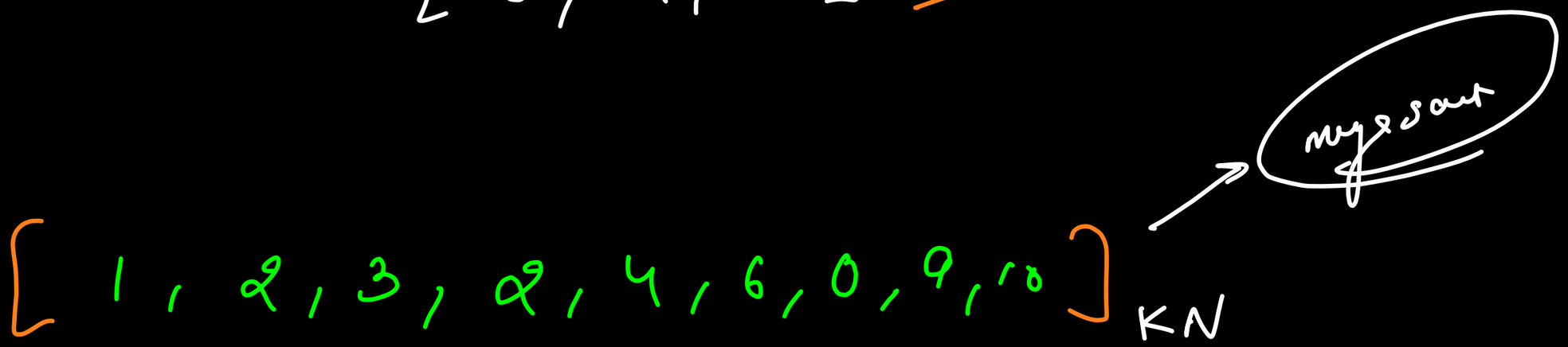
[ 1 , 2, 3 ] =

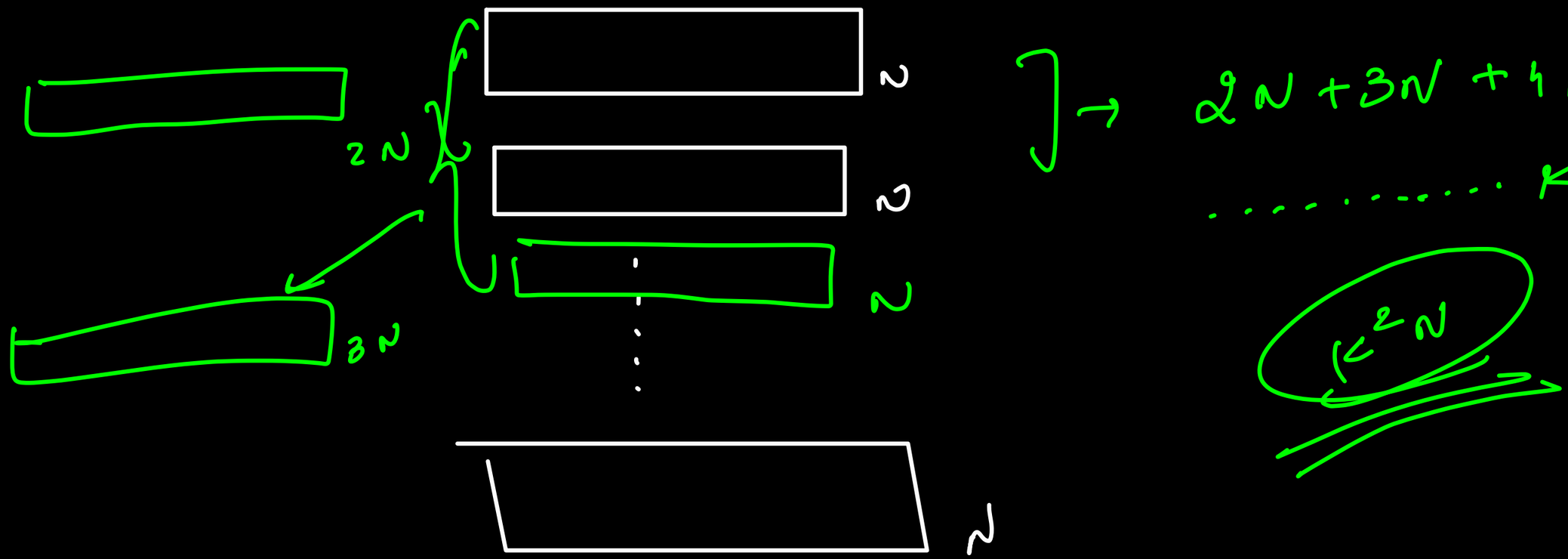
[ 2, 4, 6 ] =

[ 0, 9, 10 ] =

K → Sorted




KN log (KN)



k → smallest

min heap → [ 1, 2, 3 ] =

→ [ 2, 4, 6 ] =

→ [ 0, 9, 10 ] =

k candidates

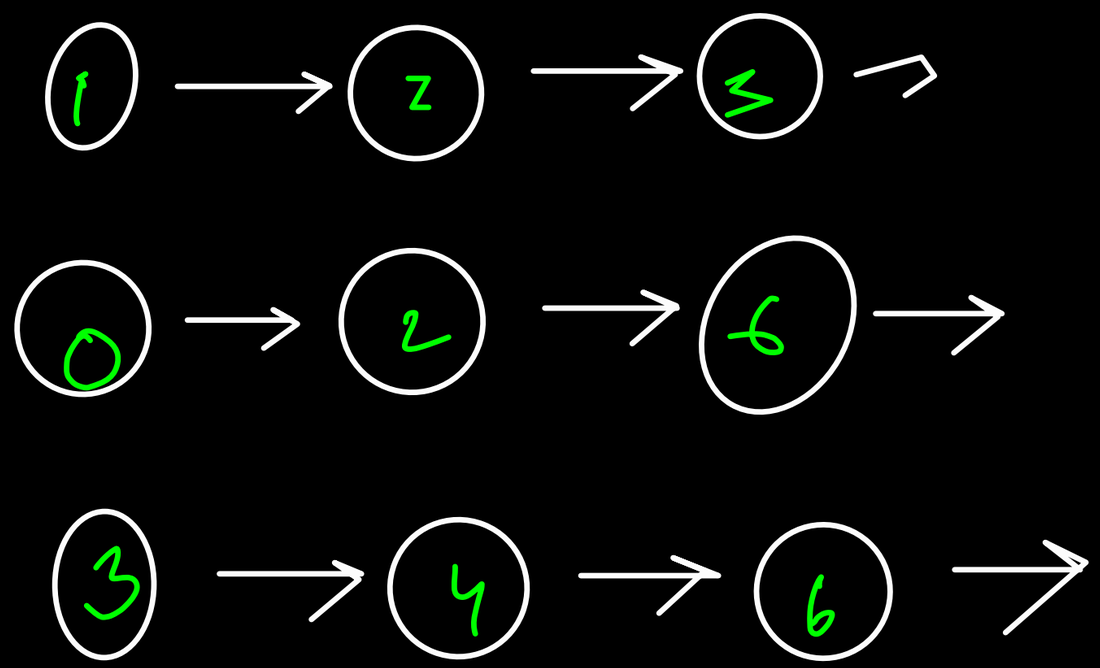
elem row index

[ 0, 1, 2, 2

j → final

O (nk log k)

```
curr = hp.top()
hp.pop();
hp.push({ row: curr.row, index: curr.index+1 })
```



```
(obj1, obj2) {
    return obj1.elem < obj2.elem;
};
```

Brute force → form all possible fractions → by making all possible pairs:

$O(n^2)$

0 1 2 3

[1, 2, 3, 5]

K th smallest

[ 1/2  2/3  3/5 ]

[ 1/3  2/5 ]

[ 1/5 ]

Min heap →

fraction
n - idx
d - idx

M

a → arr[0]
d → arr[i]
i ∈ [1, n-1]

[1, 2, 3, 5]

we don't want to generate all fractions

arrange the fractions in multiple sorted arrays

Merge k sorted arrays

[1, 2, 3, 5]
0 1 2 3

k-1 comparisons

k-1

[1/5, 1/3, 1/2, 1/1]

[2/5, 2/3, 2/2]

[3/5, 3/3]

[5/5]

Sorted

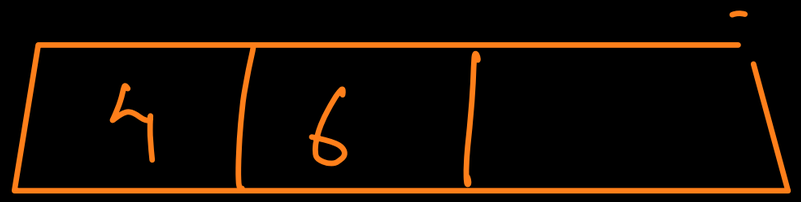
fraction
num_idx
den_idx

Min heap

O(k log k)

O(k)

[ 4, -5, 3, -2, -1, 6, -4, 8, -9 ]  k=3
0  1  2  3  4  5  6  7  8




Queue

O(n)

O(k)

-5

-2

JOIN THE DARKSIDE