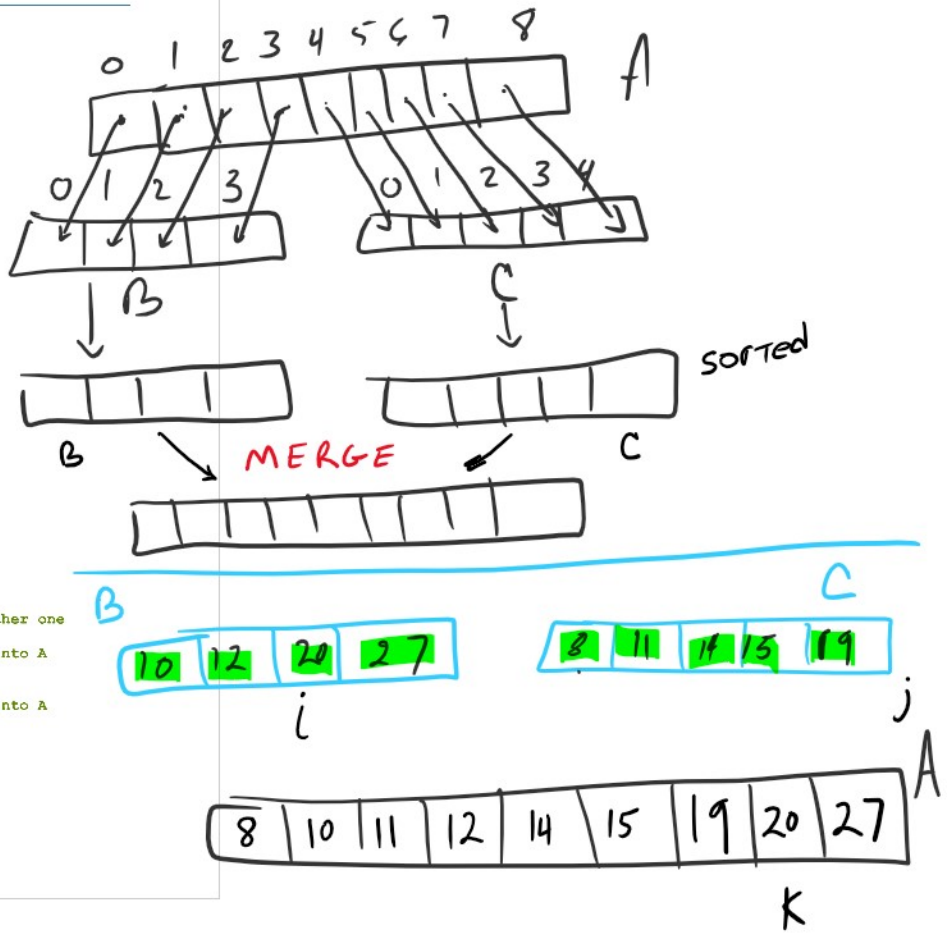
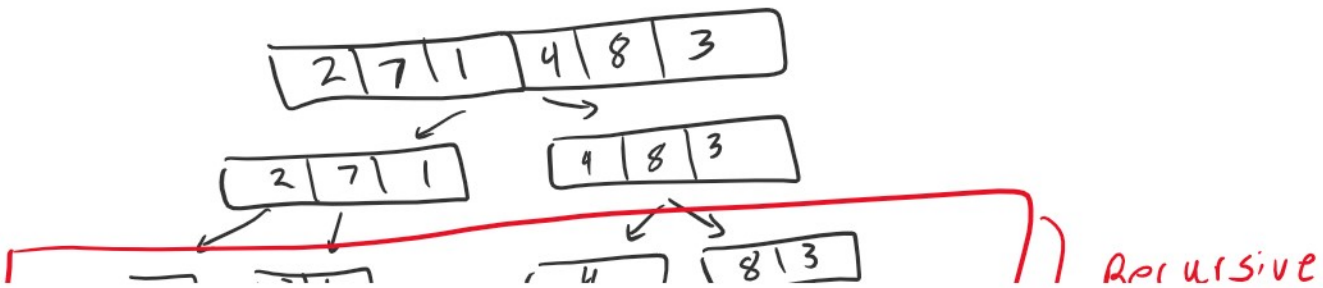
mergesort

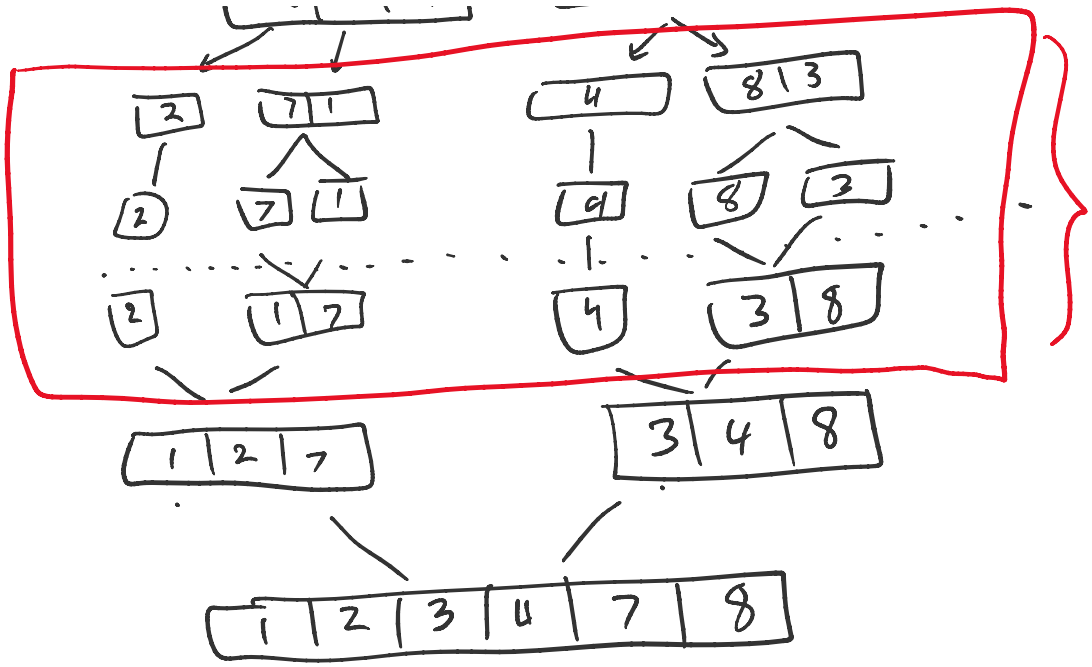
Wednesday, March 18, 2020 10:50 AM

Pseudo Code MergeSort

```
mergeSort(array A) {   // The length of A is n
     x = n / 2  // Integer division
     y = n - x

     B = new array of length x
     C = new array of length y

     B[0 .. x-1] = A[0 .. x-1]
     C[0 .. y-1] = A[x .. n-1]

     mergeSort(B)
     mergeSort(C)

     merge(B, C, A)
}


Merge(B, C, A) {
     // merge B and C into A
     // Assumes B.length + C.length == A.length

     i = j = k = 0

     while(i < B.length && j < C.length){
          if(B[i] <= C[j]) {
               A[k] = B[i]
               i++
          } else {
               A[k] = C[j]
               j++
          }
          k++
     }

     // See which one ended first & copy rest of other one
     if(i == B.length){
          // B ended first, so copy the rest of C into A
          A[k .. A.length-1] = C[j .. C.length-1]
     } else {
          // C ended first, so copy the rest of B into A
          A[k .. A.length-1] = B[I .. B.length-1]
     }
}
```







Running Time Analysis of MergeSort

$$T(n) = a\left(\frac{n}{b}\right) + f(n)$$

a = # of subproblems

b = size of each subproblem

$f(n)$ = time to divide & combine

$a = 2$ subproblems

$\frac{1}{2}n$ or $\frac{n}{2}$ $b = 2$

$f(n)$: • Divide time copying All n items

• Combine The sorted subproblem (merge) n Comparisons

n comparisons

$n + n = 2n \in \Theta(n)$

$T(n) = 2T\left(\frac{n}{2}\right) + f(n)$

$a = 2, \quad b = 2 \qquad f(n) \in \Theta(n) = \Theta(n^1) \Rightarrow d = 1$

CASE 1

$a < b^d$

$2 < 2^1$

$2 < 2$

NO

CASE 2

$a = b^d$

$2 = 2^1$

$2 = 2$

YES

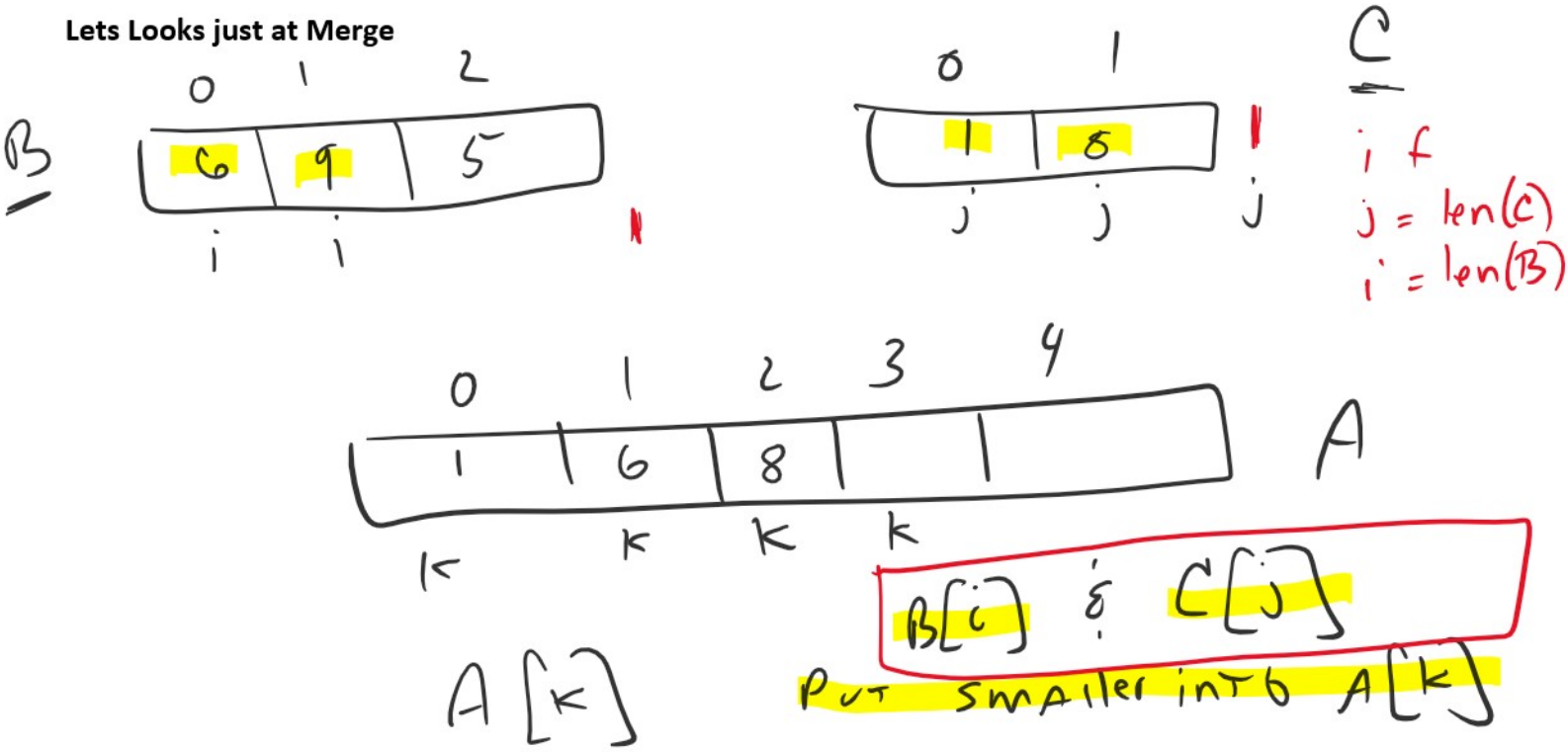
USE CASE 2

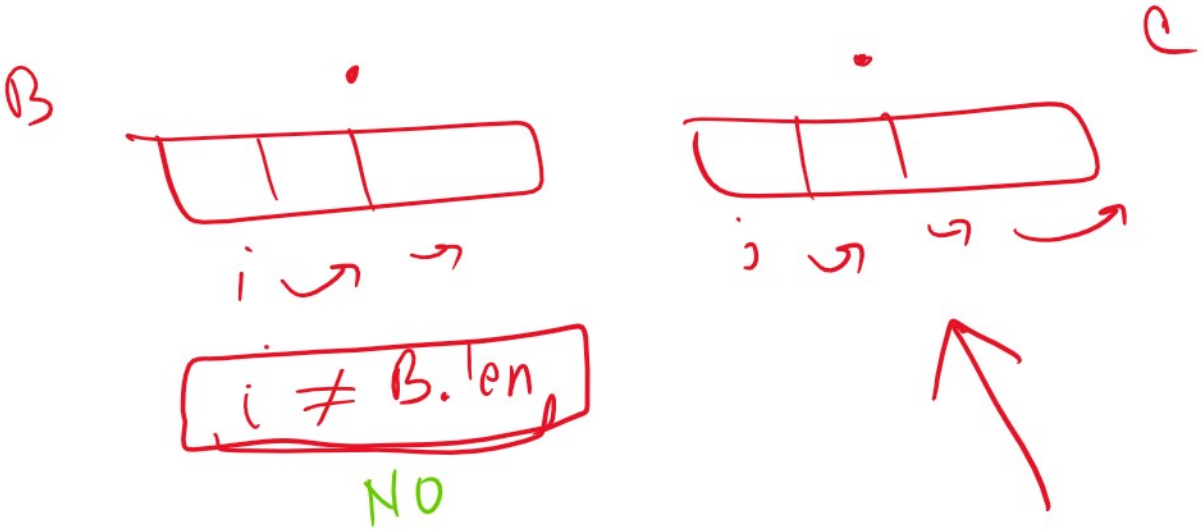
$$T(n) = \Theta(n^d \lg n)$$

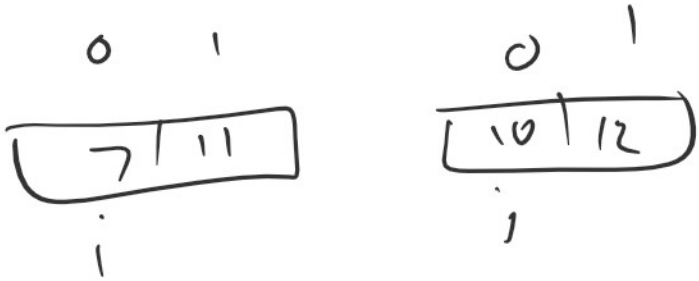
$$T(n) = \Theta(n \lg n)$$

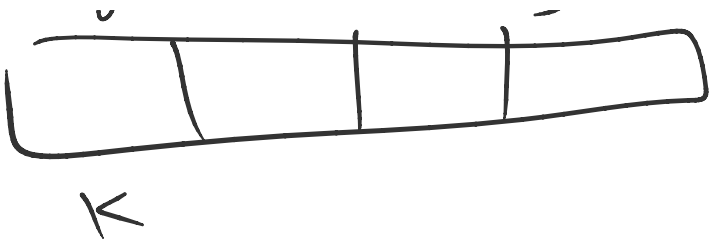
Running Time of MergeSort is $\Theta(n \lg n)$

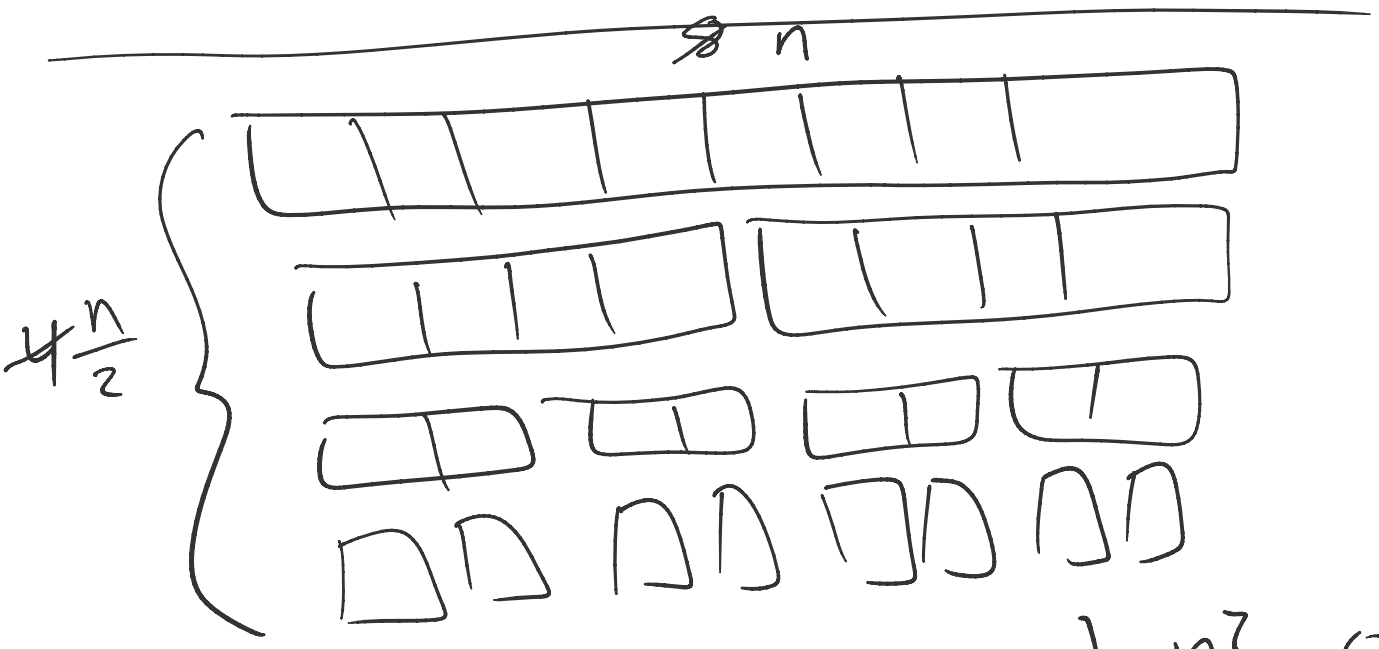
Lets Looks just at Merge











$$n \cdot \frac{n}{2} = \frac{n^2}{2} = \frac{1}{2} n^2 \in \Theta(n^2)$$