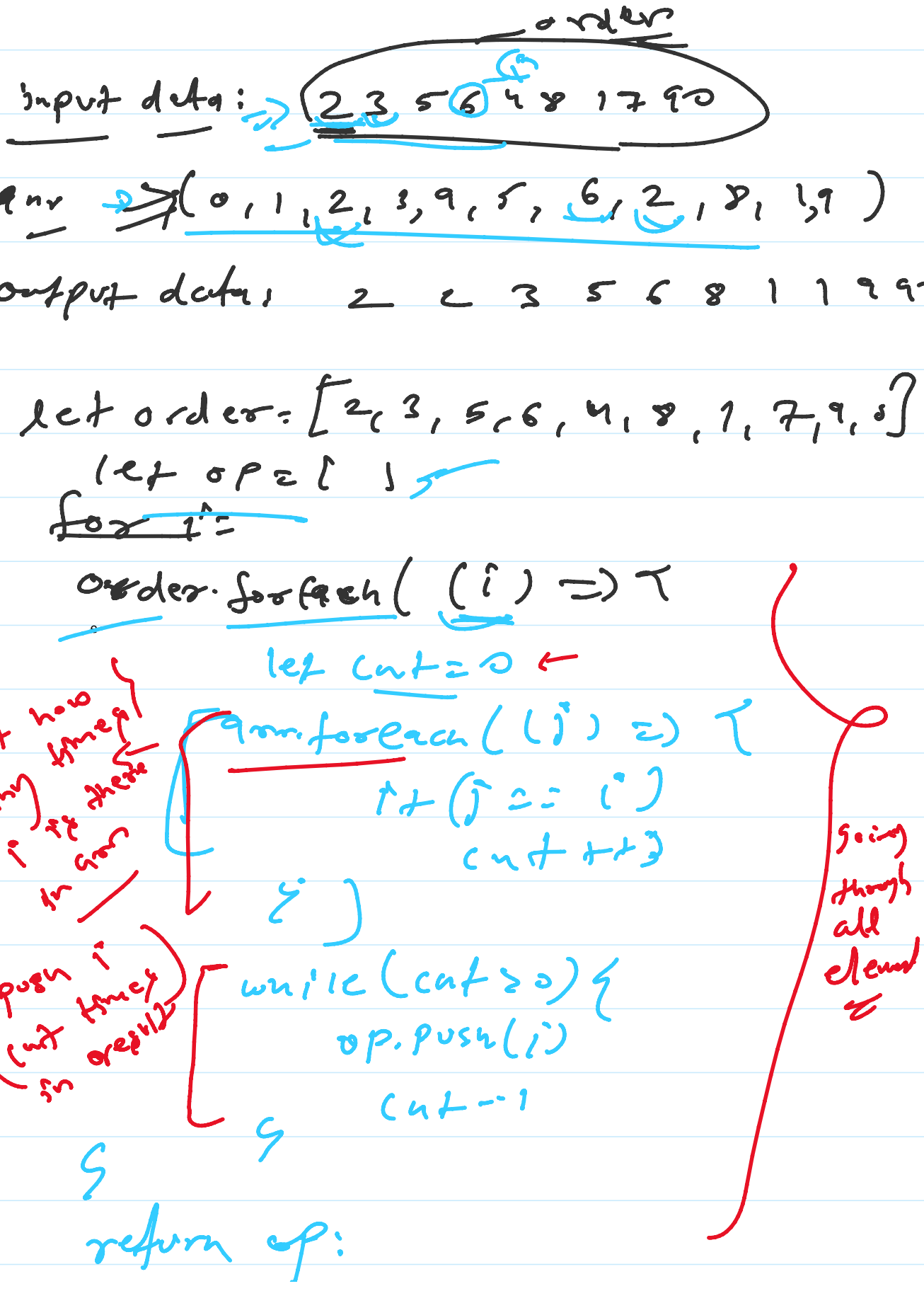
13/09/2022

13 September 2022 19:01



return of:

A → 1 ltr → K } X ltr
B → 1 ltr → L

a = [a1, a2, a3, a4 — an]

input data

A → X ltr, 1 ltr = K km
B → Y Ltr, 1 Ltr = L km
a = [ ] - values

(K, L, X, Y)

output data: no. of halts crossed by bike for each bikes

soln

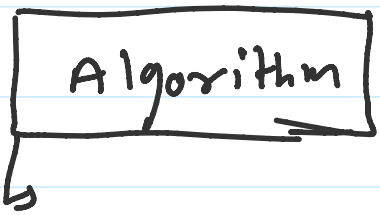
X K K = K n, km

[a1, a2, a3 - - - an]

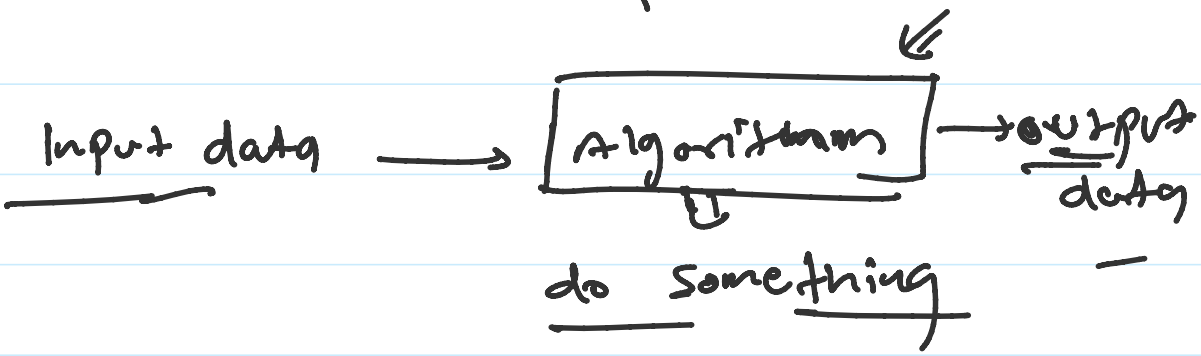
a.foreach((i) => {
if(xn - i > 0)
cnt++

```
        if(kn-1 != 0)
            cnt++
            kn = kn-1;
    }
    return cnt;
```

Time complexity:



{ a formalized way to } solve problem.



input data: list a

output data: $\sum_{i=0}^{n-1} a[i]$

```
Sum = 0
for i = 0 ---- n-1 :
    sum = sum + i;
```

```
for i = 0 ---- n-1 .
    sum += i :
print(sum) :
```

1) Is it correct ?
2) what is time complexity ?
3) Can we do better ?

Time Complexity :

→ time taken by your Program to Processor solve problem ?

in physics 1 sec, 2 sec, 1 milisec

1sec

? i5 9th generation → my laptop proc
i3 10th generation → reality

2sec

So what we can do now ? How to measure time complexity ?

We measure time complexity in terms of "no. of operations"

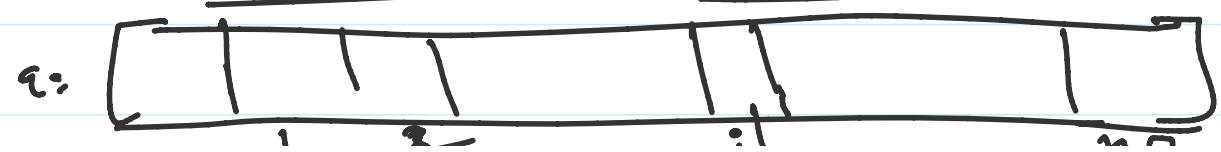
Computational model

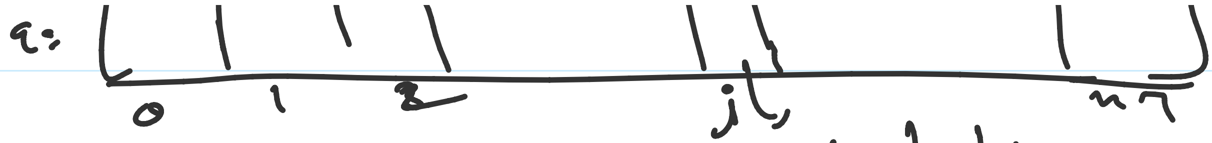
↳ A mathematical model which kind of simulate our computer processor.

- ↳ RAM - machine / model
- ↳ pointer - machine / model

RAM - machine / model :

(RANdom access memory)





constant time

a[i] -

↳ what are the opretions?

set of of oprations:

↳ (+, -, *, /, %) →

↳ (declaring a variable)

↳ (Assigning a value to variable)

↳ access any element of array (a[i])

↳ changing value of memory at location

↳ i   a[i] = n

Time complexity : "no. of opretions"

↳

2   sum = 0   for(i=0 ; i< n ; i++)

for i = 0 ---- n-1 : 2

5n [n ② sum += i ;   2 sum = (sum + i)

1 ← print (sum) :

Time complexity = 5n + 3

= 10n + 5

3n + 2

operation depends upon size of input

≥ 1000

T.C. = 5n + ~~3~~

if n ≥ 1000 (we can ignore)

T.C = 5n

is not depends on Algorithm

depends upon your programming skills

T.C = n

O(n)

notation / Asymptotic

Big-O | O( ) → upper bound / Atmost

omega | Ω( ) → lower bound / atleast

theta – Θ( ) → fixed / fixed

(omega-Ω( ) | ... at least
theta - Θ( ) | → fixed | fixed

Big-O (O(1)) ||

$f = O(g) \Rightarrow f \leq g$

$\forall$ c, n_0

$n \geq n_0$ $f \leq cg + c$

f = 4 + n 0.5
g = 4 + n^2 0 + 2.0

n = 0 0.5 g c

≥ 1 $n \leq 1$
$n^2 \geq n$

$f = O(g)$

O - upper bound / not more than g

⇒ How much time your Algo will take

→ it won't take more than n sec

it won't take more than n sec

Θ → (f = O(g) · f ≤ g

T = 5n+3 = O(n)

Ω → Algo will take at least

1 sec

t = 5n+3 = Ω(n)

when O & Ω are same → that

$\Theta(n)$

n(4n+2)

for i=0 --- n-1 — 2

4n

for j=0 --- n-1 — 2

sum += a[i][j] — 4

TC = O(n^2)

(4n+2) n

$4n^2 + 2n$

$4n^2 + 2n$

$O(n^2)$