Two Pointers

Tuesday, 9 March 2021 4:36 PM

=> int (*p) ✗

Q=> Given two sorted array. You need to merge them.

A: [1, 3, 7, 8, 15] ≈ N

B: [2, 4, 5, 6, 15, 120] ≈ N

=> C: [1 _ 3 _ 7 _ 8 _ 15 _

O(N)

A: [1 3 5 7 15 20]

B: [2 4 5 6 9 10 31]

=> C: 1 2 3 3 4 4 6 → Number of inversions

c[i] = How many elements are there in array A which are smaller than or equal to B[i]

O(N)

```
int j=0; int cnt=0;
for ( int i=0; i<n ; i++) {
    for ( ; j<n ; j++ ) {
        if (A[j] ≤ B[i]) cnt++;
        else break;
    }
    cout << cnt << endl;
}
```

for (0 ; j<n ; j++)

=> O(N^2)

A is sorted

≤ ≤ ≤ > > > >

→ O(N^2)

Two-pointer

≤ ≤ ≤ ≤

B[i] → B[i+1]

A=> [1 3, 5, 5, 9 10 12 17]

You need to check whether there exists a pair (i,j) such that i≠j && A[i]+A[j] = target

target = 17

j=n-1

```
for (i=0 ; i<n ; i++) {
    for ( ; j>i ; j-- ) {
        if (a[i]+a[j] == target) return true;
        else if (a[i]+a[j] < target) break;
```

< < < < == > > > >

B[i] → A[i]

< < < < == > > > >

B[i] A[i]

break

A[i] A[i+1]

1+3+2+5

3+2

Q → Given an array. Find max length of subarray having sum ≤ K. (Positive)

A = [1 3 2 5 1 1 6 7]

≤ K 10

1+3+2+5

3+2+5

Ans = 4

l=4, K=10

```
int ans=0; j=0 sum=0
for ( int i=0 ; i<n ; i++ ) {
    int sum=0;
    for ( ; j<n ; j++ ) {
        if (sum + a[j] ≤ K) sum += a[j]
        else break;
    }
    ans = max(ans, j-i)
    sum -= a[i]
}
```

count++

ans = (j-i)

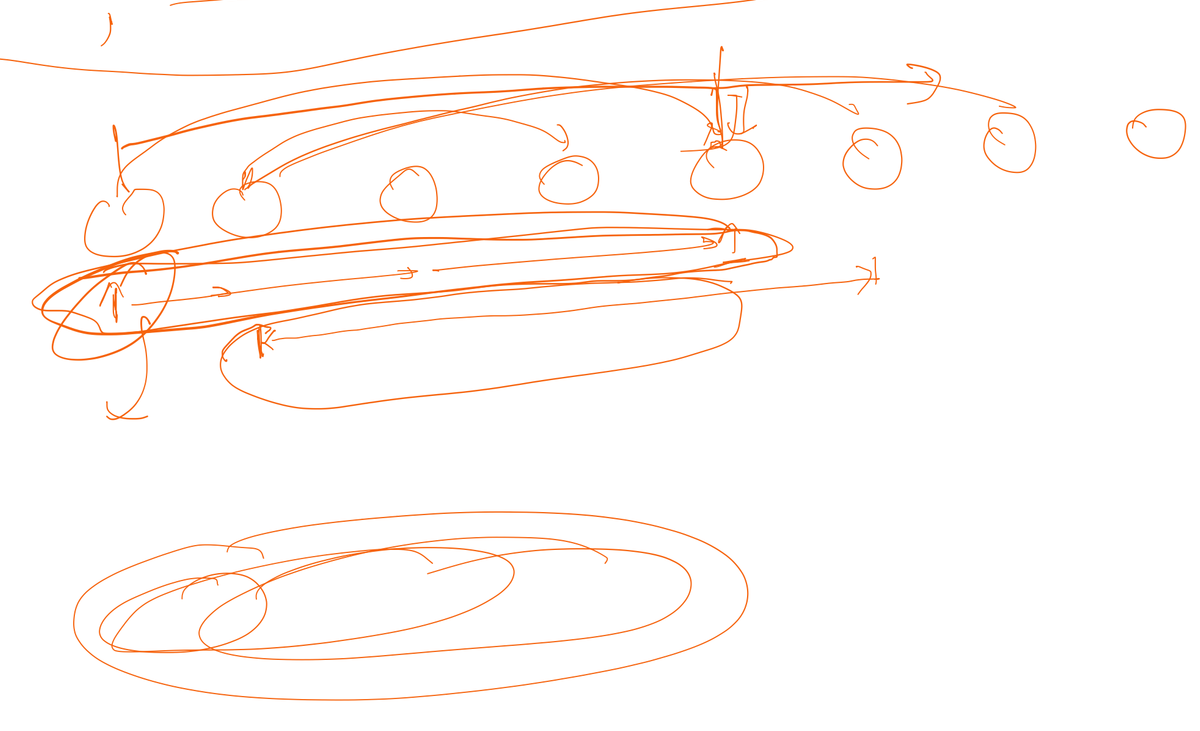
O(N)

O(N^2)

i=0 i=1

j=1

j=e ... (N)



ans = j-i

1 3 2 5

1 1 6

0 → 3

0,0

0 1

0 2

0 3

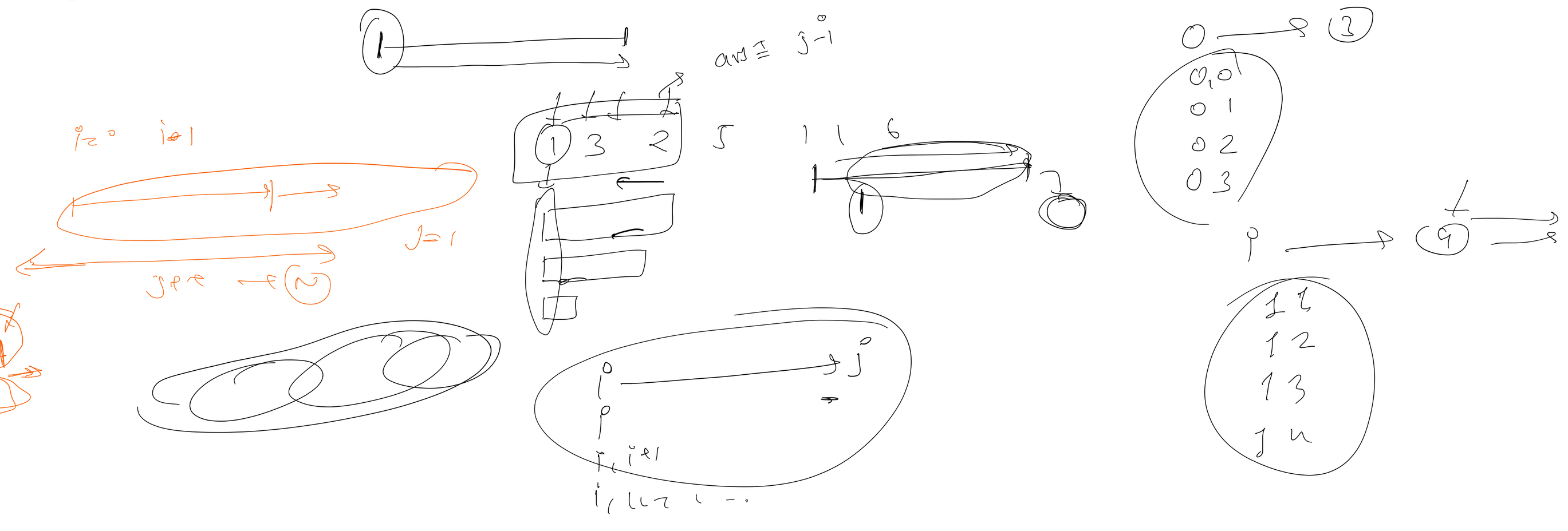
i → 4

1 1

1 2

1 3

1 4



Count number of subarrays such that sum ≤ K

Two properties:

i j

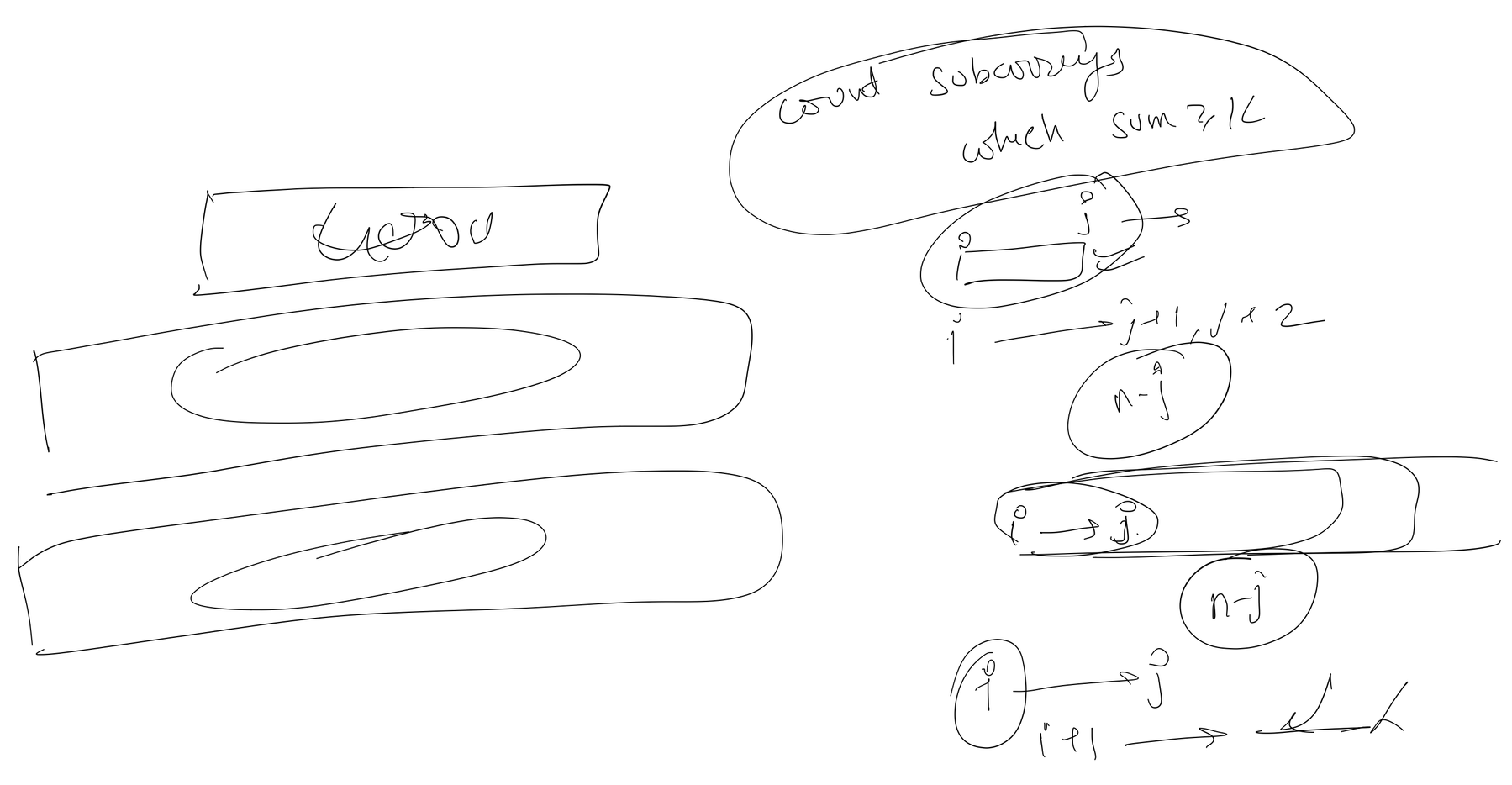
00000 → Good

i p q j

i ≤ p, q ≤ j ✓ Good

≤ K >K

≤ K



Good

Count subarrays which sum ≥ K

i j → j+1, j+2

n-j

i → j

n-j

i → j

i+1

i i+1 → j

≥ K

Q => Find the shortest subarray with ≥ K distinct element.

K=3

=> 2 6 4 4 3 6 8 3 — ans=3

≥ K == K

d = 1 2 1

A = ~~1 1 1~~ 2 2 1 1 3 3 3 1 1 2 2 3 3 1 1 2

K=3

count[0] = ~~0~~ 1 2 3 ~~2~~ 1 0

ans = 4

```
d=0, count[N]={0}; j=0;
for ( int i=0; i<n ; i++ ) {
    for ( ; j<n ; j++ ) {
        count[a[j]]++;
        if (count[a[j]]==1) d++;
        if (distinct == K) {
            ans = min(ans, j-i+1);
            break;
        }
    }
    count[a[i]]--
    if (count[a[i]] == 0) d--;
}
```

set log(N)

map<int,int> → log(N)

N

int distinct = 0;

int count[N+1] = {0};

0 1 2 3

count[2] → 0 1 2 3

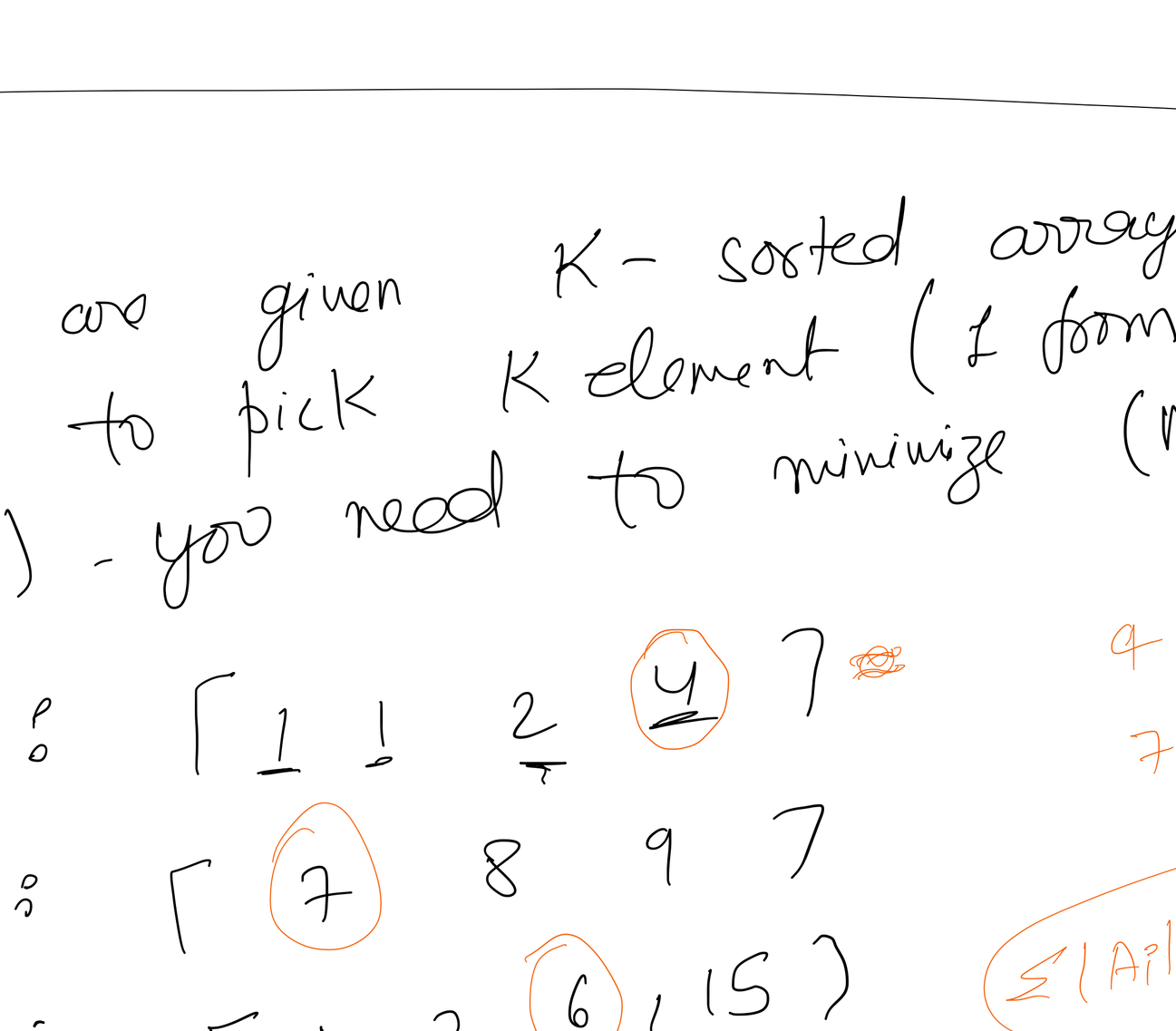
count[a[i]]++;

if (count(a[i]) == 1) distinct++;

count[a[i]]--

if (count(a[i]) == 0) distinct--;

1 1 1 2 3 3



== K

Good

Good

Good

< K

Good = false

i j j+1

i-1

You are given K-sorted arrays. You need to pick K element (1 from each of them). You need to minimize (max-min).

A1 : [1 1 2 4 7]

A2 : [7 8 9 7]

A3 : [1 3 6, 15]

4 6 7

7-4 = 3

Σ|Ai| ≈ 10^5

count[a[i].second]++;

ans=

[1, 1, 1, 2, 3, 4, 6, 7, 8, 9, 15]

[{1,1} {1,2} {1,3} {2,1} {3,3} {4,1} {6,3} {7,2} {8,2} {9,2} {15,3}]

ans = min (ans, j-i)

ans = min (ans, a[j].first - a[i].first)

7-1=6

7-2=5

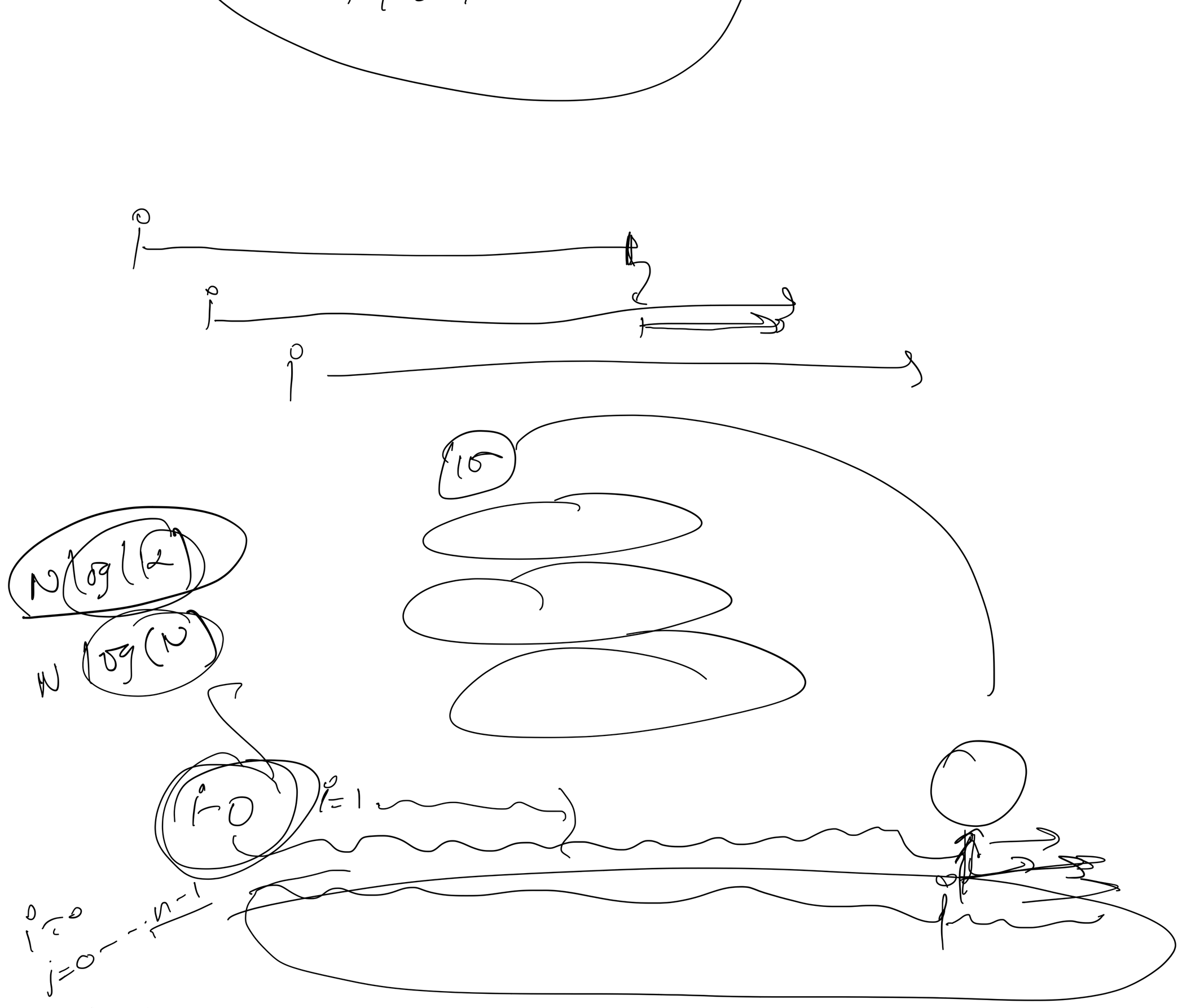
one

at least one element from K

{1, 1, 2, 3}

1, 1, 6, 15

7, 8, 9



N log(K)

N log(N)

i=0

i=1

i=0

j=0 - - - n-1

i=1

i=2

O(N) => O(N)