Count Sort

Sort array of even & odd

Merge Sort

Stable & In place Sorting

Q. Find smallest no. that can be formed by rearranging digits given in an array.

A = < 6, 3, 4, 2, 7, 2, 1 >

ans → < 1, 2, 2, 3, 4, 6, 7 >

A = < 4, 2, 7, 3, 9, 0 >

ans → < 0, 2, 3, 4, 7, 9 >

Approach 1: Sort the array

Arrays.sort()

↓

TC: O(N log₂N)

Approach 2: Can we use the info that digits are 0 → 9

< -, -, -, -, -, -, -, - >

0 00... 111... 222... 3..... 9

freq of every digit

0 → -

1 → -

2 → -

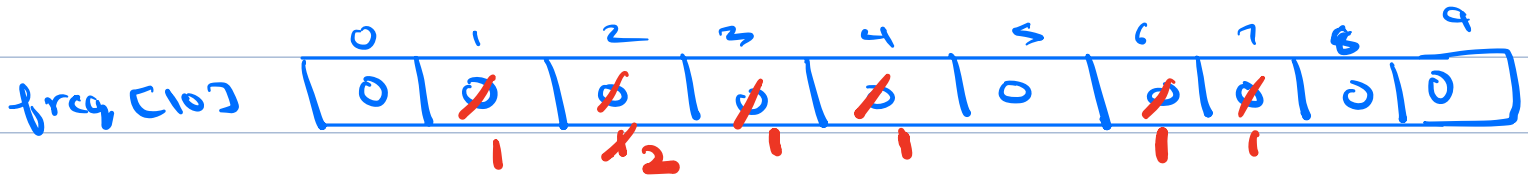
⋮

9 → -

① Count freq of all the digits int freq[10]

② Use freq array to fill the original array

A = < 6, 3, 4, 2, 7, 2, 1 >



A = < 6, 3, 4, 2, 7, 2, 1 >
0 1 2 3 4 5 6
1 2 2 3 4 6 7

Sort based on counting frequency

↓

Count sort

```
// int A[], int N
int freq[10] = <0>
① for (i=0 ; i<N ; i++) {
      freq[A[i]] = freq[A[i]] + 1
  }

int k = 0
② for (dig = 0 ; dig <= 9 ; dig++) {
      for (cnt = 1 ; cnt <= freq[dig] ; cnt++) {
          A[k] = dig      k++
      }
  }
return A
```

N

10

2

dig → cnt
0 → for
1 → for
2 → for

TC : $O(N + 10 + N)$

$\simeq O(N)$

SC : $O(10) \simeq O(1)$

↓

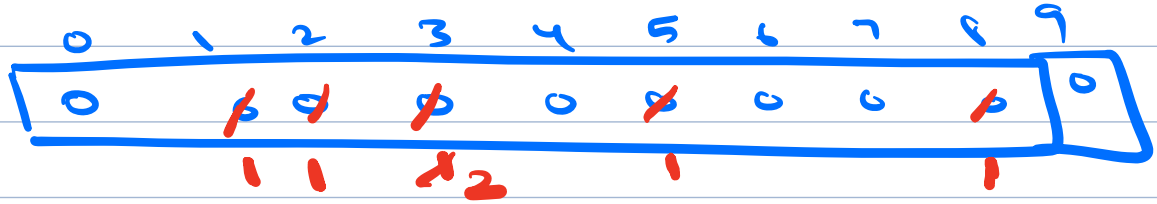
No. of digits (0-9)

3 → ⋮ 9

N

$A = [1, 3, 8, 2, 3, 5]$

freq[10] =



$A = [1, 3, 8, 2, 3, 5]$

1 2 3 3 5 8

Will count sort work if range of A[i] is more than 10^9 ?

$< 10^9, 10^7 + 2, 10^8, 10^9 >$

$10^9 \rightarrow 2$

$10^7 + 2 \rightarrow 1$

$10^8 \rightarrow 1$

data → 0 to 10^9

int freq[10^9+1]

int → 4 B

10^9 integers → 4 B × 10^9 = 4 GB

Count sort works until till 10^6

Range → 0 to 10^6

int freq[10^6]

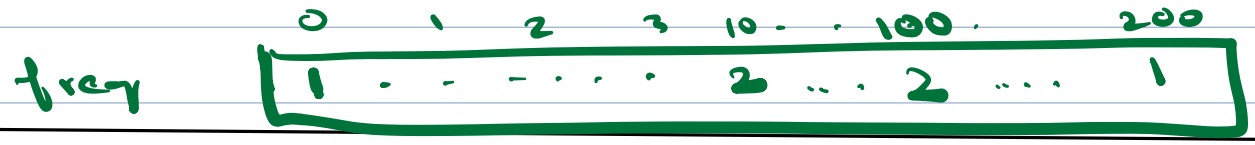
↓

4×10^6 B = 4 MB

< 10, 0, 100, 200, 100, 10 >

range → 0 to 200

int freq[201]



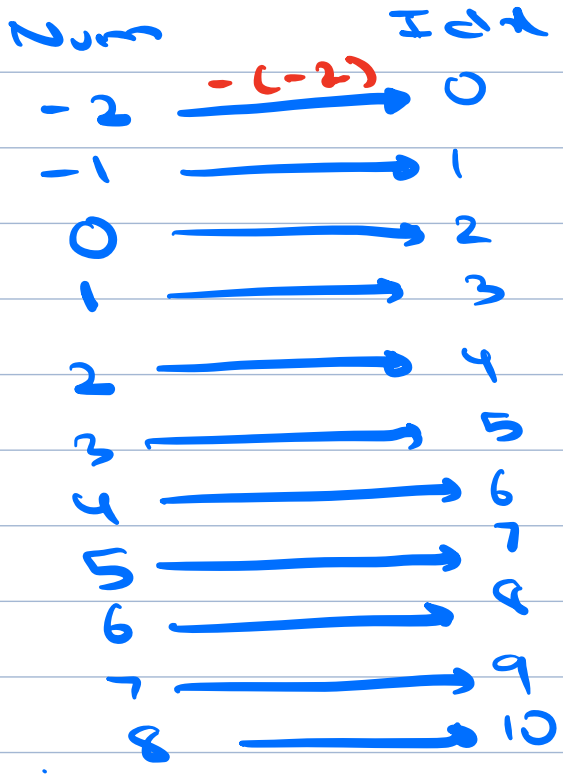
Case II A = [-2 3 8 3 -2 3]

Range → -2 to 8

int freq[11]

No. of el in range =

8 - (-2) + 1 = 11



Num - (-2) = Idx

Num - Min = Idx

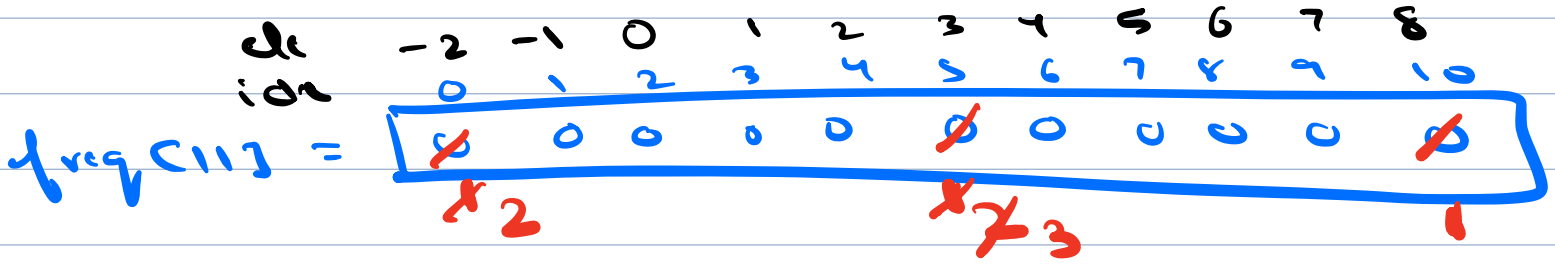
Idx = Num + Min

A = [-2 3 8 3 -2 3]

Range → -2 to 8

int freq[11]

Min = -2 Max = 8



A = [-2 -2 3 3 3 8]

① Iterate in array and get min and max → N

② int freq[max - min + 1] = <0>

for (int i = 0 ; i < N ; i++) → N

idx = A[i] - min

freq[idx] ++

int k = 0

for (int i = 0 ; i < max - min + 1 ; i++) → Range

num = i + min → N

```
for (cnt = 1 ; cnt <= freq[i]; cnt++){
    A[k] = num    k++
}
return A
```

TC: O(3N + Range)
≃ O(N + Range)

SC: O(Range)
freq[max-min+1]

Gmail → All inboxes

A1 → [9, 8:30, 8:10]

A2 → [10, 9, ...]

Merge 2 sorted arrays.

A → [1, 5, 6, 9]

B → [2, 4, 8]

ans → [1, 2, 4, 5, 6, 8, 9]

Q. Given an integer array where all odd elements are sorted and all even elements are sorted. Sort entire array.

A = [2 5 4 8 11 13 10 15 21]

Approach : Seperate out even and odd
Merge two arrays

odd = [5 11 13 15 21]

even = [2 4 8 10]

ans = [2 4 5 8 10 11 13 15 21]

```
void Sort (int A[], int N) {
    list <int> odd, even
    for (i=0 ; i< N ; i++) {
        if (A[i] % 2 == 0)
            even.add(A[i])
        else
            odd.add(A[i])
    }

    int oddN = odd.size()
    int evenN = even.size()
```

```
int o=0, e=0, a=0

while (o < oddN && e < evenN) {
    if (odd[o] < even[e]) {
        A[a] = odd[o]
        a++   o++
    }
    else {
        A[a] = even[e]
        a++   e++
    }
}

while (o < oddN) {          // when even[] ended
    A[a] = odd[o]
    a++   o++
}

while (e < evenN) {         // when odd[] ended
    A[a] = even[e]
    a++   e++
}

return A
```

2N iterations

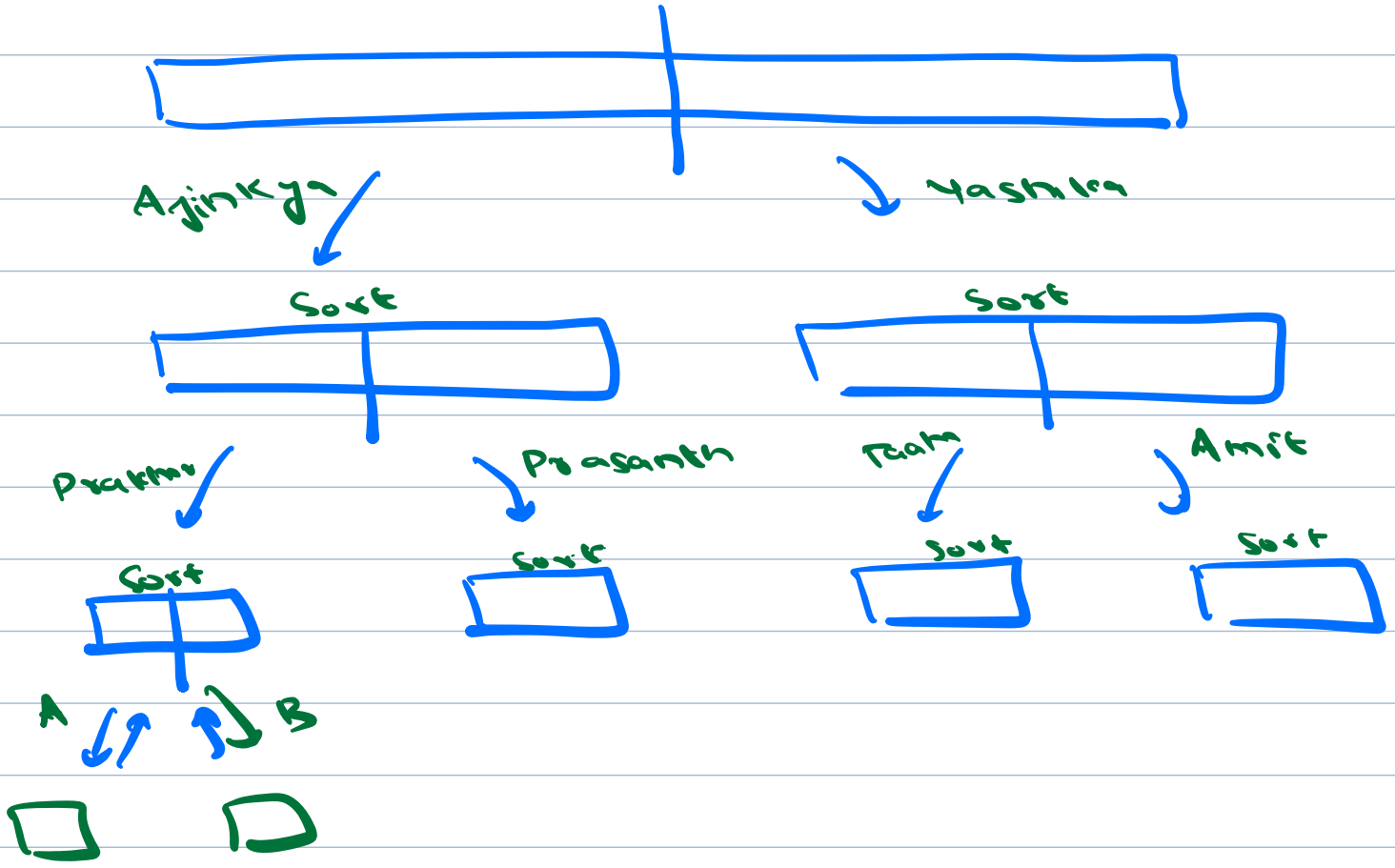
TC : O(N)

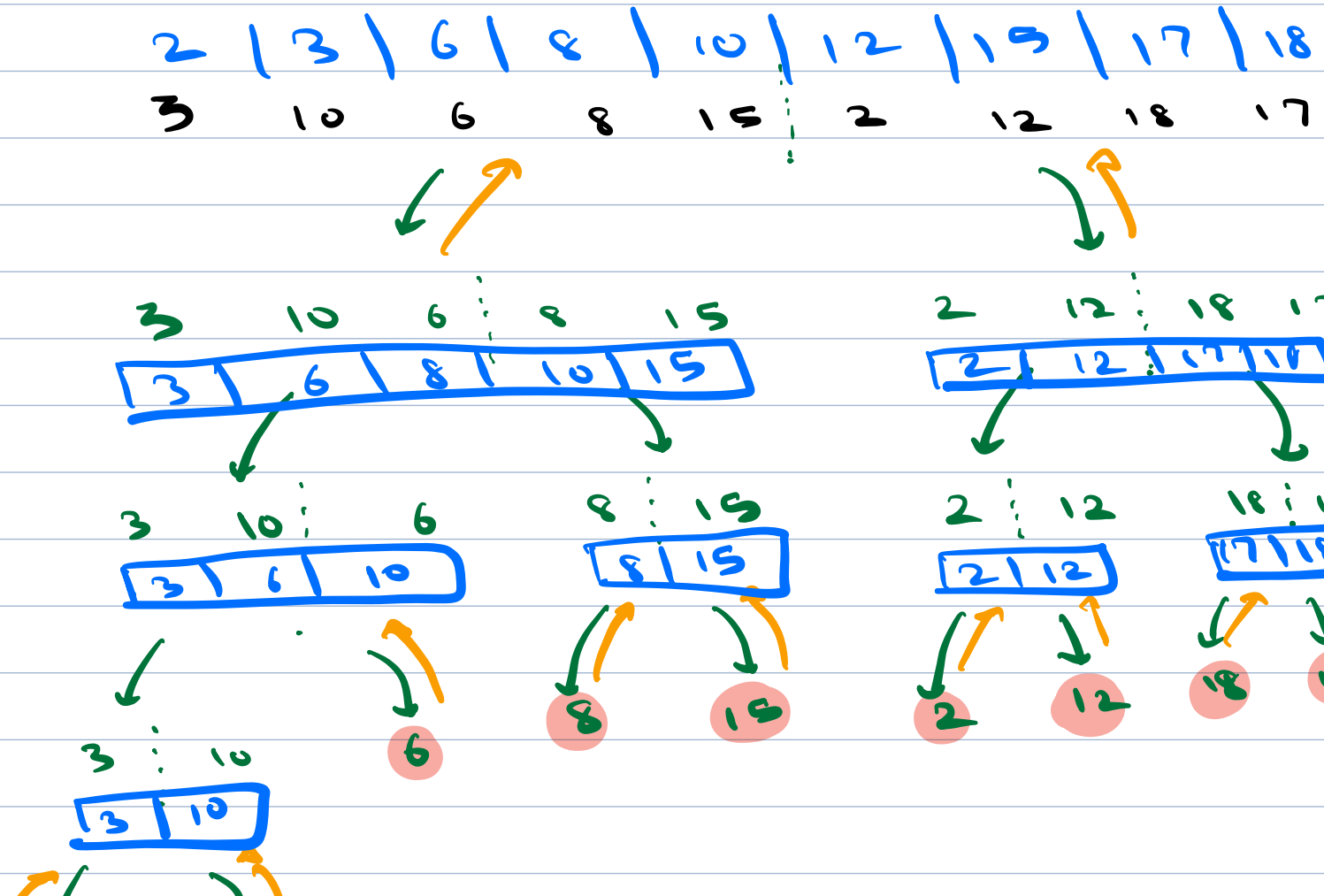
SC : O(N)

odd and even list

7

Merge Sort


Ajinkya
Yashika
Sort
Sort
Prathmesh
Prasanth
Ram
Amit
Sort
Sort
Sort
Sort
A
B


2 3 6 8 10 12 13 17 18
3 10 6 8 15 2 12 18 17
3 10 6 8 15
3 6 8 10 15
2 12 18 17
2 12 17 18
3 10 6
3 6 10
8 15
8 15
2 12
2 12
18 17
17 18
3 10
3 10
6
8
15
2
12
18
17

3 10

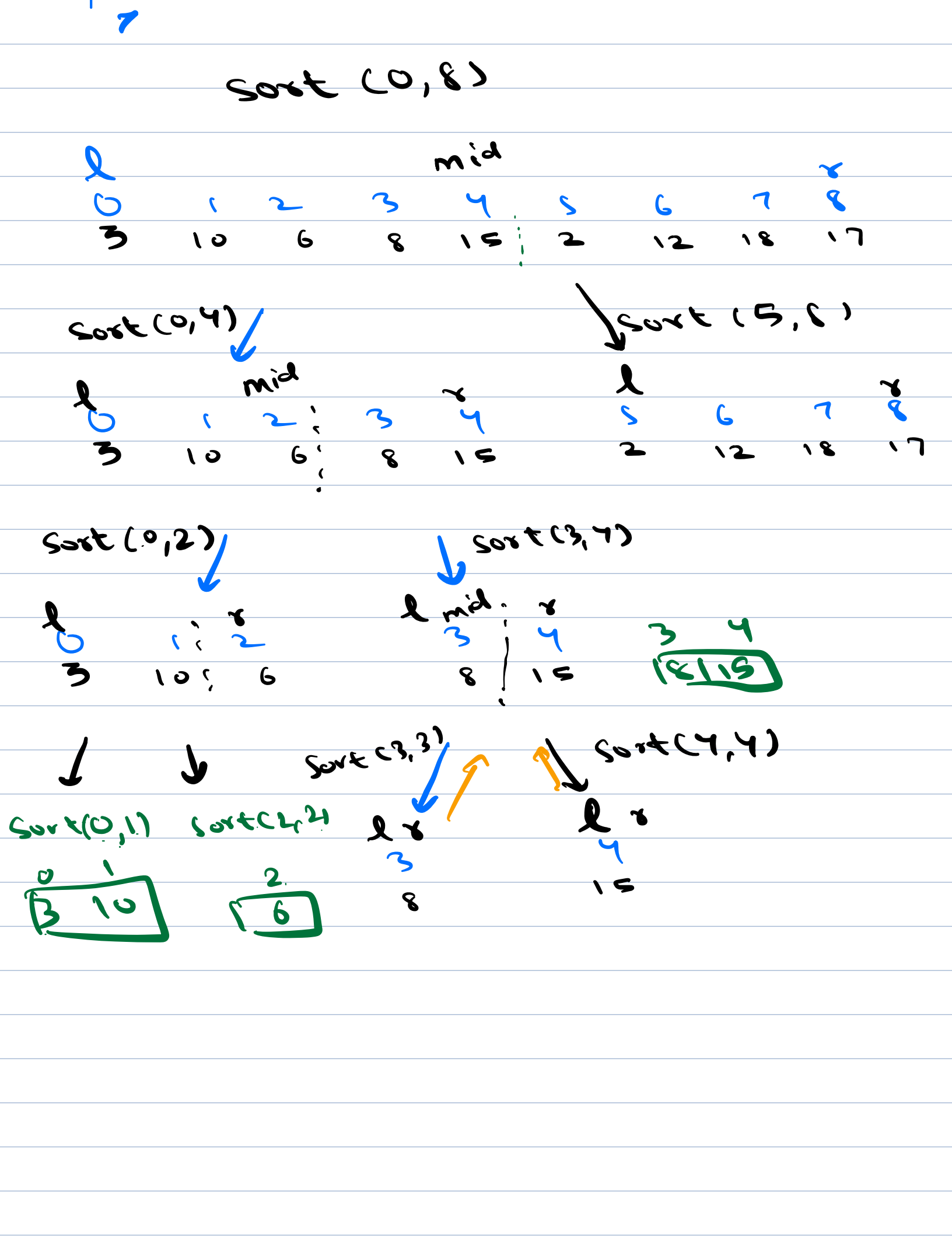
sort (A, 0, n-1)

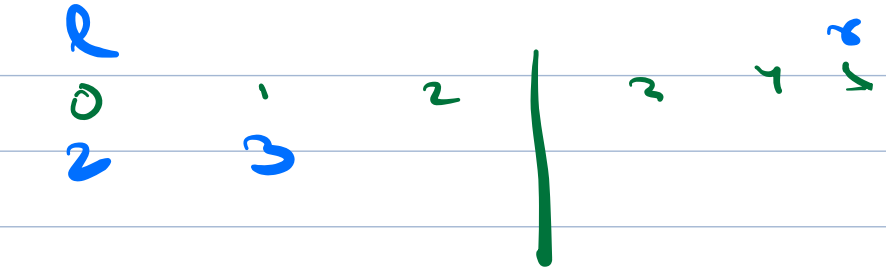
// given an A[], l and r ; sort array from idx l → r

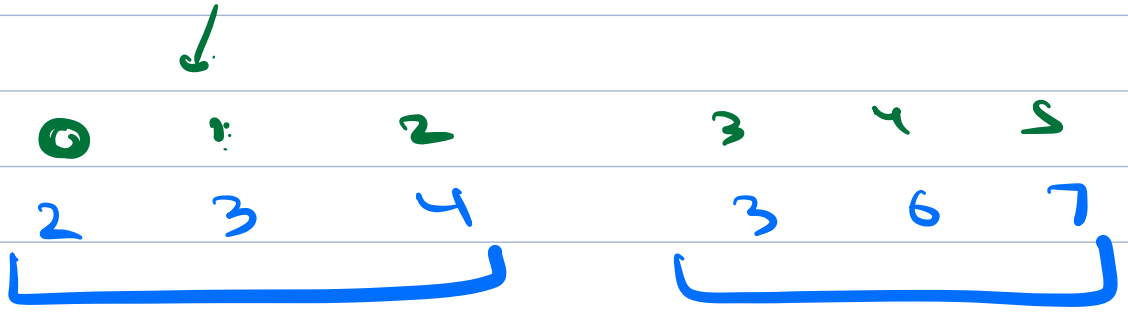
```
void sort (int A[], int l, int r) {
    if (l == r)  return
    int mid = (l + r) / 2
    sort (A, l, mid)
    sort (A, mid+1, r)
    merge (A, l, mid, r)
}
```

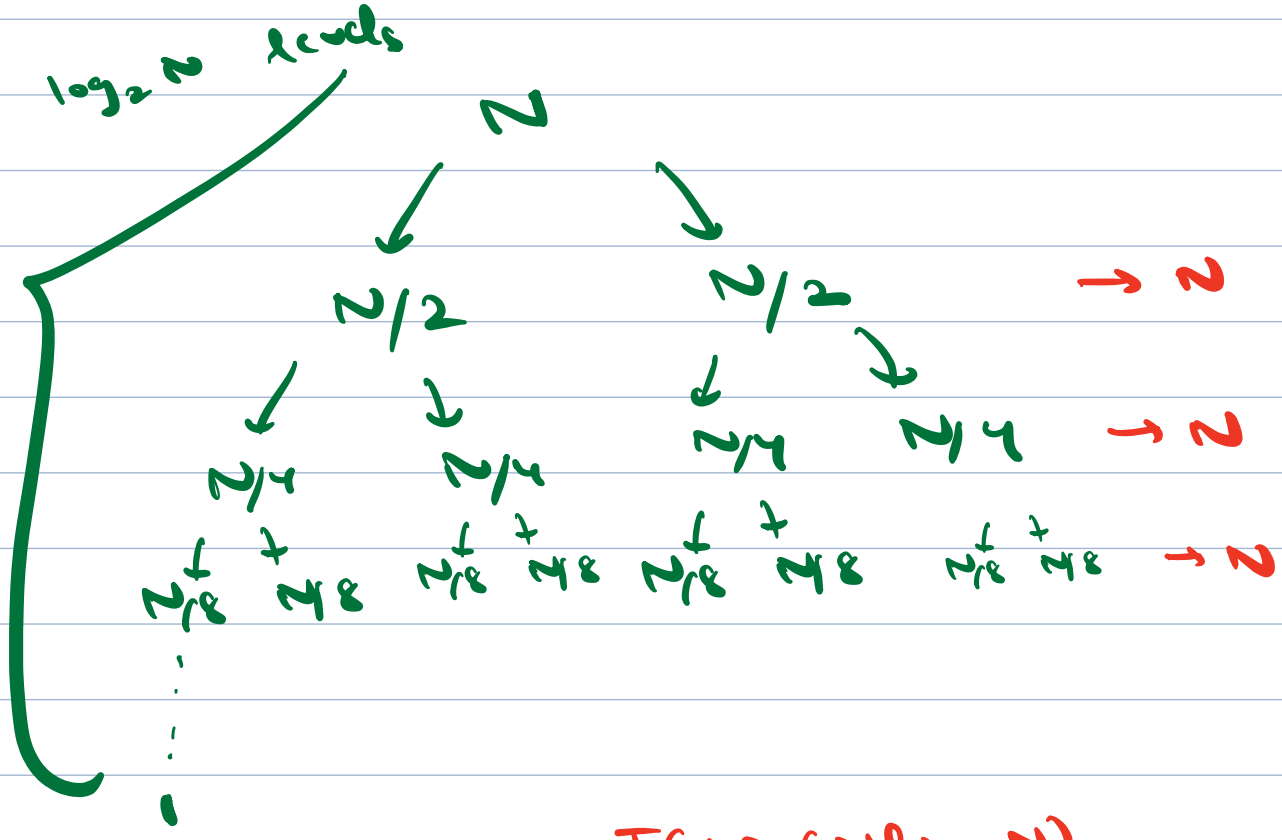

```
void merge (int A[], int l, int mid,
            int r) {
    int n1 = mid - l + 1
    int n2 = r - (mid+1) + 1
    int B[n1], C[n2]
    int k = 0
    for (idx = l ; idx ≤ mid ; idx++)
        B[k] = A[idx]   k++
    k = 0
```

```
for (idx = mid+1 ; idx <= r ; idx++)
    C[k] = A[idx]      k++

// Merging
int b=0, c=0, k=l
while (b < n1 && c < n2)
    if (B[b] <= C[c])
        A[k]=B[b]
        k++   b++
    else
        A[k]=C[c]
        k++   c++

while (b < n1)
    A[k]=B[b]
    k++   b++

while (c < n2)
    A[k]=C[c]
    k++   c++
return
```


Sort (0, 8)
l
mid
r
0 1 2 3 4 5 6 7 8
3 10 6 8 15 2 12 18 17
Sort(0,4)
Sort (5, 8)
mid
Sort(0,2)
Sort(3,4)
Sort(0,1)
Sort(2,2)
Sort(3,3)
Sort(4,4)
3 10
6
8
15
3 4
8 15







TC: $O(N \log_2 N)$

SC: $O(\log_2 N + N)$

↓ Recursive space

↓ Merge space of B and C

Inplace → No extra space, SC: O(1)

Merge sort ✗

Insertion, Selection ✓

Stable → Relative order of equal elements should not change

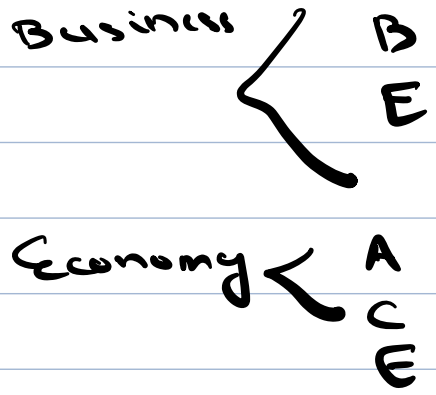
A → Economy

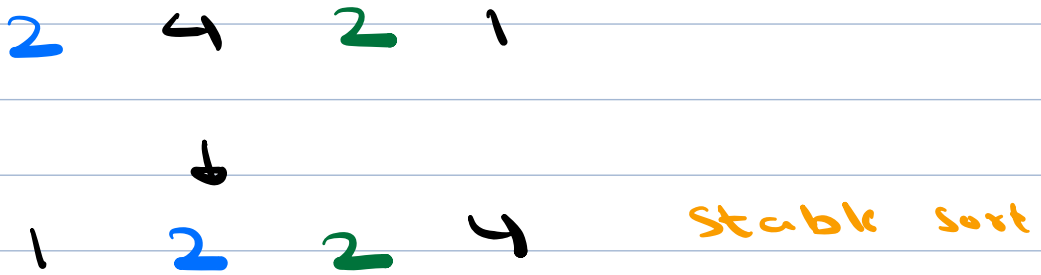
B → Business

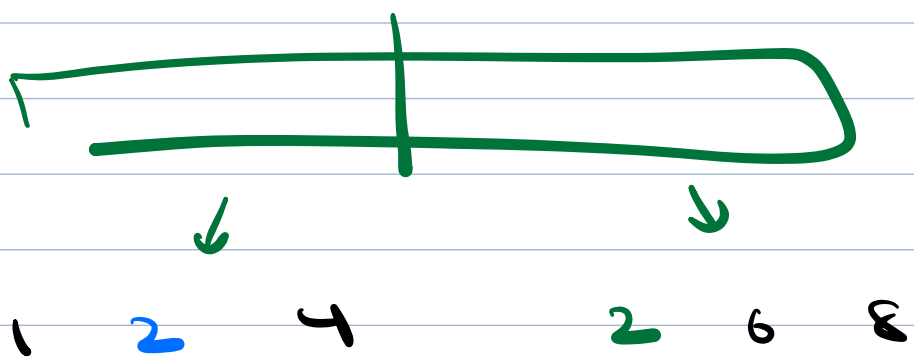
C → Economy

D → Economy

E → Business







Merge sort ✓ selection ✗