Data Structures

Heaps

CS 225

March 9, 2026

Brad Solomon

UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN

Department of Computer Science

Exam 3 (3/23 — 3/25)

Autograded MC and one coding question

Manually graded short answer prompt

Practice exam on PL soon*

Topics covered can be found on website

Registration started March 5

https://courses.engr.illinois.edu/cs225/exams/

Learning Objectives

Introduce the heap data structure

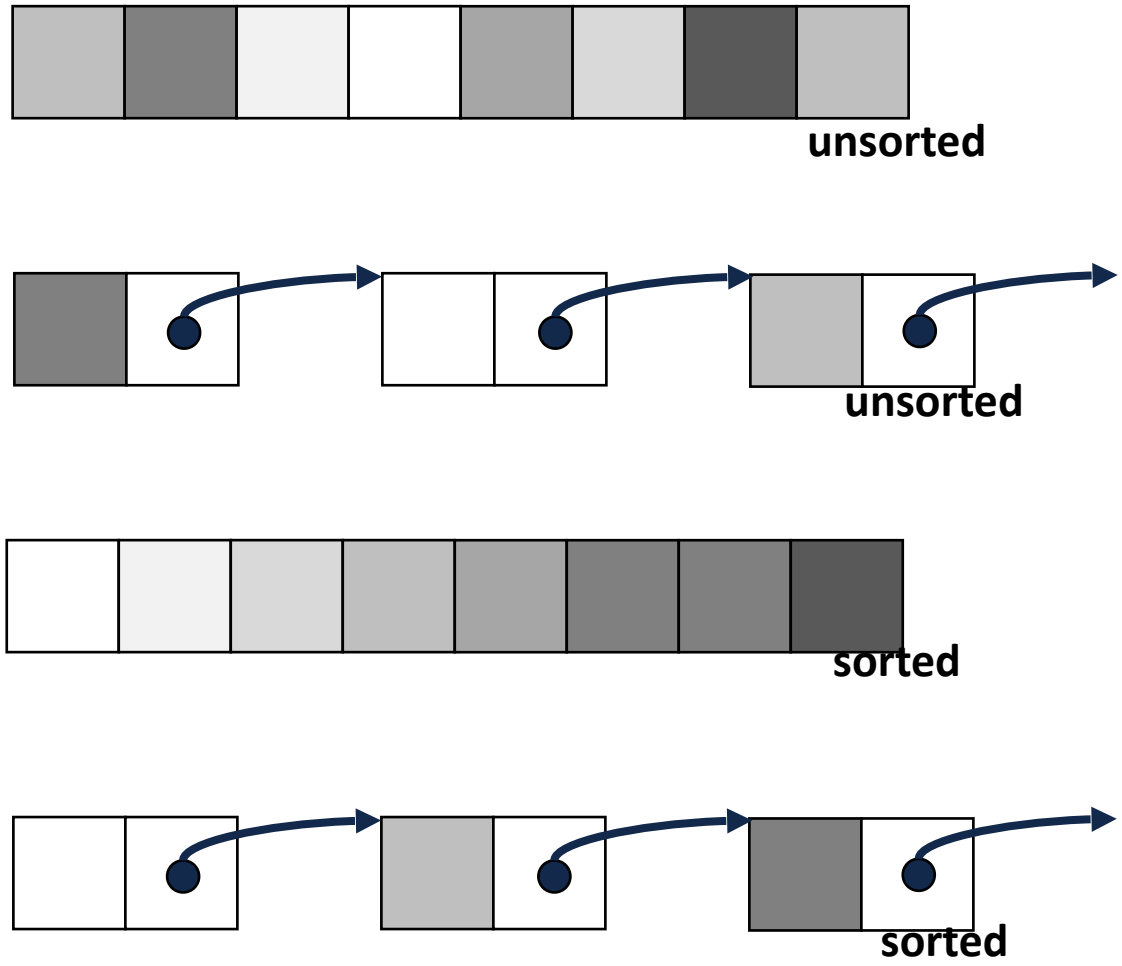
Discuss heap ADT implementations

Thinking conceptually: Sorting a queue

How might we build a ‘queue’ in which our front element is the min?

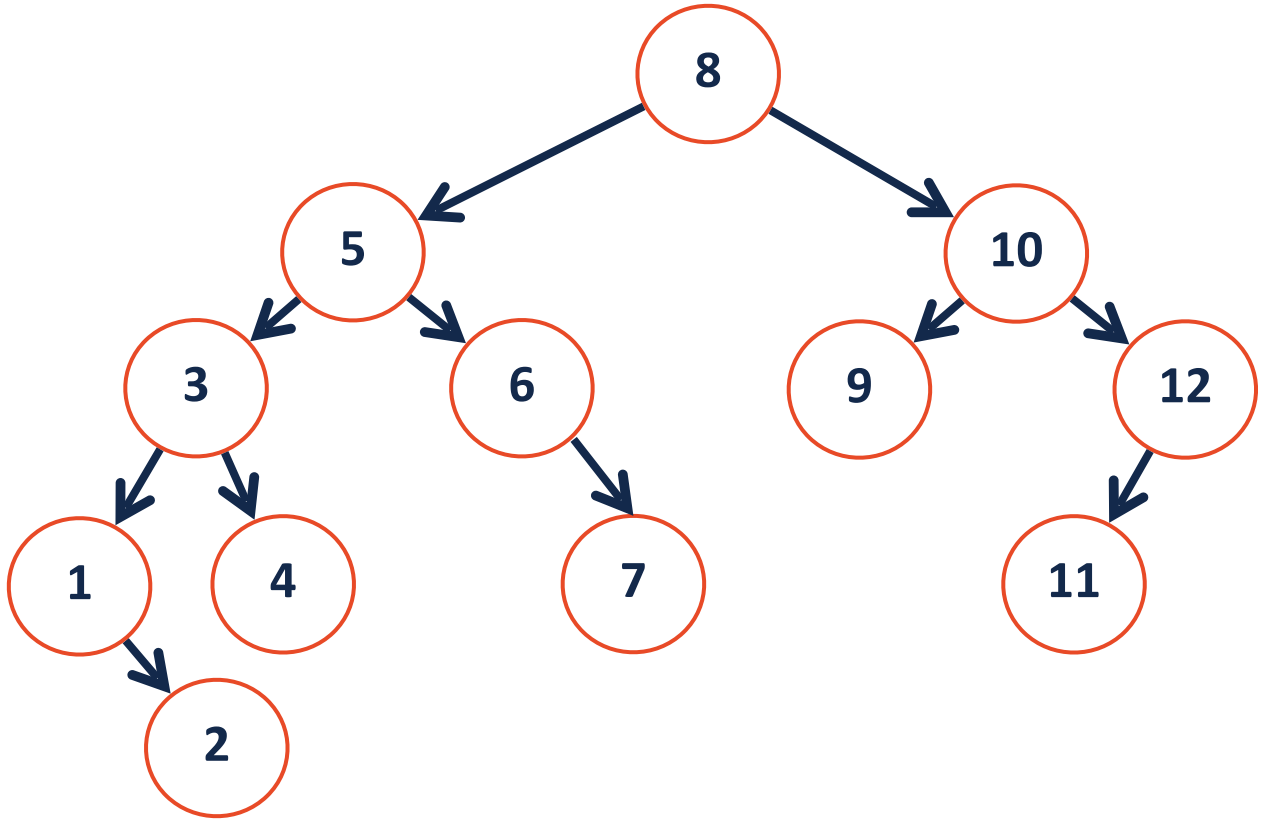
Priority Queue Implementation

insert	removeMin
O(1)*	O(n)
O(1)	O(n)
O(n)	O(1)
O(n)	O(1)

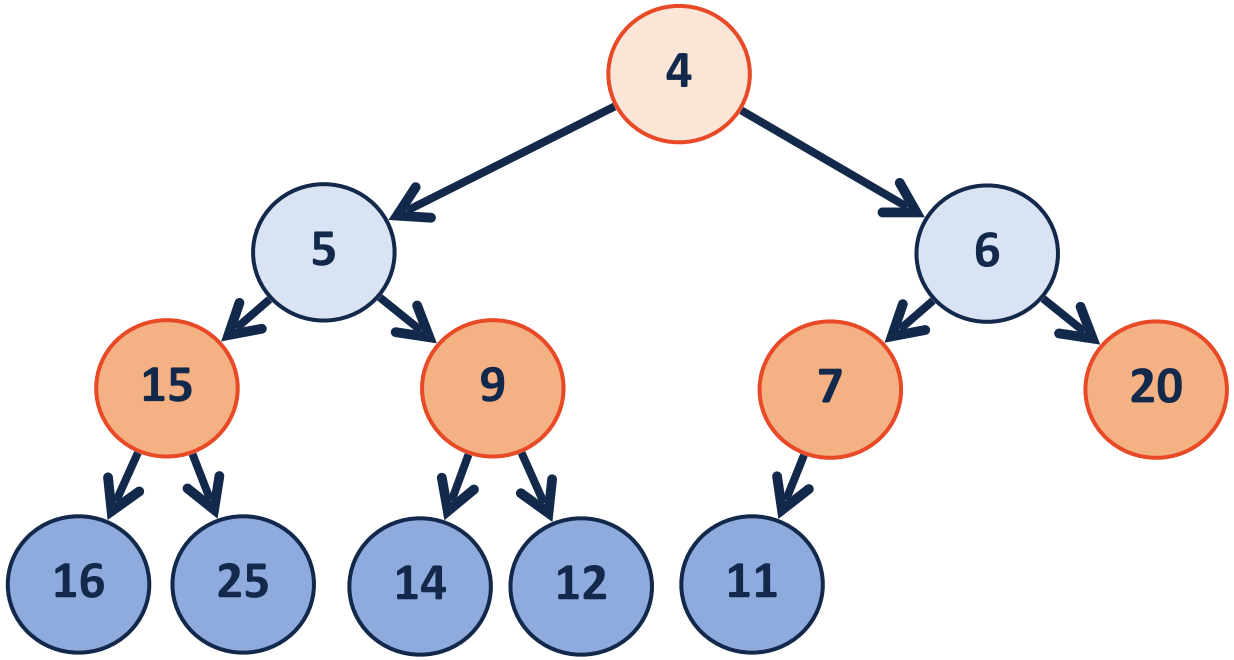


Priority Queue Implementation

insert	removeMin



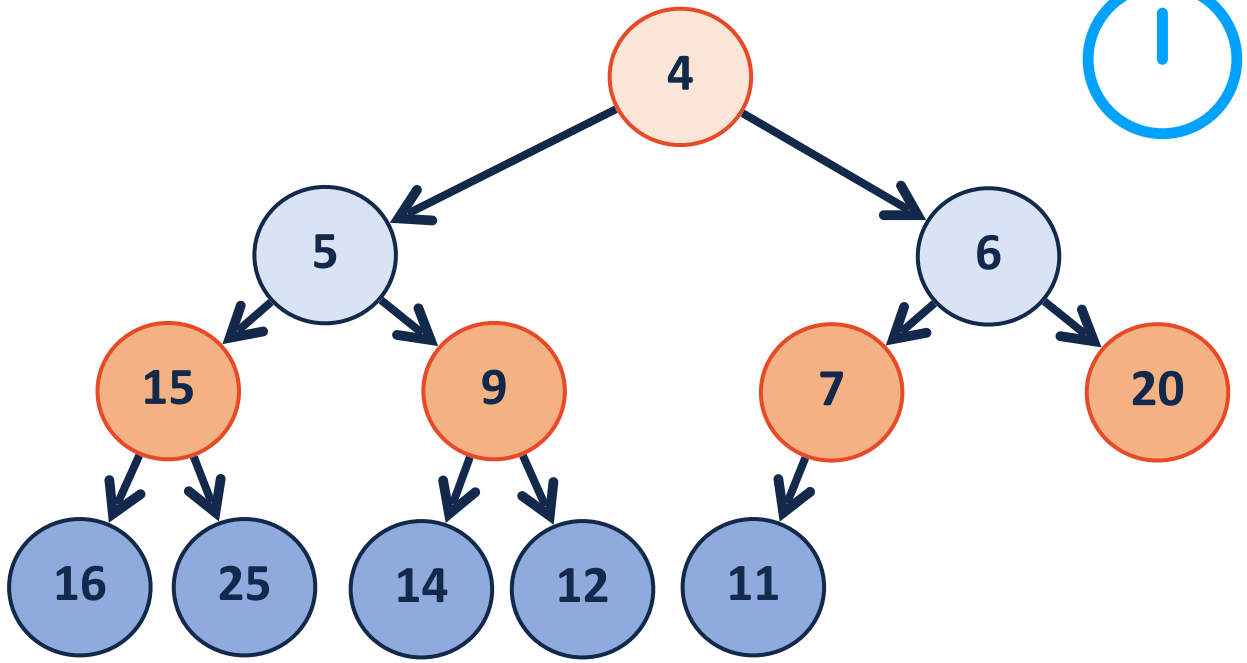
A different priority queue implementation…



(min)Heap

A complete binary tree T is a min-heap if:

- **T = {}** or
- **T = {r, T_L, T_R}**, where **r** is less than the roots of {T_L, T_R} and {T_L, T_R} are min-heaps.

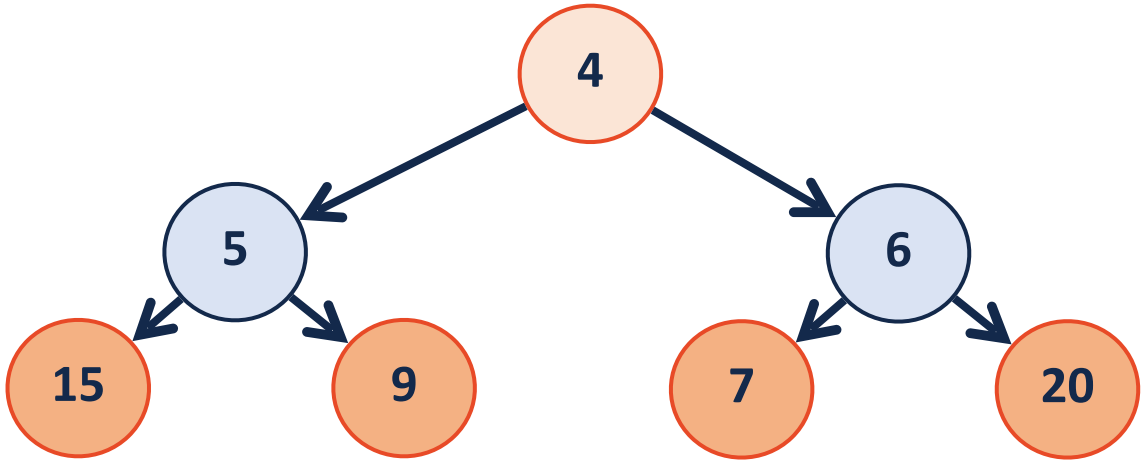


Thinking conceptually: A tree without pointers

What class of (non-trivial) trees can we describe without pointers?

What is the relationship between nodes and height for these trees?

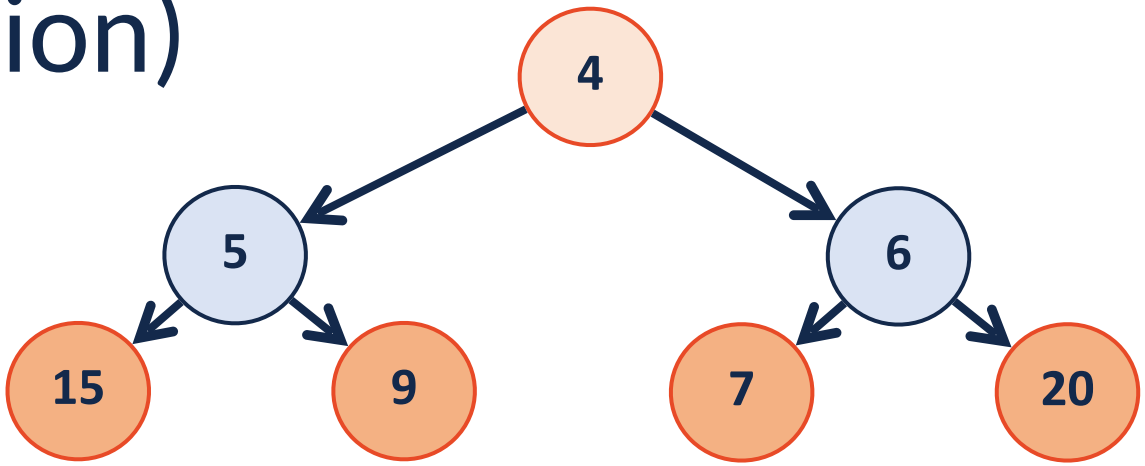
(min)Heap



4	5	6	15	9	7	20

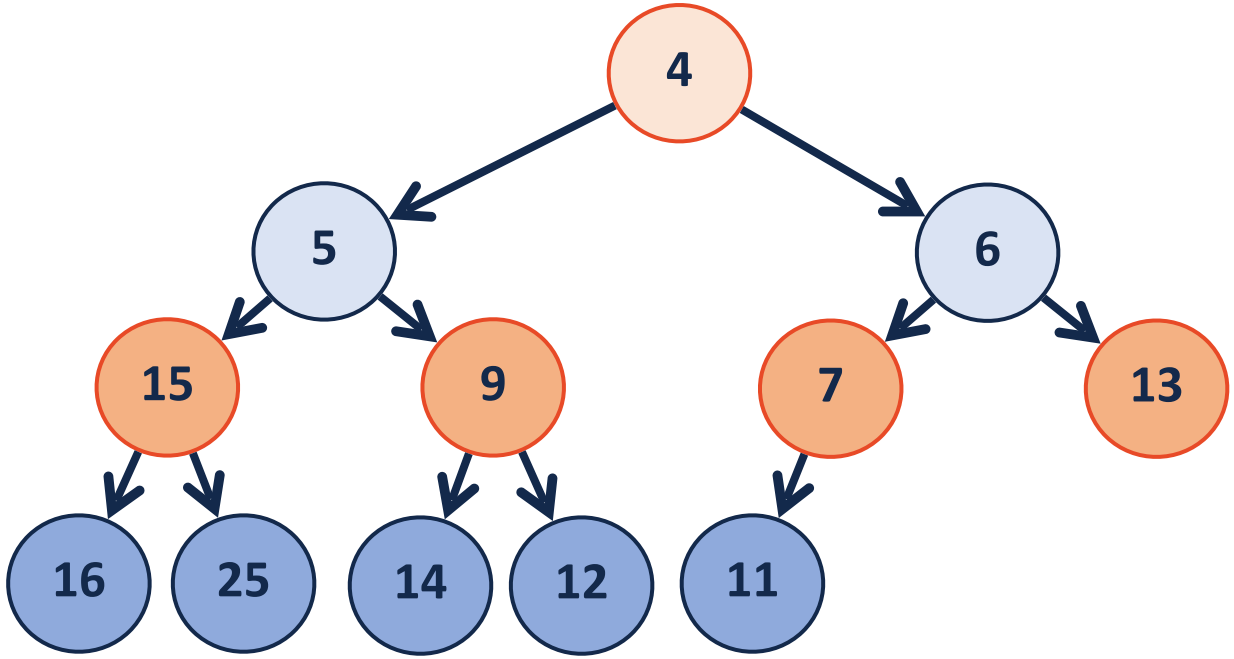
(min)Heap (Design Decision)

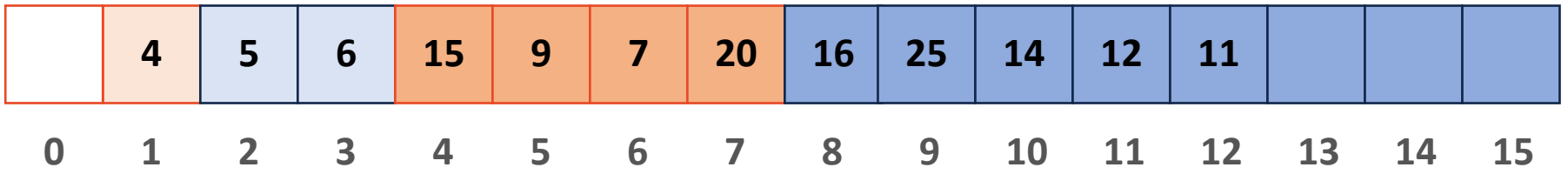
Would you rather have A or B?



A	4	5	6	15	9	7	20	B

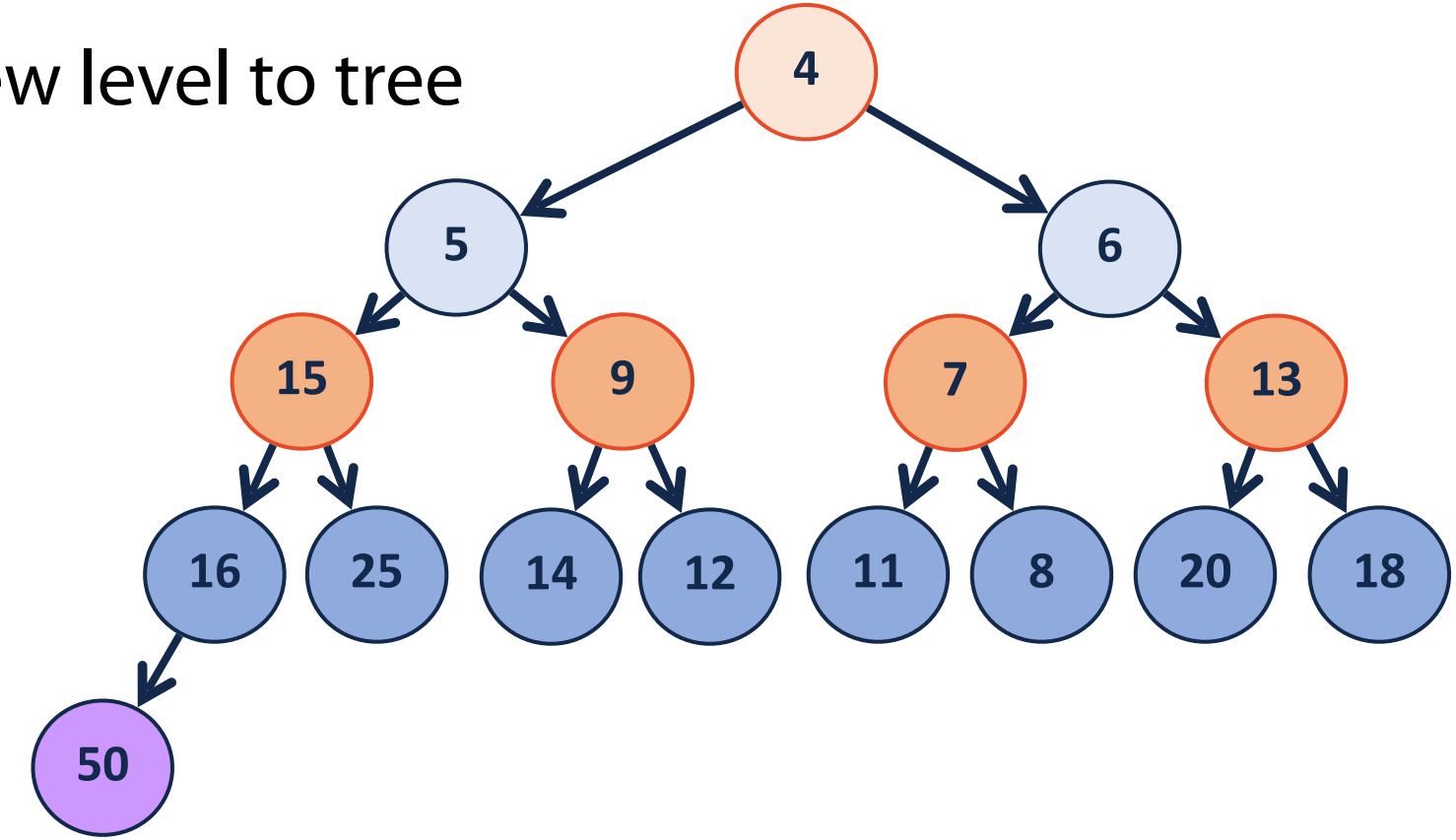
growArray

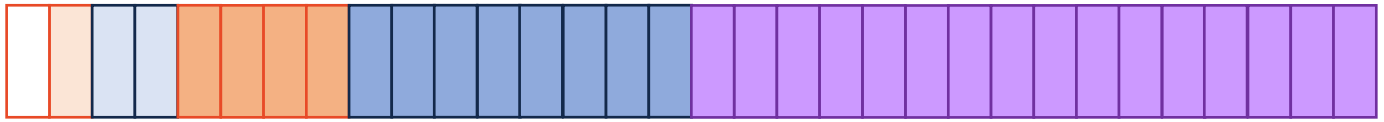

4
5
6
15
9
7
13
16
25
14
12
11


4
5
6
15
9
7
20
16
25
14
12
11
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

growArray

Array doubling adds a new level to tree



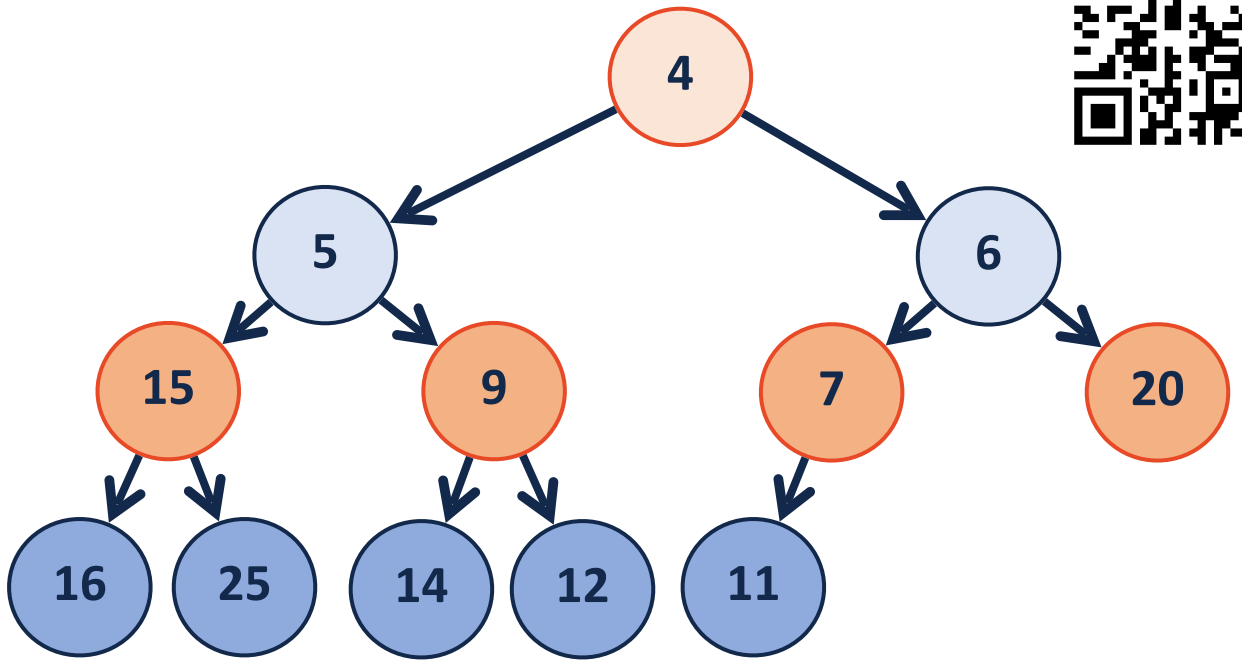


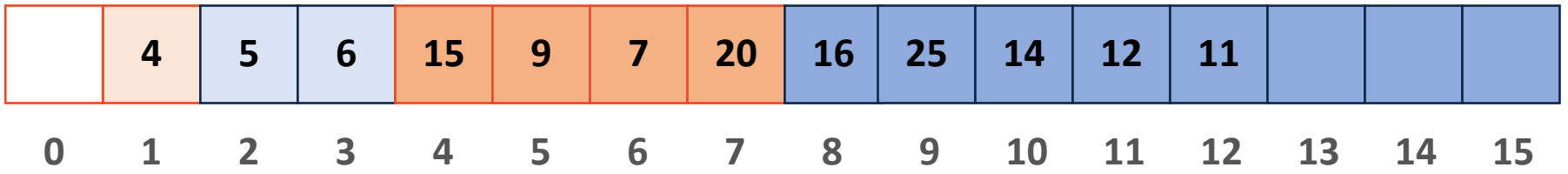
(min)Heap

Join Code: 225

`leftChild(i):`

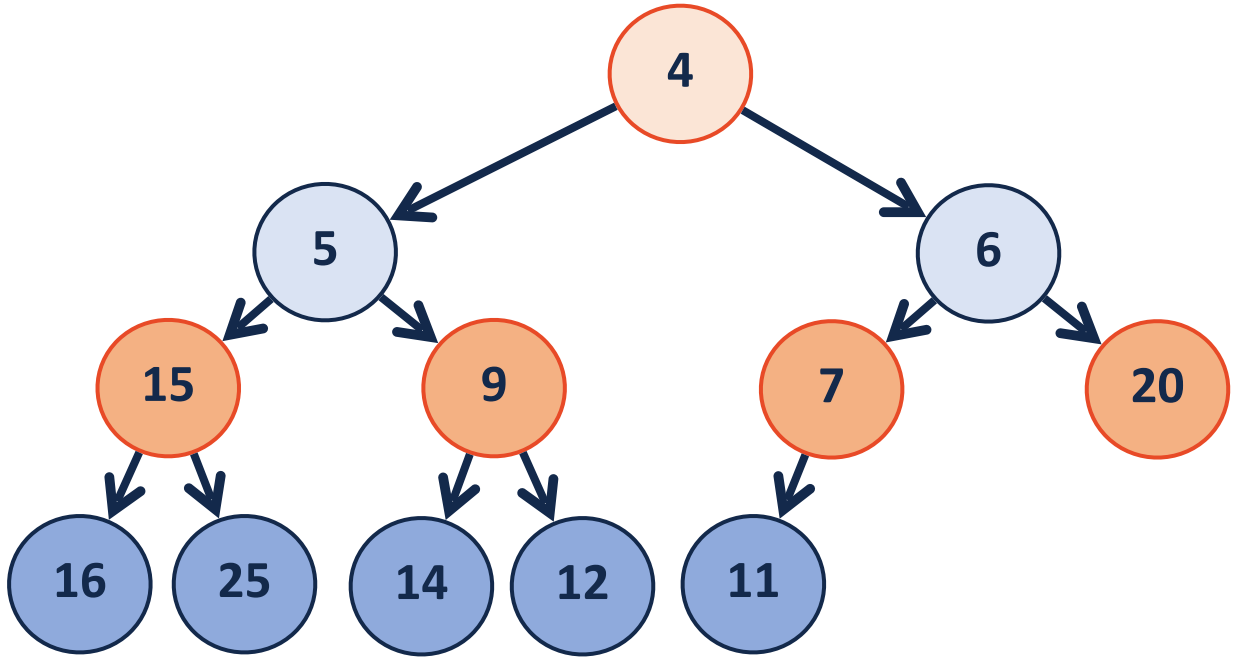
`rightChild(i):`





(min)Heap

`parent(i):`



	4	5	6	15	9	7	20	16	25	14	12	11			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

(min)Heap

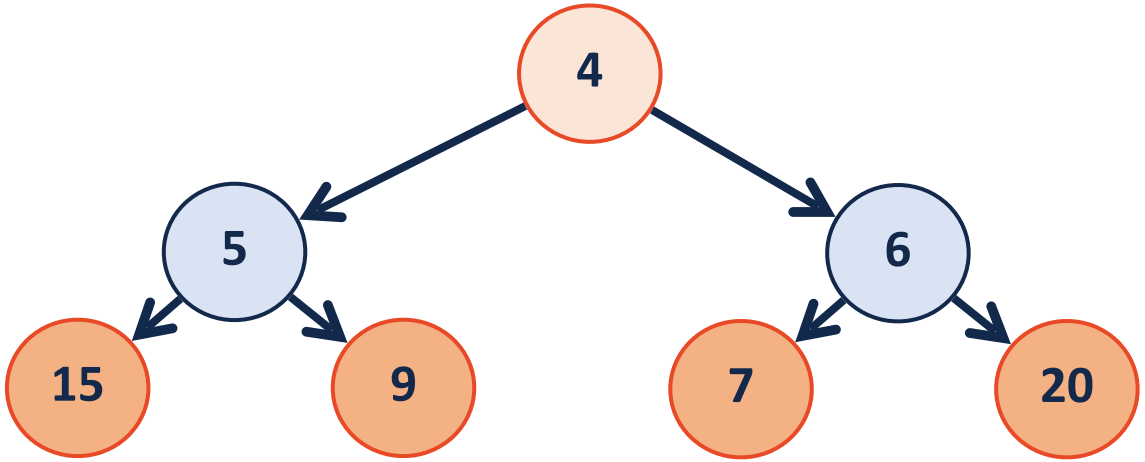
By storing as a complete tree, can avoid using pointers at all!

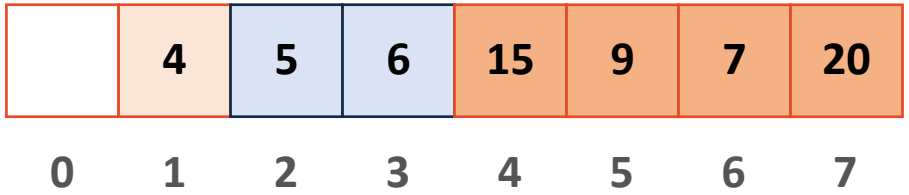
Can index from 0 or 1 (we will index from 1 in slides)

```
leftChild(i): 2i
```

```
rightChild(i): 2i+1
```

```
parent(i): floor(i/2)
```





(min)Heap ADT

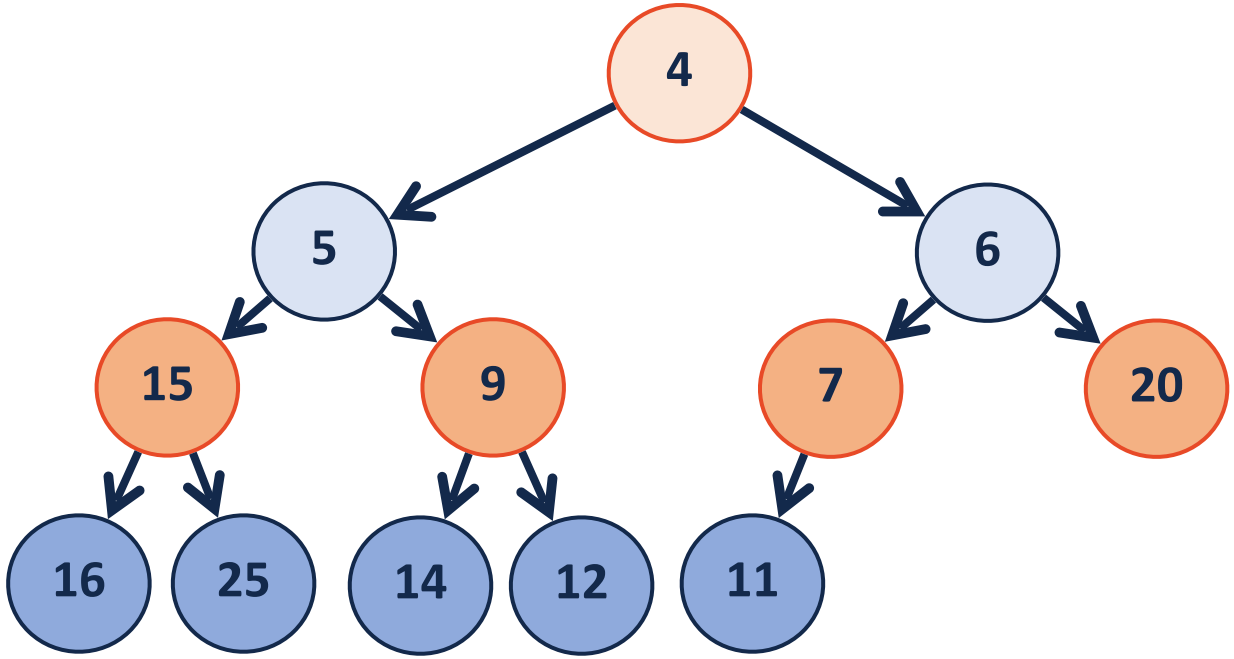
Insert

RemoveMin

Constructor

insert

Insert(8)



	4	5	6	15	9	7	20	16	25	14	12	11			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

insert

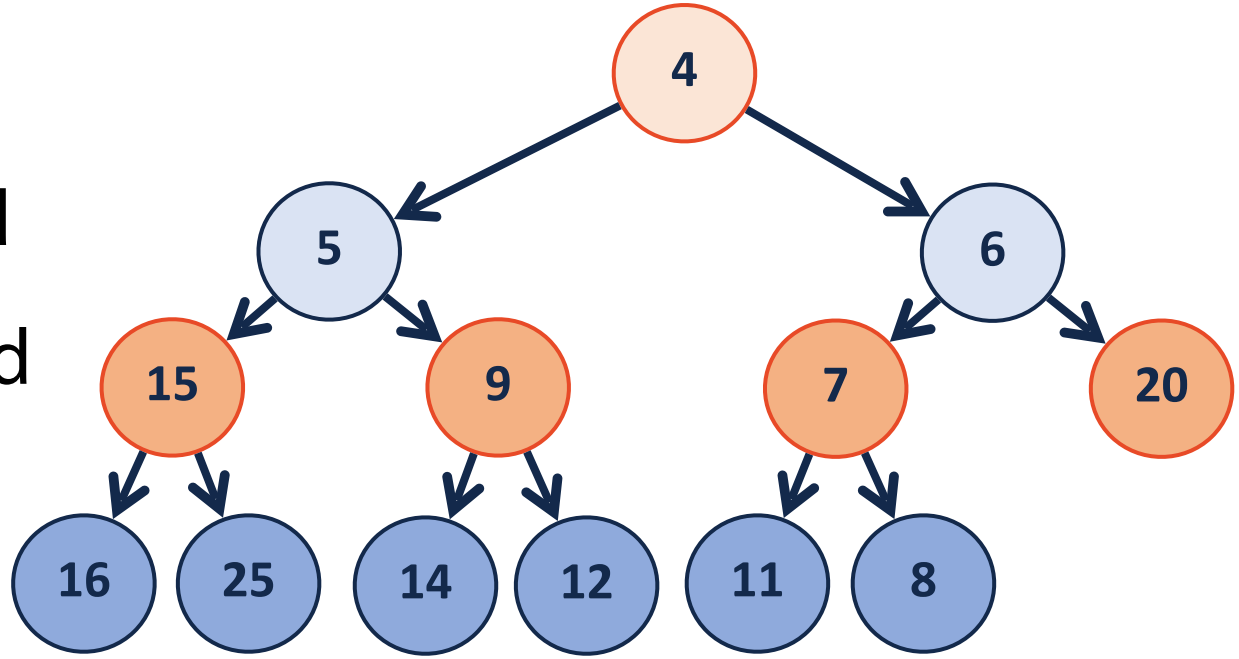
Insert(2)

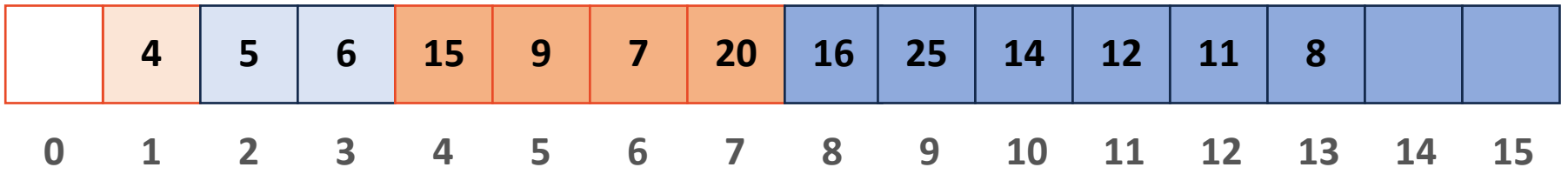
1) Insert at end of array

2) Check if minHeap still valid

3) Swap with parent if needed

Steps 2 and 3 are recursive!





insert

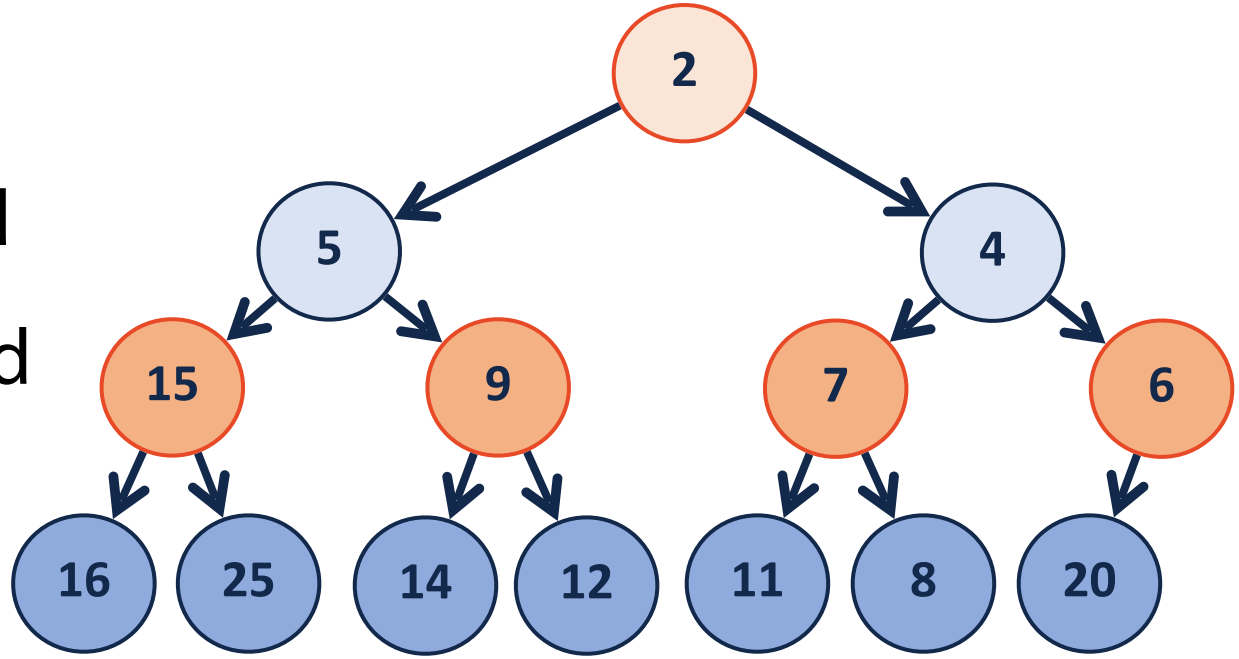
[After] Insert(2)

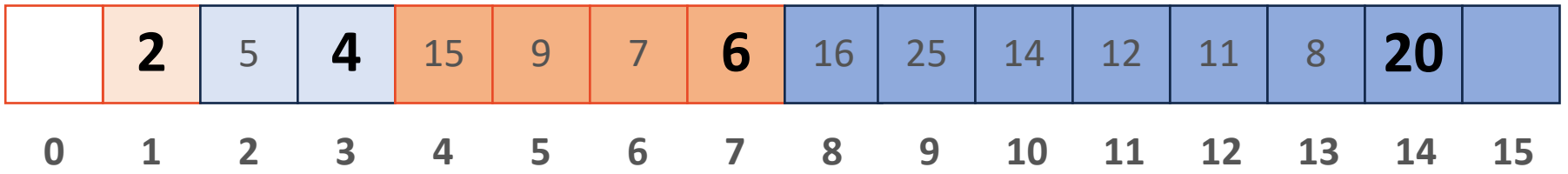
1) Insert at end of array

2) Check if minHeap still valid

3) Swap with parent if needed

Steps 2 and 3 are recursive!



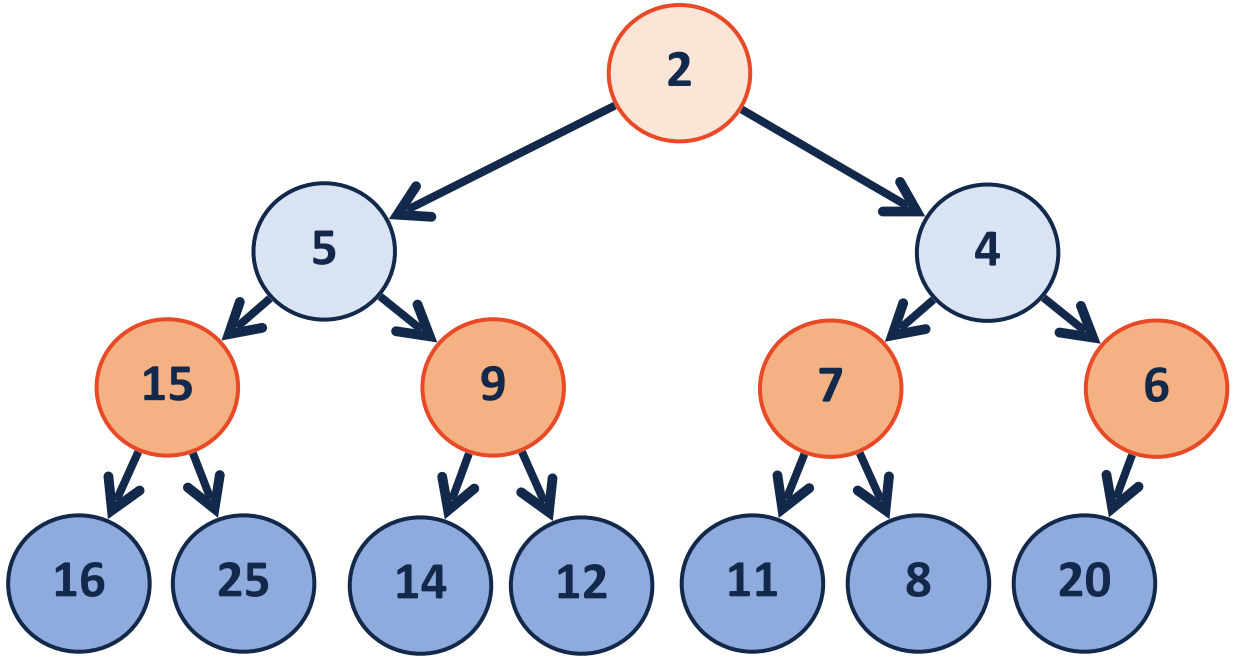


insert

[After] Insert(2)

What is my height?

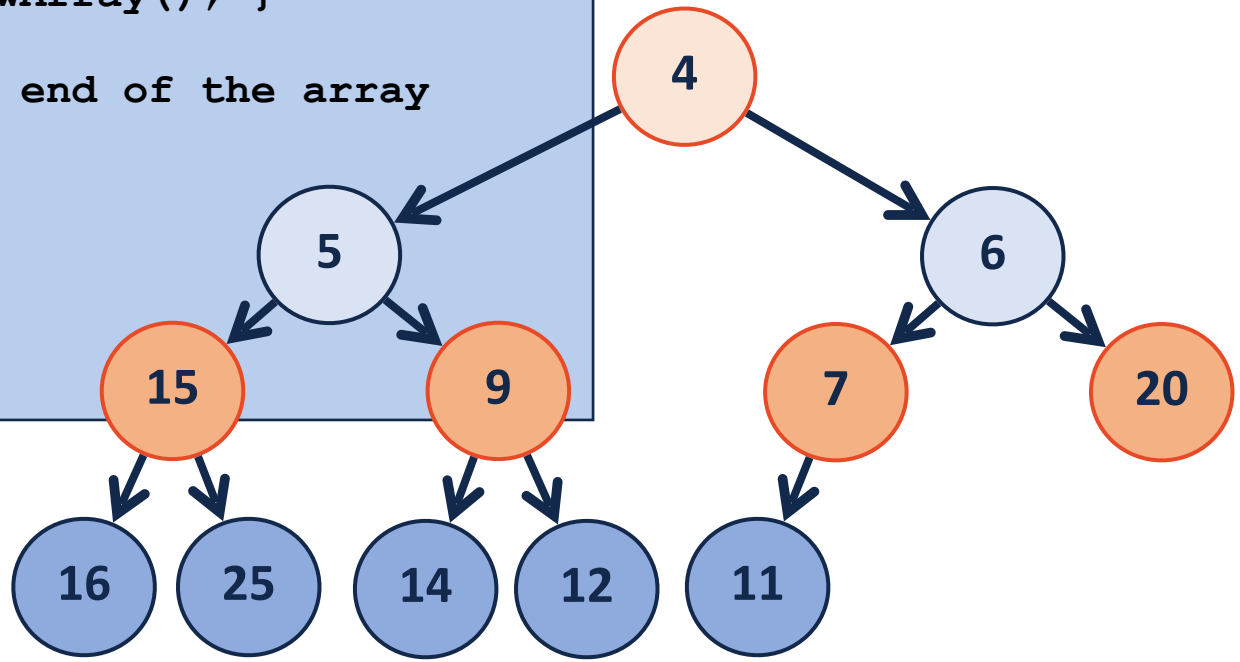
Number of swaps?



	2	5	**4**	15	9	7	**6**	16	25	14	12	11	8	**20**	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

insert

```
 1  template <class T>
 2  void Heap<T>::_insert(const T & key) {
 3    // Check to ensure there's space to insert an element
 4    // ...if not, grow the array
 5    if ( size_ == capacity_ ) { _growArray(); }
 6
 7    // Insert the new element at the end of the array
 8    item_[size_++] = key;
 9
10    // Restore the heap property
11    _heapifyUp(size_ - 1);
12  }
```



insert - heapifyUp

Join Code: 225


```
1  template <class T>
2  void Heap<T>::_insert(const T & key) {
3    // Check to ensure there's space to insert an element
4    // ...if not, grow the array
5    if ( size_ == capacity_ ) { _growArray(); }
6
7    // Insert the new element at the end of the array
8    item_[size_++] = key;
9
10   // Restore the heap property
11   _heapifyUp(size_ - 1);
12 }
```

```
1  template <class T>
2  void Heap<T>::_heapifyUp( ____________________________ ) {
3
4    if ( index > _____________ ) {
5      if ( item_[index] < item_[ parent(index) ] ) {
6         std::swap( item_[index], item_[ parent(index) ] );
7
8         _heapifyUp( ____________________ );
9      }
10   }
11 }
```