Sorting 1 Oct

Friday, 8 October 2021 8:57 PM

Sorting 1

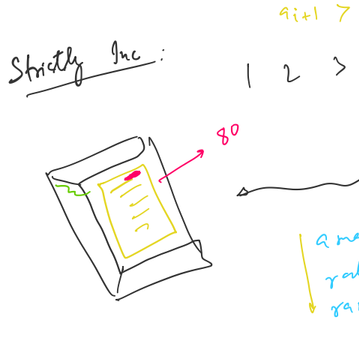
→ Arranging data in a particular order!
→ ascending / increasing
→ descending / decreasing ..

⊙ Increasing : $a_{i+1} \geq a_i$

1 2 2 3 4 4 7 8 9 9

⊙ Strictly Inc : $a_{i+1} > a_i$

1 2 3 5 7 8



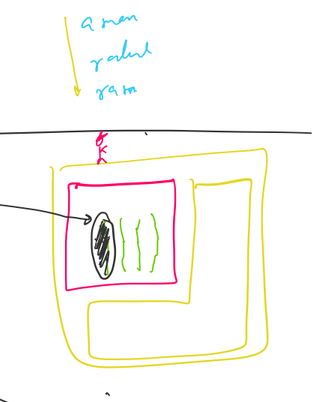
80

Rahul

aman
rahul
ram

⊙ Real life Scenario

1) 1 5 7 12 27



⊙ Maintain a sorted list.
⊙ Add element one by one..
→ finding the correct place
→ Inserting it ...

→ 3, 5, 2, 9, 6

⊙ Insertion Sort

2 3 5 6 9

n

n

N

TC: O(N²) : Worst Case

TC: O(N) : Best Case

1 2 3 4 5

1 2 3 4 5

n/2 → O(1)

n/2 → O(n)

⑥

① — ③ — ⑤ — ⑦ — ⑧

~ 10⁵

~ 10⁶

10¹⁰

10¹²

① Search → O(N)

② Insert → O(1)

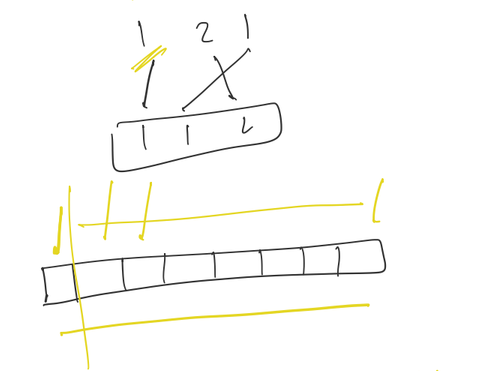
⊙ In place sorting : Not use any extra space..

⊙ Stable sorting :

✓

3 2 3 5

2 3 3 5



Rahul 1

Aman 2

Ketan 1

⊙ 2 3 5 8 18 ✓

2) 2 3 5 8 10

O(N) → O(N²)

1 2 3 4 5

⊙ Select the smallest element
→ put it in front...

TC: O(N²) : Worst

⊙ Selection sort

- stable ✓
- Inplace ✓

✓ ✓ ✓

1 2 1

1 1 2

N + N-1 + N-2 + ... + 1

$\frac{N(N+1)}{2}$ → O(N²) ✓

3)

5 10 3 2 8

2 5 10 3 8

2 3 5 10 8

2 3 5 8 10

TC: O(N²) : Worst Case

TC: O(N) : Best Case.

1 2 3 4 5 6

⊙ Bubble Sort

- No swaps → Already sorted!

stable ✓

inplace ✓

- quick
- merge
- heap

} hybrid

Tim

Intro

⊙ TC: O(N log N)

⊙ Count Sort

Q Given an array of 0 & 1 ..
Sort the array!

1 1 0 1 1 0 0 0 1 0 1 0 1 1

Cnt 0 : 6

cnt 1 : 8

0 0 0 0 0 0 1 1 1 1 1 1 1 1

TC: O(N)

⊙ Sort a string

S: "banana"

"aaabnn"

0 1 2 ... 25

3 1 ... 2

K

N log N

"aaabnn"

TC: O(N+K)

Alphabet Set Size

SC: O(K)

Q Array of integers. $-20 \leq A_i \leq 20$

Sort the array

20 + Ai

0 1 ... 40

-20 -19 ... 20

3 2

-20, -20, -19, ... -19, 19 ...

N log N

R: Max Ai – Min Ai

N = 10⁶

A : [10⁻⁹, 10⁹]

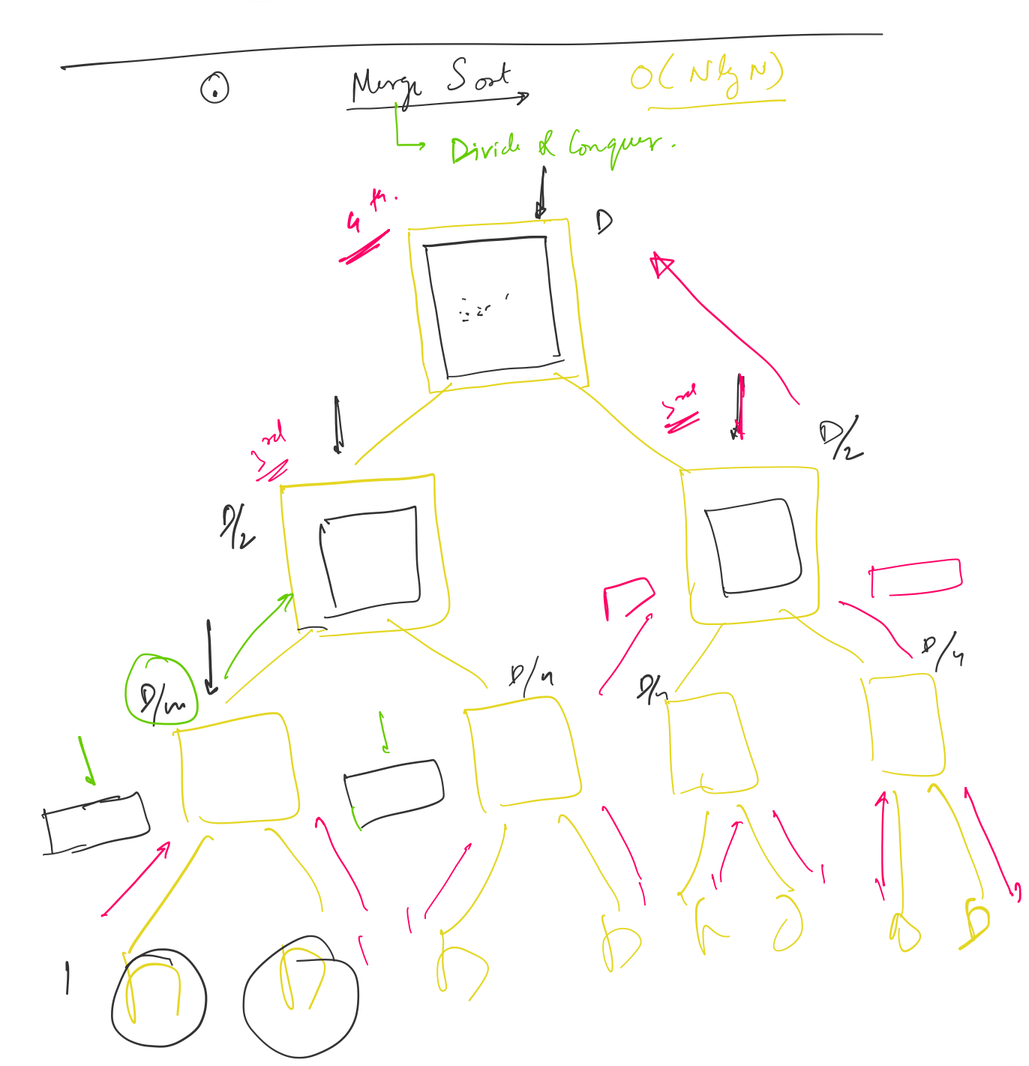
TC: O(N+R)

SC: O(R)

⊙ If the range of values is less
→ Use count sort!

⊙ Merge Sort O(N log N)

→ Divide & Conquer.



D

N/2

N/4

1

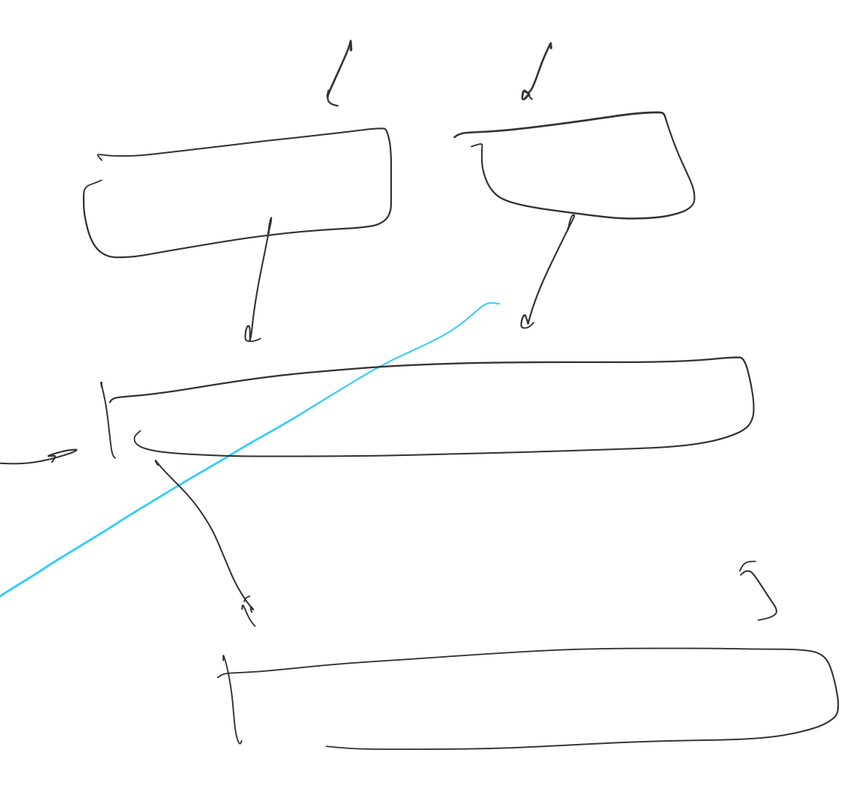
⊙ Merging 2 sorted arrays into 1 sorted array!

A: 2 3 5 (n)

B: 1 4 8 (m)

C: 1 2 3 4 5 8

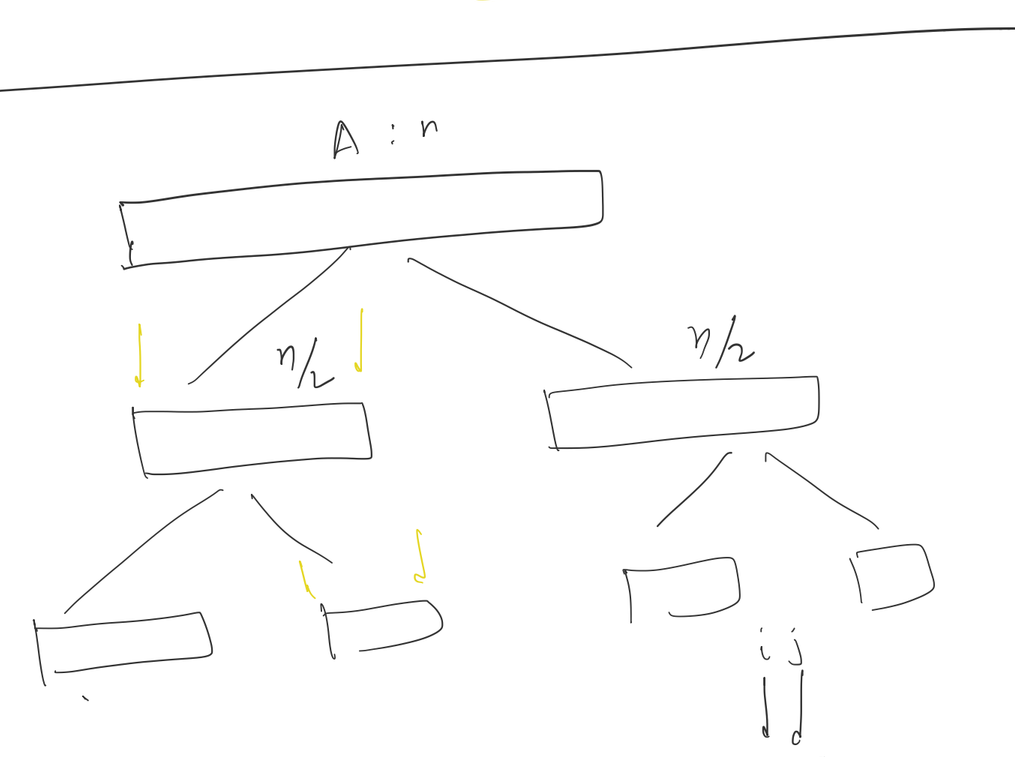
O(n+m)



```
i=0, j=0,      k=0
while( i < n && j < m) {
    if( Ai < Bj) {
        Ck = Ai ;
        i++; k++;
    }
    else {
        Ck = Bj ;
        j++; k++;
    }
}
while ( i < n ) {
    Ck = Ai
    i++; k++;
}
while( j < m) {
    Ck = Bj
    j++; k++;
}
```

O(m+n)

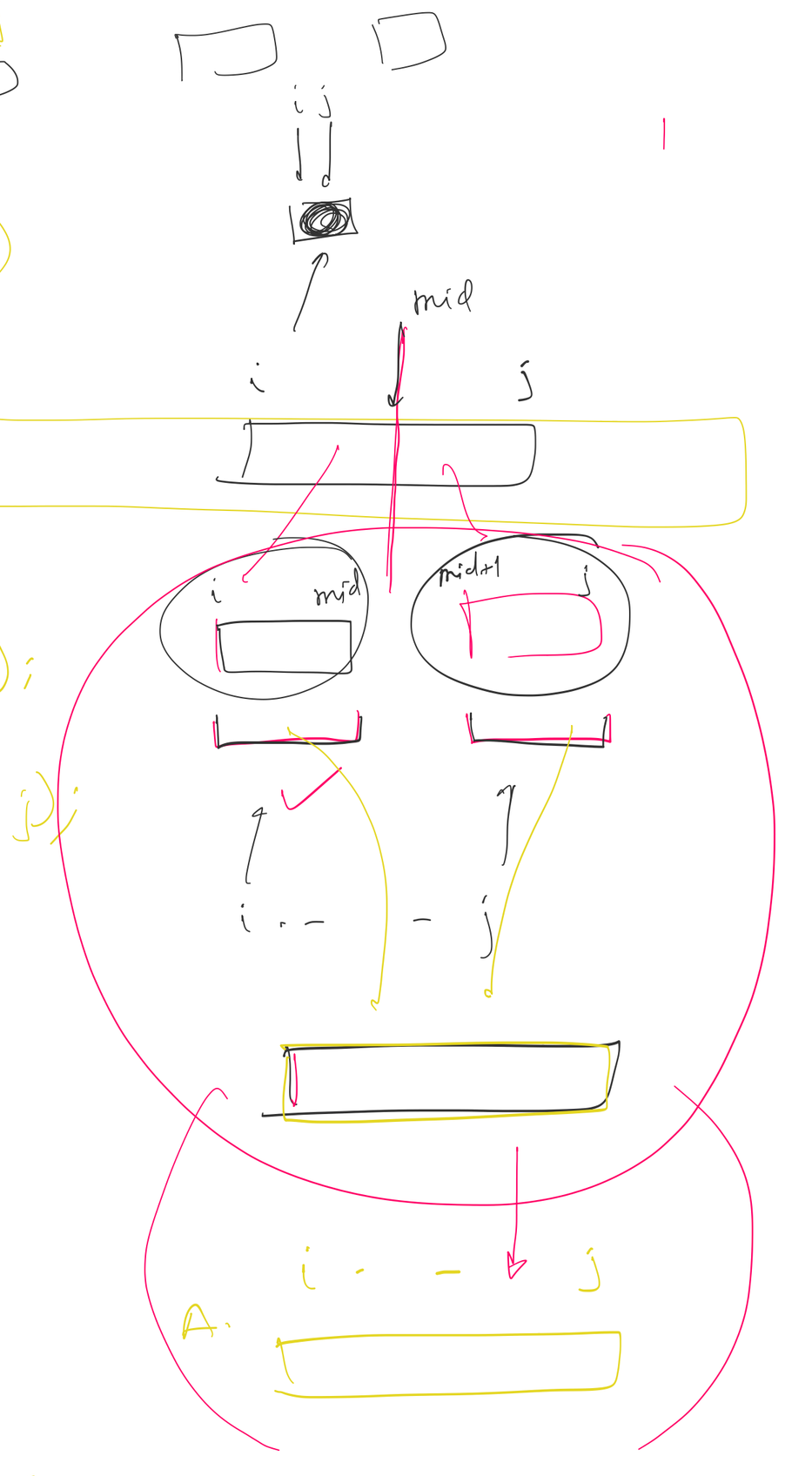
A : n



n/2

n/2

```
mergeSort( A, i, j)
    if( i==j)
        return;
    mid = (i+j)/2
    ✓ mergeSort( A, i, mid);
    ✓ mergeSort( A, mid+1, j);
    merge (A, i, j);
}
```



i mid j

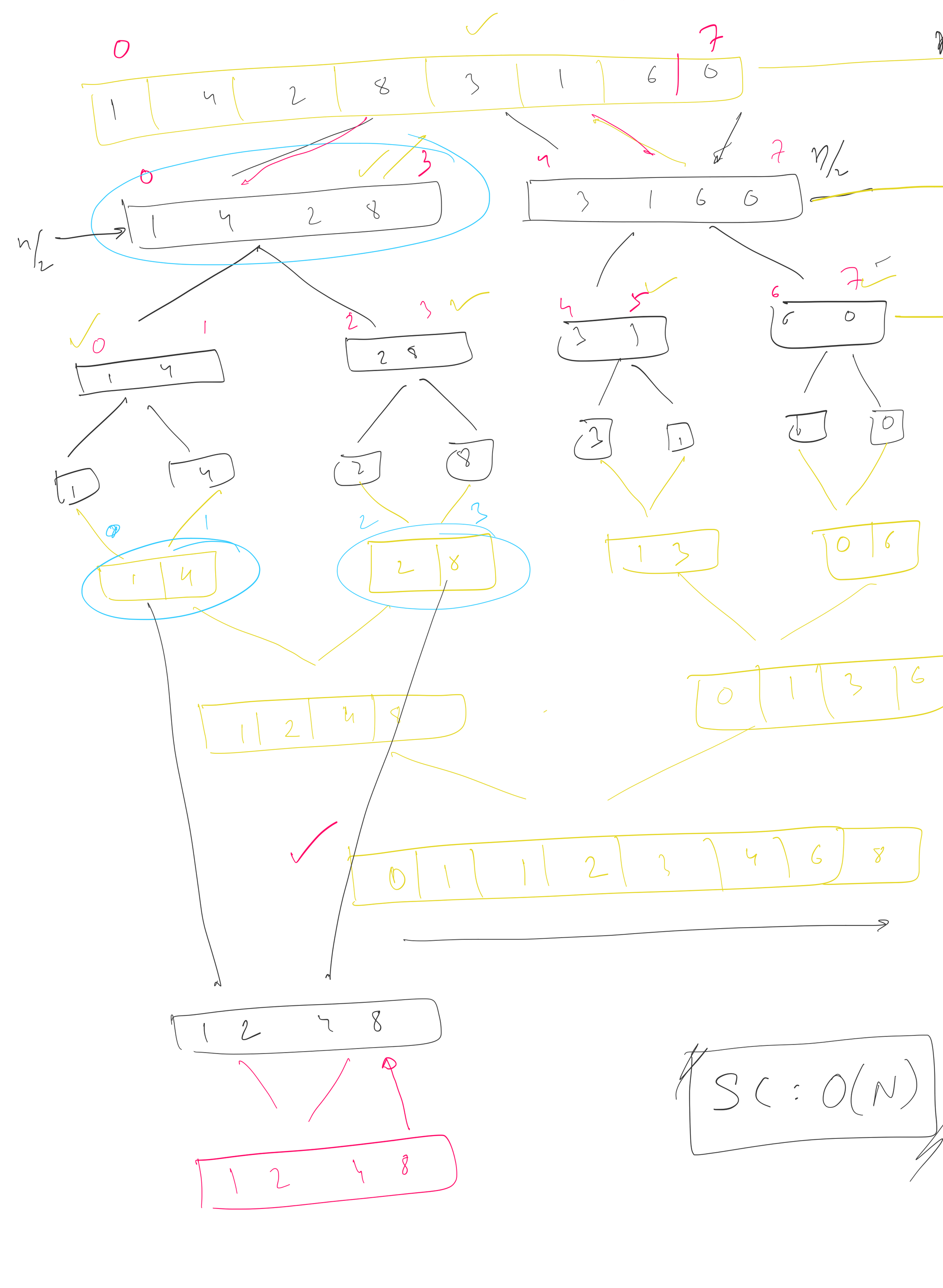
i mid mid+1 j

i .. j

A

0 7

1 4 2 8 3 1 6 0



n/2

n → n

n/2 → 2 × n/2

n/4 → 4 × n/4

1

log n

TC: O(N.log N)

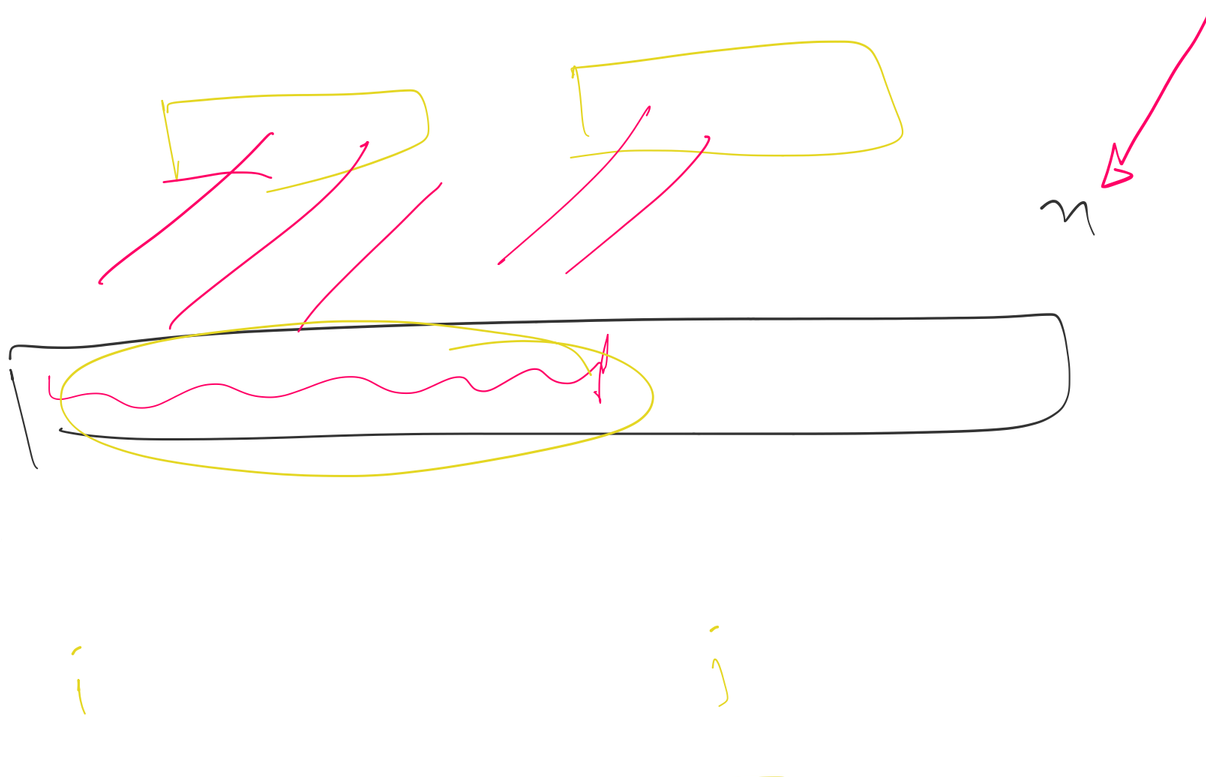
T(N) = 2×T(N/2) + N

0 1 1 2 3 4 6 8

1 2 4 8

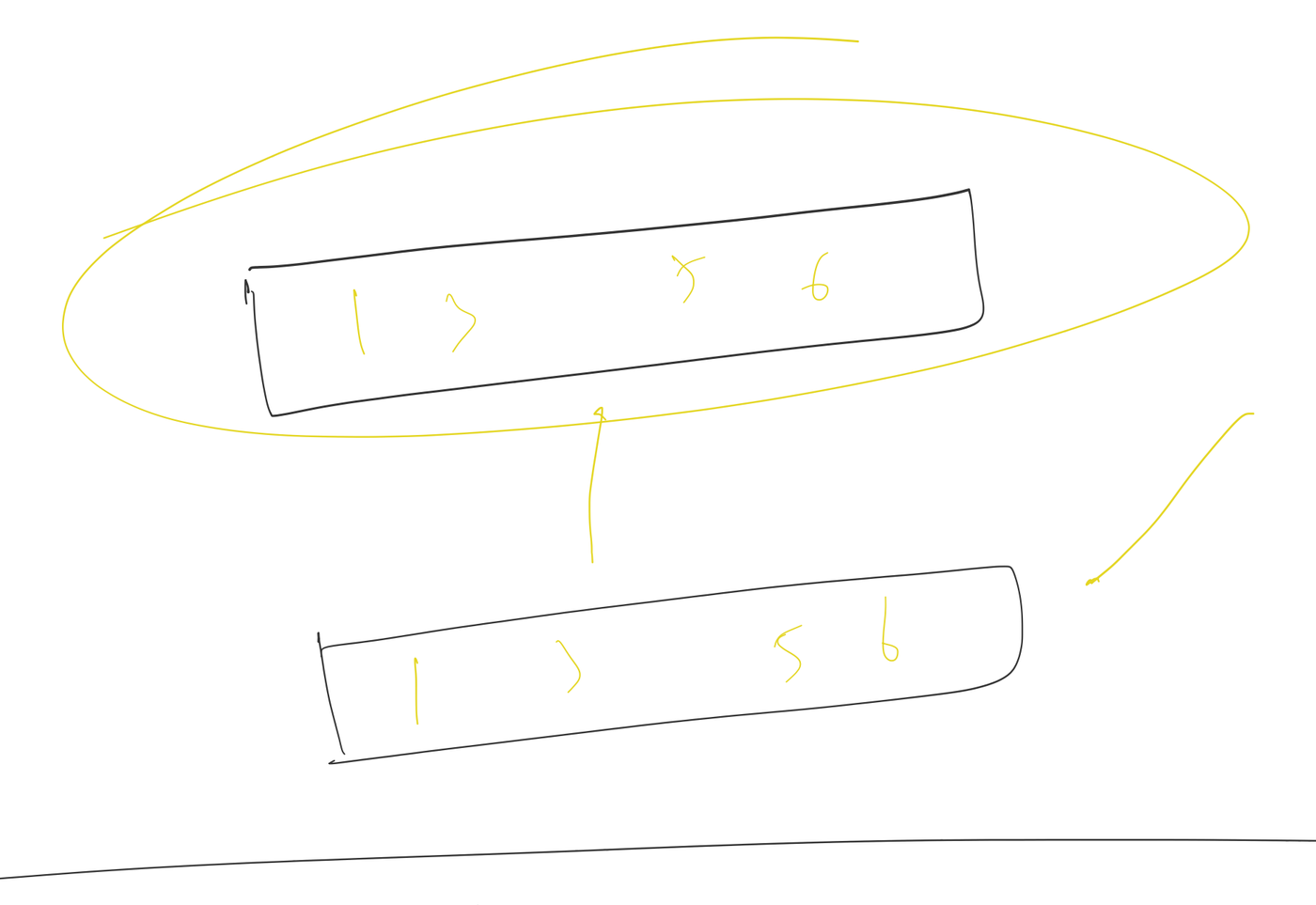
1 2 4 8

SC: O(N)



i j

A.



1 3 5 6

1 3 5 6

Inversion Count

Q

A: Cnt (i, j)

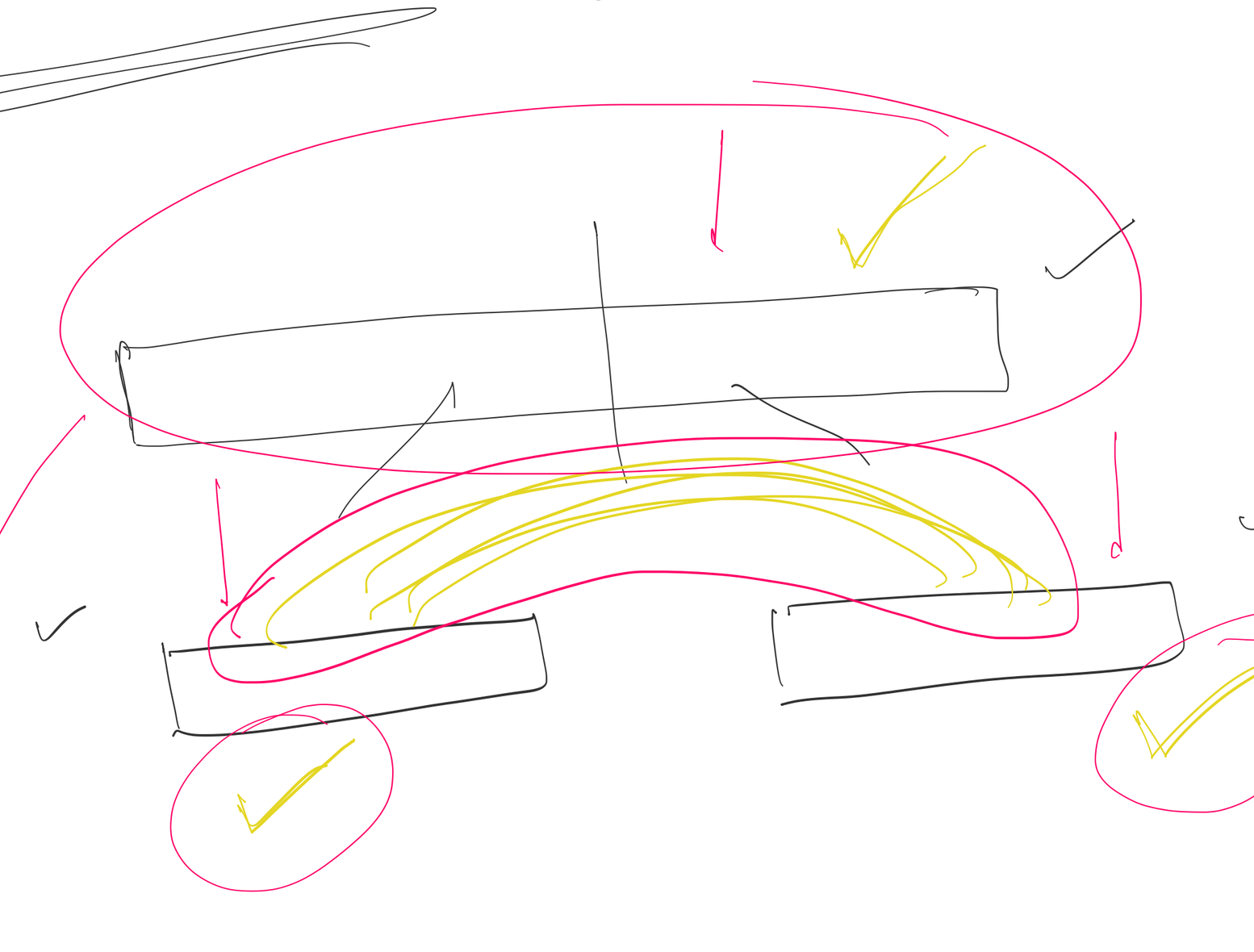
i < j

$A_i > A_j$

5 4 8 3

~~1~~ ~~2~~ ~~3~~ ④

BF: O(N²) ✓



① Count inv

② Sort

4 5 8

1 3 6

~~2~~ ~~4~~ 7

1 3 4 5 6 8