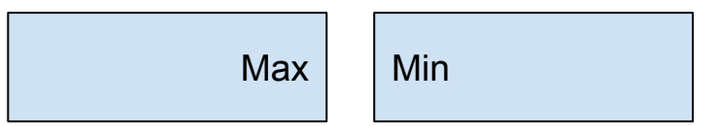
Input: n1, n2, n3, …..

Median: m1, m2, m3, …..

Pattern:

- Constantly insert new number
  - What data structure? Heap
- Return the midden from an array
  - Trick: max-heap, min-heap

The Whole Array



Keys:

- How to keep the balance of two heaps

  Cases:
  - n = odd

    if we want return max[0], we need to make sure the maxheap has one element more.
  - n = even

    two heaps should have equal lengths

Add()

N compare to the min[0], max[0]
Insert(N)
If maxheap, minheap not balance:
balance()

Balance():
choose the heap to extract and another to insert
heapify() both heaps

N compare to the min[0], max[0]
Insert(N)
If maxheap, minheap not balance: -> maxheap - 1 > minheap
balance()

FindMedian():
if odd:
return maxheap[0]
if even:
return (maxheap[0] + minheap[0]) / 2