CS310
Data Structures

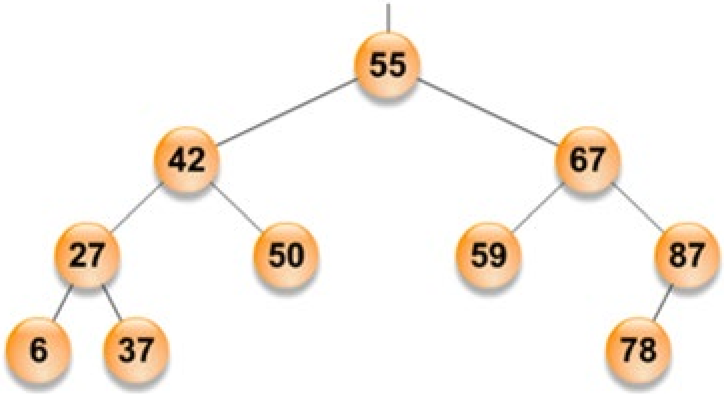
K. Raven Russell
krusselc@gmu.edu
George Mason University

Today

- **Last Lecture(s)**
  - Tree Traversals
- **Today**
  - More advanced data structures!
  - Heaps

New Schedule

“Lightning” Warm-up



30sec: Write the code for a linked binary tree node.

```
class Node<T> {
    T data;
    Node<T> left;
    Node<T> right;
}
```

30 sec: Draw the tree above stored as an array.

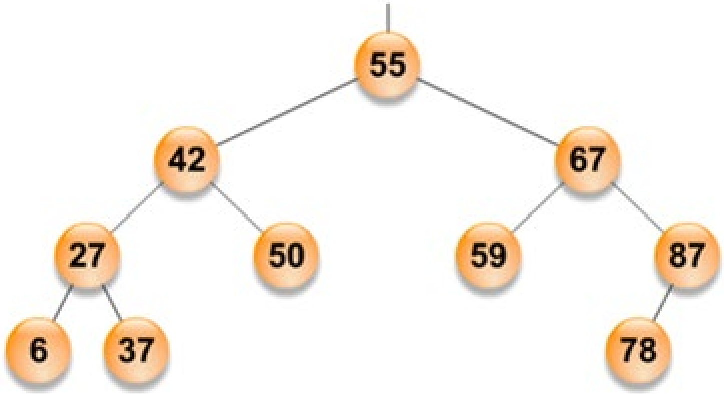
55	42	67	27	50	59	87	6	37	null	null	null	null	78	null

1 min: Show the result of printing with a "post order walk".

6,37,27,50,42,59,78,87,67,55

1 min: Write the recursive code for this walk.

```
void print(Node<T> n) {
    if(n == null) return;
    print(n.left);
    print(n.right);
    S.o.p(n.data);
}
```



30sec: Write the code for a linked binary tree node.

```
class Node<T> {
    T data;
    Node<T> left;
    Node<T> right;
}
```

30 sec: Draw the tree above stored as an array.

55	42	67	27	50	59	87	6	37	null	null	null	null	78	null

1 min: Show the result of printing with a "post order walk".

6,37,27,50,42,59,78,87,67,55

1 min: Write the recursive code for this walk.

```
void print(T[] tree, int index) {
    if(index >= tree.length) return;
    if(tree[index] == null) return;
    print(tree, (index*2)+1);
    print(tree, (index*2)+2);
    S.o.p(tree[index]);
}
```

Trees at Work #1: Heaps

(NOT related to memory heaps, sorry!)

Priority Queues and Sorting

Priority Queue Intro

– n = number of items put in the queue

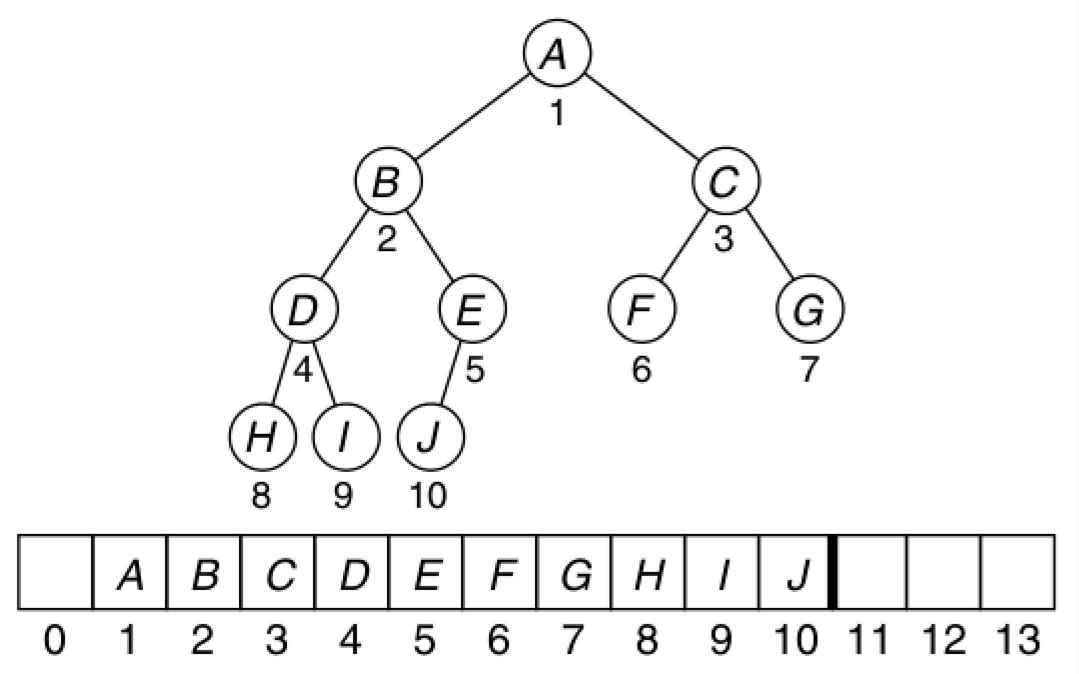
Operation Implementation	add	remove (max/min)	peek (max/min)
Unordered List	O(1)	O(n)	O(n)
Sorted Array List	O(n)	O(1)	O(1)

Heaps

- A type of tree!
  - Usually binary when learning... K-ary professionally
- Relationship maintained between... parent and child
- Operations:
  - Deleting items removes the root (“top” item)
  - Adding items adds to the “end” / “bottom”
  - Items “swim” up and “sink” down to maintain order
- Maintains two properties
  - structure property
  - heap order property

Heap Structure

- Want the logical tree represented by the heap to be balanced
- a "nearly complete binary tree"



Note: some books have 1 as the root, we're using 0, but this image is from a book that uses 1

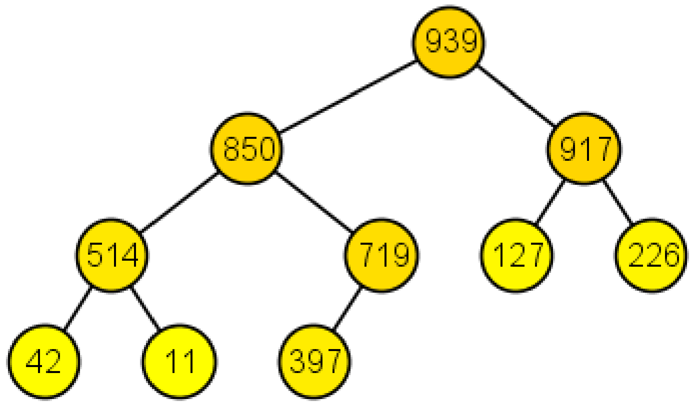
Heap Order Property

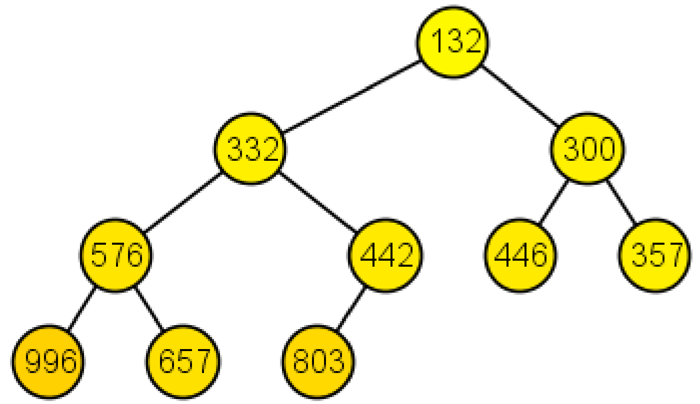
Weiss 21.1.2

- Max Heap
  - node is always larger than (or equal to) any of its descendants
- Min Heap
  - node is always smaller than (or equal to) any of its descendants
- Idea: we want to find max (or min) quickly
  - keep at the root of the tree
  - every subtree should have the largest (or smallest) item at the subtree root

Heap Order Property

– Are either of these a heap?





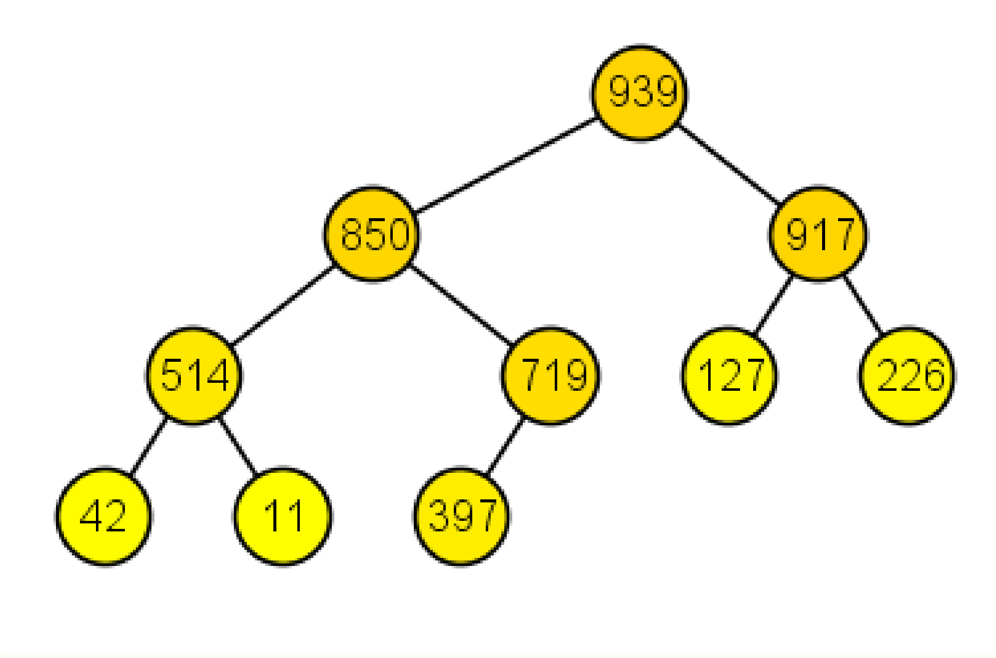
Heap Demo

(Whiteboard → GnarlyTrees)
add & remove

(Max) Heap Implementation

- Insert
  - adding item at end of the logical tree
  - increment the size
  - likely violates the heap order property
    - *fix by swimming the value up the tree*
- Delete (the **max value**)
  - remove item at start of the logical tree
  - decrement size
  - violates the heap structure property
    - *fix by swapping the last value and the root value*
      - likely violates the heap order property
        - *sink the new root value down the tree*
    - *null out the last value (prevent loitering)*

How Tall Are Trees?



Level 0

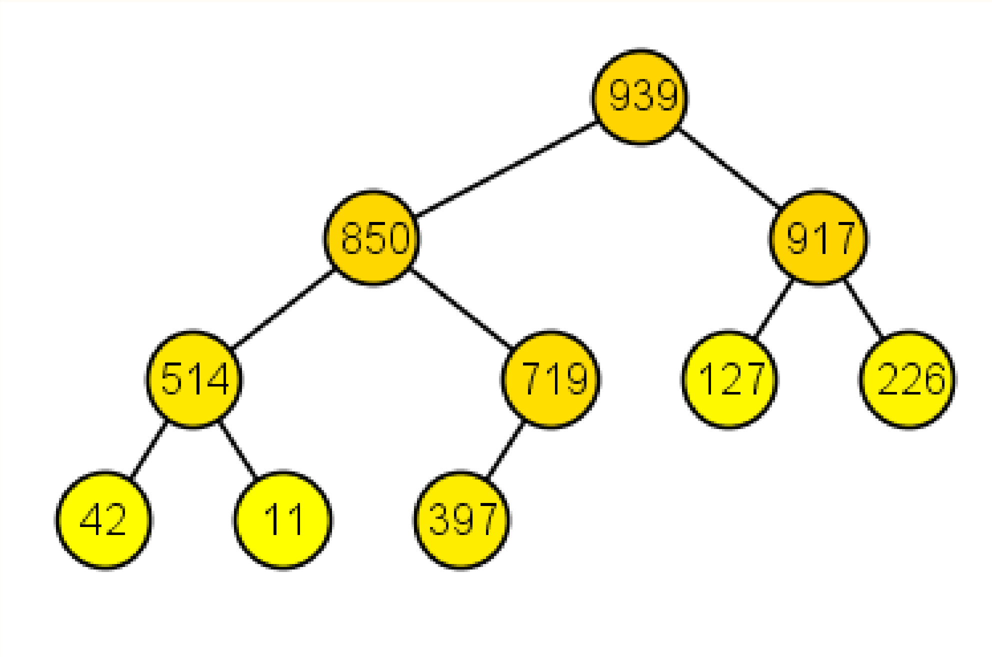
Level 1

Level 2

Level 3

Height = 3

How Tall Are Heaps?



← Fits 1 item

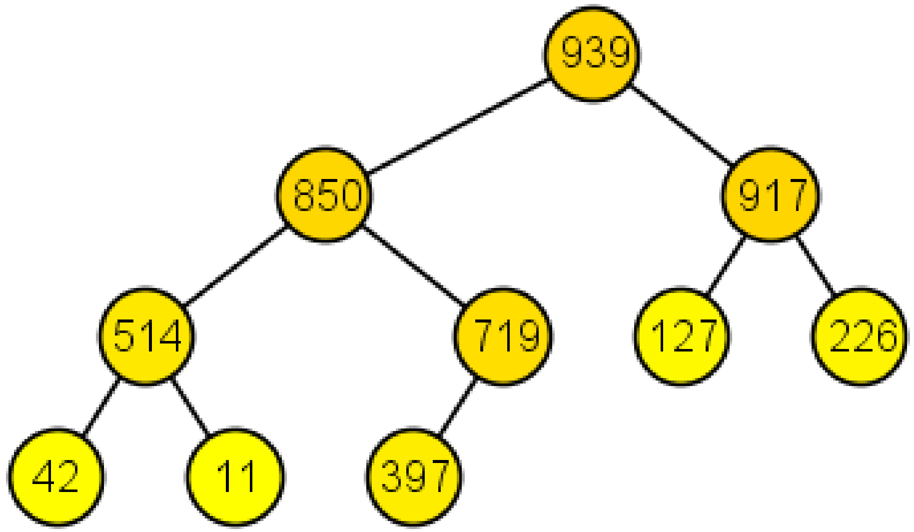
← Fits 2 items

← Fits 4 items

← Fits 8 items

How Tall Are Heaps?

Remember, $\lg(n)$ is just used to question: $2^? = n$ or
"how many times do I need to multiply by 2 to get n?"



← Fits 1 item (2^0)

← Fits 2 items (2^1)

← Fits 4 items (2^2)

← Fits 8 items (2^3)

So if I need to fit n nodes in the tree, I'll need this to be true:

$$2^{h+1} - 1 \geq n$$

And h must be $O(\lg n)$

Each level fits 2^h items

$$\sum_{k=0}^{h} 2^k = 2^{h+1} - 1$$

Priority Queue Summary

– n = number of items put in the queue

– m = number of priorities

Operation Implementation	add	remove (max/min)	peek (max/min)
Unordered List	?	?	?
Ordered List	?	?	?
Binary Heap	?	?	?

Priority Queue Summary

- n = number of items put in the queue
- m = number of priorities

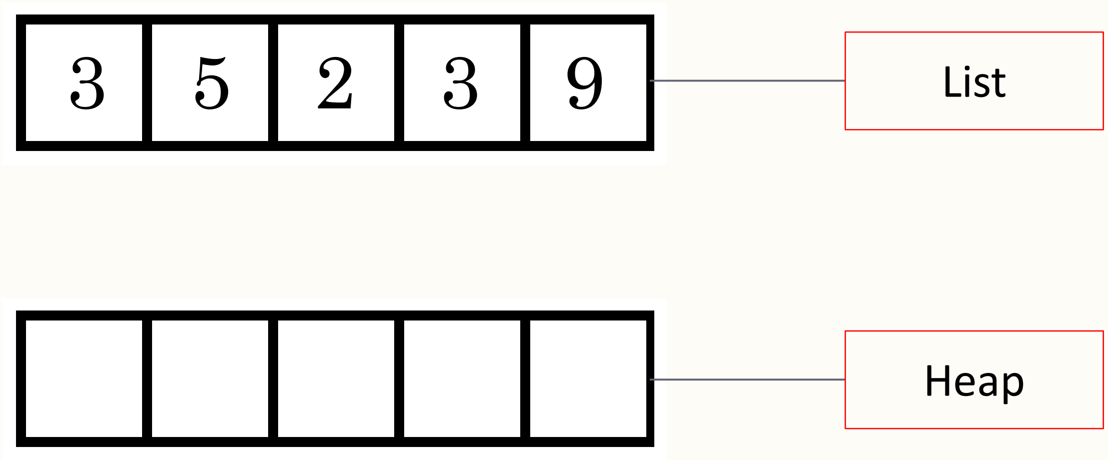
Operation Implementation	add	remove (max/min)	peek (max/min)
Unordered List	O(1)	O(n)	O(n)
Ordered List	O(n)	O(1)	O(1)
Binary Heap	O(lg n)	O(lg n)	O(1)

Heap Practice

Add the following numbers onto a max heap and show the steps when you delete the max 5 times. Draw the ARRAY at each step.

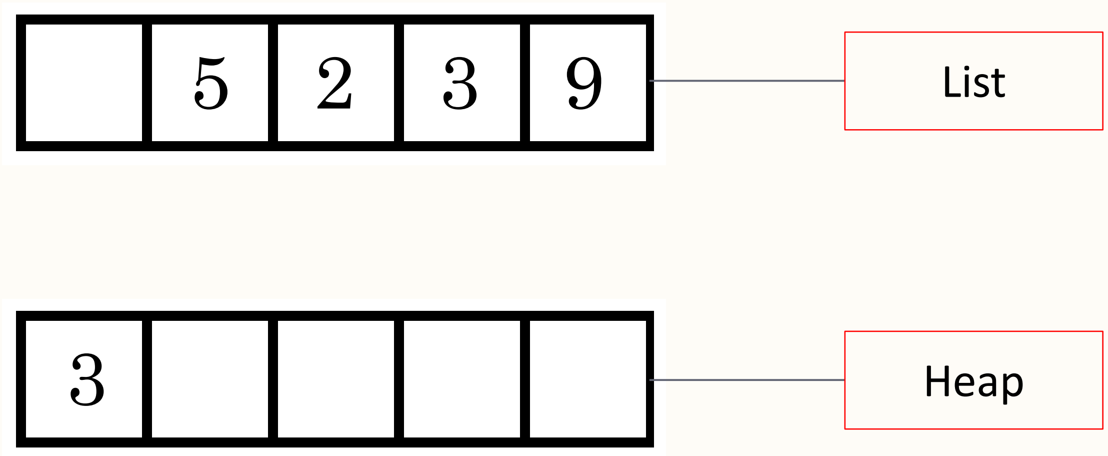
3, 5, 2, 3, 9

Heap Practice



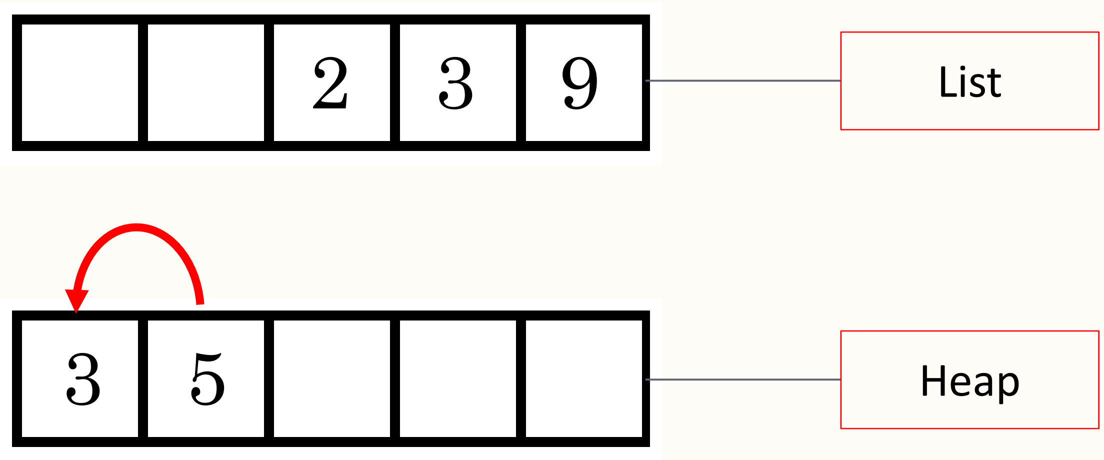
Heap Practice

Put 3
On Heap



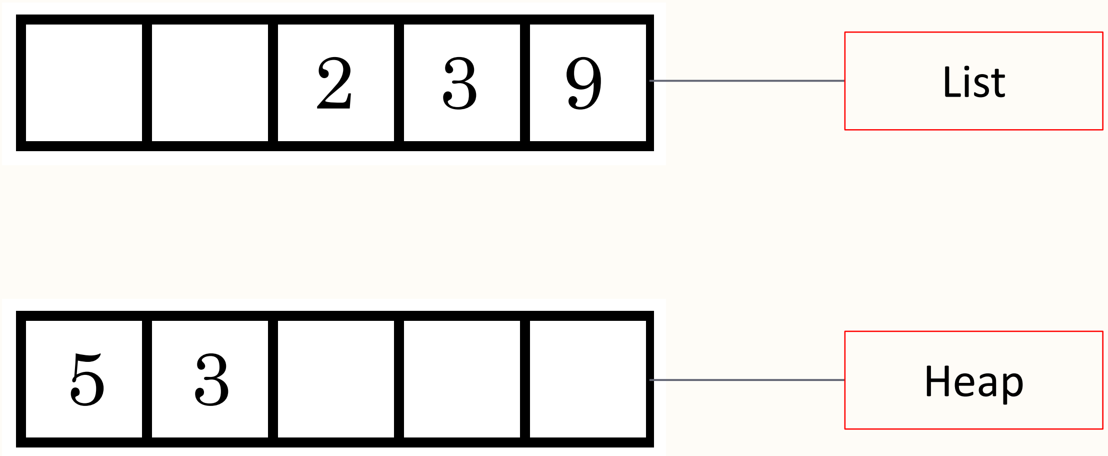
Heap Practice

Put 5
On Heap
(and fix)



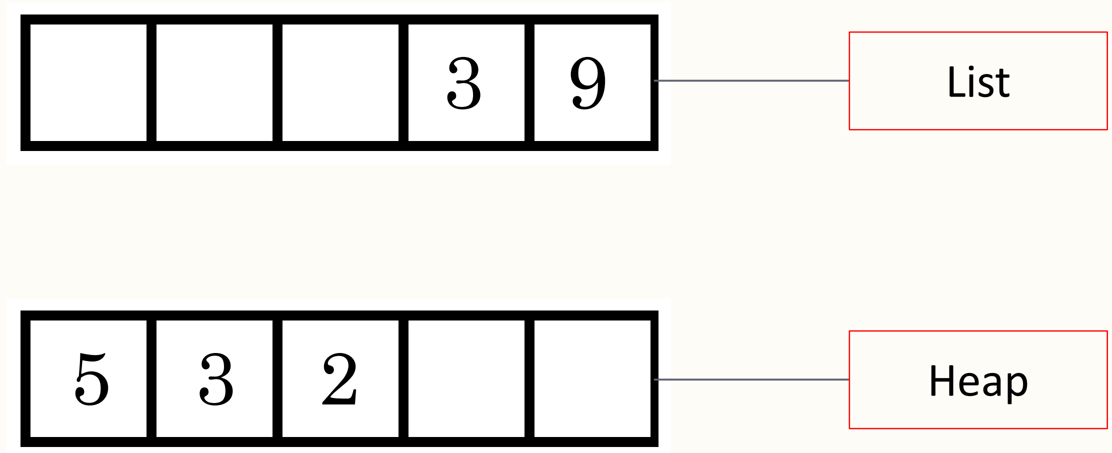
Heap Practice

Put 5
On Heap
(and fix)

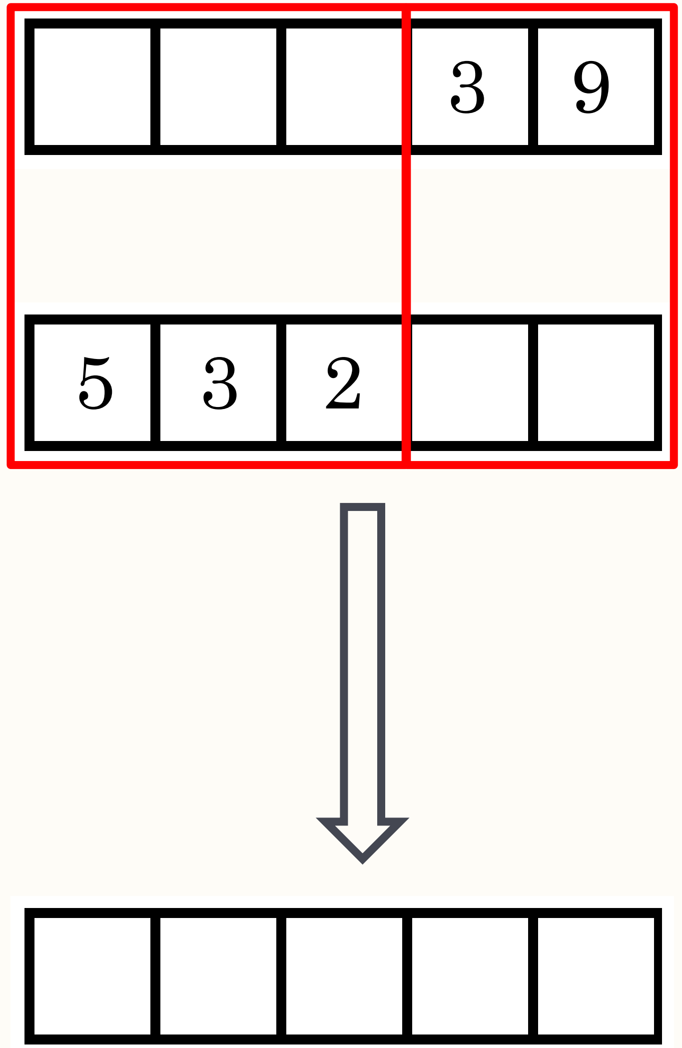


Heap Practice

Put 2
On Heap
(no fix)

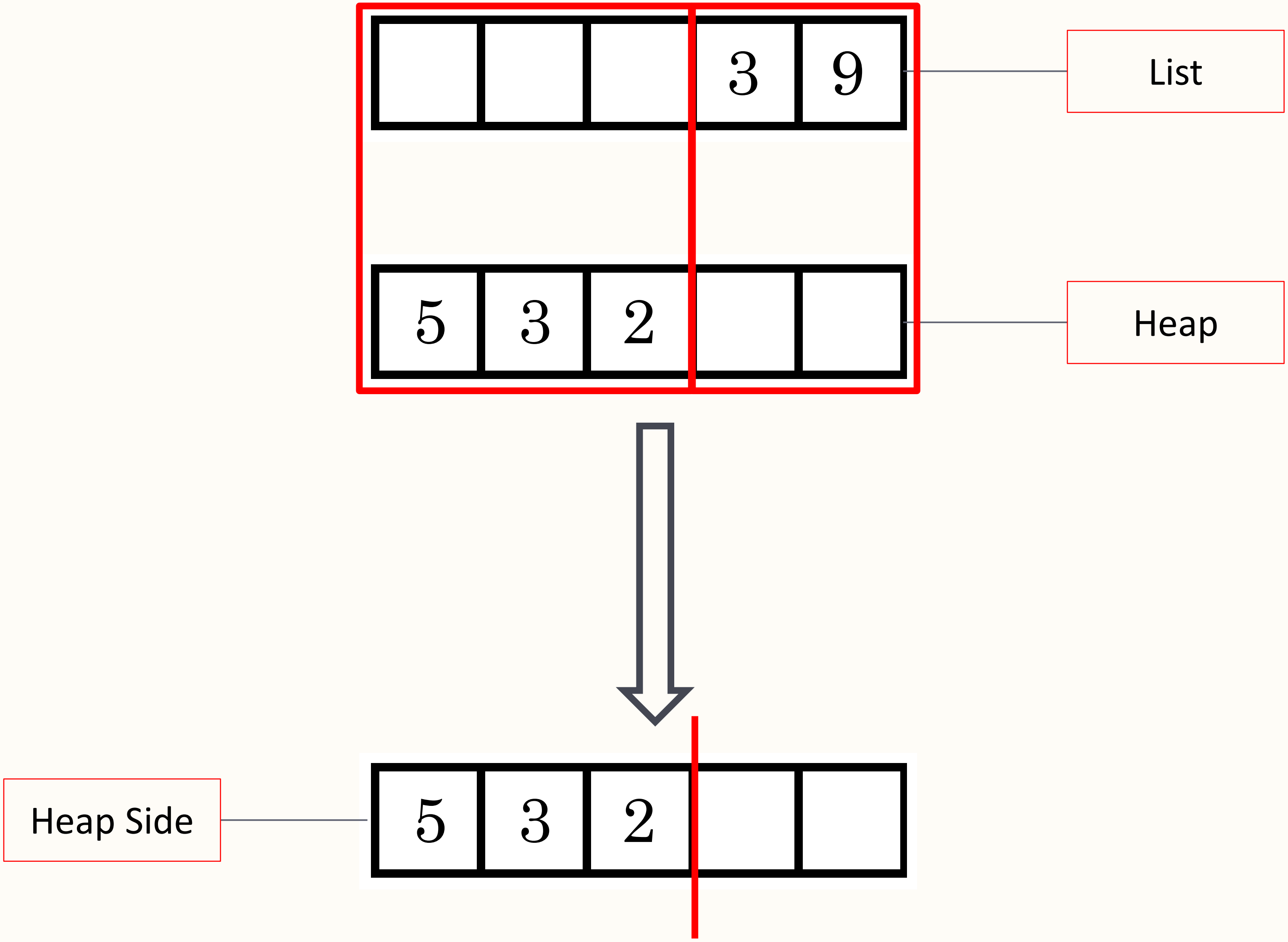


Heap Practice

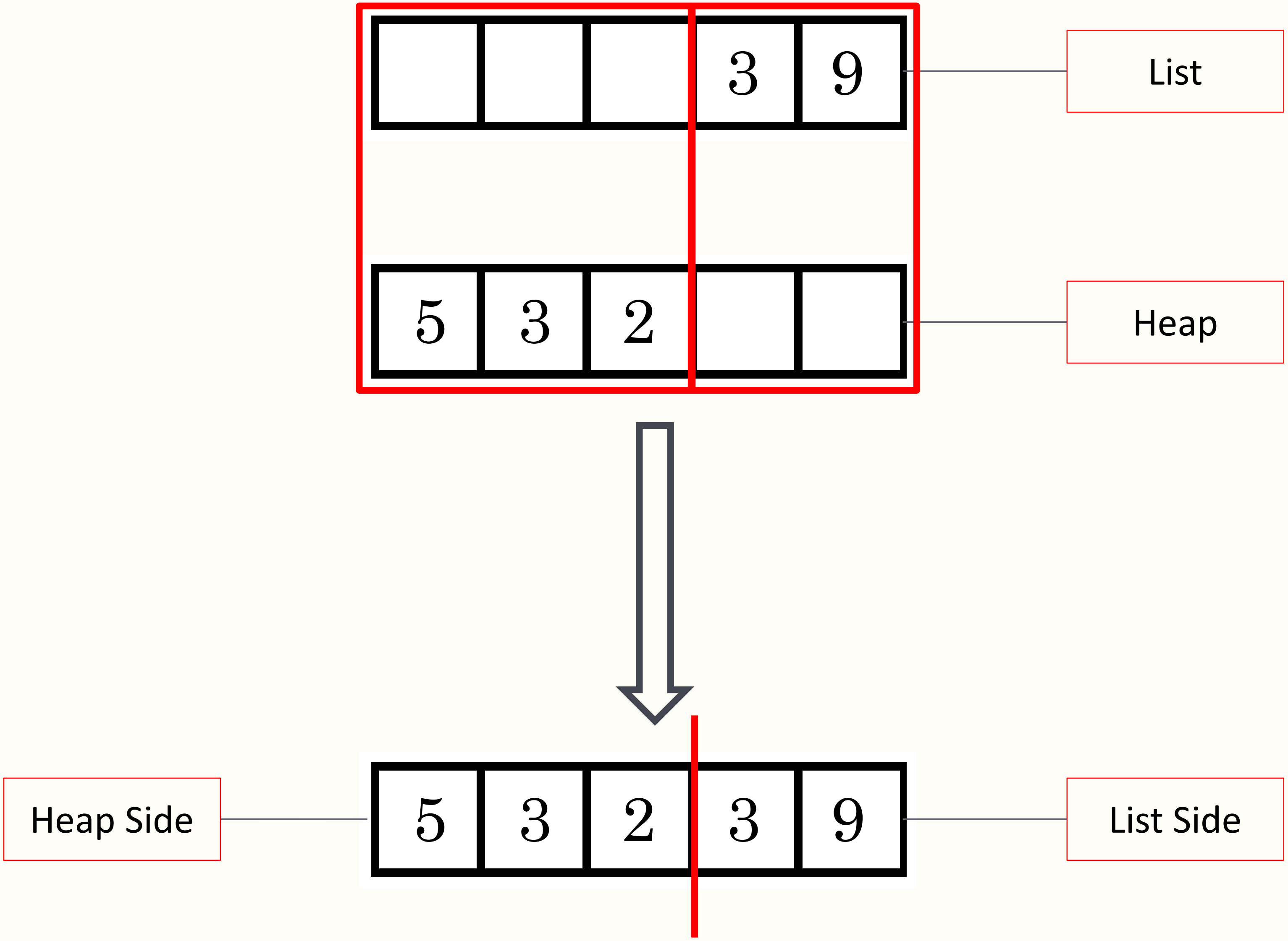


What do you notice about these two arrays?

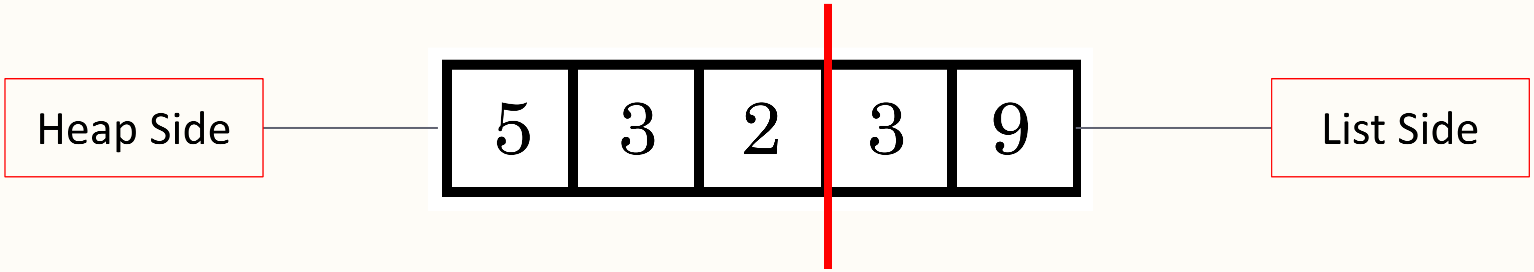
Heap Practice



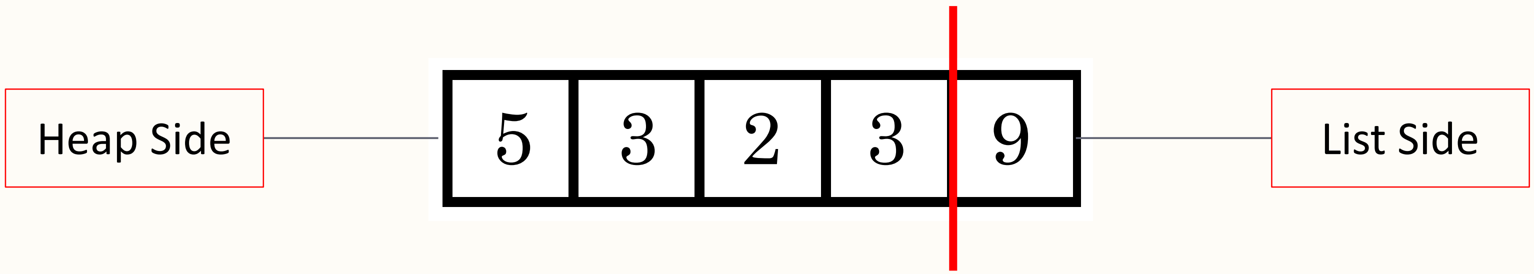
Heap Practice



Heap Practice

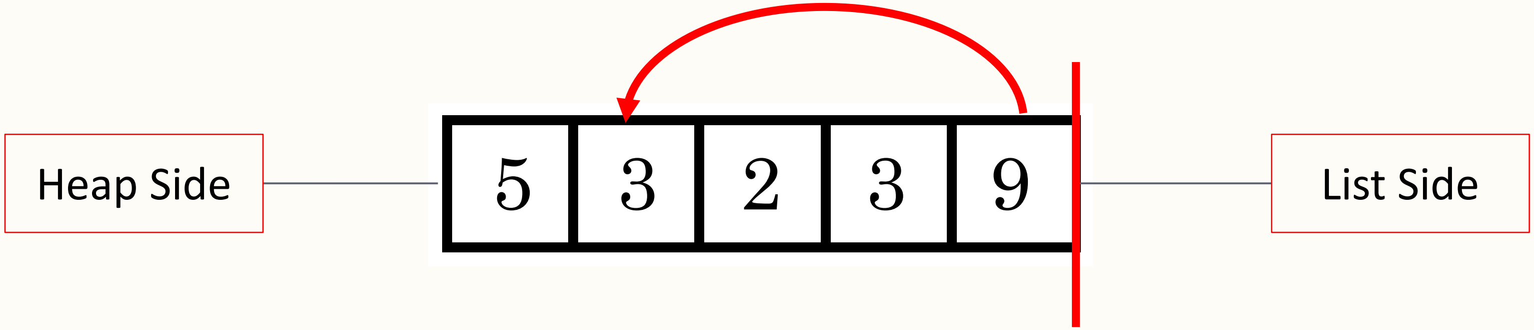


Heap Practice



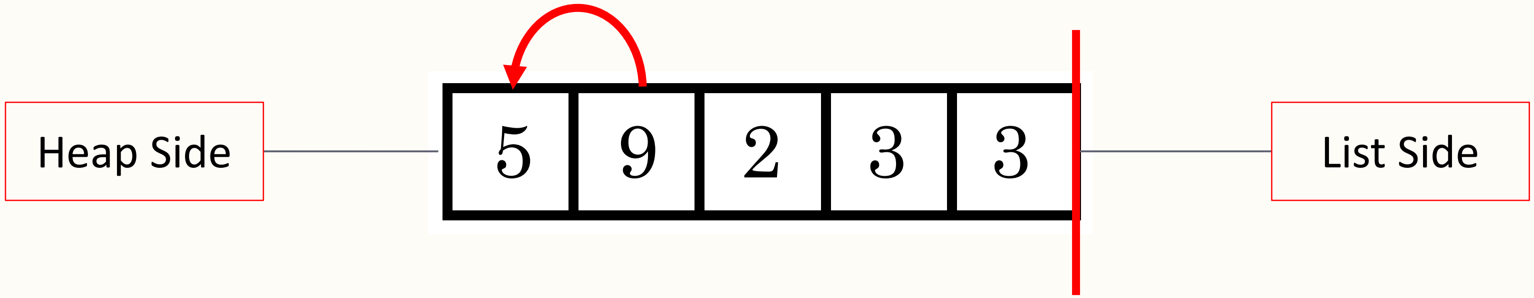
Put 3
On Heap
(no fix)

Heap Practice



Put 9
On Heap
(and fix)

Heap Practice



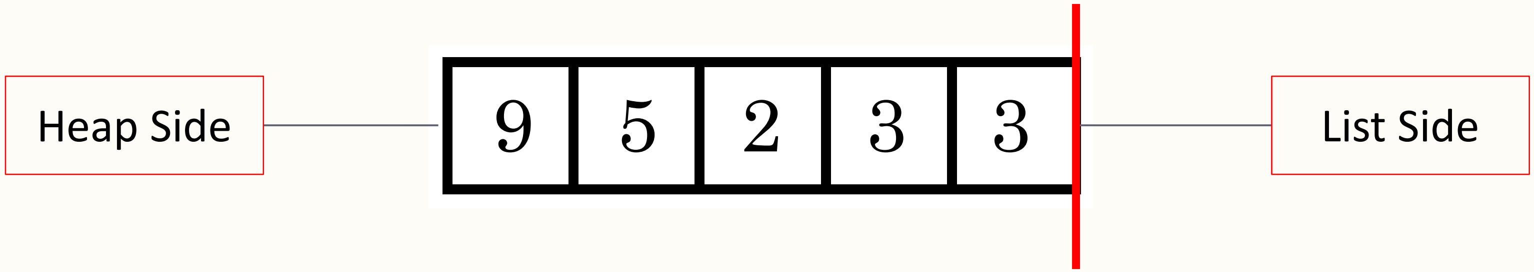
Put 9

On Heap

(and fix)

(and fix more)

Heap Practice



Put 9

On Heap

(and fix)

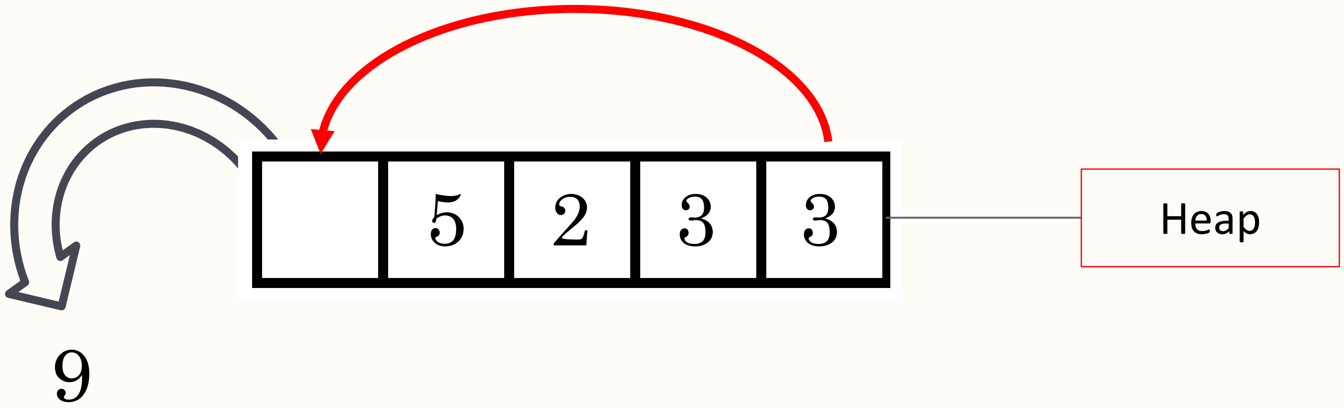
(and fix more)

Heap Practice

9	5	2	3	3

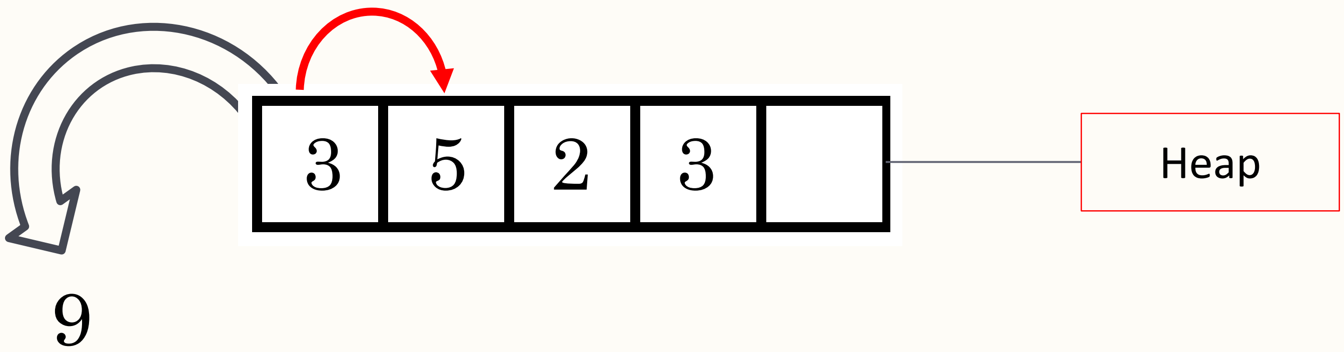
Heap

Heap Practice



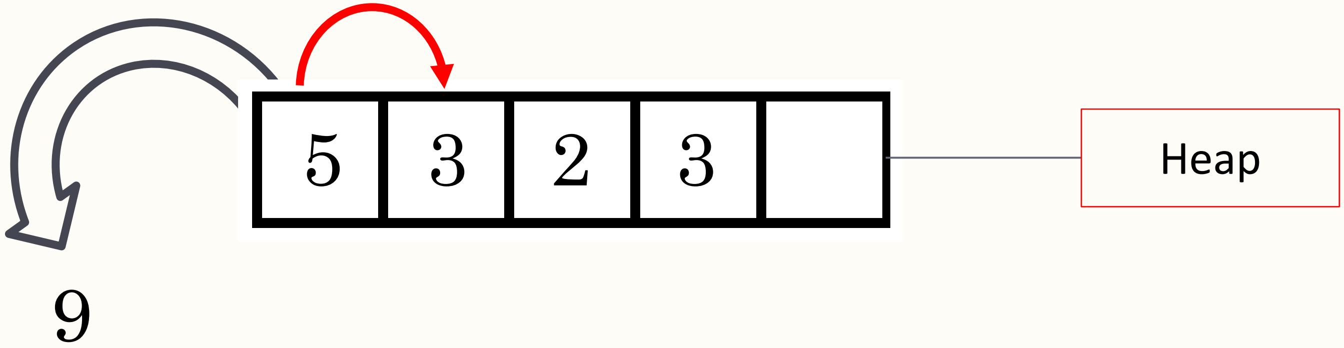
Remove 9
From Heap
(and fix)

Heap Practice



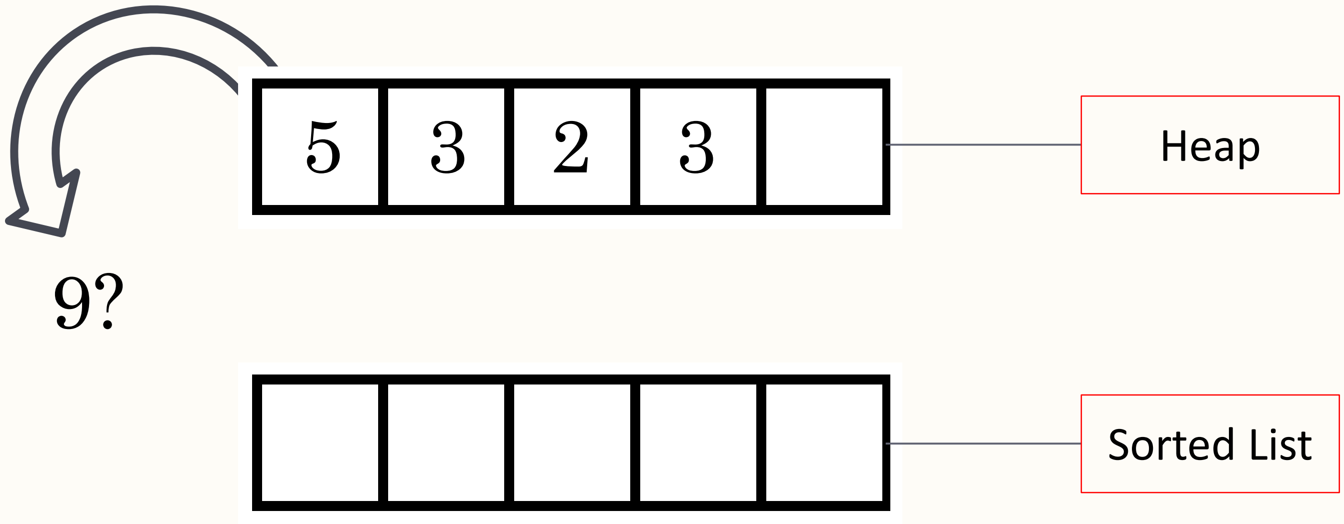
Remove 9
From Heap
(and fix)
(and fix more)

Heap Practice

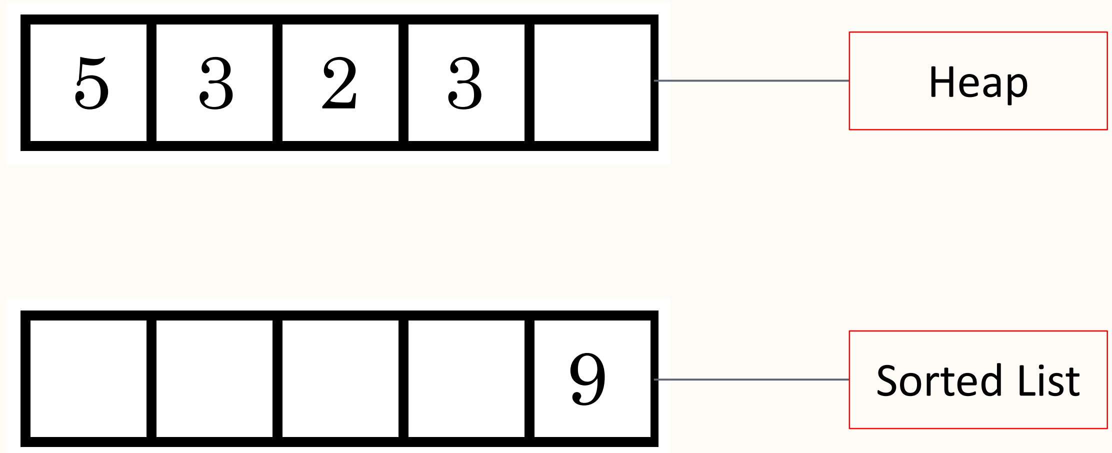


Remove 9
From Heap
(and fix)
(and fix more)

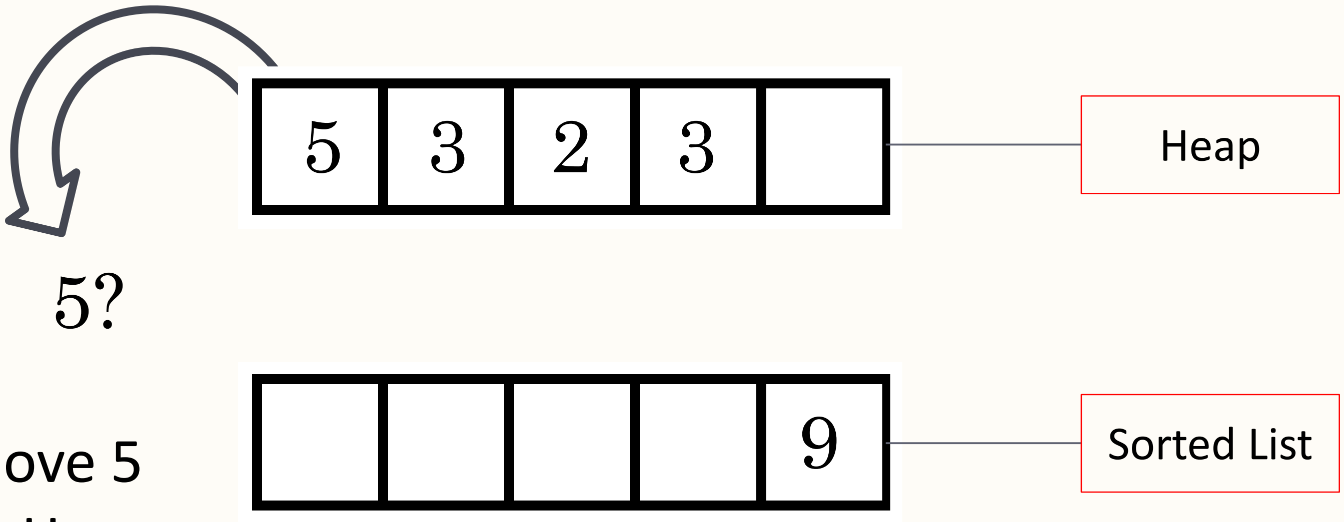
Heap Practice



Heap Practice

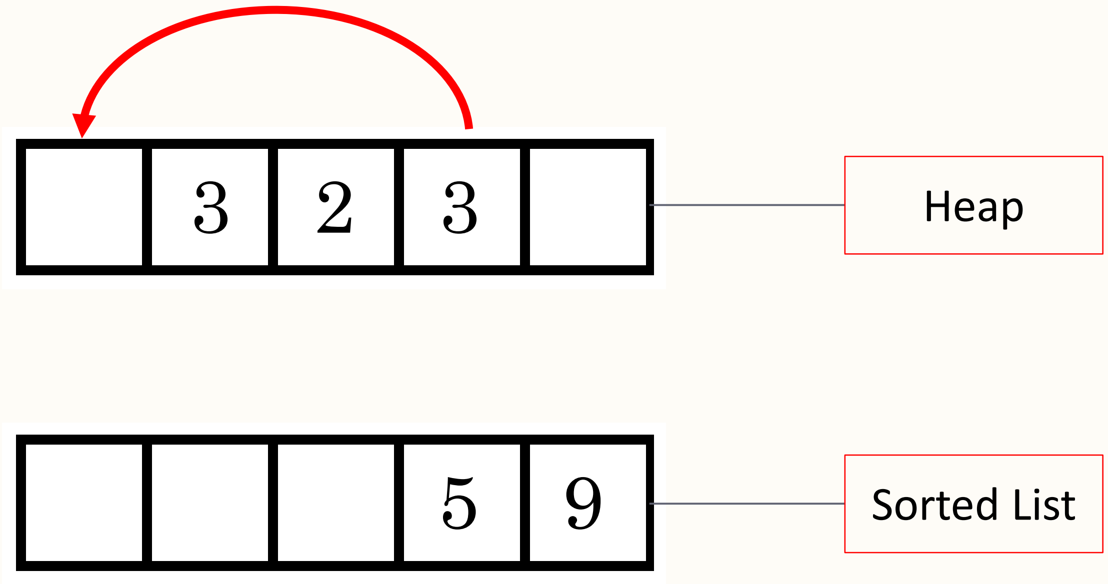


Heap Practice



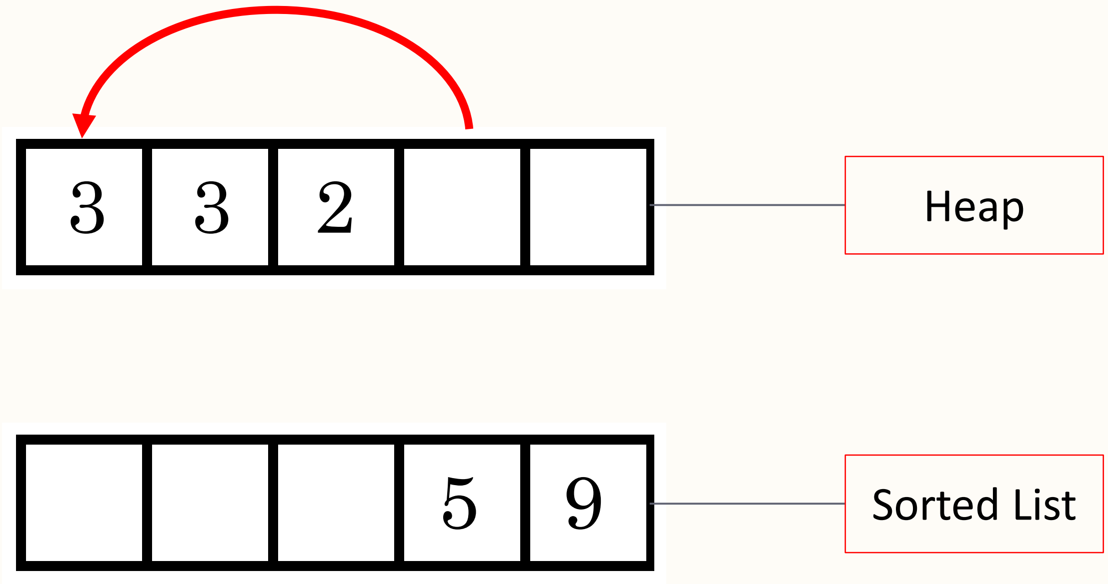
Remove 5
From Heap

Heap Practice



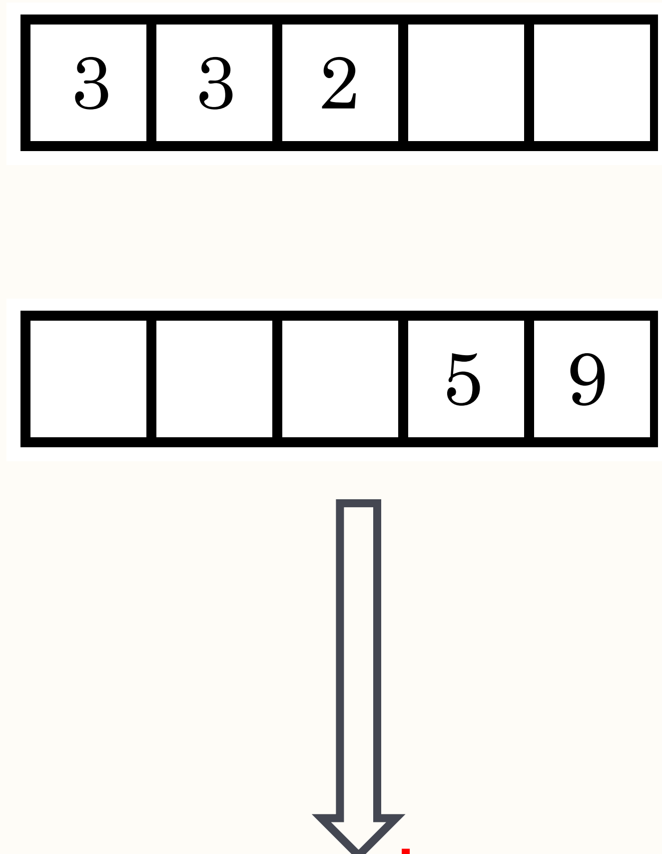
Remove 5
From Heap
(and fix)

Heap Practice

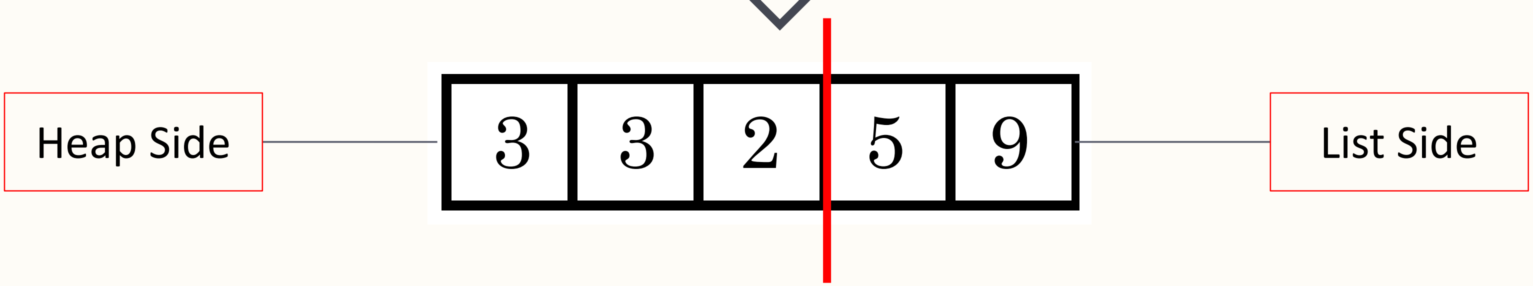


Remove 5
From Heap
(and fix)

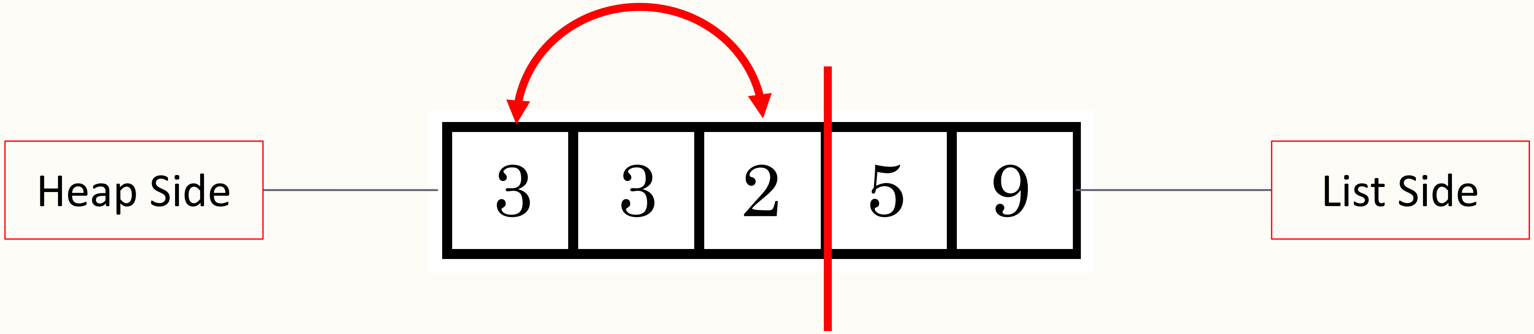
Heap Practice



What do you notice about these two arrays?

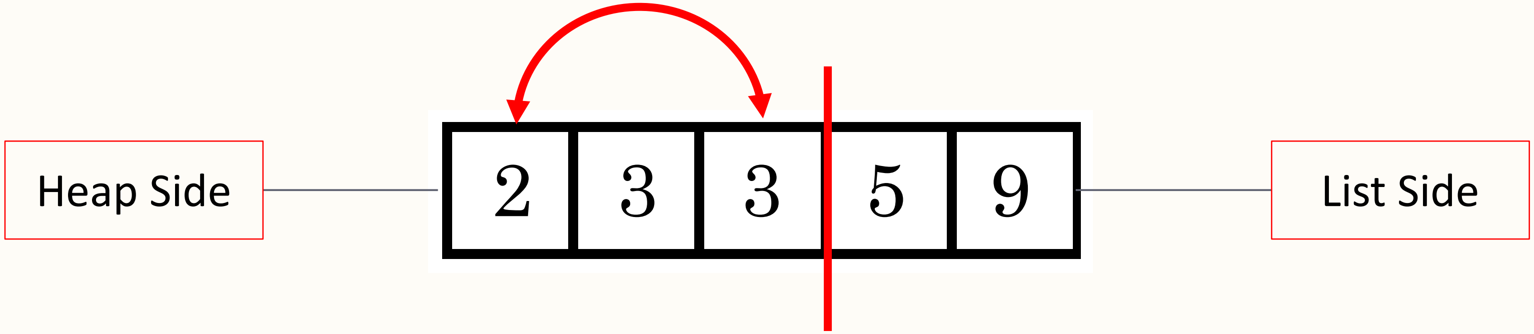


Heap Practice



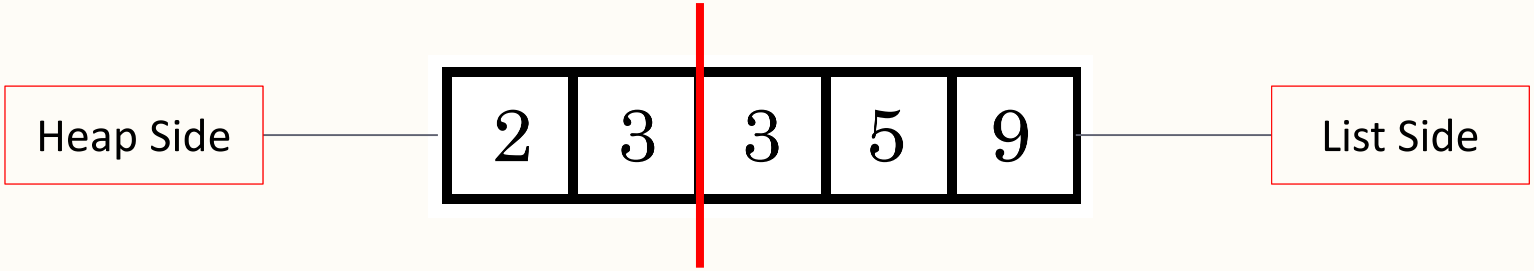
Remove 3
From Heap
(and fix-part 1!)

Heap Practice



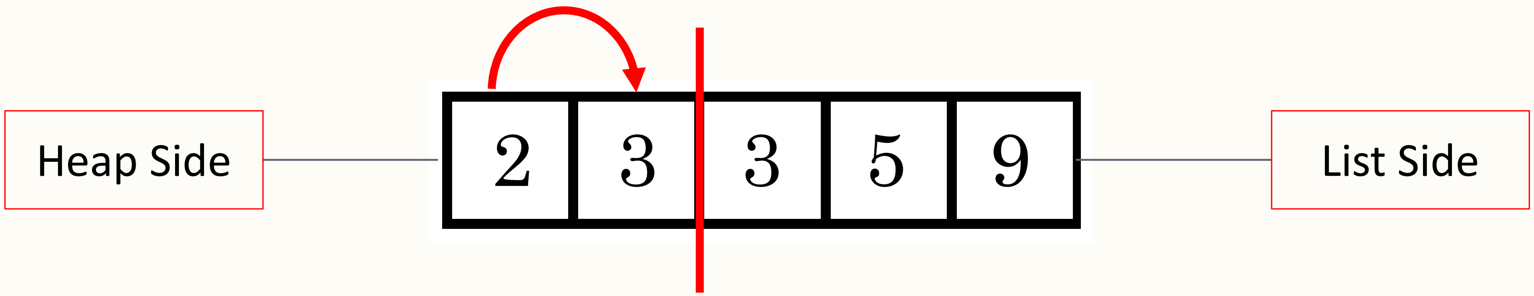
Remove 3
From Heap
(and fix-part 1!)

Heap Practice



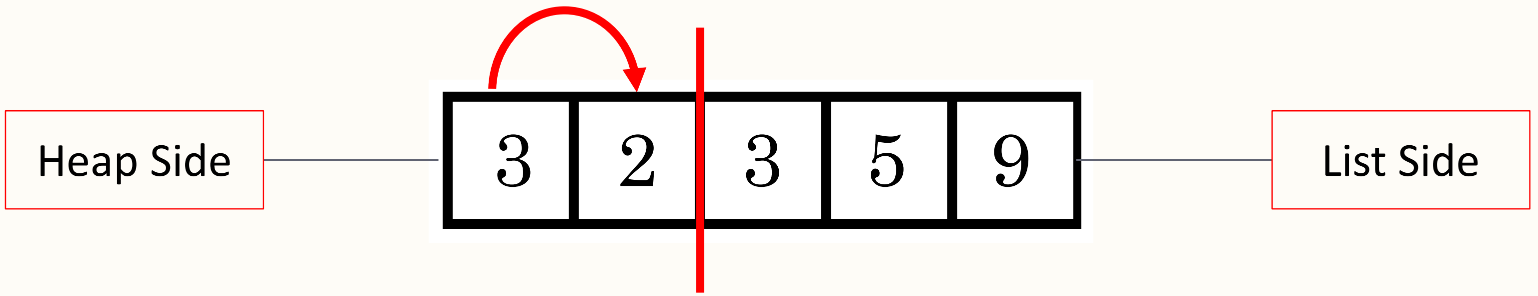
Remove 3
From Heap
(and fix-part 1!)

Heap Practice



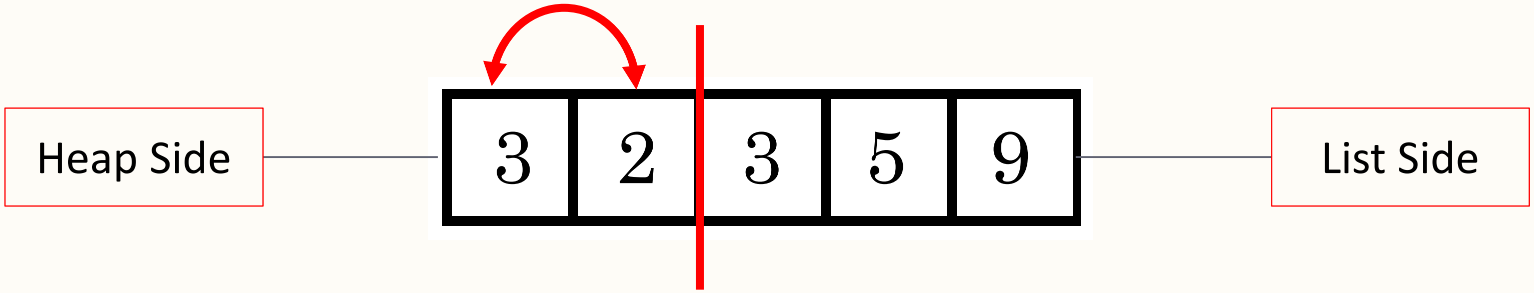
Remove 3
From Heap
(fix-part 2)

Heap Practice



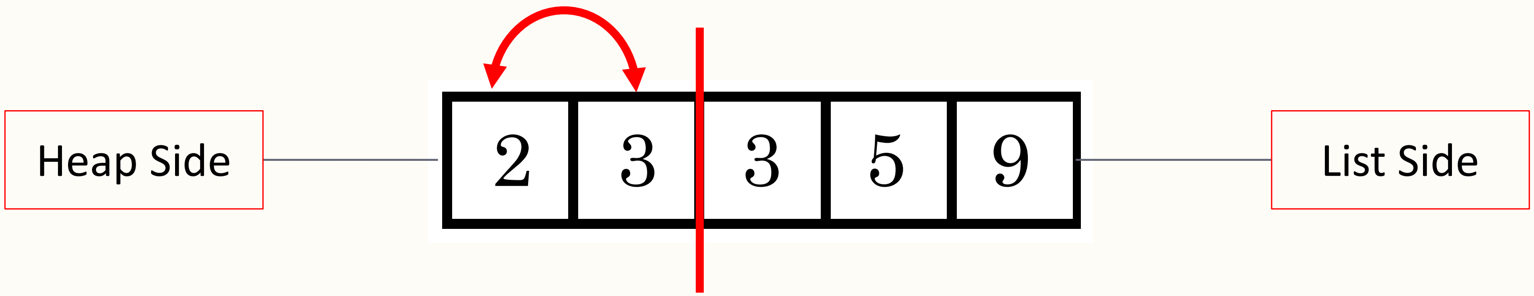
Remove 3
From Heap
(fix-part 2)

Heap Practice



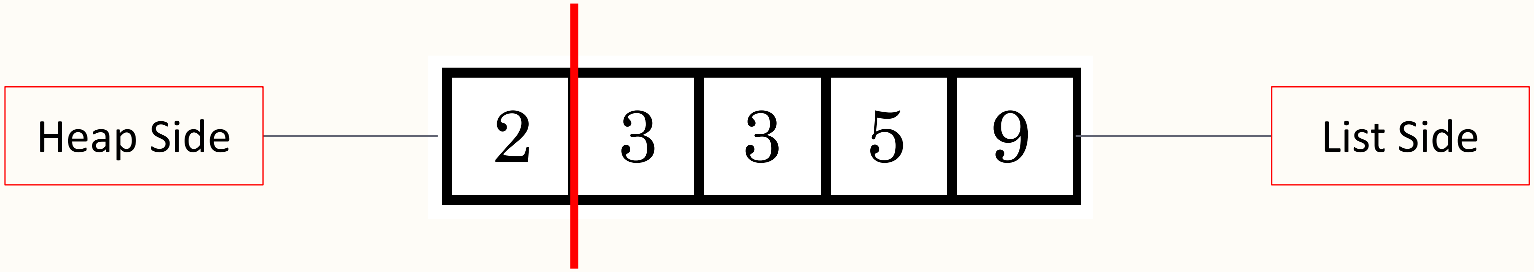
Remove 3
From Heap
(and fix!)

Heap Practice



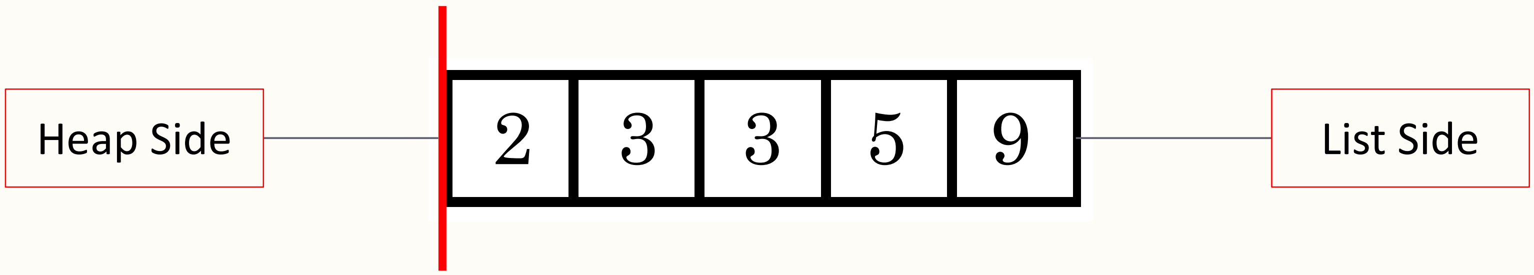
Remove 3
From Heap
(and fix!)

Heap Practice



Remove 3
From Heap
(and fix!)

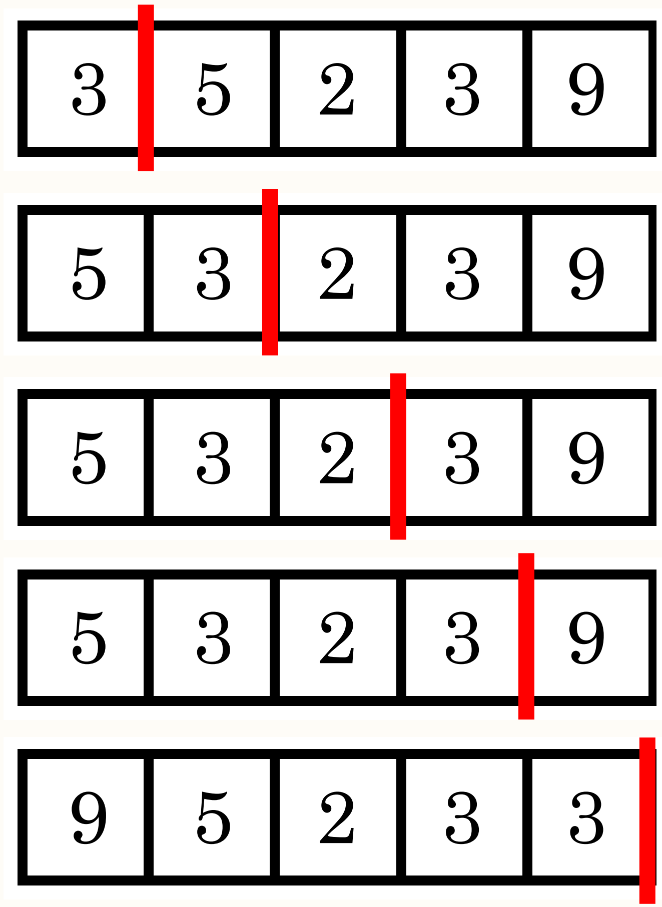
Heap Practice



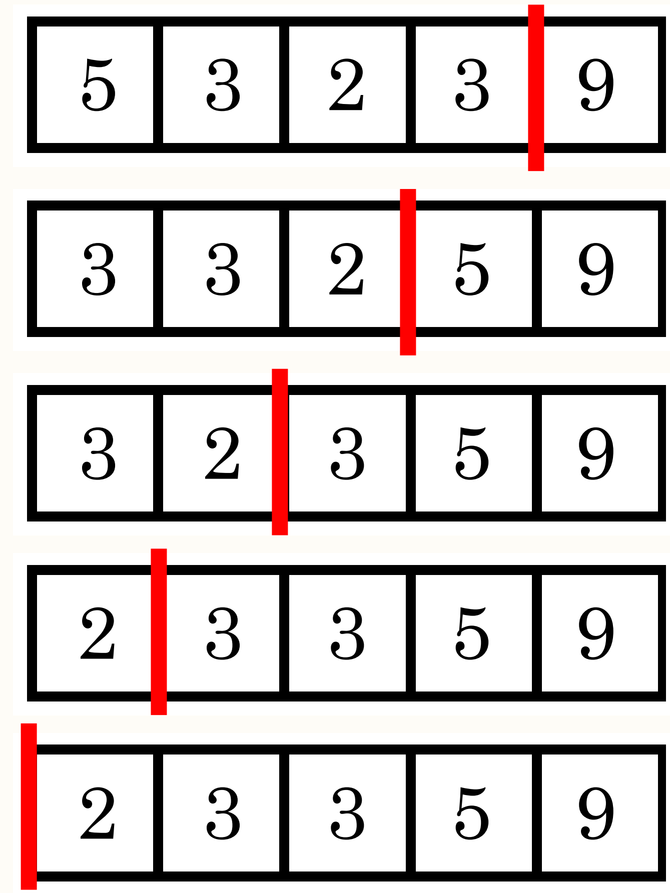
Remove 2
From Heap

Heap Sort Version 1 Summary

O(n lg n) Build Heap



Sort



Uses for a Heap?

- Two things we've already seen!

- Priority Queue
  - maintain order by “priority”
  - highest priority at the top
  - removing an item puts the next highest priority at the top

- Sorting
  - Max heap for ascending order
  - Min heap for descending order

Using Heaps for Sorting

- Given a binary heap and n items to sort
- Option 1:
  - insert each item into the heap
  - remove until no more elements
    - *place elements at the "back" of the array used to store the heap*
- Option 2: Next class!

Gnarley Trees