Problem: given array of n integers, find the median value.

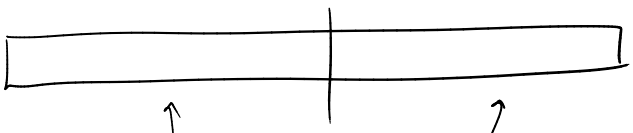
e.g. 1 7 3 10 5 $\longrightarrow$ 5

8 4 2 6 $\longrightarrow$ 4

Approach 1: sort the list, then look in the middle, ie. at index $\lfloor n/2 \rfloor$. $\Theta(n \lg n)$.

Divide + Conquer?

- split in half? $\rightarrow$



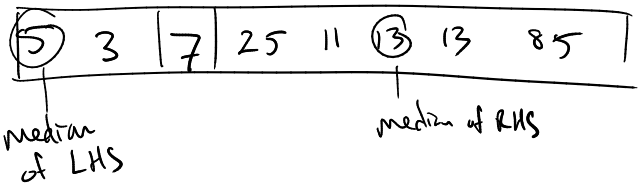
?

Doesn't seem to work?

- partition?

e.g. (7) 25 5 3 11 13 13 85

↓ partition

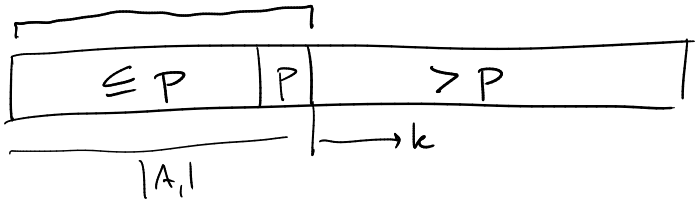


Median of both sides do not give us enough info to find overall median.

Solution: solve a harder problem, ie. generalize the recursion!

Instead of finding the median, we want to find any index. ie. find the k^{th}-smallest value in an array.

Given array A_1 and $0 \le k < |A|$. want to find sort(A)[k].



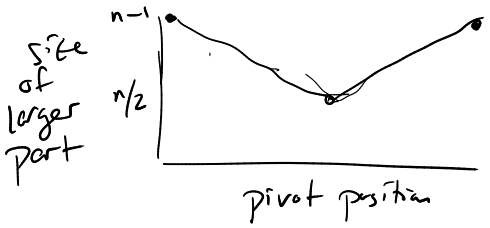
assume $0 \le k < |A|$

```
QuickSelect(A, k):
    if |A| = 1:
        return A[0]
    else:
        A1, A2 ← partition(A)
        if k < |A1|:
            return QuickSelect(A1, k)
        else:
            return QuickSelect(A2, k − |A1|)
```

Master Theorem:

$Q(1) = 1$

$Q(n) = Q(\frac{3}{4}n) + O(n)$



On average, bigger partition is 3/4 n.

$a = 1$
$b = 4/3$
$d = 1$

$a ? b^d$

$1 < (4/3)^1$

Case 1.

$\rightarrow Q(n) = O(n)$.