TUTORIAL-6

Q1 DAQ algo to merge k sorted arrays

K sorted arrays — n elements

MERGE $(A_1 \ldots A_K)$:

$B1 = Merge(A_{1 \ldots K/2})$ — size $nk/2$

$B2 = Merge(A_{\frac{K+1}{2} \ldots A_K})$ — $nk/2$

B = Merge B1 & B2 in $O(nk)$ time using 2 pointers as merge sort.

return B.

TIME COMPLEXITY : Time to merge k-arrays of size 'n' — $T(n,k)$

$T(n,k) = 2T(n, \frac{k}{2}) + cnk$

$$= 2\left(2T\left(n, \frac{k}{4}\right) + \frac{cnk}{2}\right) + cnk$$

$$4T\left(n, \frac{k}{4}\right) + 2cnk$$

$$= 8T\left(n, \frac{k}{8}\right) + 3cnk$$

$$\vdots$$

i^{th} step: $2^i T\left(n, \frac{k}{2^i}\right) + icnk$

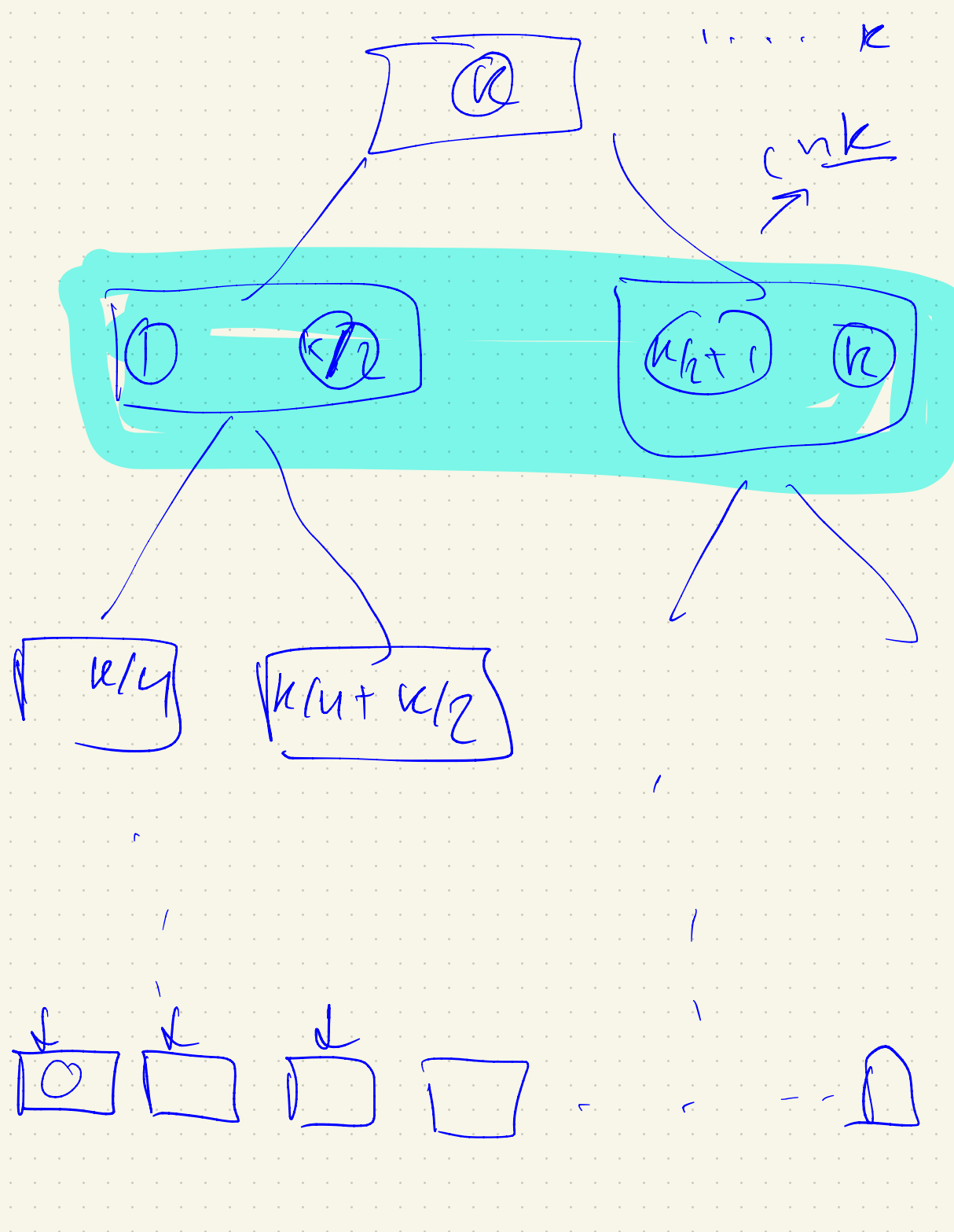
How does this recursion end

$$k = 2^i \Rightarrow i = \log_2 k$$

Put $i = \log_2 k$ in

$$= kT(n, 1) + cnk \log_2 k$$

$$= O(nk \log k)$$


k
n
k
1
k/2
k/4
k/4+k/2

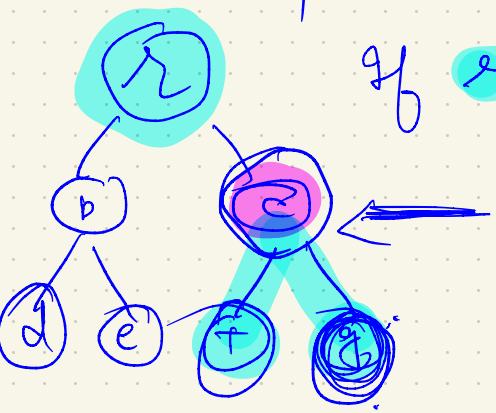
Q2 :- n-node tree CBT height h, so $n = 2^h - 1$

Nodes labelled with distinct real nos.

Node in T is local minimum if its label is smaller than label of its neighbours.

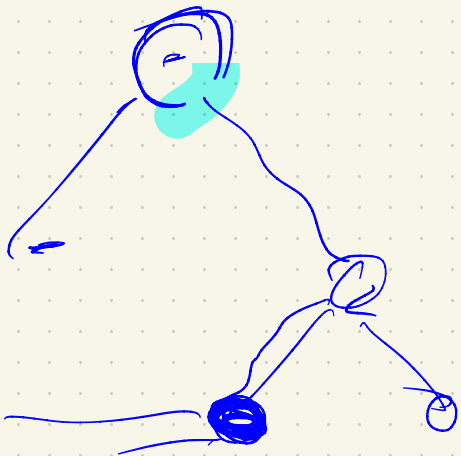
- Design algo to find local minimum of T in O(log n) time.

use Top-Down search



If r is root start by invoking Minima(r)

root



MINIMA(x): →

① If (x is leaf) Return x.

② L = x.leftchild

③ R = x.rightchild

④ If (label(x) < label(L), label(R))
return x

else

If (label(L) < label(x)):
return Minima(L)

else
Return Minima(R)

Time Complexity = Height of Tree
O(log n)
∵ We've Complete Binary Tree.

↳ find out the min^m among L & R
& use min^m.

Corollary of Lemma follows:-

Lemma :-

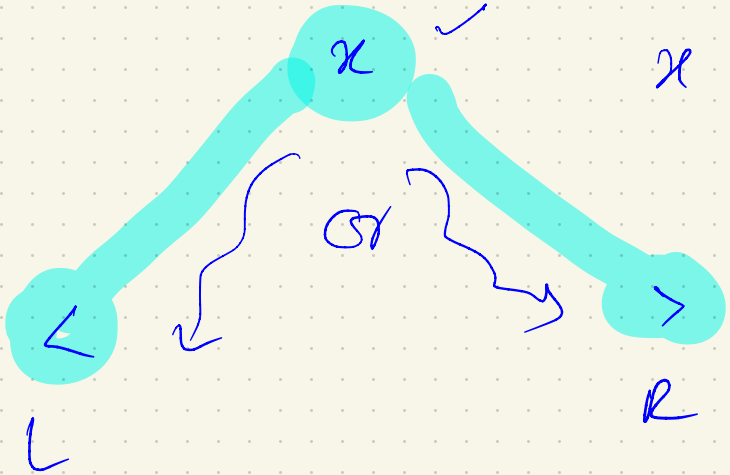
Say Minima(x) invoked for $x \in T$

Then :-

→ Either x is a root

(OR)

→ label(x) < label(Parent(x)).



$R > x > L$

$R < x < L$

A	>	B

Q: 3 n-sized array

Inversion count is #. of pairs (i, j) such that A[i] > A[j].

Question 3

Use a divide + conquer approach.

Our algo counts no of inversions and also sorts the array.

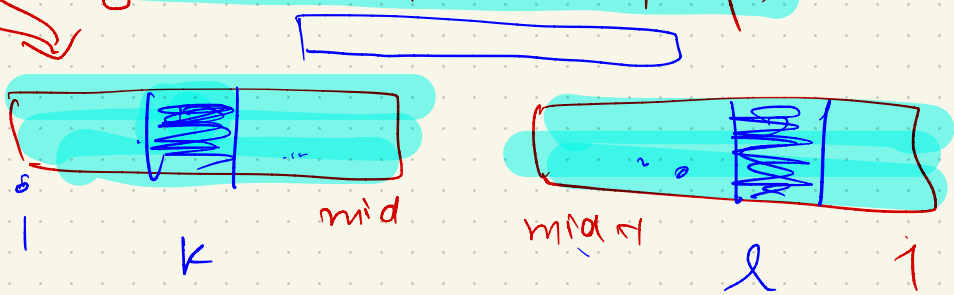
```
Def InvCount (A)
    n = len(A)
    If (n = 1) return 0.
O(n) { Copy in B1 the array A[1, n/2]
       Copy in B2 the array A[n/2 + 1, n]
    ans = InvCount(B1) + InvCount(B2)        } 2T(n/2)
    /* B1 and B2 now are sorted */
    Set x, y, pos = 1
    While (x ≤ len(B1) or y ≤ len(B2)):        → Now merge B1 & B2
        If (B1[x] ≤ B2[y] and x ≤ len(B1)):
            A[pos] = B1[x]
            Increment x and pos by 1.
        Else:
            A[pos] = B2[y]
            Increment y and pos by 1.
            ans = ans + (1 + len(B1) - x)
                        No of elements in B1 greater than B2[y].
```

O(n)

→ Do a merge. Soft. → while combining 2 merged arrays.

if a swap takes place, that is counted as a inversion.

```
MergSort (A, i, j)
  mid = ⌊(i+j)/2⌋
  MergSort (A, i, mid)
  MergSort (A, mid+1, j)
  Merge (A, i, mid, j)
```



No. of swaps to get to final sorted array is the # of inversions here.

Q4 Multiply 2 n bit positive integers x and y.

$x \rightarrow x_1 \cdot 2^{n/2} + x_0$ for some x_1, x_0

$y \rightarrow y_1 \cdot 2^{n/2} + y_0$ for some y_1, y_0

Soln is in section 1.5.

ON INTEGER MULTIPLICATION BY K.T.

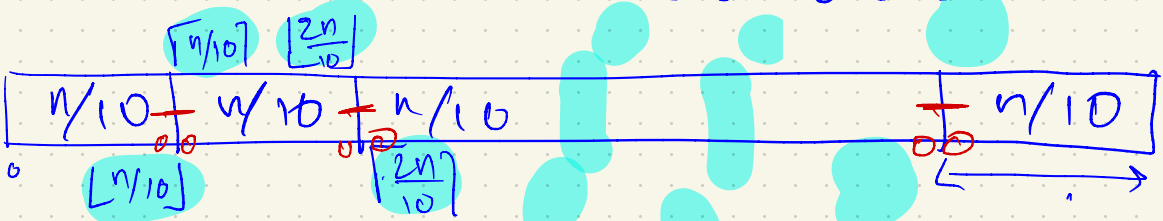
Q.5 A → array of size n : n integers

$O(n)$ time algorithm to check

if $\exists x \in A$ s.t.

$$\text{count}(x) \geq \lceil n/10 \rceil$$

→ use k^{th} smallest algo covered in class with linear TC.



sorted (A)

If count $(x, A) \geq \lceil \frac{n}{10} \rceil$, atleast one of the highlighted pos^n is occupied by x.

If count $(x, A) \geq \lceil \frac{n}{10} \rceil$, then at least one of highlighted pos^n is occupied by x.

① Let $J = \bigcup_{i=1}^{10} \left\{ \lfloor \frac{in}{10} \rfloor, \lceil \frac{in}{10} \rceil \right\}$ be a set of size at most 20. } O(1)

② $S \leftarrow$ A set initialized to ϕ.

③ For $j \in J$, compute j^{th} smallest integer in A and add to S. } O(n)

④ For $s \in S$: compute count (s, A) } O(n)

⑤ Return True if $\max_{s \in S}$ count$(s, A) \geq \lceil n/10 \rceil$ and False otw. } O(1)

Say $n = 100$ $A \rightarrow [\ \ ]$

$J = \{9, 10, 19, 20, 29, 30, \ldots, 99, 100\}$

9th smallest, 10th smallest, ..., 99th smallest, 100th smallest

$S = \{$ count, count, ..., count $\}$

$\{10, 3, 7, \ldots, 11\}$

Now, just return true if any element in the count array has value greater than $\lceil \frac{n}{10} \rceil$ else return False

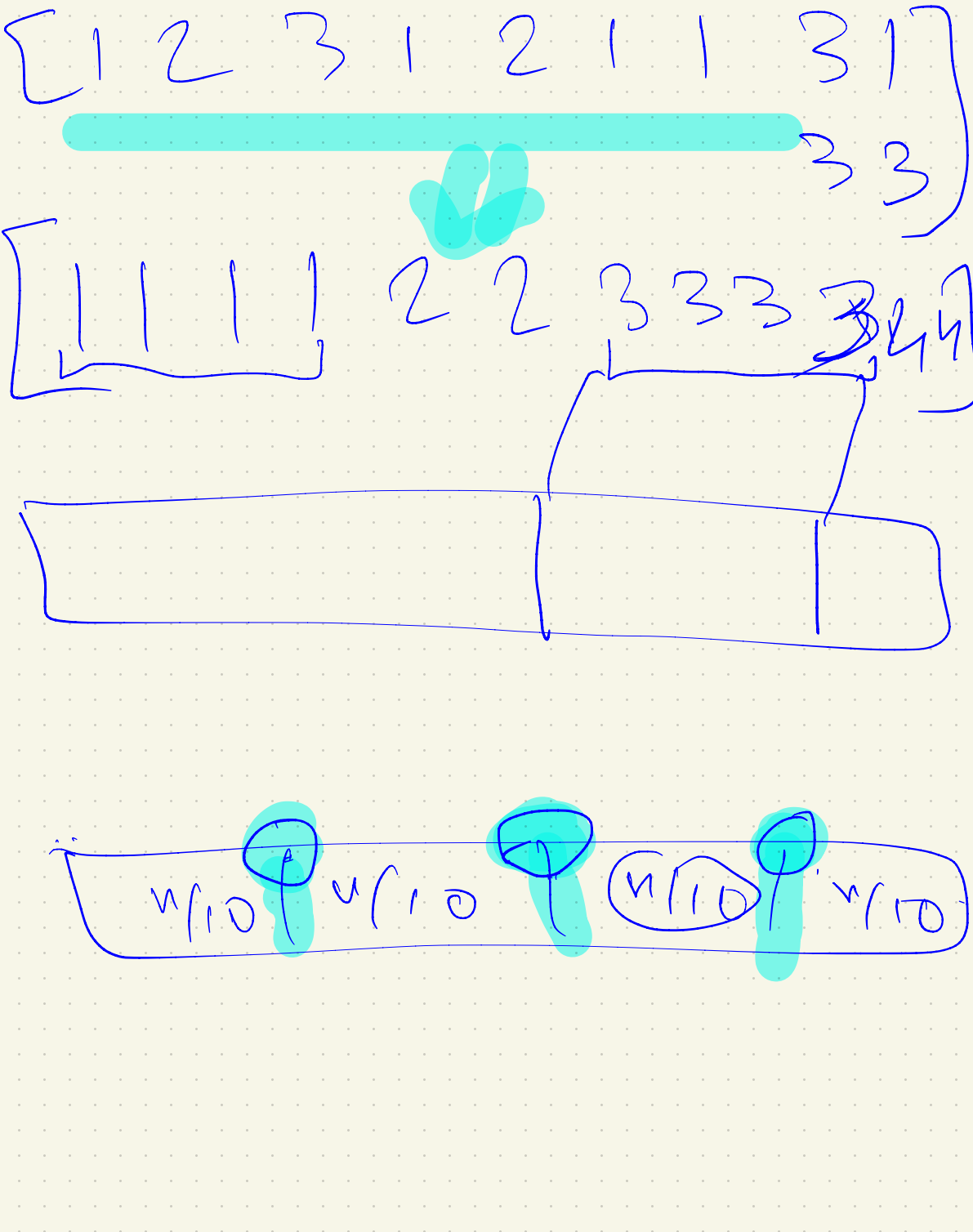
Analysis of Time Complexity

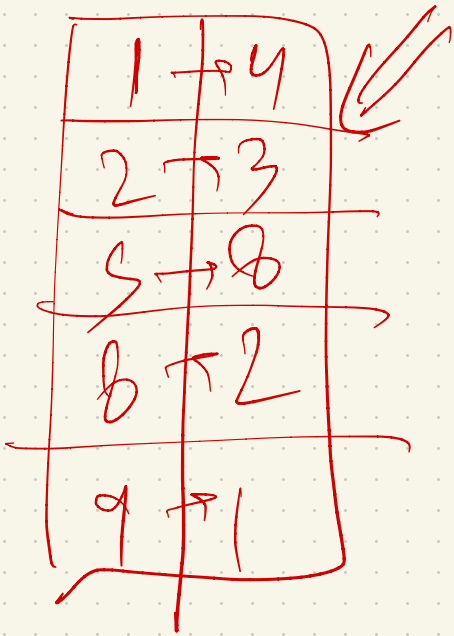
Lemma (lec 20): k-th smallest integer in n-sized array is computable in $O(n)$ time

$\Rightarrow$ Step 3 has $O(n|J|) = O(n)$ time

Correctness follows from following fact:

Fact: If $\exists x \in A$ with count $(x, A) \geq \lceil n/10 \rceil$, then such an x must definitely lie at index in J in sorted A.





19

1- c

SOLUTIONS :-

Ques:-1 Solution

Question 1

Def Merge($A_1,\ldots,A_k$):

1. B1=Merge($A_1,\ldots,A_{k/2}$)
2. B2=Merge($A_{1+k/2},\ldots,A_k$)
   /*size of B1 and B2 is (nk/2) */
3. B = Merge B1 and B2 in $O(nk)$ time by using two pointers as in merge-sort. (see Ques 3 for pseudocode).
4. Return B.

T(n,k):= Time to merge k arrays of size n.

$$T(n,k) = 2T\left(n,\frac{k}{2}\right) + cnk$$

$$= 2\left(2T\left(n,\frac{k}{4}\right) + cn\frac{k}{2}\right) + cnk$$

$$= 4T\left(n,\frac{k}{4}\right) + 2cnk$$

$$= 8T\left(n,\frac{k}{8}\right) + 3cnk$$

$$\vdots$$

Put $2^i = k$, $i = \log_2 k$

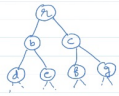
$$= 2^i T\left(n,\frac{k}{2^i}\right) + i\,cnk$$

$$= kT(n,1) + cnk\log_2(k)$$

$$= O(nk\log k)$$

Q 2. Solution

Question 2



Idea: Top down search

If r is root then invoke Minima(r)

```
Def Minima(x):
① If (x is leaf) Return x
② L = x. left child
③ R = x. right child
④ If ( label(x) < label(L), label(R) ): Return x
   Else If (label(L) < label(x)): Return Minima(L)
   Else: Return Minima(R)
```

Time Complexity = O(height of tree) = O(log n)

Correctness follows by lemma below

Lemma: If Minima(x) is invoked/called for $x \in T$, then

- Either x is a root
- Or, label(x) < label(parent(x)).

Q3 Solution

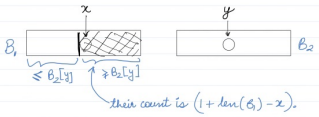
Question 3

Use a divide & conquer approach.
Our algo counts no of inversion and also sorts the array.

```
Def InvCount ( A)
    n = len(A)
    If (n = 1) return 0.
O(n) { Copy in B1 the array A[1, n/2]
       Copy in B2 the array A[n/2 + 1, n]
    ans = InvCount(B1) + InvCount(B2)        } 2T(n/2)
    /* B1 and B2 now are sorted */
    Set x, y, pos = 1
    While (x ≤ len(B1) or y ≤ len(B2)):
        If ( B1[x] ≤ B2[y] and x ≤ len(B1)):
            A[pos] = B1[x]
            Increment x and pos by 1.
        Else:
            A[pos] = B2[y]
            Increment y and pos by 1.
            ans = ans + (1 + len(B1) - x)
                        No of elements in B1 greater than B2[y]
    Return ans.
```

O(n)

Else cond^n:



Time complexity

$$T(n) = 2T(n/2) + O(n)$$

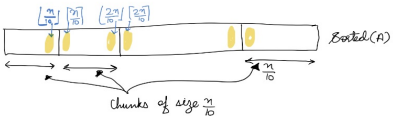
$$\Rightarrow T(n) = O(n \log n)$$

Q4 Solution :- Available as Section 5.5 on Kleinberg, Tardos

Question:- 5

Question 5

To check if $\exists\, x \in A$ with $count(x, A) \geq \lceil n/10 \rceil$ we use the "k^{th}-smallest" algo covered in the class with linear time complexity.



If $count(x, A) \geq \lceil \frac{n}{10} \rceil$, then at least one of highlighted pos^n is occupied by x.

① Let $J = \bigcup_{i=1}^{10} \left\{ \lceil \frac{in}{10} \rceil, \lceil \frac{in}{10} \rceil \right\}$ be a set of size at most 20.	} $O(1)$
② $S \leftarrow$ A set initialized to ϕ.	
③ For $j \in J$, compute j^{th} smallest integer in A and add to S.	} $O(n)$
④ For $s \in S$: compute $count(s, A)$	} $O(n)$
⑤ Return True if $\max_{s \in S} count(s, A) \geq \lceil n/10 \rceil$ and False else.	} $O(1)$

Analysis of Time Complexity

Lemma (Lec 20): k-th smallest integer in n-sized array is computable in $O(n)$ time

$\Rightarrow$ Step 3 has $O(n|J|) = O(n)$ time

Correctness follows from following fact:

Fact: If $\exists\, x \in A$ with $count(x, A) \geq \lceil n/10 \rceil$, then such an x must definitely lie at index in J in sorted A.