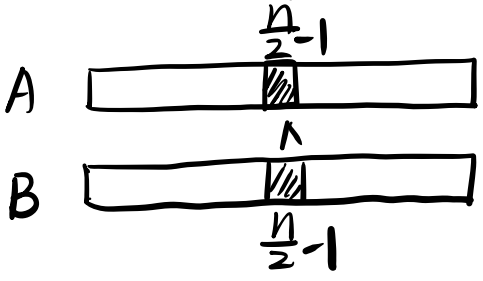
1. You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values—so there are $2n$ values total—and you may assume that no two values are the same. You'd like to determine the median of this set of $2n$ values, which we will define here to be the n^{th} smallest value.

   However, the only way you can access these values is through *queries* to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the k^{th} smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

   Give an algorithm that finds the median value using at most $O(\log n)$ queries.

(assume n is even, if not follow the similar steps)

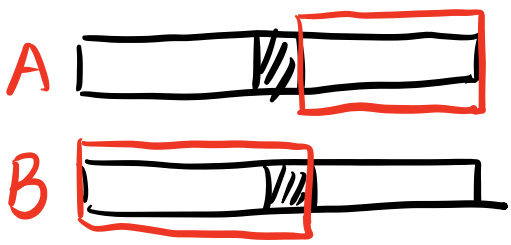


. n^{th} smallest means $(n-1)$ elements smaller than it.

① find $A[\frac{n}{2}-1]$ $B[\frac{n}{2}-1]$ compare $A[\frac{n}{2}-1]$ and $B[\frac{n}{2}-1]$

if $A[\frac{n}{2}-1] < B[\frac{n}{2}-1]$, then for $B[\frac{n}{2}-1]$, there are at least $(n-1)$ elements smaller than it

for $A[\frac{n}{2}-1]$, there are at most $(n-2)$ elements smaller than it



range shrinked to $\frac{n}{2}$

we should find $\frac{n}{2}$ th smallest value

② repeat the above procedure until A. B only left one elements. then $n=2$ $\frac{n}{2}=1$, we find the smaller one.

we need $2 \times \log n$ times queries.

$= O(\log n)$

X partially wrong

2. Recall the problem of finding the number of inversions. As in the text, we are given a sequence of n numbers $a_1, \ldots, a_n$, which we assume are all distinct, and we define an inversion to be a pair $i < j$ such that $a_i > a_j$.

We motivated the problem of counting inversions as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let's call a pair a *significant inversion* if $i < j$ and $a_i > 2a_j$. Give an $O(n \log n)$ algorithm to count the number of significant inversions between two orderings.

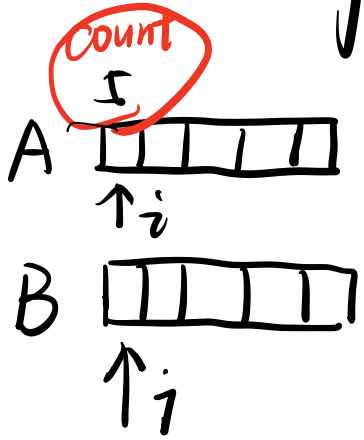
use the same Algorithm but change some condition

assume $A[1, 2 \cdots n]$

divide → $A_1[1 \cdots \frac{n}{2}]$, $A_1[\frac{n}{2}+1, \cdots n]$ ... until length = 1

while merge two sub-sorted Array

merge →



if i, j out of range.
break;

else if A[i] < B[j]
j++

```
else
    if A[i] > 2B[j]
        inverse += count
    i++; count--;
```

Then when we finish merging, the inverse is the number of significant inversions.