**Two heaps**

In many problems, we are given a set of elements such that we can divide them into two parts. To solve the problem, we are interested in knowing the smallest element in one part and the biggest element in the other part. This pattern is an efficient approach to solve such problems.

This pattern uses two heaps; A Min Heap to find the smallest element and a Max Heap to find the biggest element. The pattern works by storing the first half of numbers in a Max Heap, this is because you want to find the largest number in the first half. You then store the second half of numbers in a Min Heap, as you want to find the smallest number in the second half. At any time, the median of the current list of numbers can be calculated from the top element of the two heaps.

**Ways to Identify the pattern:**

* Useful in situations like Priority Queue, Scheduling
* If the problem states that you need to find the smallest/largest/median elements of a set
* Sometimes, useful in problems featuring a binary tree data structure

**Common problems you use the sliding window pattern with:**

* Find the Median of a Number Stream (medium)

**Strategies to Solve the Problem:**

* Spend some time thinking if maxheap or minheap will be used.