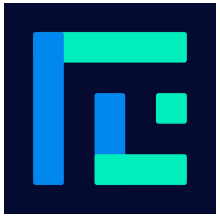


Advanced Binary Search - 1

on Answer

- Priyansh Agarwal

Binary Search Revision

Time Complexity Discussion

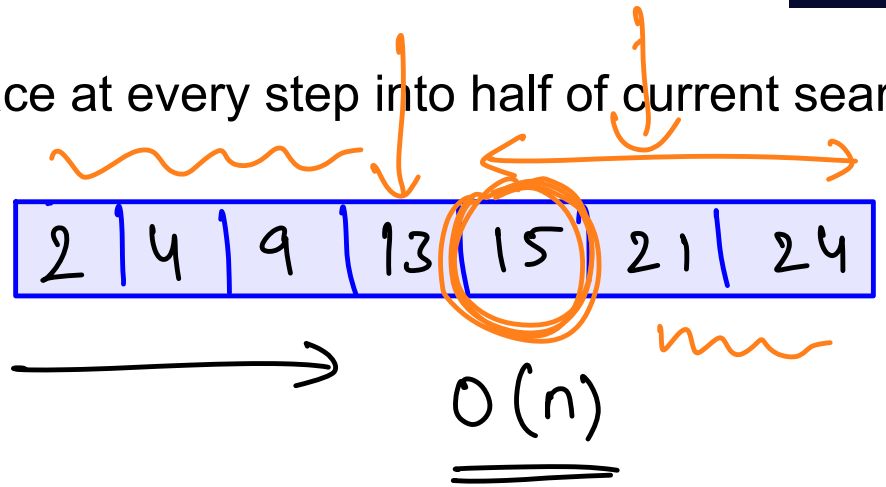
We are reducing our search space at every step into half of current search space

Recurrence:

$$T(n) = T(n / 2) + 1$$

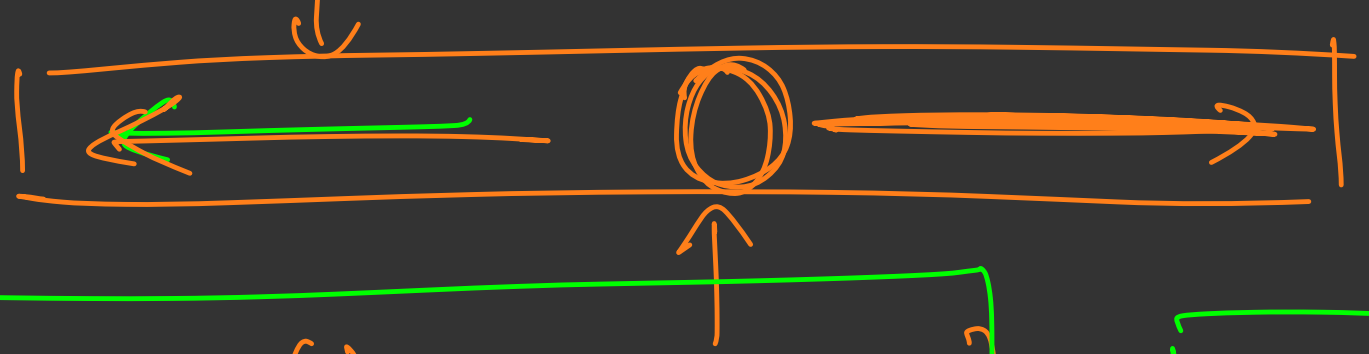
$$T(1) = 1$$

Time Complexity: O(logn)



$$T(n) = T(n/2) + 1$$

I am given an array of size (n), time to search for an element in this array of size (n) = $T(n)$



$T(n) = O(1) + T(n/2)$, $T(1) = O(1)$

$$T(n) = O(\log n)$$

$$T(n) = T(n/2) + 1$$

$$= T(n/4) + 1 + 1$$

$$= T(n/8) + 1 + 1 + 1$$

$$= \boxed{T(n/2^k) + k}$$

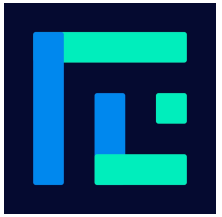
$$\boxed{T(1) = 1} \Rightarrow \boxed{T(n/2^k) = 1}$$

if $2^k = n$

$$k = \log n$$

$$\boxed{T(n) = T(n/2^k) + k} = \boxed{1 + \log(n)} = O(\log n)$$

Binary Search Revision



Requirement for using Binary Search

Monotonicity

- f(x) ≥ f(y) iff x > y (increasing monotonic)
- f(x) ≤ f(y) iff x > y (decreasing monotonic)

Monotonicity

$$f(x) = x^2 + 2x + 1$$

$x = 5$

$$f(x) = 5^2 + 10 + 1 = 36$$

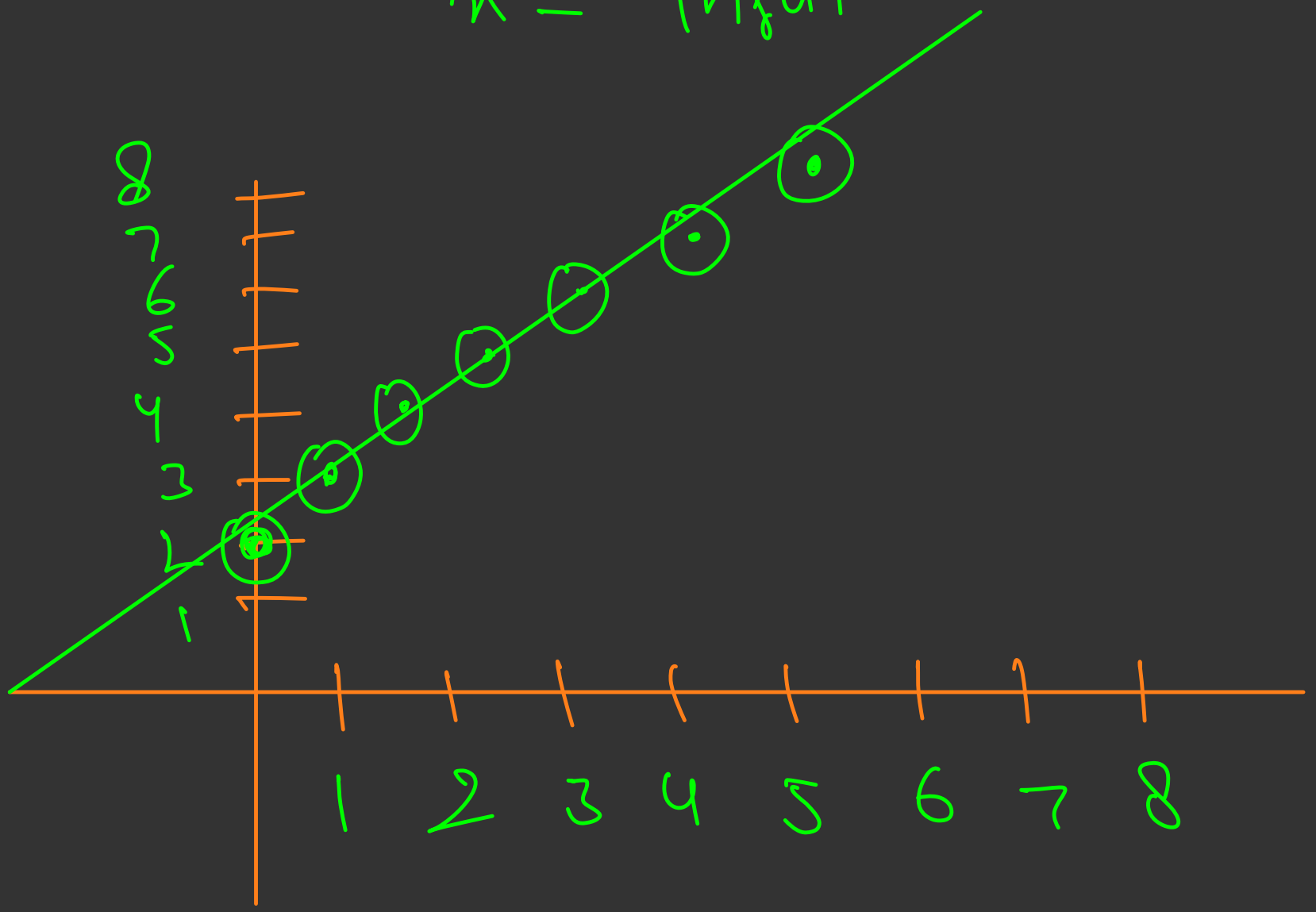
Input $\longrightarrow$ function (expression) $\longrightarrow$ output

$y = f(x)$

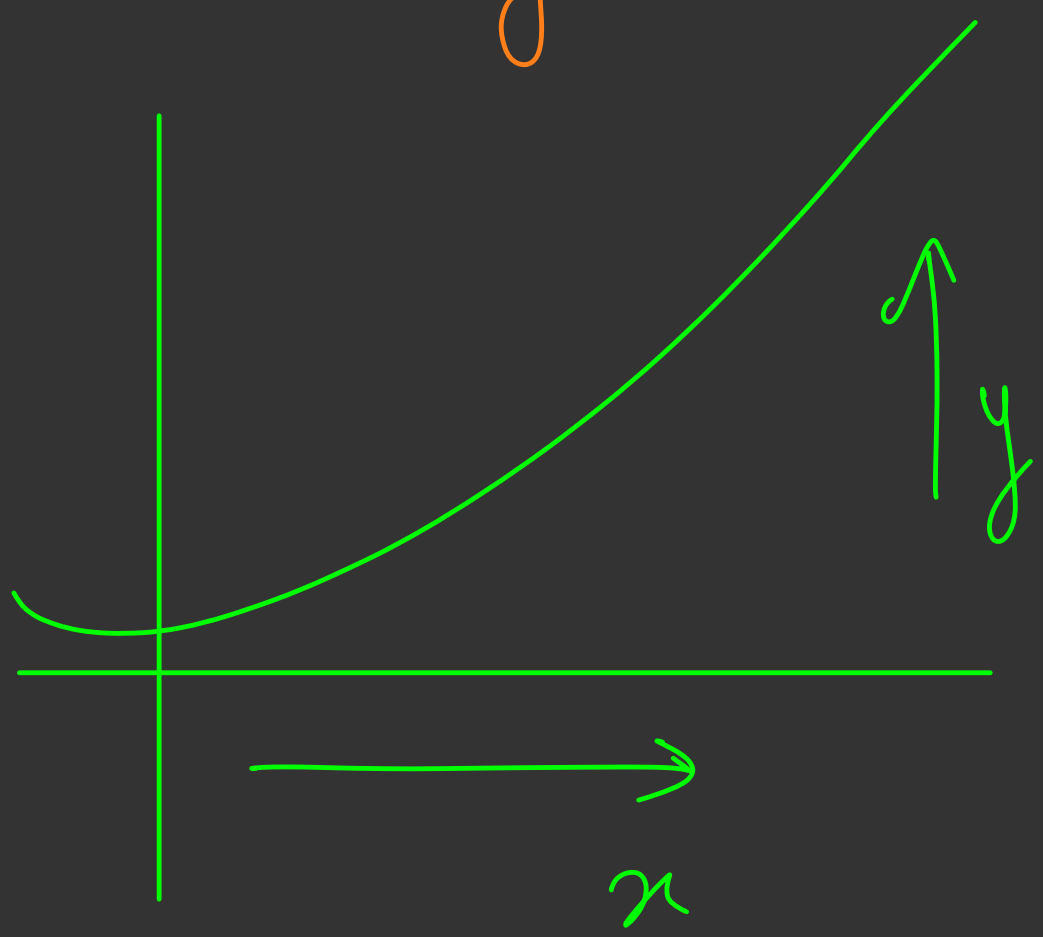
$f(x) = x + 2$

y = output

x = input



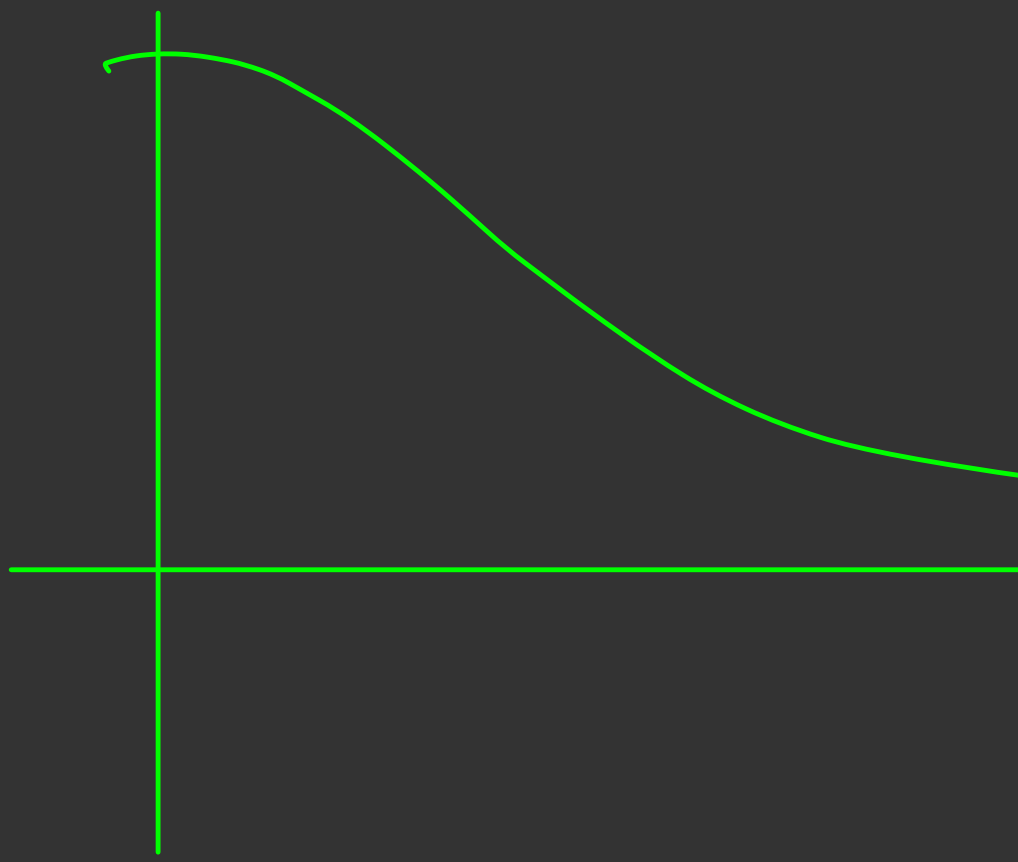
increasingly



$x \uparrow$

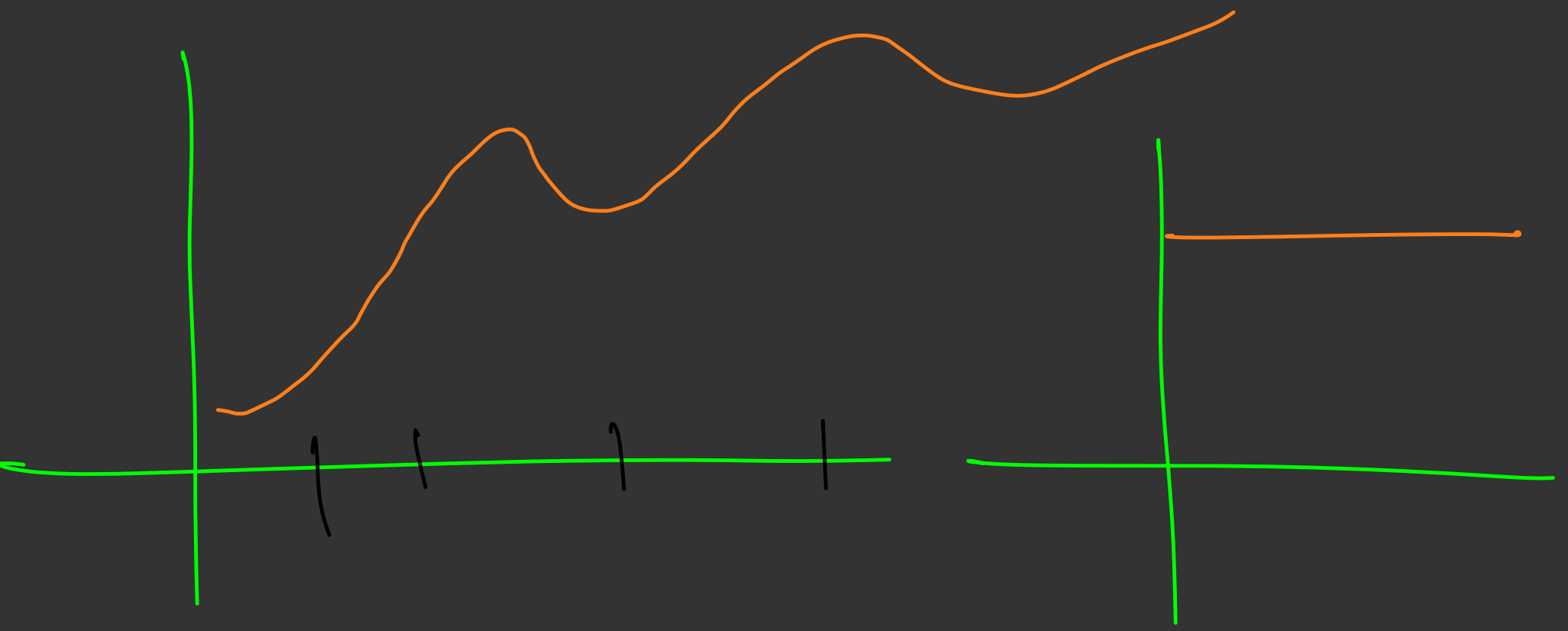
$y \uparrow$ $\quad y = f(x)$

decreasingly

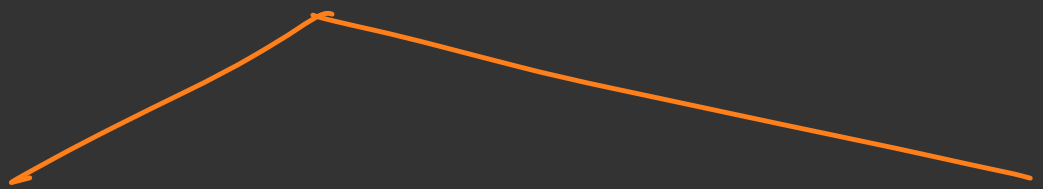


$x \uparrow$

$y \downarrow$



$\forall$ input x_1, x_2



$x_1 > x_2 \Rightarrow y_1 > y_2$

$x_1 > x_2 \Rightarrow y_2 > y_1$

① Strictly increasing

$x_1 > x_2$

$\Downarrow$

$y_1 > y_2$

② Strictly decreasing

$x_1 > x_2$

$\Downarrow$

$y_2 > y_1$

③ non decreasing

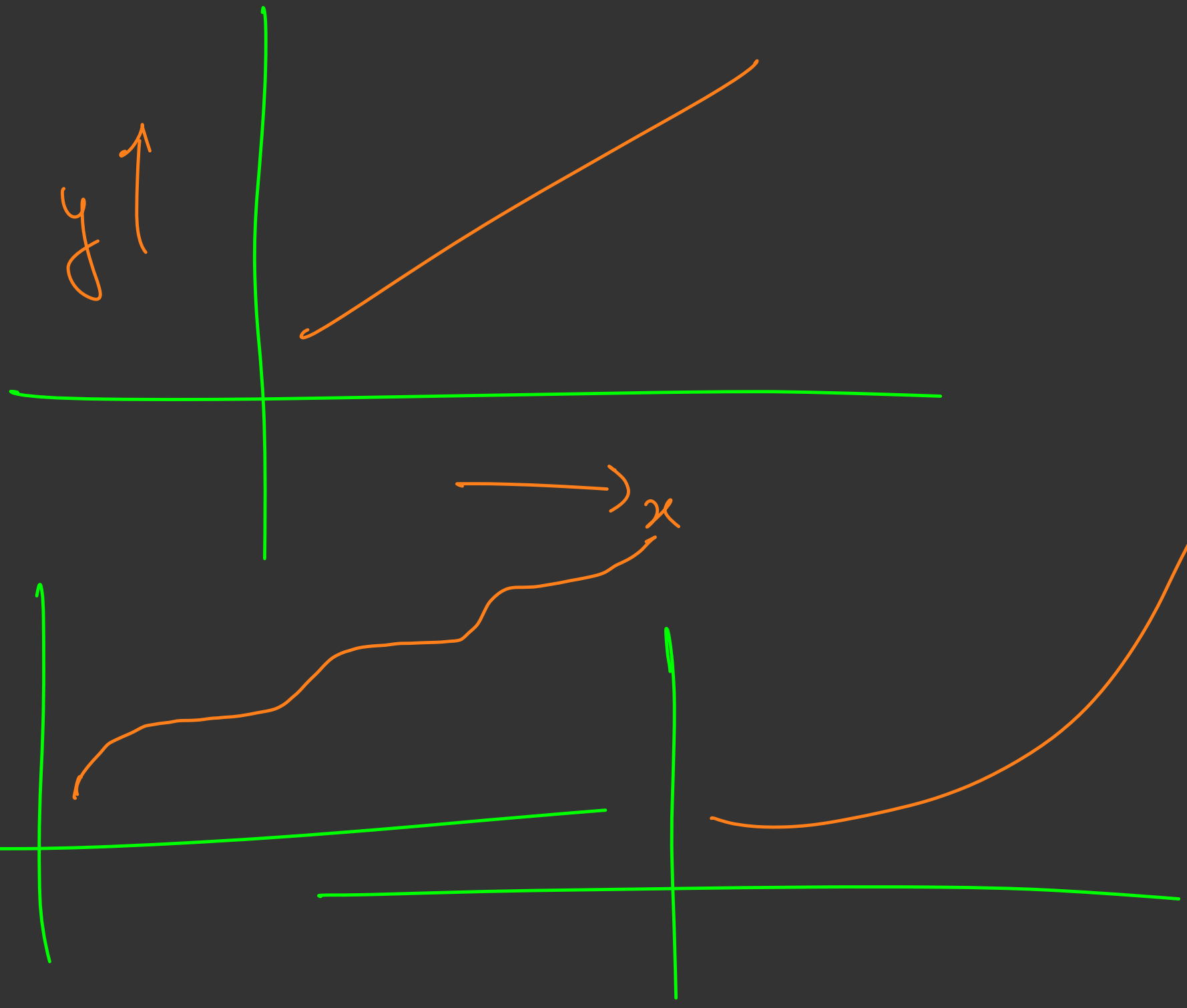
$x_1 > x_2$

$y_1 \geq y_2$

④ non increasing

$x_1 > x_2$

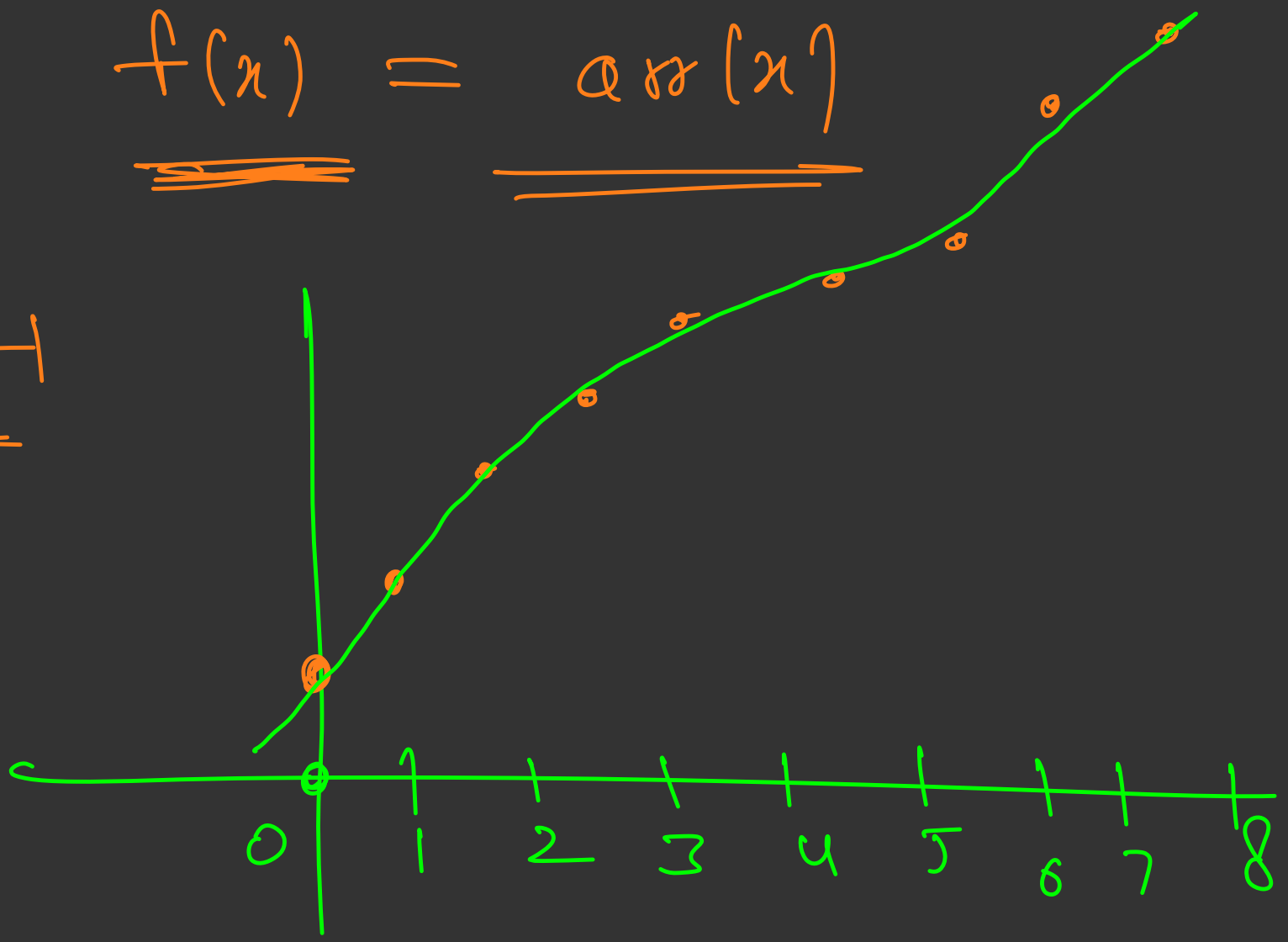
$y_1 \leq y_2$


y
x

2	3	5	7	9	11	12	20	25
0	1	2	3	4	5	6	7	8

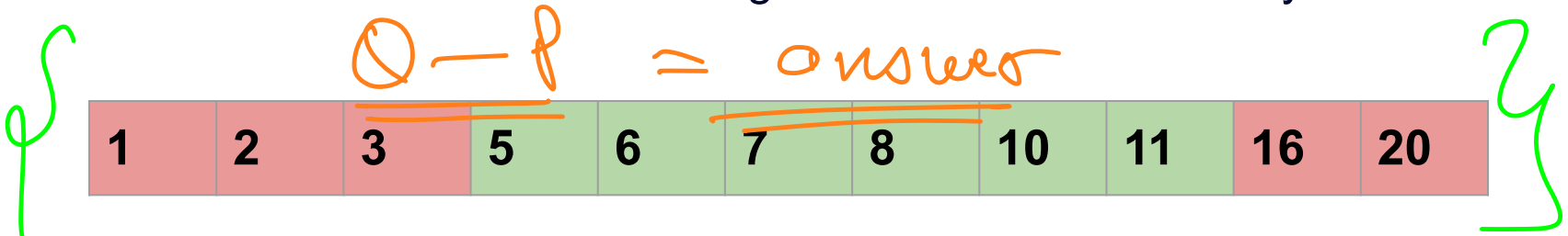
$x \longrightarrow f(x) = arr[x]$

$0 \longrightarrow n-1$



Binary Search Revision

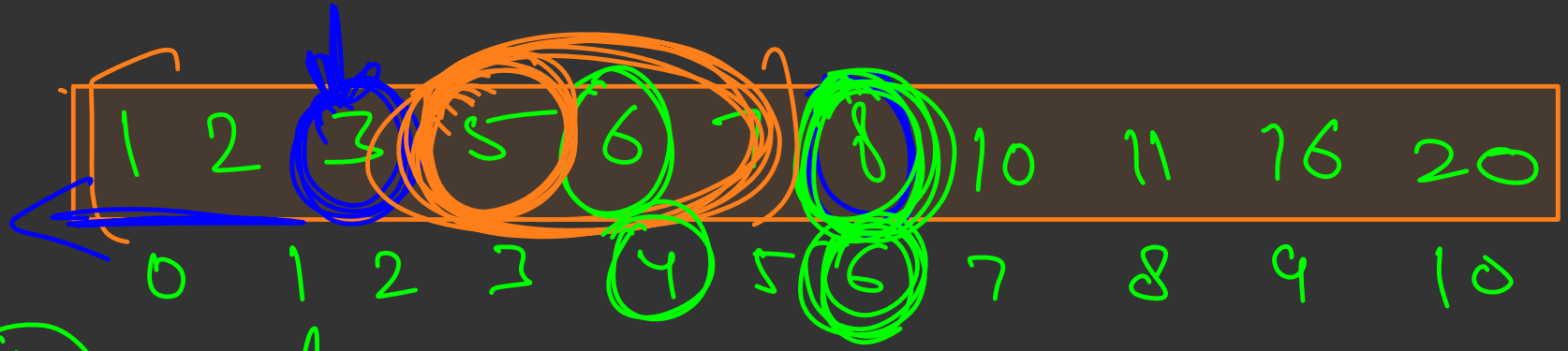
Find Number of elements in the range of l to r in a sorted array



Ex: Number of Elements in the range 4 to 12 are 6

first index for which arr[x] >= L → P

first index for which arr[x] > R → Q



(L) first index (x) in the array such that arr(x) ≥ L

L = 4

6 ≥ 4

8 ≥ 4

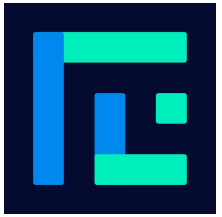
ans = 6

ans = min (ans, x)

= min (6, 4) = 4

ans = min (4, 3) = 3

Predicate Functions



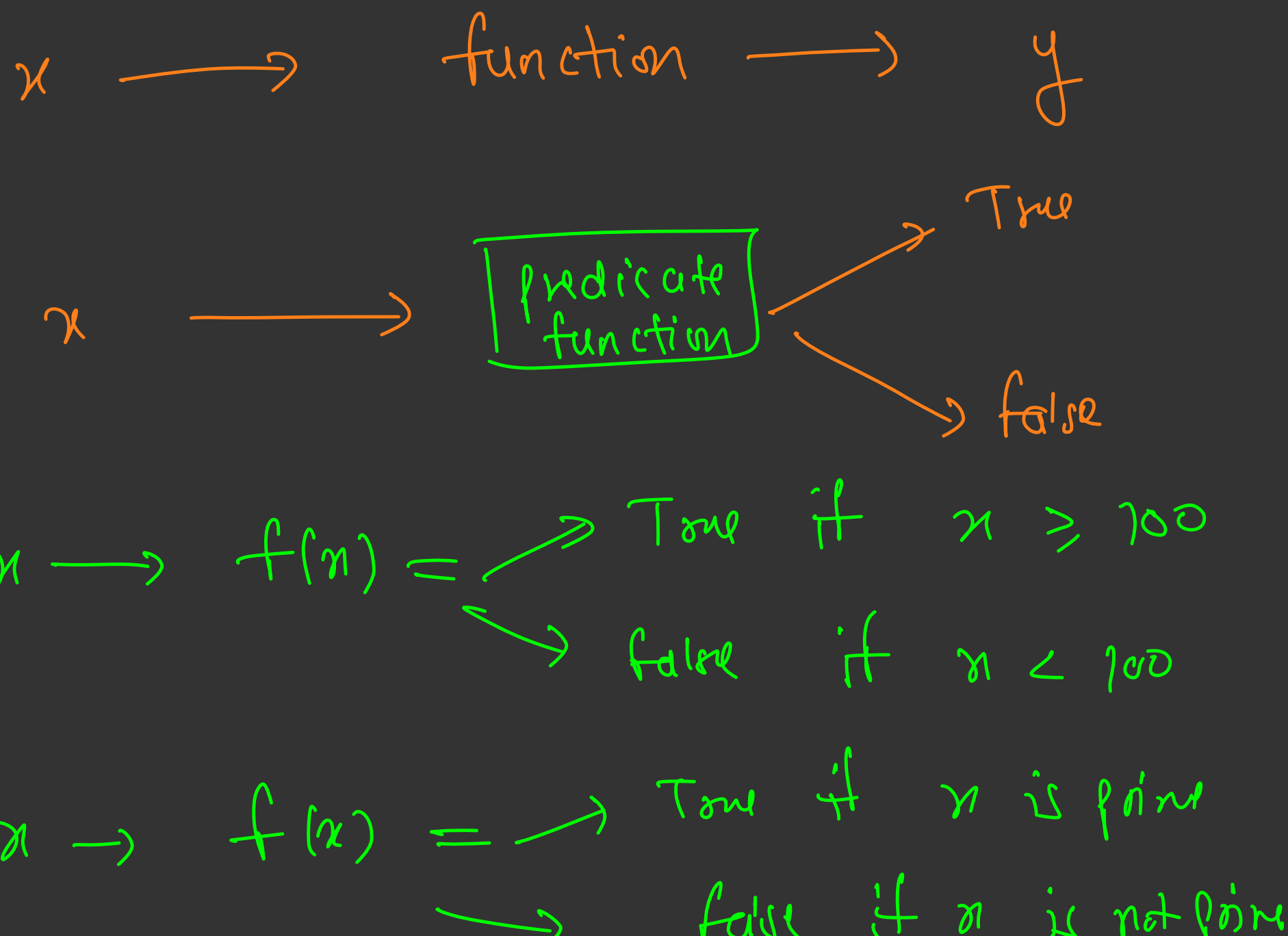
Functions that return a single TRUE or False for every input. You use predicate functions to check if your input meets some condition or not.

Examples:

- F(x) = True if x > 10 otherwise False
- F(x) = True if x is a character otherwise False
- F(x) = True if x^2 is an odd number otherwise False

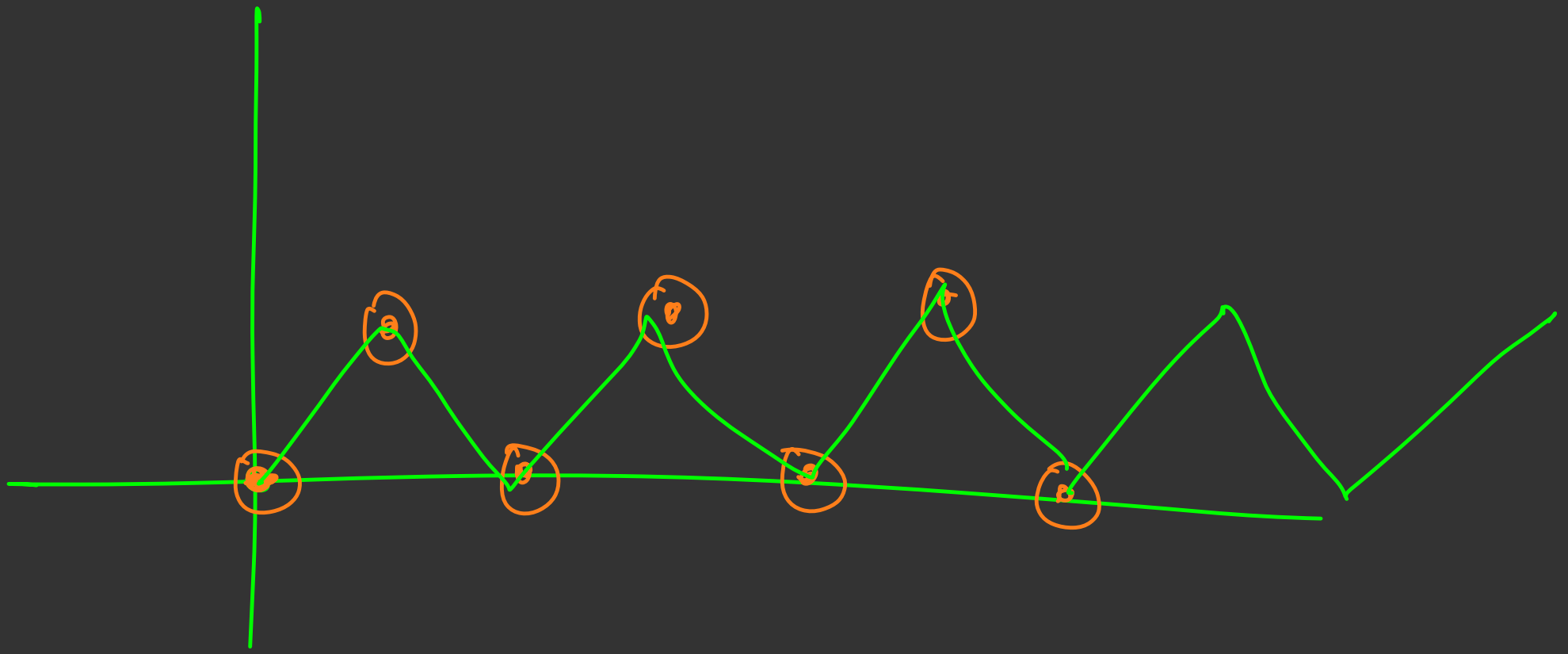
True = 1

False = 0


x
function
y
x
predicate function
True
false
x
f(x)
True if x ≥ 100
false if x < 100
x
f(x)
True if x is prime
false if x is not prime

$f(x)$ = True if x^2 is odd

false o/w



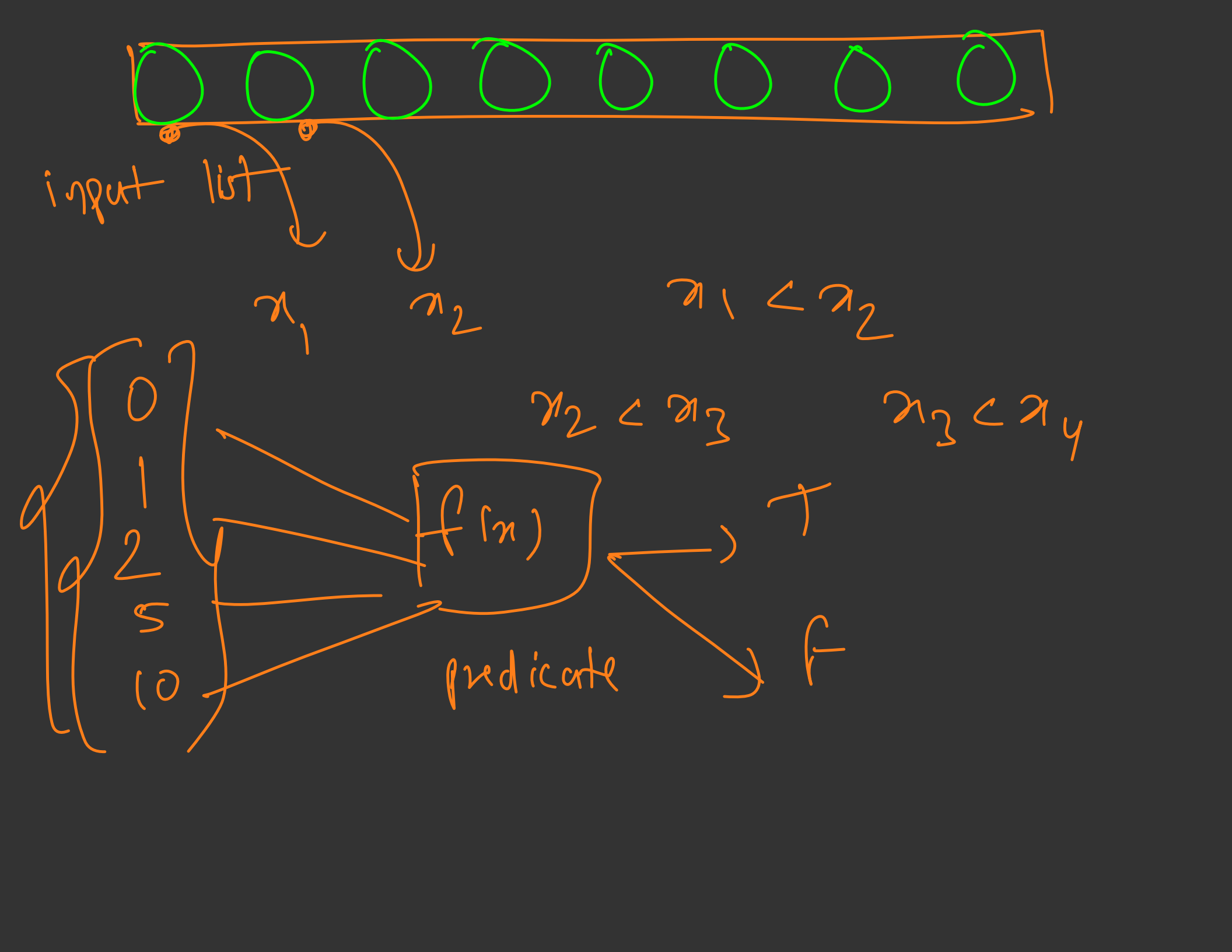
Binary Searching on Answer

- Consider a predicate P defined over some ordered set S (the search space).
- The search space consists of candidate answers to the problem.
- We use the predicate to verify if a candidate answer is legal or not.

Example: We have the set of numbers {1, 2, 3, 4, 5, 6}.

Our predicate function could be following:

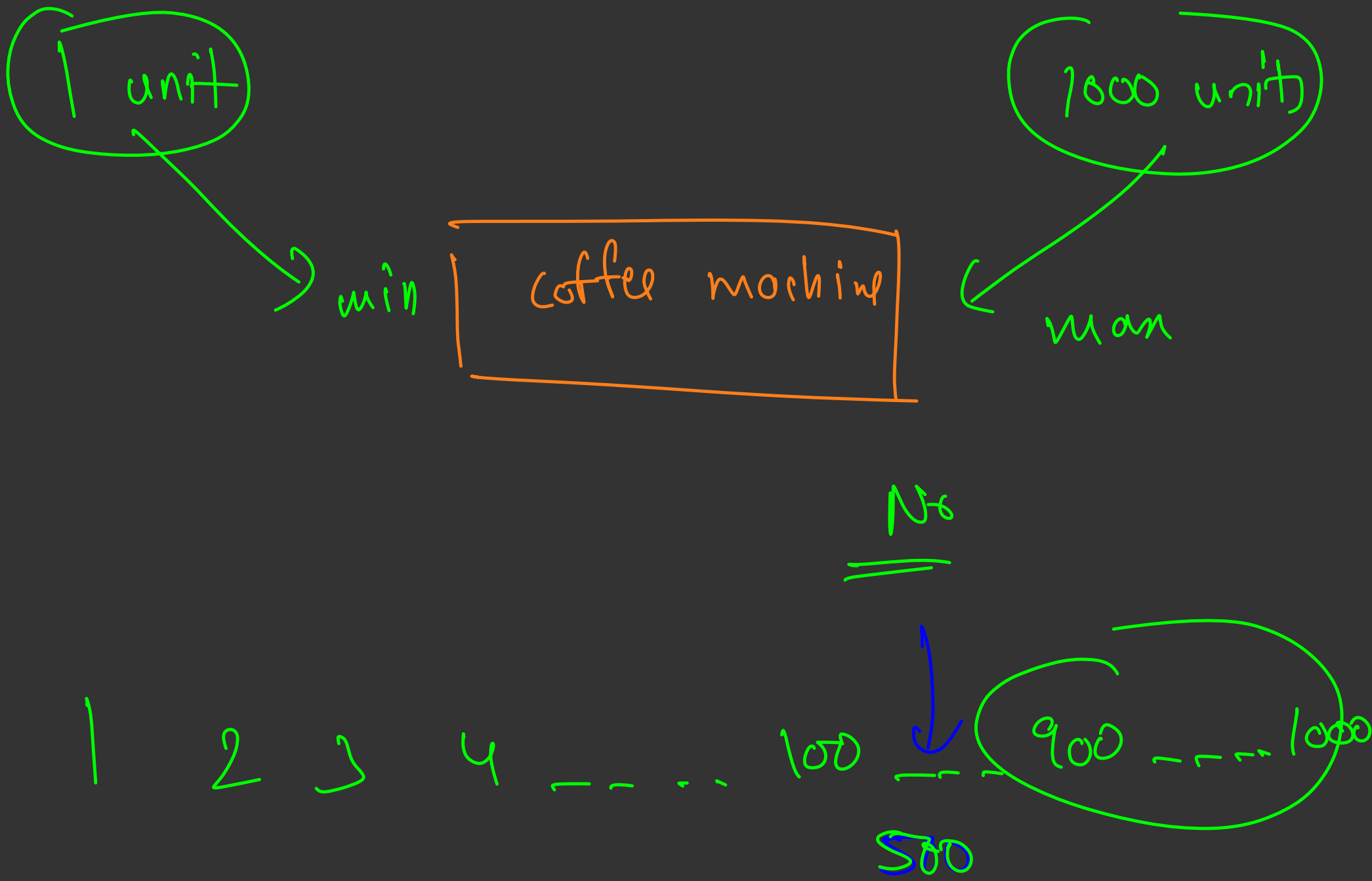
- Return TRUE if the number is less than 3 and FALSE otherwise

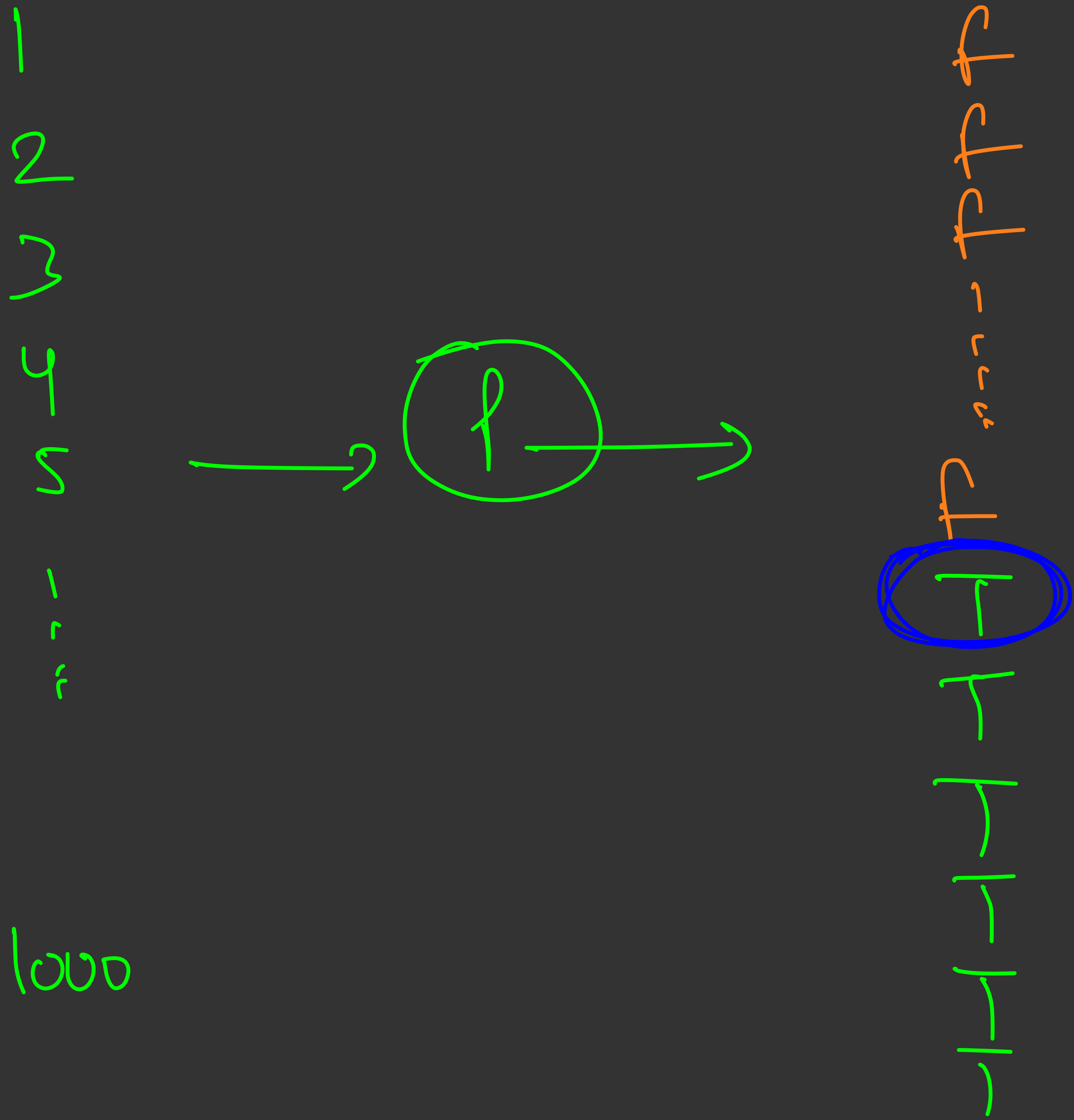

input list
x_1
x_2
$x_1 < x_2$
$x_2 < x_3$
$x_3 < x_4$
0
1
2
5
10
f(x)
T
F
predicate

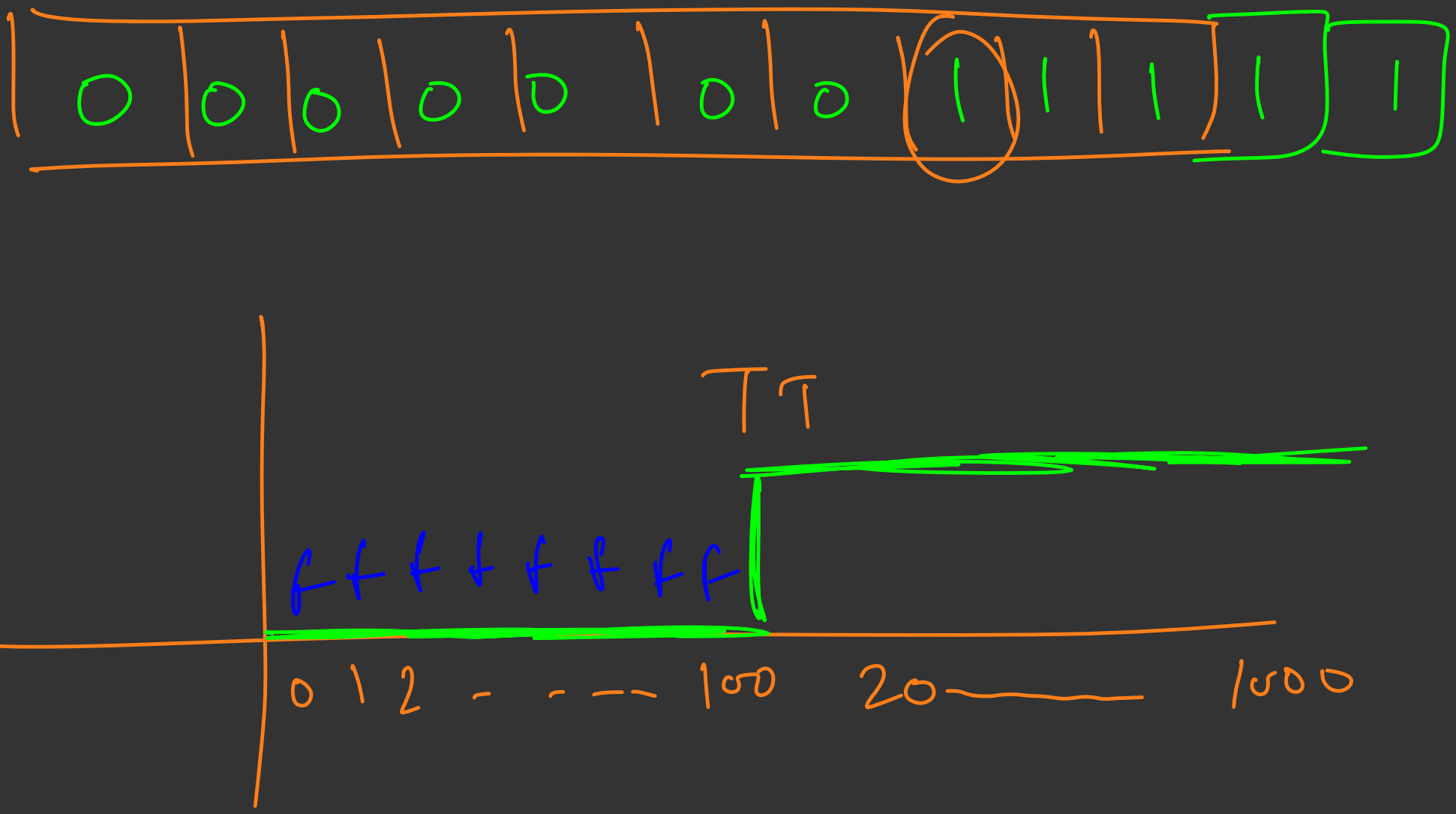
Codeforces problem

$f(n) \to T$ if the CM can run you with n power

Print the minimum power you need to run a step

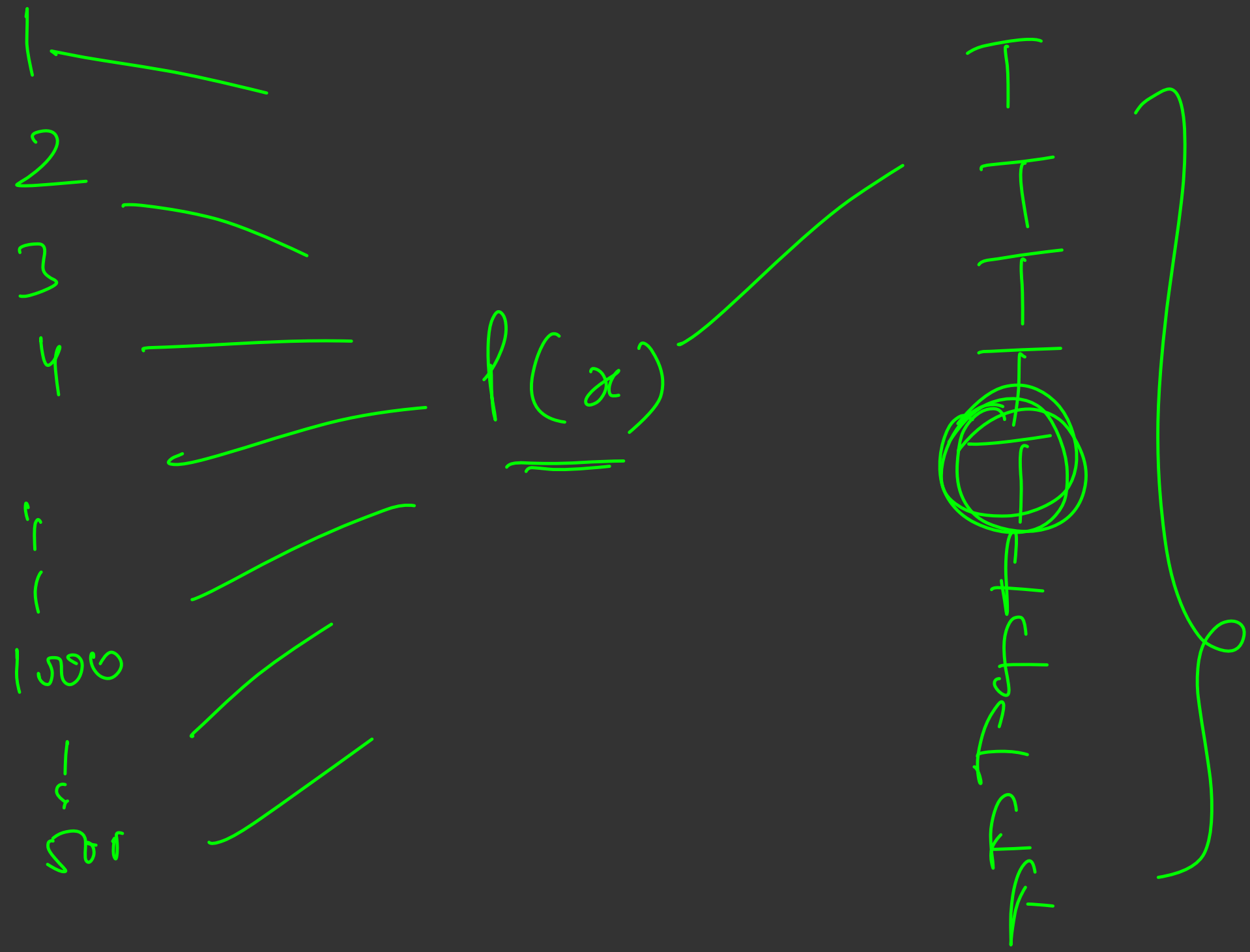

1 unit
1000 unit
min
coffee machine
max
No
1 2 3 4 ... 100 ... 900 ... 1000
500


1
2
3
4
5
1000
f
F
F
F
F
T
T
T
T
T

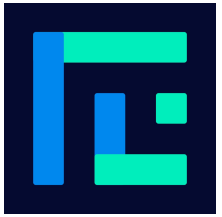

T T
f f f f f f f f
0 1 2 - - -- 100 20 1000

find out the maximum power upto
which the coffee machine will
not Breah

1 unit 1000 unit 3000 p


1
2
3
4
f(x)
T
T
T
T
F
F
F
F

Monotonic Predicate Functions



1. TTTTTTTTTFFFFFFFF

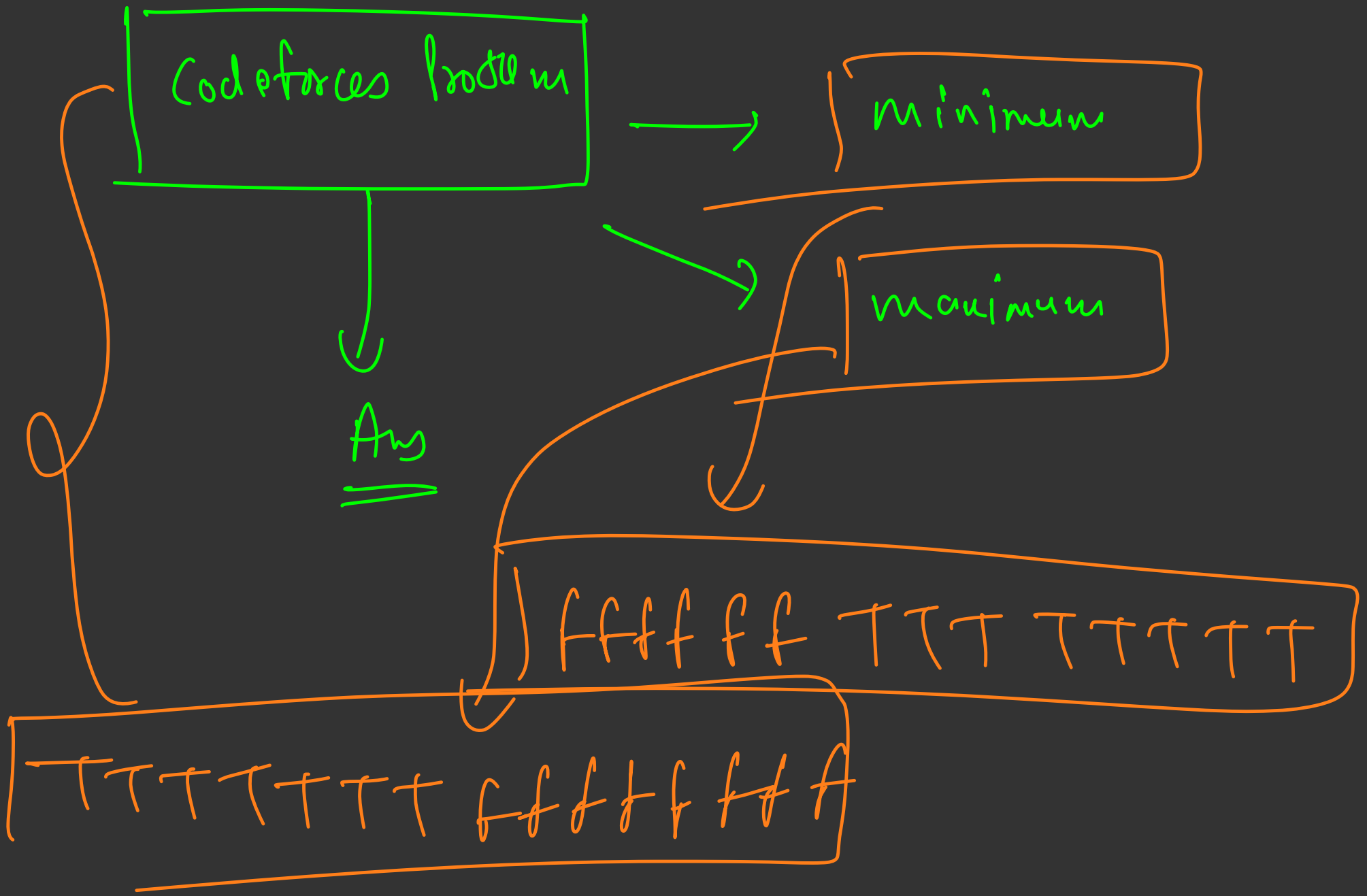
T	T	T	T	F	F	F	F	F

2. FFFFFFFFTTTTTTTTT

F	F	F	F	F	F	T	T	T

ans = 4 → ans = 7

① first T
② last T
③ first f
④ last f


Codeforces Problem
Minimum
Maximum
Any
fffff ff TTT TTTTT
TTTTTTTT fffffff

FairWorkload Problem

Given an array of workloads, split it among **K** workers, such that the maximum work that any worker has to do is minimised (can't change order of workloads).

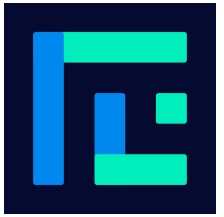
Eg. [10, 20, 30, 40, 50, 60, 70, 80, 90]

- Solution : 10 20 30 40 50 | 60 70 | 80 90

  First worker - 150, Second worker - 130, Third worker - 170

Is it possible to partition workload in a way that the highest workload of any worker is less than 170?

Sqrt(X)



Given a number X ($1 <= X <= 10^{18}$), find the biggest number Y such that $Y^2 <= X$

Eg:

X = 10000, Y = 100

X = 1024, Y = 32

X = 26, Y = 5