n repeated 2n size array.

1

n

n exactly

8

2 1 3 2 2 4 5 2

2

2 → 4

exactly

n+1 unique

1 → n

n → 1

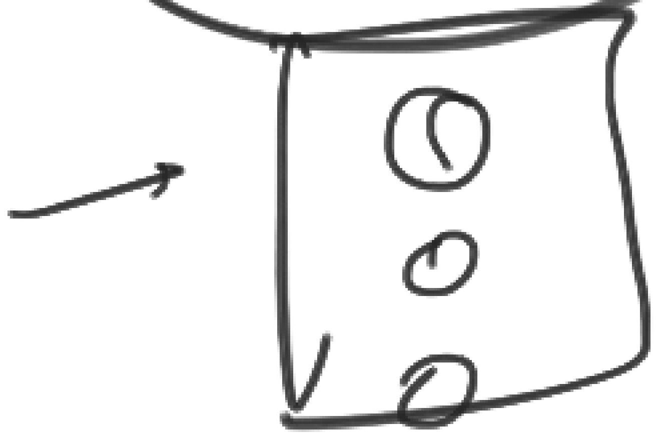
Searching

Hash Table

if (Arr[i])

HashSet

HashMap



key → value

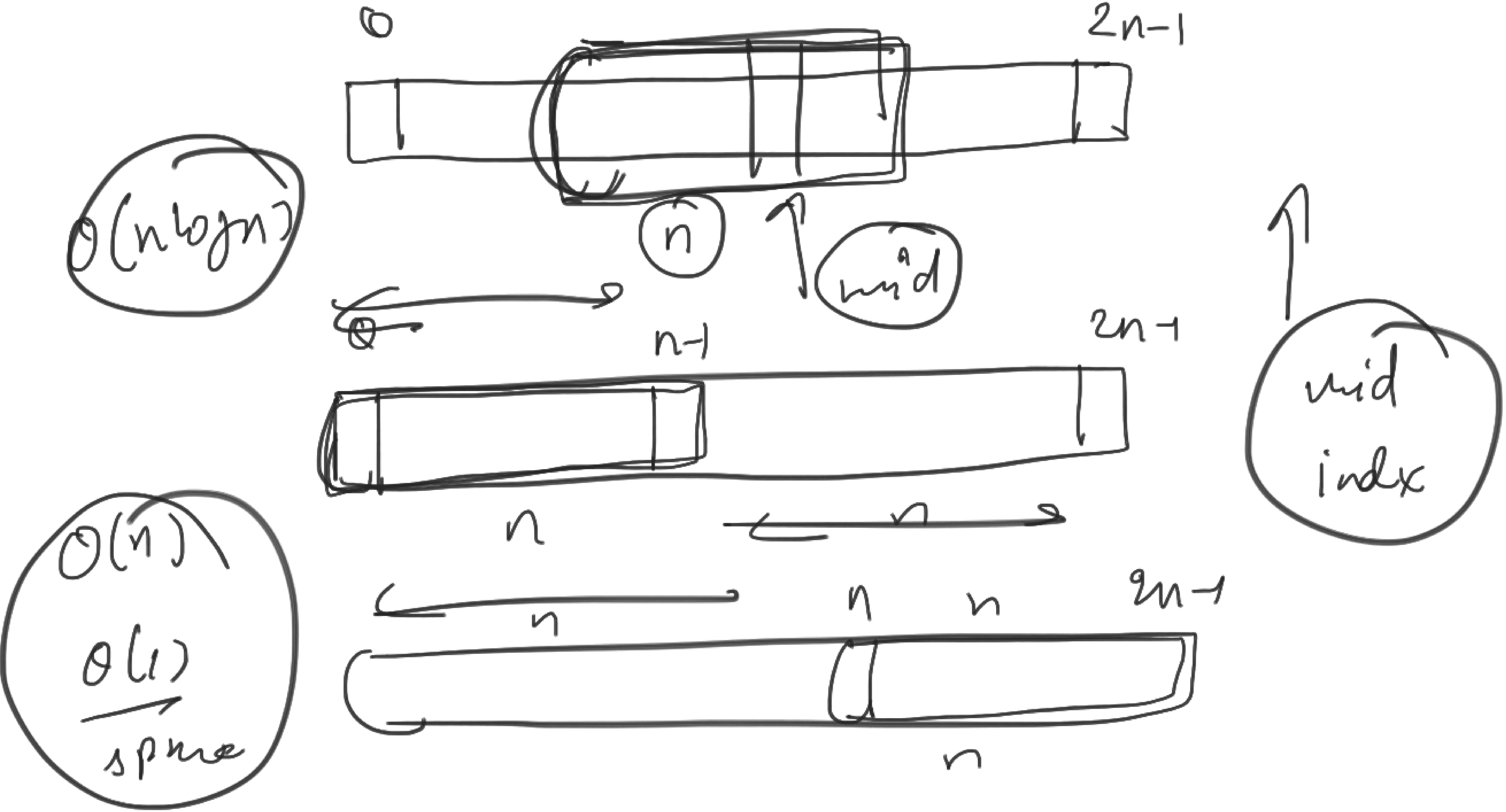
O(n)

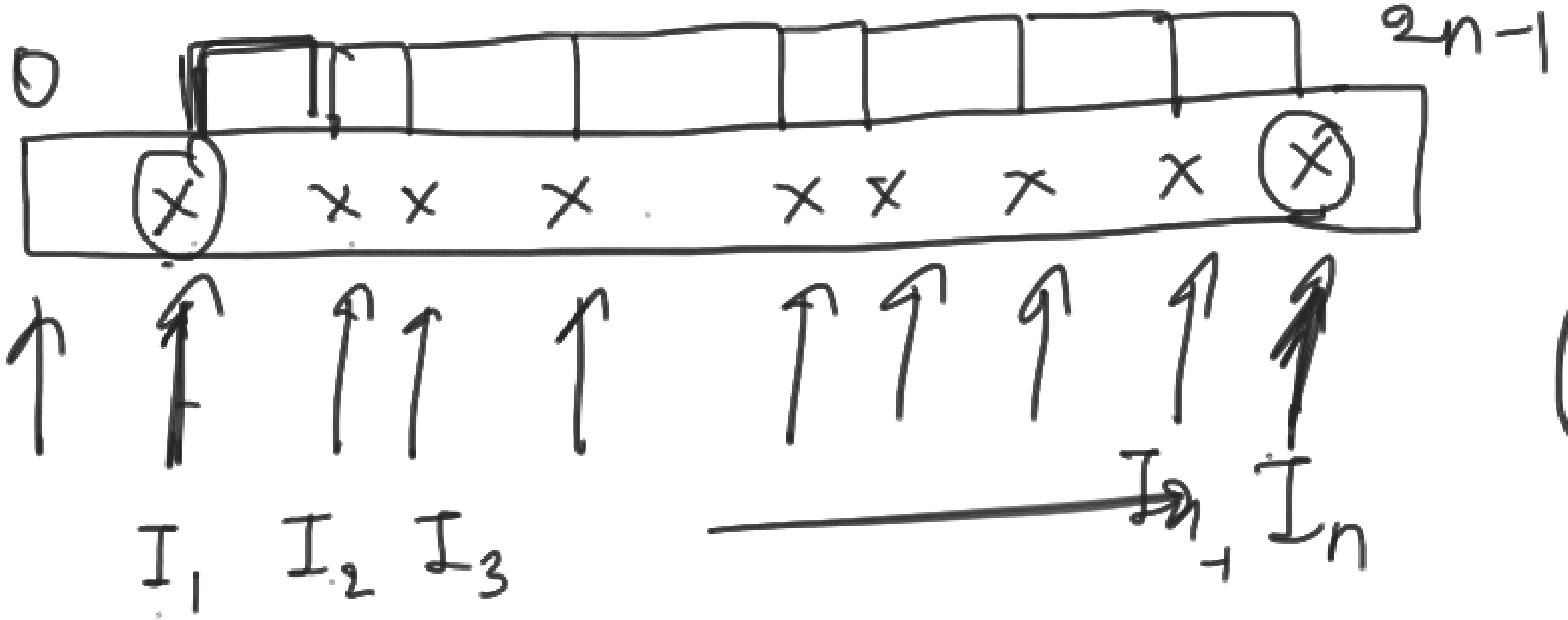
O(n)

O(n) span

O(1) span

M


0
2n-1
O(nlogn)
n
mid
0
n-1
2n-1
mid
indx
n
n
O(n)
O(1)
space
n
n
n
2n-1
n



x → n

4

$$\text{Avg gap} = \frac{(\cancel{I_2} - I_1) + (\cancel{I_3} - \cancel{I_2}) + (I_4 - \cancel{I_3}) + \cdots + (I_n - \cancel{I_{n-1}})}{n-1}$$

$$= \frac{(I_n - I_1)}{n-1}$$

Avg gap

Some gap will be less than Avg

Some gap will be greater then

2n

$$\text{Avg gap} = \frac{I_n - I_1}{n-1}$$

$$\frac{\max(I_n)}{\min(I_1)}$$

$$\frac{4\times 2-1}{4-1} = 7/3$$

$$\frac{2\times 3-1}{3-1} \Rightarrow 5/2 = \frac{2n-1-0}{n-1}$$

$n=1$

$$\frac{2n-1}{n-1}$$

$$\frac{2\times 2-1}{2-1} = 3$$

$$= \frac{2n-1}{n-1}$$

$n=2 \rightarrow 3$

$n=3 \rightarrow 5/2$

$n=4 - 7/3$

$n \rightarrow n - 2$

$n \rightarrow \infty$

$$\frac{2n-1}{n-1} = \frac{2-1/n}{1-1/n}$$

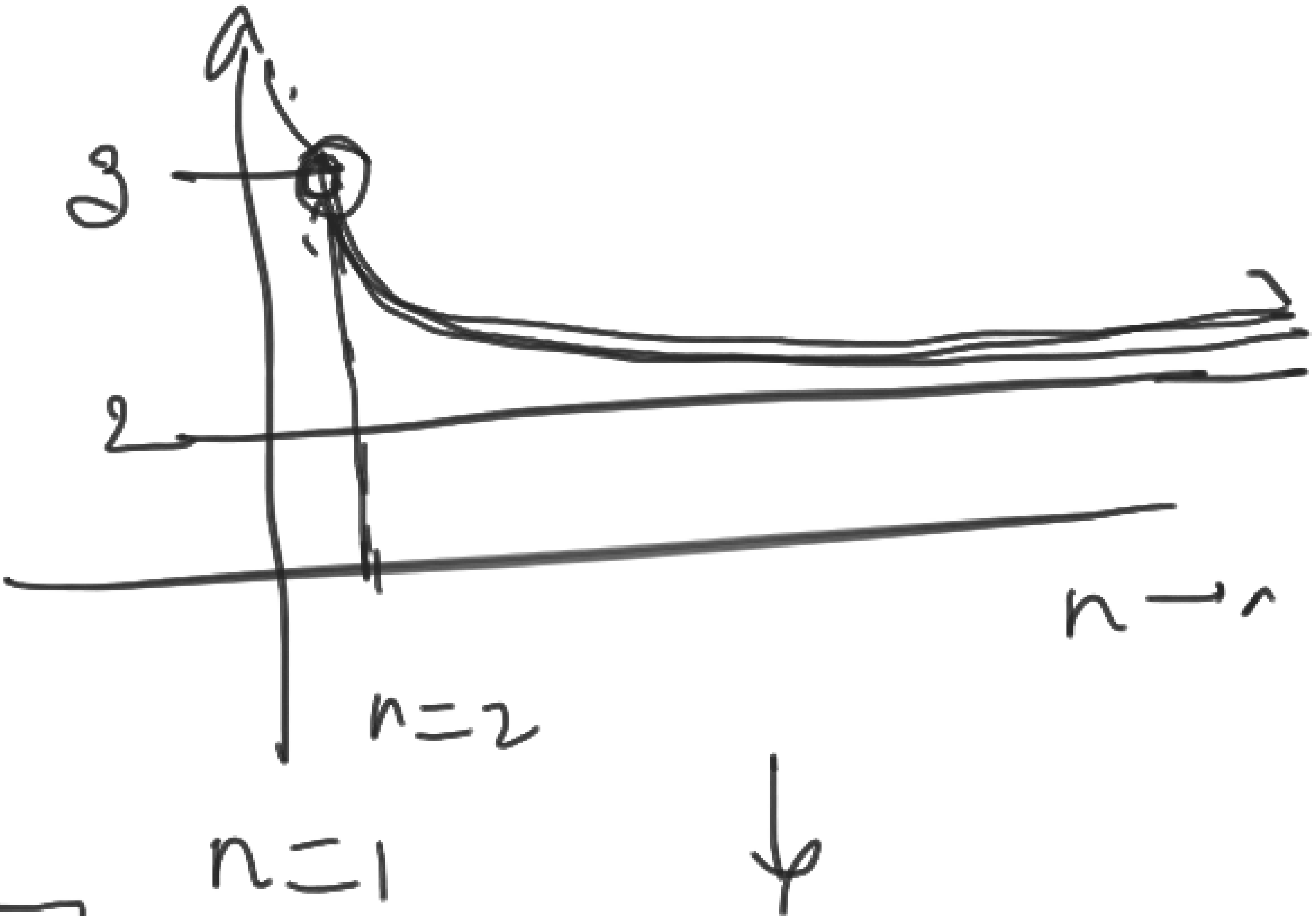
$n \to \infty$

$= 2$

x — y

3

Avg gap = 3

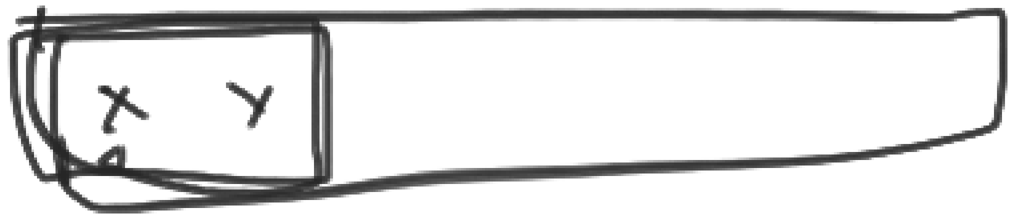


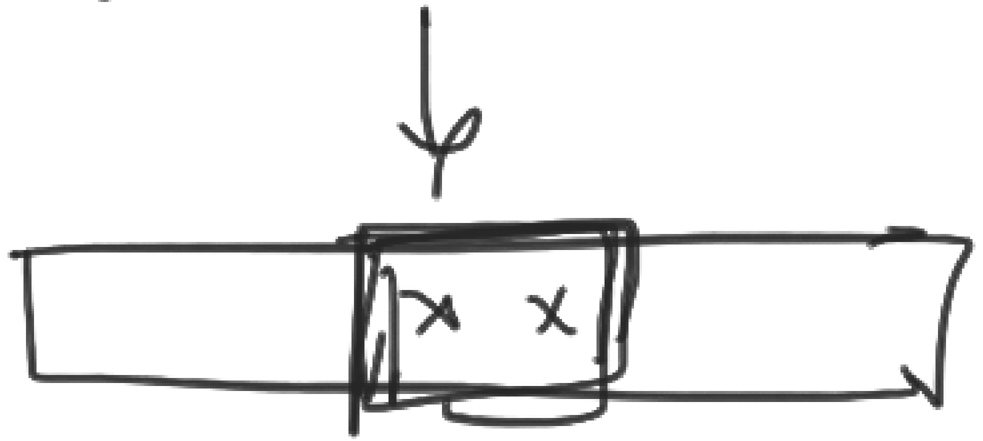
x x

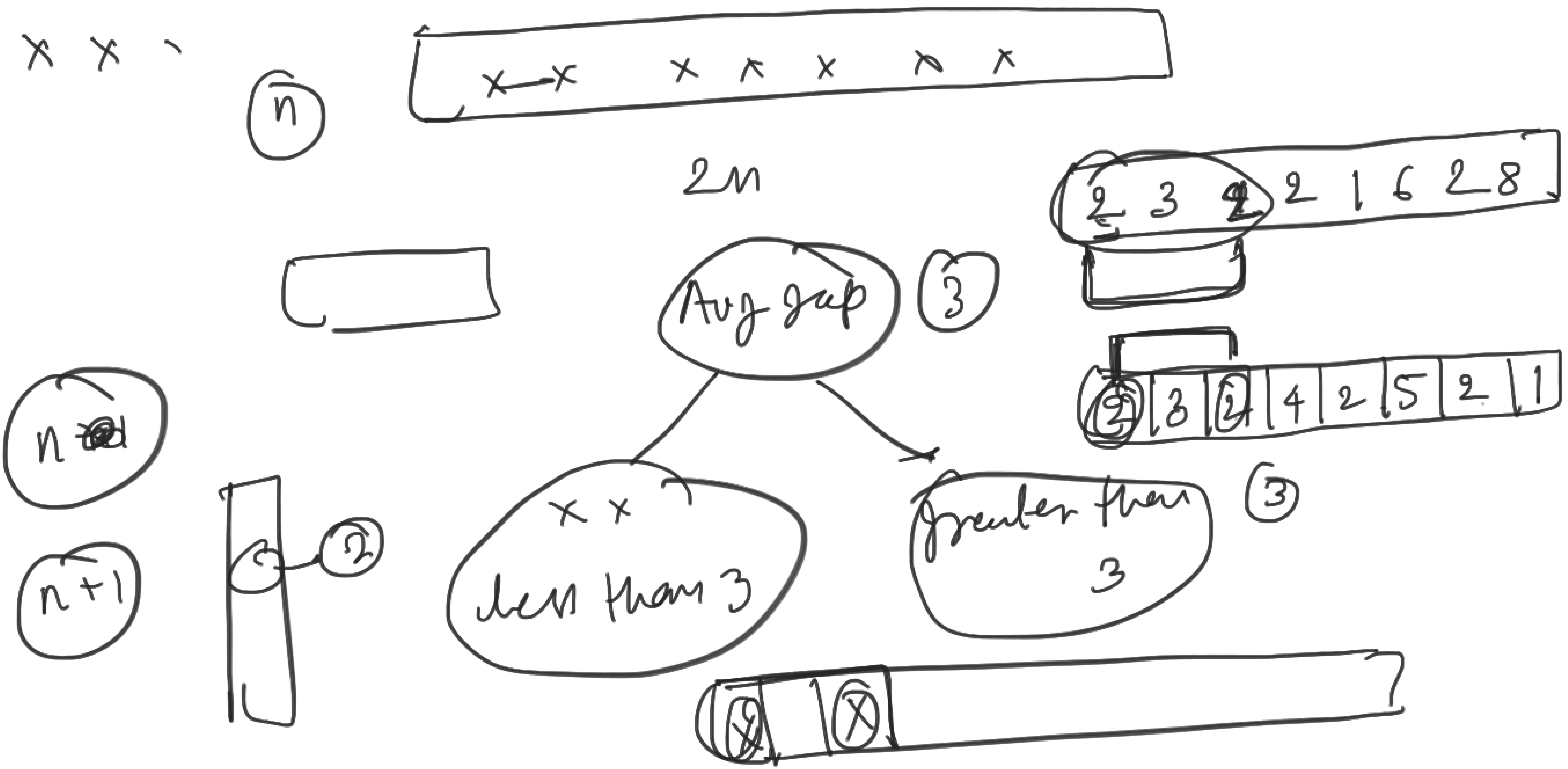
less than 3

greater than 3

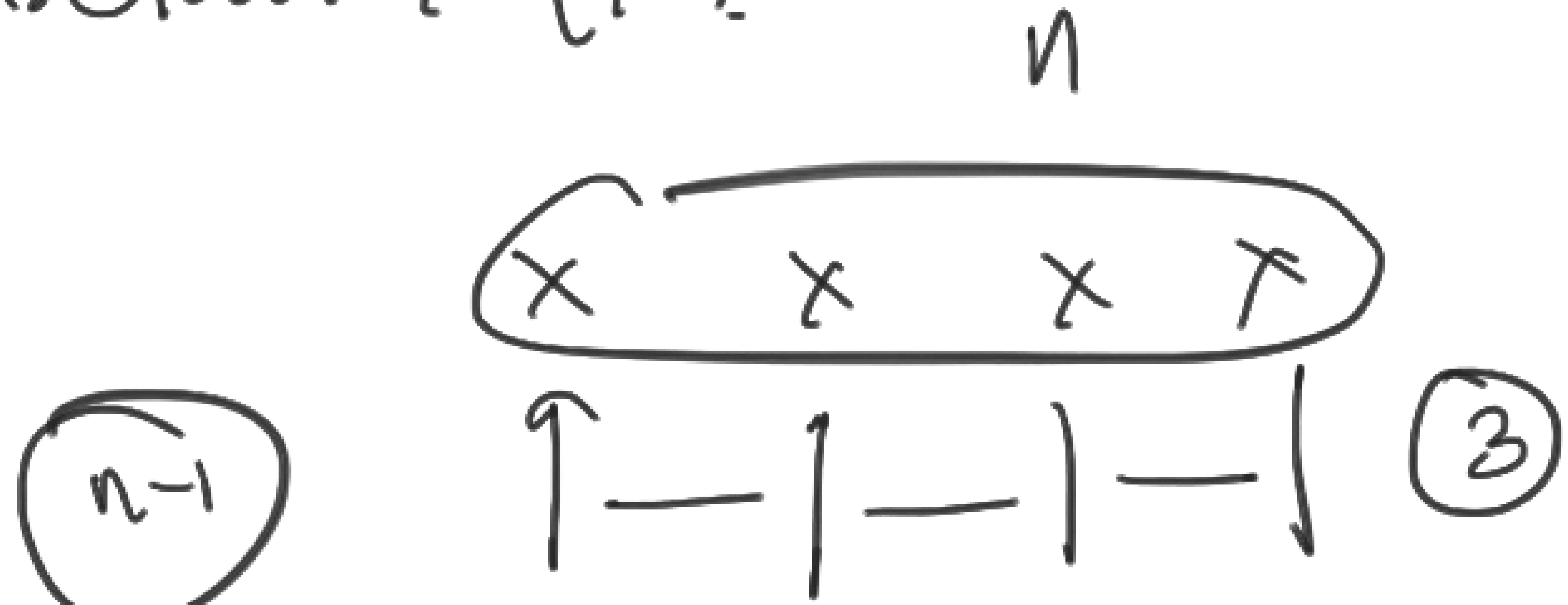
3



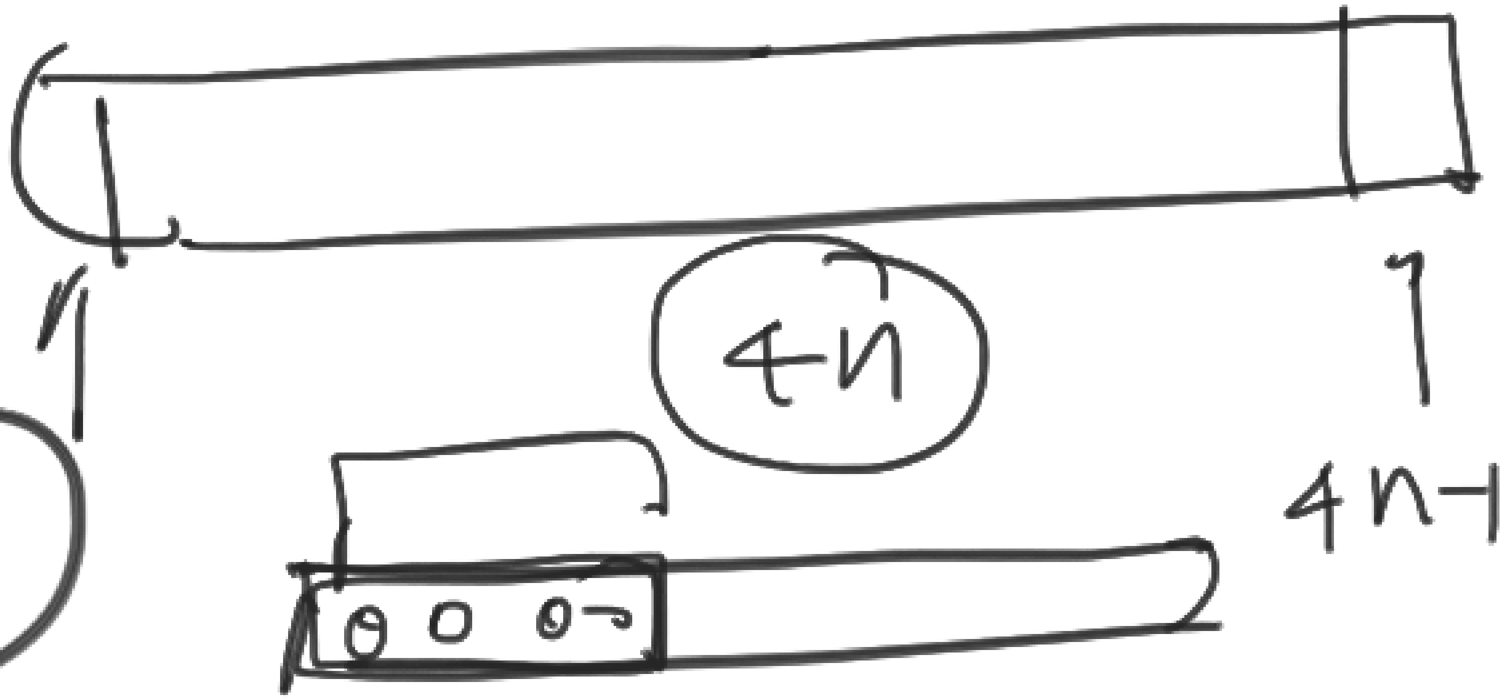


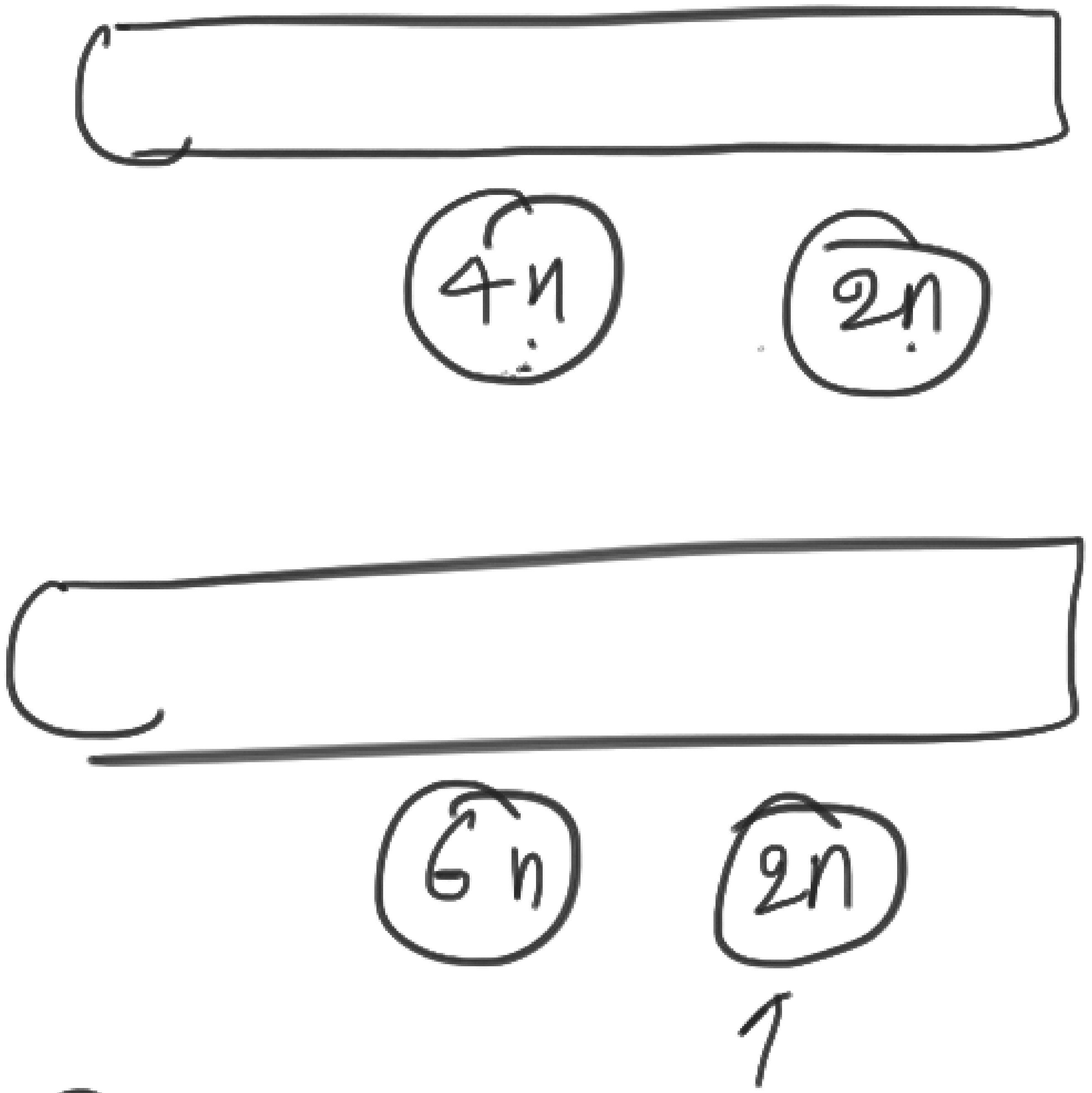

n
2n
2 3 2 2 1 6 2 8
Avg gap
3
2 3 2 4 2 5 2 1
n+1
2
less than 3
greater than 3
3
3

```
for( gap = 1 to 3)        O(1)·O(n) = O(n)
{   for( i = 0  to n-gap-1)
    {
        if ( x[i] == x[i+gap])
            return x[i]
    }
}
```



n times





$$\text{Avg gap} = \frac{I_n - I_1}{n-1} = 7$$

4

$$= \frac{4n-1}{n-1} \Rightarrow n=2 \rightarrow 10$$

4

$$\frac{4 \times 2 - 1}{2-1} = 7$$

$$\frac{4 - 1/n}{1 - 1/n}$$

$$\text{Avg gap} = \frac{I_{2n} - I_1}{2n-1} = \frac{6n-1-0}{2n-1}$$

$$= \frac{6n-1}{2n-1}$$

$n=2$

$$\frac{12-1}{4-1} \Rightarrow 11/3$$

$n=1$

$$\frac{6\times1-1}{2\times1-1}$$

$\Rightarrow 5$

2n size array

more than n times

n+1, n+2 — —

n+1 time

Boyer-Moore voting algo

majority element

n = 10

2 → 5

3 → 2

4 → 2

2	3	2	3	2	3	4	2	2	2

O(n)

O(1)

n → more than n/2 times

kadane → O(n) O(1)

Concepts

Problem-solving

→ Analysis of Quicksort (Avg)

→ Hash Table

Direct address table

Chaining

Counting sort

Propr

Hash funct

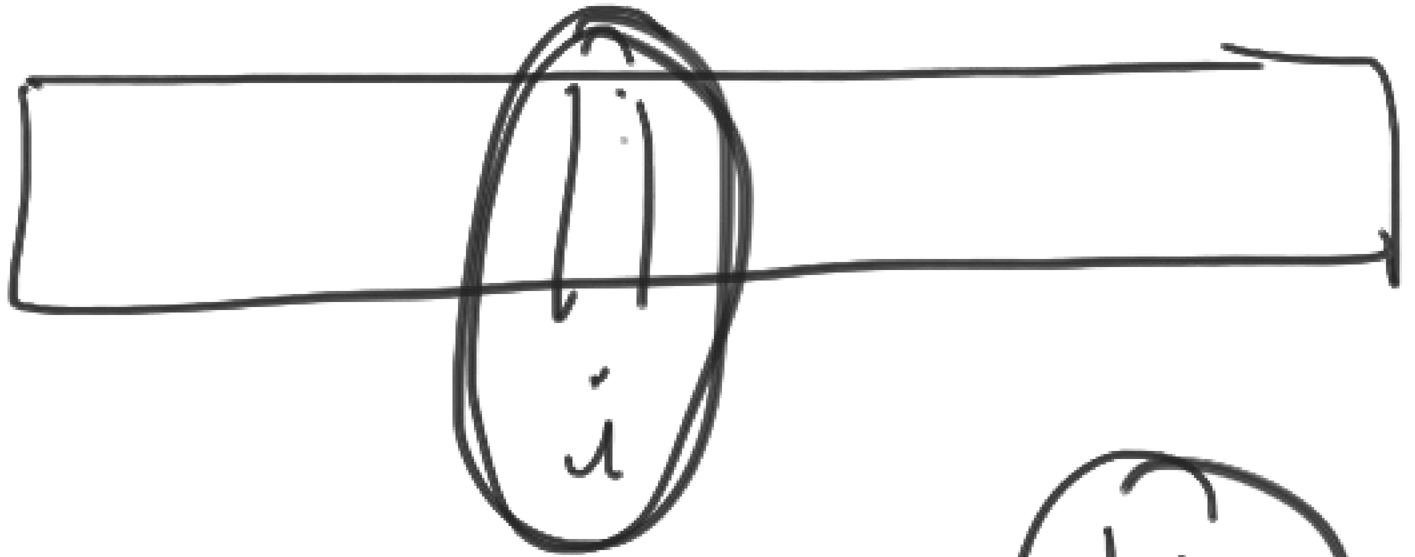
Open addressing

Bucket sort

→ Comparison of sorting algorith

n $\xrightarrow{O(n)}$ $2n$

randomised algo



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

$\frac{1}{2} \times \left(\frac{1}{2}\right)$

$k \Rightarrow \left(\frac{1}{2}\right)^k$

$k = \log n$

99.9%

$k \Rightarrow \left(\frac{1}{2}\right) \times \left(\frac{1}{2}\right) \text{ --- } k \Rightarrow \left(\frac{1}{2}\right)^k$

$\log n \times O(n) \Rightarrow O(n \log n)$

$\left(\frac{1}{2}\right)^{\log_2 n} = 1/n$

0.000...

$n \to \infty$

3 $\Rightarrow \left(\frac{1}{2}\right)^3 = \frac{1}{8} \Rightarrow$ 12.8 %

of 90 %

$3 \times 0 (n)$

4

$\left(\frac{1}{2}\right)^4 = \frac{1}{16} \Rightarrow$ 6 %