FIND THE MEDIAN IN A STREAM OF ELEMENTS

FIND THE MEDIAN IN A STREAM OF ELEMENTS

THE STREAM IMPLIES THAT WE HAVE NO IDEA WHICH ELEMENTS ARE GOING IN TO COME IN NEXT - WE HAVE TO KEEP TRACK OF THE MEDIAN AS THE ELEMENTS STREAM IN

USE A **MAX HEAP** TO STORE THE **SMALLER (FIRST) HALF** OF ELEMENTS SEEN IN THE STREAM

USE A **MIN HEAP** TO STORE THE **LARGER (SECOND) HALF** OF ELEMENTS SEEN IN THE STREAM

FIND THE MEDIAN IN A STREAM OF ELEMENTS

USE A MAX HEAP TO STORE THE SMALLER (FIRST) HALF OF ELEMENTS SEEN IN THE STREAM

USE A MIN HEAP TO STORE THE LARGER (SECOND) HALF OF ELEMENTS SEEN IN THE STREAM

NOW IF THE SIZE OF THESE HEAPS ARE SUCH THAT THEY DIFFER BY NO MORE THAN 1 ELEMENT

THEN THE MINIMUM ELEMENT OF THE MIN HEAP AND THE MAXIMUM ELEMENT OF THE MAXIMUM HEAP ARE THE MIDDLE ELEMENTS OF THE STREAM

GETTING THE MEDIAN IS THEN A SIMPLE CALCULATION!

FIND THE MEDIAN IN A STREAM OF ELEMENTS

5 6 7 9 10 2 3 13 15 17 20 1 8

PLACE THE FIRST ELEMENT IN
THE MAX HEAP

MAX HEAP

MIN HEAP

FIND THE MEDIAN IN A STREAM OF ELEMENTS

6 7 9 10 2 3 13 15 17 20 1 8

COMPARE THE NEXT ELEMENT IN THE STREAM WITH THE ELEMENT IN THE MAX HEAP

PLACE IT IN THE MIN HEAP IF IT'S GREATER THAN THE MAXIMUM OF THE MAX HEAP

MAX HEAP

MIN HEAP

FIND THE MEDIAN IN A STREAM OF ELEMENTS

COMPARE THE MAX ELEMENT IN THE MAX HEAP WITH THE NEXT ELEMENT IN THE STREAM

7 9 10 2 3 13 15 17 20 1 8

5

6

PLACE IT IN THE MIN HEAP IF IT'S GREATER THAN THE MAXIMUM OF THE MAX HEAP

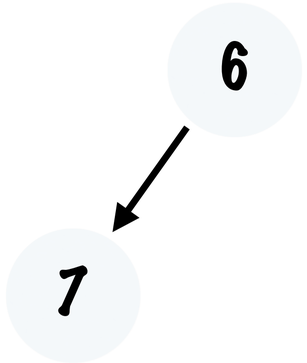
REBALANCE THE HEAPS IF THEY DIFFER BY MORE THAN 1 - THEY DO NOT SO WE CAN CONTINUE

MAX HEAP

MIN HEAP

FIND THE MEDIAN IN A STREAM OF ELEMENTS

COMPARE THE MAX ELEMENT IN THE MAX HEAP WITH THE NEXT ELEMENT IN THE STREAM



PLACE IT IN THE MIN HEAP IF IT'S GREATER THAN THE MAXIMUM OF THE MAX HEAP

REBALANCE THE HEAPS IF THEY DIFFER BY MORE THAN 1 - THEY DO NOT SO WE CAN CONTINUE

MAX HEAP

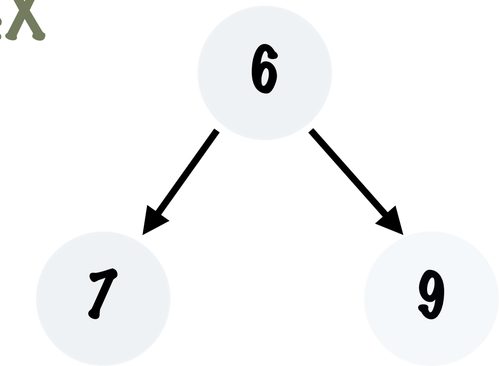
MIN HEAP

FIND THE MEDIAN IN A STREAM OF ELEMENTS

10 2 3 13 15 17 20 1 8

GET THE MINIMUM IN THE MIN HEAP AND ADD IT TO THE MAX HEAP

5



REBALANCE THE HEAPS IF THEY DIFFER BY MORE THAN 1

MAX HEAP

MIN HEAP

FIND THE MEDIAN IN A STREAM OF ELEMENTS

COMPARE THE MAX ELEMENT IN THE MAX HEAP WITH THE NEXT ELEMENT IN THE STREAM

10 2 3 13 15 17 20 1 8

PLACE IT IN THE MIN HEAP IF IT'S GREATER THAN THE MAXIMUM OF THE MAX HEAP

REBALANCE DONE!

MAX HEAP

MIN HEAP

FIND THE MEDIAN IN A STREAM OF ELEMENTS

COMPARE THE MAX ELEMENT IN THE MAX HEAP WITH THE NEXT ELEMENT IN THE STREAM

2 3 13 15 17 20 1 8

PLACE IT IN THE MAX HEAP IF IT'S SMALLER THAN THE MAXIMUM OF THE MAX HEAP

REBALANCE THE HEAPS IF THEY DIFFER BY MORE THAN 1 - THEY NOT SO WE CAN CONTINUE

MAX HEAP

MIN HEAP

FIND THE MEDIAN IN A STREAM OF ELEMENTS

NOTE THAT THE MAX OF THE MAX HEAP AND MIN OF THE MIN HEAP ARE ALWAYS THE MIDDLE ELEMENTS

3 13 15 17 20 1 8

REBALANCE THE HEAPS IF THEY DIFFER BY MORE THAN 1 - THEY DO NOT SO WE CAN CONTINUE

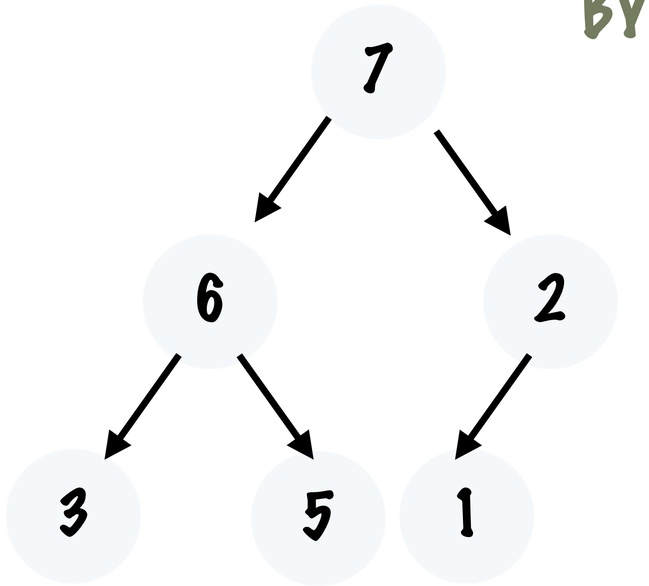
MAX HEAP

MIN HEAP

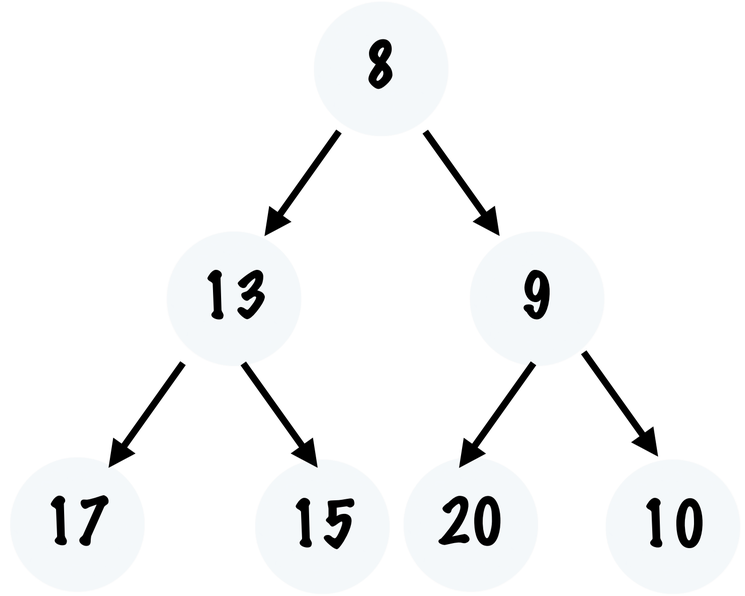
FIND THE MEDIAN IN A STREAM OF ELEMENTS

THE HEAP GROWS AS THE ELEMENTS FROM THE STREAM ARE ADDED

THE SIZE OF THE HEAPS DIFFER BY NO MORE THAN 1



MAX HEAP

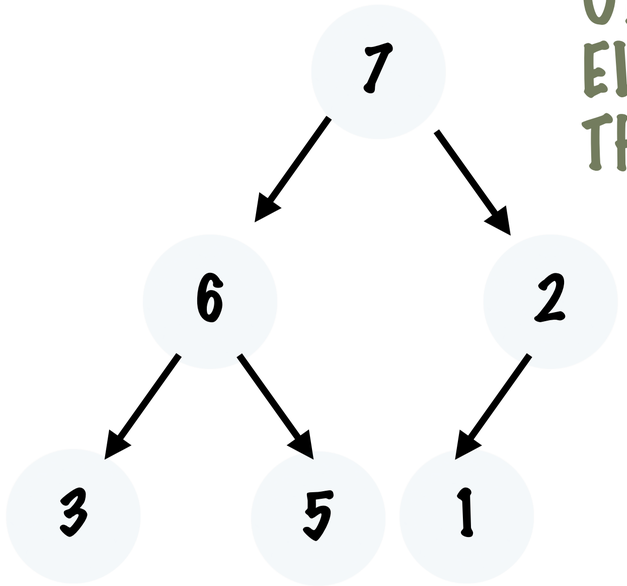


MIN HEAP

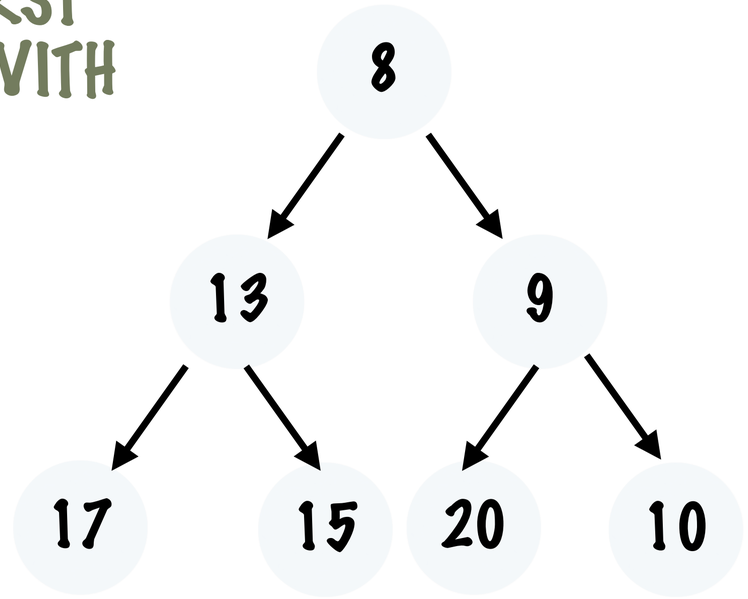
FIND THE MEDIAN IN A STREAM OF ELEMENTS

IF BOTH HEAPS ARE EQUAL THEN THE MEDIAN IS THE AVERAGE OF THE TWO MIDDLE ELEMENTS

OTHERWISE IT IS THE FIRST ELEMENT OF THE HEAP WITH THE LARGER SIZE



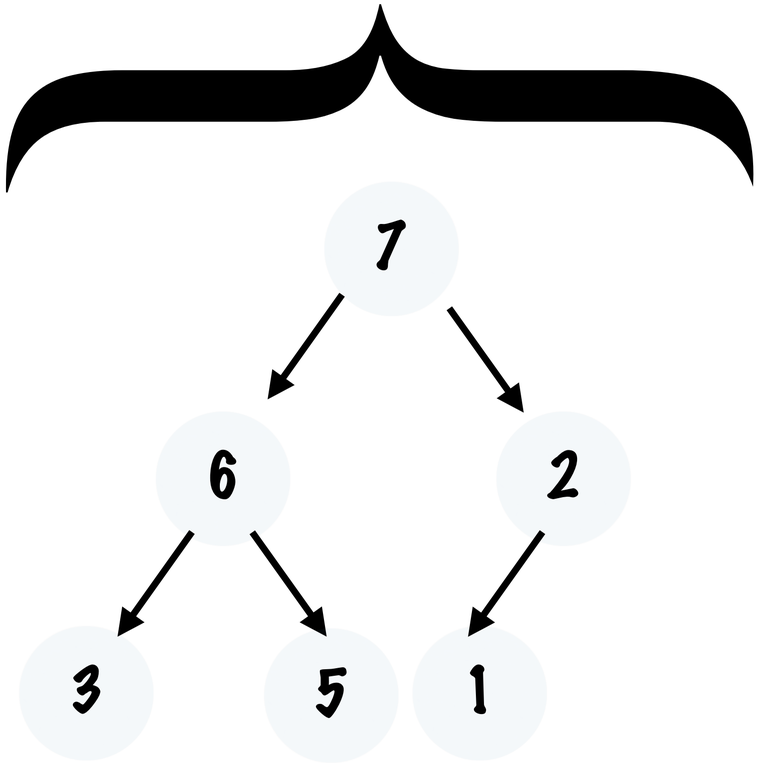
MAX HEAP



MIN HEAP

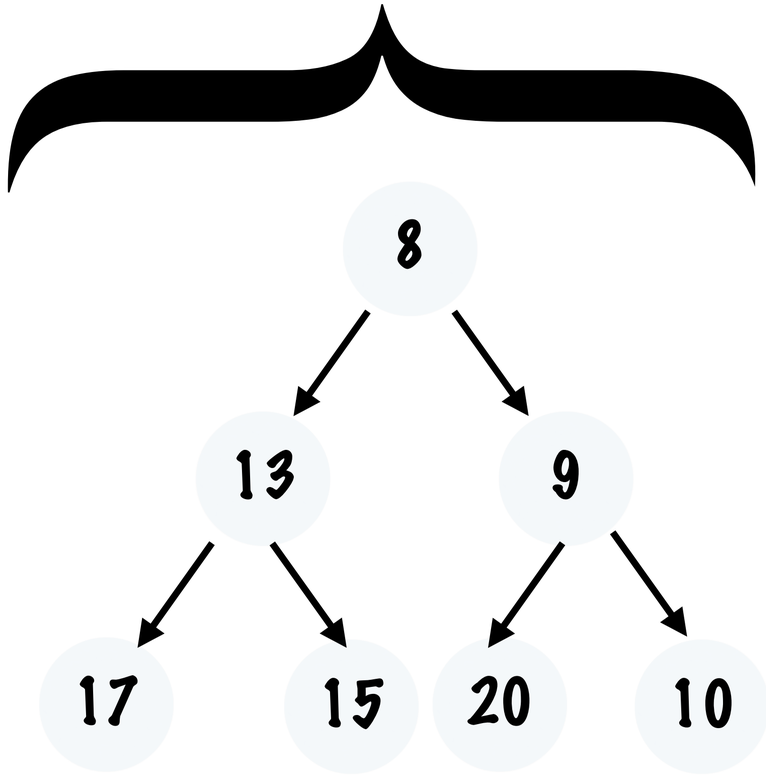
FIND THE MEDIAN IN A STREAM OF ELEMENTS

HAS ALL THE ELEMENTS
SMALLER THAN THE MEDIAN



MAX HEAP

HAS ALL THE ELEMENTS
GREATER THAN THE MEDIAN



MIN HEAP

GET STREAMING MEDIAN

```
public static double getStreamingMedian(int number)
        throws MinHeap.HeapEmptyException, MinHeap.HeapFullException {

    if (!maxHeap.isEmpty() && number > maxHeap.getHighestPriority()) {
        minHeap.insert(number);
    } else {
        maxHeap.insert(number);
    }

    if (maxHeap.getCount() > minHeap.getCount() + 1) {
        minHeap.insert(maxHeap.removeHighestPriority());
    } else if (minHeap.getCount() > maxHeap.getCount() + 1) {
        maxHeap.insert(minHeap.removeHighestPriority());
    }

    if (maxHeap.getCount() == minHeap.getCount()) {
        return 0.5 * (maxHeap.getHighestPriority() + minHeap.getHighestPriority());
    }

    return minHeap.getCount() > maxHeap.getCount()
            ? minHeap.getHighestPriority()
            : maxHeap.getHighestPriority();
}
```

THE NEXT NUMBER IN THE STREAM

COMPARE THE ELEMENT FROM THE STREAM TO THE MAX HEAP AND ADD TO THE MIN HEAP IF THE CURRENT NUMBER IS GREATER THAN THE MAX OF THE MAX HEAP

REBALANCE THE HEAPS IF THEY DIFFER BY MORE THAN 1

IF THE HEAP SIZES ARE EQUAL THEN THE AVERAGE OF THE MIDDLE ELEMENTS IS THE MEDIAN

IF THE COUNTS ARE DIFFERENT GET THE ELEMENT FROM THE LARGER SIZED HEAP