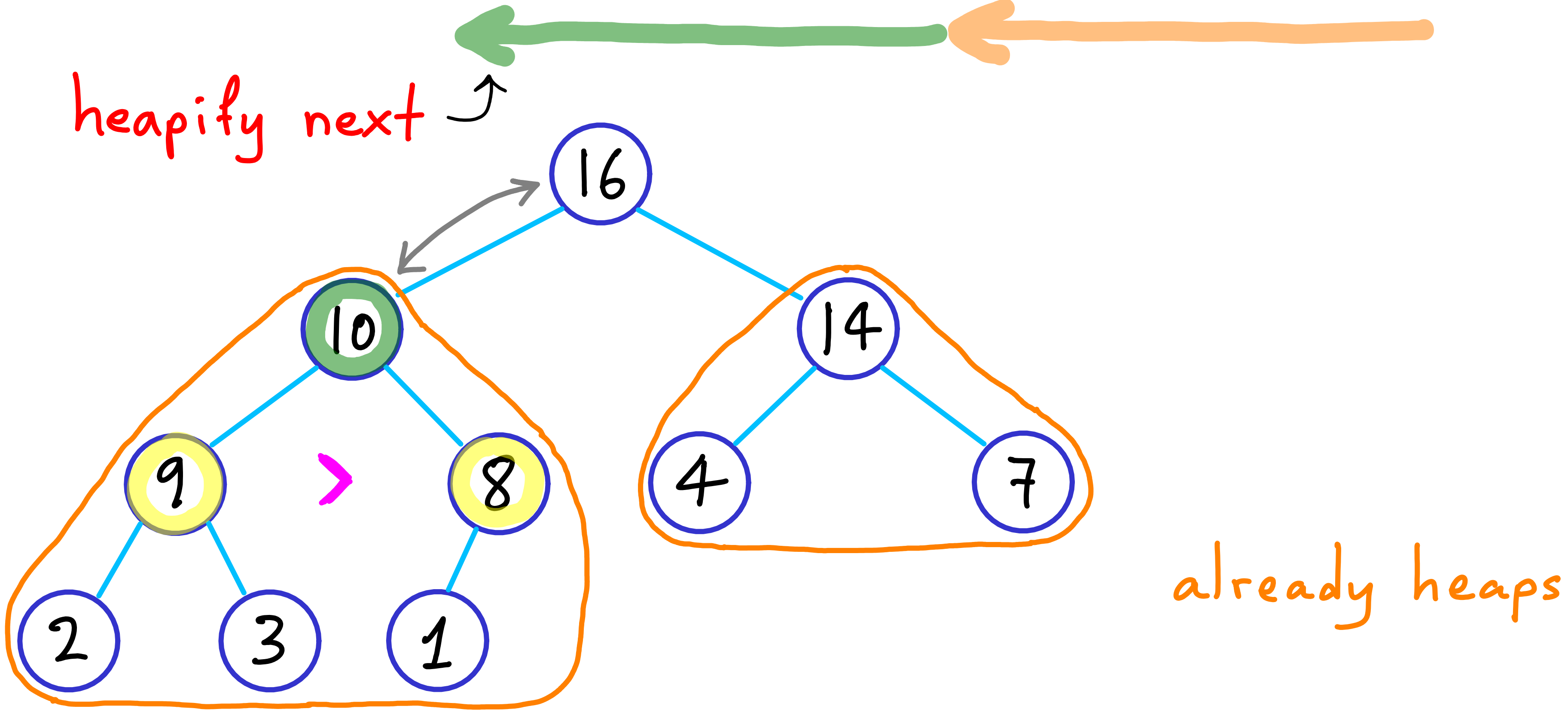
heapify next



already heaps

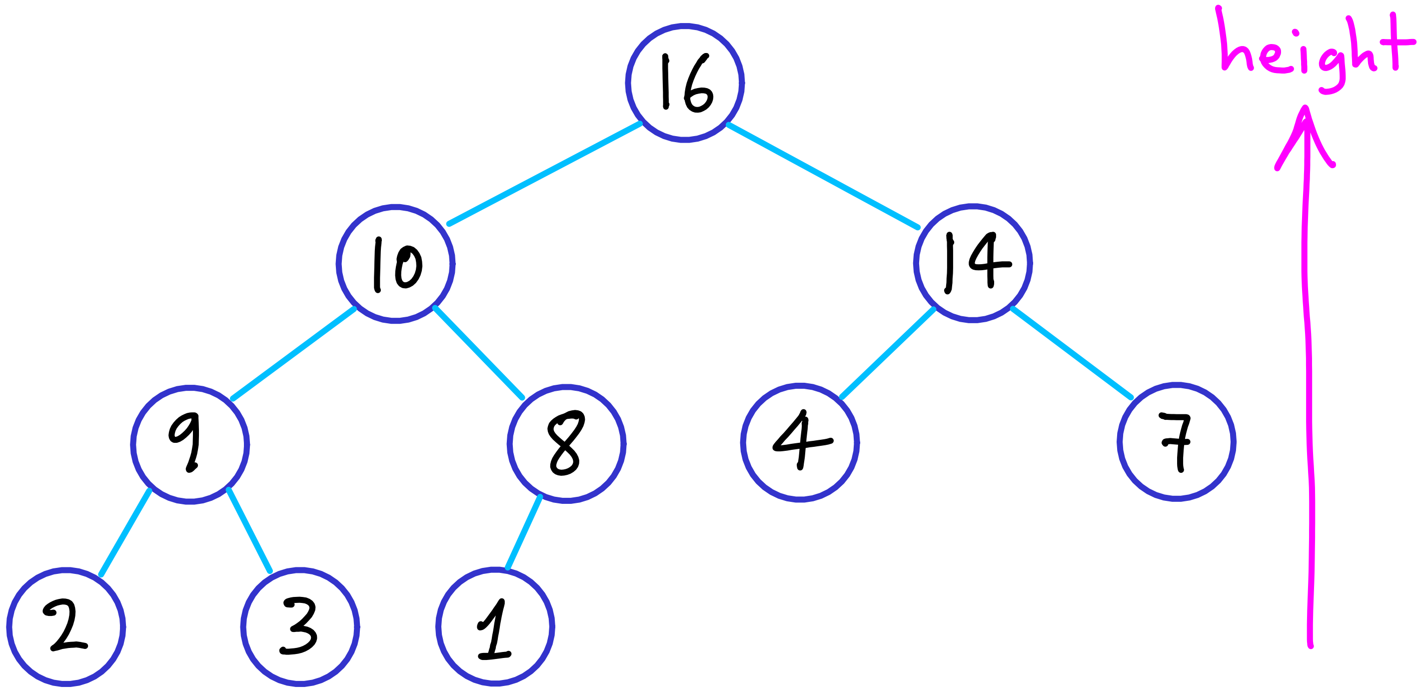
Heap building: the REVERSE METHOD (right to left)

1	2	3	4	5	6	7	8	9	10
16	10	14	9	8	4	7	2	3	1



Heap building: the REVERSE METHOD (right to left)

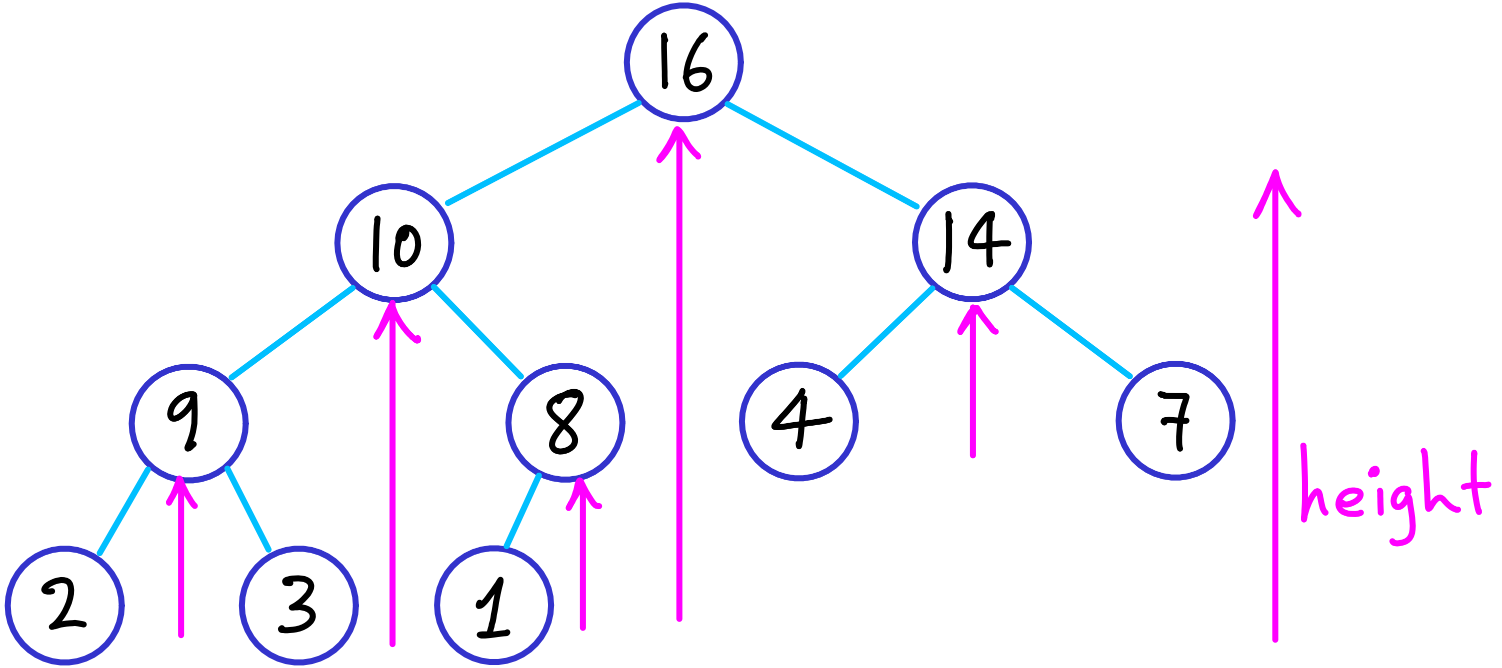
1	2	3	4	5	6	7	8	9	10
16	10	14	9	8	4	7	2	3	1



Time ?

$$\text{heapify}(x) = O(\text{height}(x))$$

$$\sum_{\text{all } x} \text{height}(x) = O(n \log n)$$



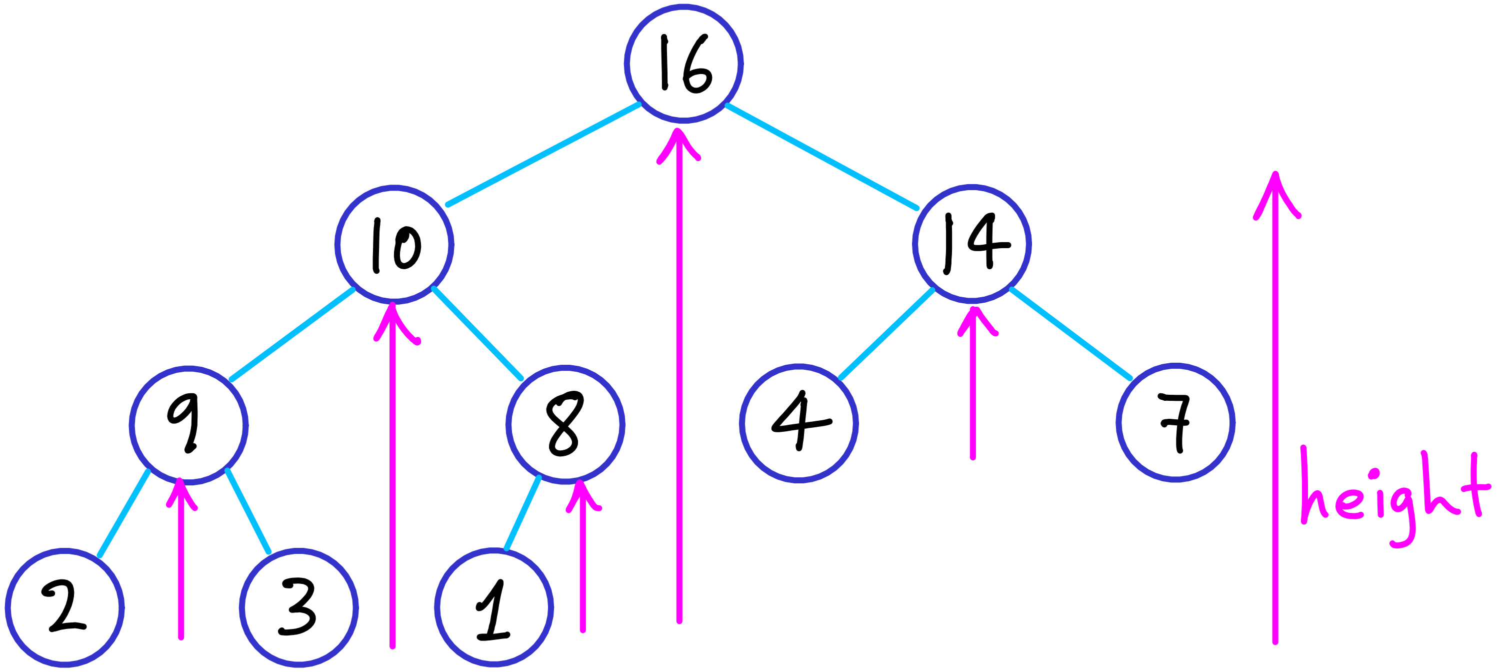
better calculation

$$\sum_{\text{all } x} \text{height}(x)$$

$$\sum \le \underbrace{\frac{n}{2}}_{\#\text{nodes}} \cdot \underbrace{1}_{\text{height}} + \underbrace{\frac{n}{4}} \cdot \underbrace{2} + \underbrace{\frac{n}{8}} \cdot \underbrace{3} + \cdots + \underbrace{2} \cdot \underbrace{((\log n) - 1)} + \underbrace{1}_{\#\text{nodes}} \cdot \underbrace{\log n}_{\text{height}}$$

lowest level

root level



better calculation

$$\sum_{\text{all } x} \text{height}(x)$$

$$\sum \leq \frac{n}{2}\cdot 1 + \frac{n}{4}\cdot 2 + \frac{n}{8}\cdot 3 + \cdots + 2\cdot((\log n)-1) + 1\cdot \log n$$

$$= \sum_{h=1}^{\log n} \frac{n}{2^h}\cdot h = n \cdot \sum \frac{h}{2^h} \leq n\,\frac{1/2}{(1-1/2)^2} = O(n)$$

CLRS 1148

[use $\sum_{0}^{\infty} kx^k$]