Time Complexity II

→ Why are iterations important?

→ How can we compare iterations

→ Asymptomatic Analysis → Big O
→ Ω, Θ

→ Big O

→ Problem with Big O

→ Observations / Limits

→ Space Complexity

→ Questions

Headings

Base

Definitions

Extra Stuff

Questions

Use Case

→ Sort An array

Jaya	Saneth
Jaya Sort	Dilrgent Sort
10s	15s
i9 11700k	Duel Core
→ 10s	6s
Python	C++
C++ → 4s	6s
Kashmir	Delhi

N = 10 thousand

$= 10^4$

Hardware ✓

Software

External

ITERATIONS

→

$100000 < 10^7 \Rightarrow$ Jeya Sort is more optimised than DSort

10^5 10^7

N

10^5 10^{10} 10^8 → Dsort will be better

10^6 10^9 10^{11}

$N^2+N < N^{10}$

$N=1$ $2 < 1$

$N \geq 2$ ≥ 6 $\geq 2^{10}$ $\xrightarrow{N \geq 2}$ Jeya Sort > DSort

Jeya Sort → N^2 N^2

N → input size

DSort → $N^2 - 7N$ → N : Input size

$N^2 - 7N$ $\quad N \geq 0$

Iterations ≥ Iterations

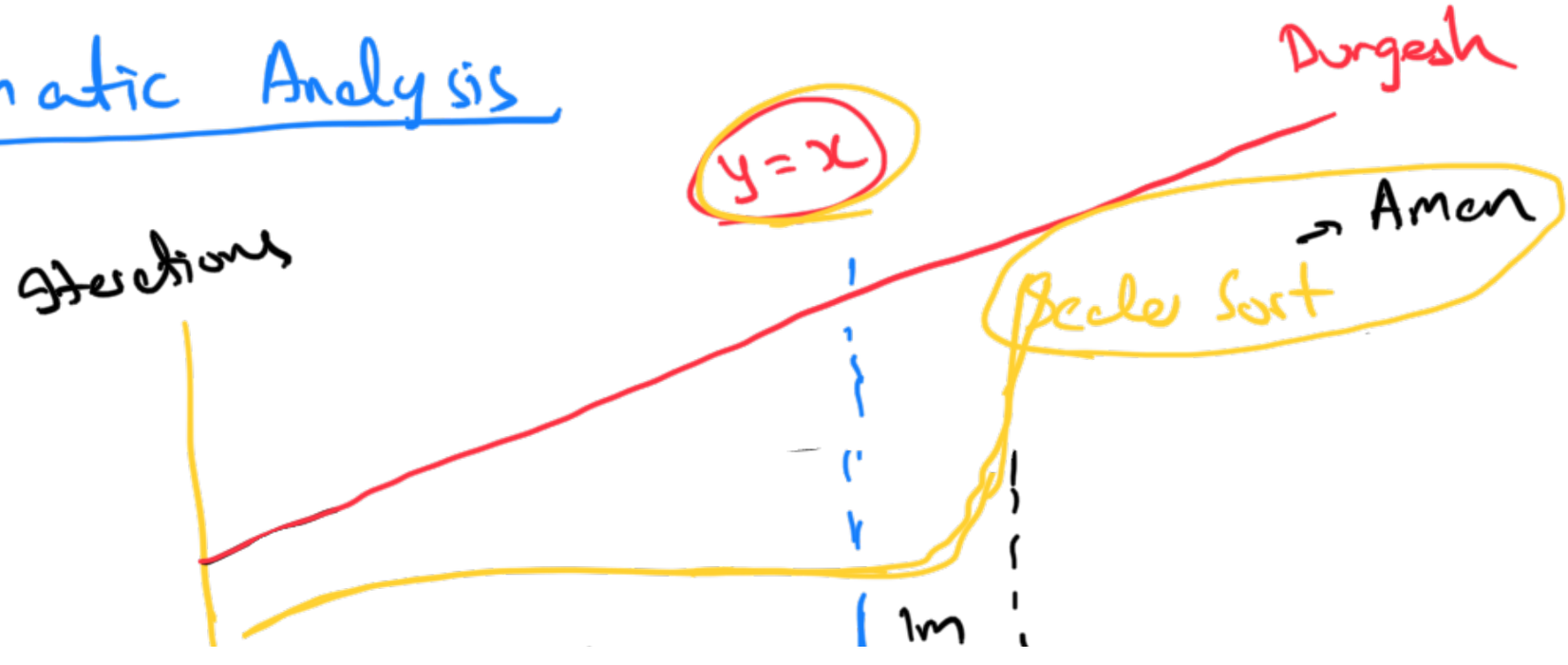
JayaSort < DSort (Optimization)

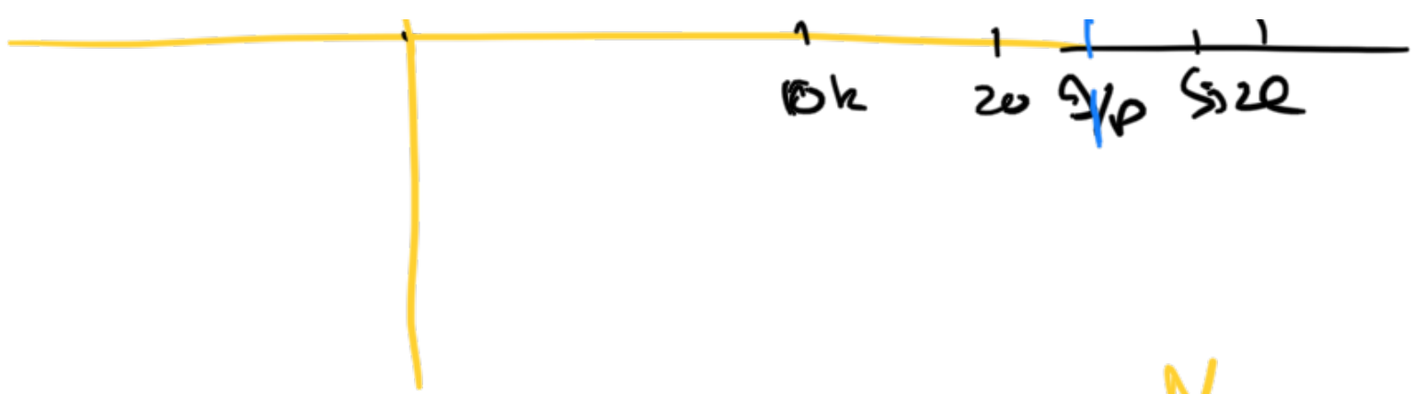
Chandra

↳ C-Sort → 100 log (x)

or 100 log (N) ~

Asymptomatic Analysis





N

Q → I have an algo (like → y=x) → 10s

→ AnkitSort ⟶ f(N) → iteration (log N

N^2

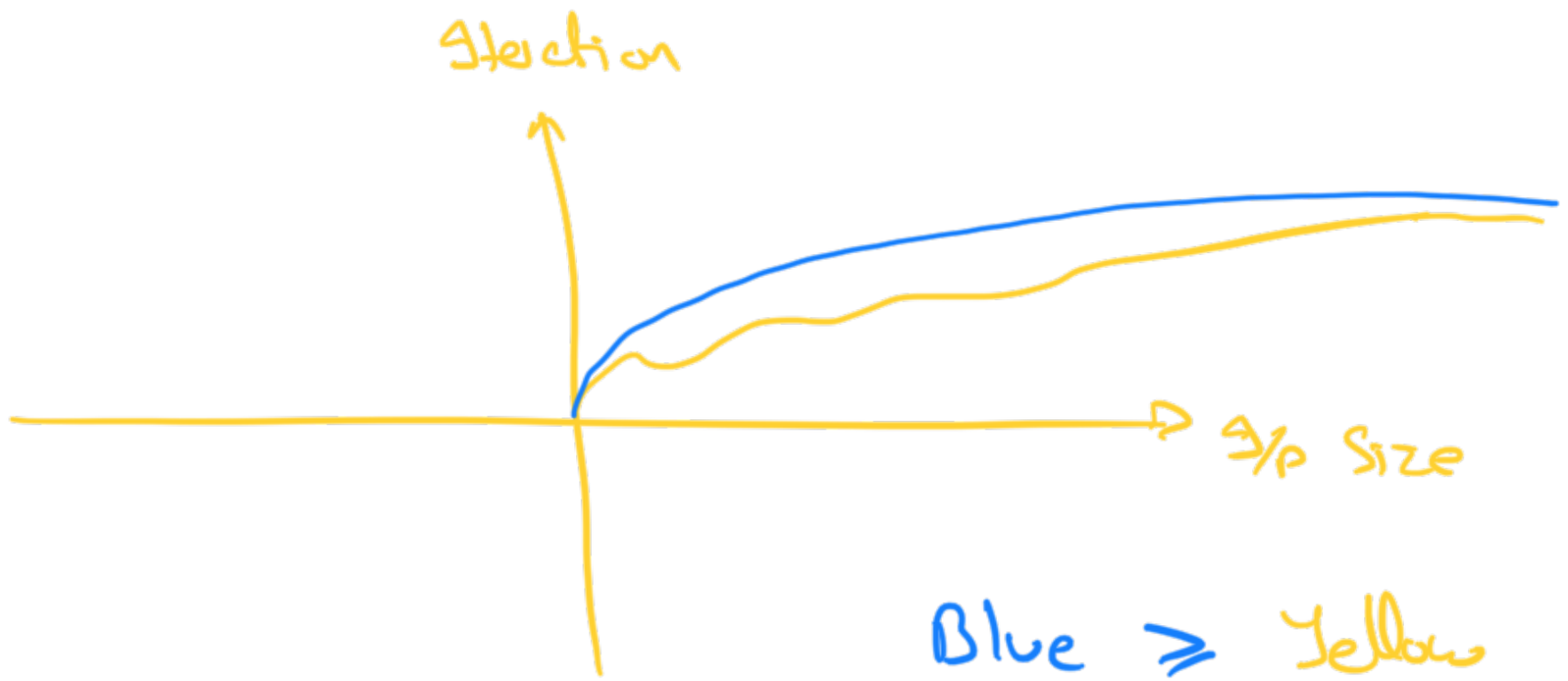
N^3)

f(N) ≤ N

↳ Ankit Sort ≤ 10s

Blue

Yellow



Blue ≥ Yellow

=> Blue is upper bound to yellow

Linear Search → O(N)

Bubble Sorts → O(N^2)

Devesh

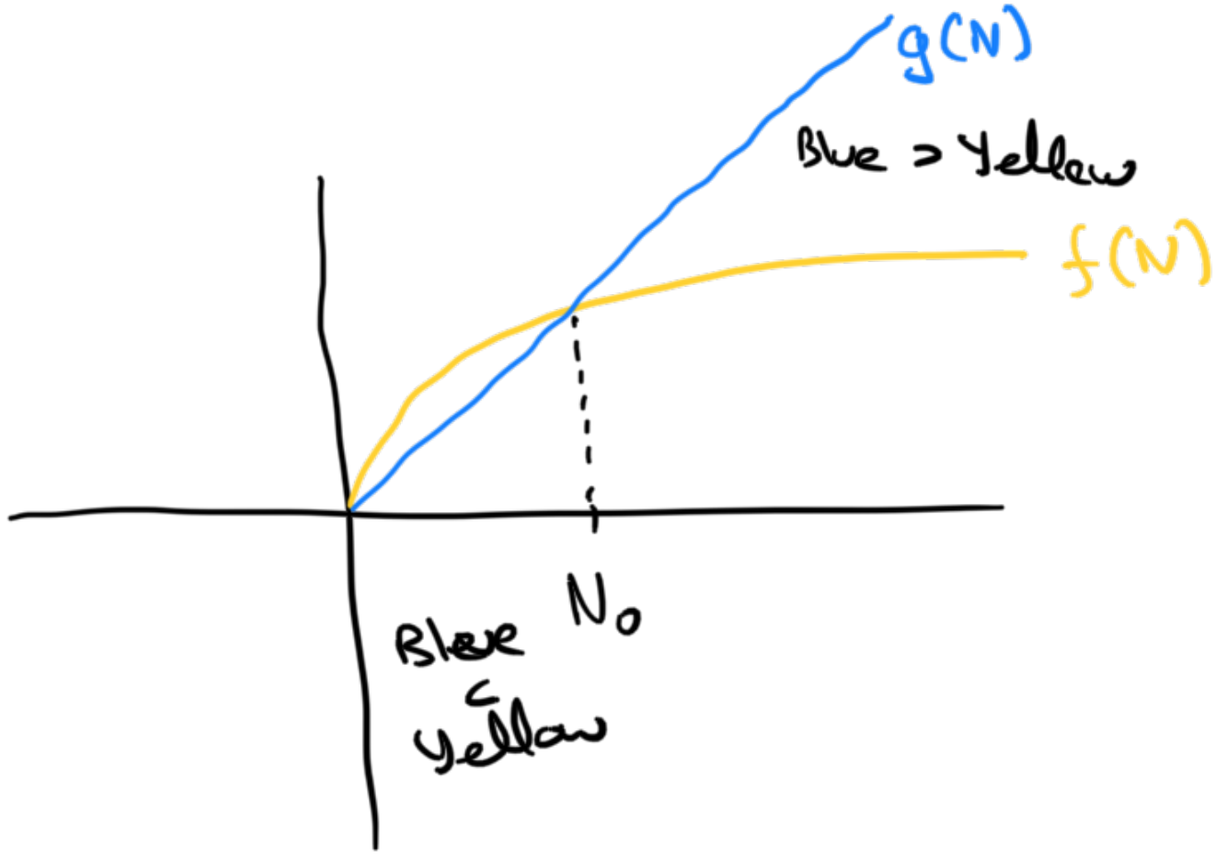
Mahesh → Bleech

Big O, Big Ω, Big Θ

Formal

Given a function f(N) where N ≥ 0 (positive), it is

$c \times g(N) \geq f(N) \quad , \forall\ N \geq N_0$



Break 10:18

Blue upper bounds

Yellow

N_0

$g(N) := N$ ✓

$f(N) = \log(N) + 10$ ✓

$N \geq \log(N) + 10$

$\forall\ N \geq 12$

$N_0 = 12$

$c = 1$

Examples

$N \geq 0$

1. $f(N) = N^2 + N$. $g(N)$

$f(N)$ → $O(g(N))$

$O(N^2)$

$O(N^3)$

$f(N) = N^2 + N$

$N^2 \leq N^2 \quad \forall N$

$N \leq N^2 \quad \forall N \geq 1$

$N^2 \leq N^3 \quad N \geq 1$

$N \leq N^3 \quad N \geq 1$

$\Rightarrow N^2 + N \leq N^3 + N^3$

$N^2 + N \leq N^2 + N^2 \quad \forall N \geq 1$

$N^2 + N \leq 2N^2 \quad \forall N \geq 1$

$N_0 = 1$

f(N) g(N)

$f(N) = N^2 + N$ → $O(N^2)$ ★ Tight Upper Bound

→ $O(N^3)$

$O(N^{1.9})$ ∝

$x^2 \to \infty$ → $O(N^2)$

$N^2 + N$

$x^2 + x \to \infty$ → $O(N^{10})$

$N^{1.9} \geq N^2$ ∝

as $x \to \infty$

Algo1 → $N^2 + N$ $O(N^2)$

Algo2 → N^2 $O(N^2)$ ✓

$N^2 + N$

$10^2 + 10$ $N = 10$

$10^6 + 10^3$

10^{20} $+10^{10}$

10^{20}

$10^{90} + 10^{20}$

$N = 10^3$

$N = 10^{10}$

$N = 10^{20}$

2. $f(N) = N^2 + 5$

$O(N^2)$

3. $f(N) = N^3 + 8N^2 + 4\log(N)$

$O(N^3)$

4. $5N^3 + N$

$O(N^3)$

Way to Calculate

→ Pick highest Power ✪

→ Ignore constants

Issues with the Big O

1. No Flaw

Algo 1	Algo 2
N^2	$\log(N) + 10$
$O(N^2)$	$O(\log(N))$

$\longleftrightarrow$ → Green is better

For $N_0 = 3.2$

If our $N < 3.2$ → Red is better

2. Algo 1 Algo 2

→ $100N^2$ N^2

$O(N^2)$ ⟷ $O(N^2)$ Blue always better

Our Purpose

Time Complexity

↳ a way to approximate how much time will your code as a fx of Ip Size

Code forces / chefs hacc. —

Time constraint = 1s

≤≃ 10^7 iterations per second

```
for (int i=0; i<N; ++i)
  {
    assignment
  }
```

↳ O(N)

Is → codeforces

$N \leq 10^7$ ✓

else <u>TLE</u>

Yp Size

$N \sim 10^3 \Rightarrow$ O(N²)
O(N)
O(N log N)
O(log N)

$N \sim 10^6 \Rightarrow$ O(N)
O(N log N)

$N \sim 10^5 \Rightarrow$ O(N log N)

- $N \sim 10^9 \Rightarrow O(\log N)$

$N \sim 10 \Rightarrow O(2^N)$

$O(N!)$

1. Understand the question
2. Decide algo
3. Calculate OC
4. Check if TLE
5. Code

Space Complexity

→ Big O

```
void foo (int N) {

    int a;          // 4 bytes
    long b;         // 8 bytes          T => 4 + 8 = 12 bytes
    int arr[N];     // How many?  N elements
    int arr2[N][N]; // N x N elements
                    ↳ 4 bytes
                    ↳ 4N^2 bytes

}
```

$$Space = \underline{12 + 4N + 4N^2}$$

$$SC = O(N^2)$$

```
for (N) {
    for (int i=0; i<=N; ++i)
    {
        int a=0;
    }
}
```

T.C $\rightarrow$ $O(1)^3$

$O(N)^3$ ✗

T.C = $O(N)$

↳ Re declaring (a) n times

```
void sortArray (int arr[], int N) {
    for (int i=0; i<N; ++i)
        cout << arr[i];
```

int a = 2;

}

T.C = O(n)

S.C = O(1)

Best Case vs Worst Case

Linear Search

```
foo (      ){
for (int i = 0; i < N ; ++i)
    {
        if (arr[i] == ele)
            return true;
    }
return false
```

S.C = O(1)

T.C =

Best Case TC = O(1)

Worst Case TC = O(n)

}

{

3 2 1 6 7

ele = 3 → N=1

ele = 7 → N = 5 , N

Bubble Sort

↳ O(n²)

→ 1 2 3 4 5 6 Best → O(N)

→ 6 5 4 3 2 1 Worst → O(N²)

Avg →

N=1 N=2

1 2 21

1 3 31

x iteration

$x \sim \sqrt{n}$

$\frac{x(x+1)}{2} > n$

x^{th} term is $s = \frac{x(x+1)}{2}$

1st 2nd 3

$s \rightarrow 1\ 3\ 6\ 10\ 15\ 21 \ldots$

$> n$

```
fun(n)
{  int i=1, s=1;
   while (s <= n)
   {  i++ → 1,2,3,4
      s += i;
   }
}
```

sqrt(n)

s=1

1+1+2

i	s
1	1
2	3
3	6
4	10
⋮	⋮

I II III

3 T.C $O(\sqrt{n})$

1 +2 +3 +4
1, 3, 6, 10 . .

15, 21 . . .

i^{th} term in series $> n$

$\frac{i(i+1)}{2}$

$i=1 \rightarrow 1$

$i=2 \rightarrow 3$

$i=3 \rightarrow 6$

$i=4 \rightarrow 10$

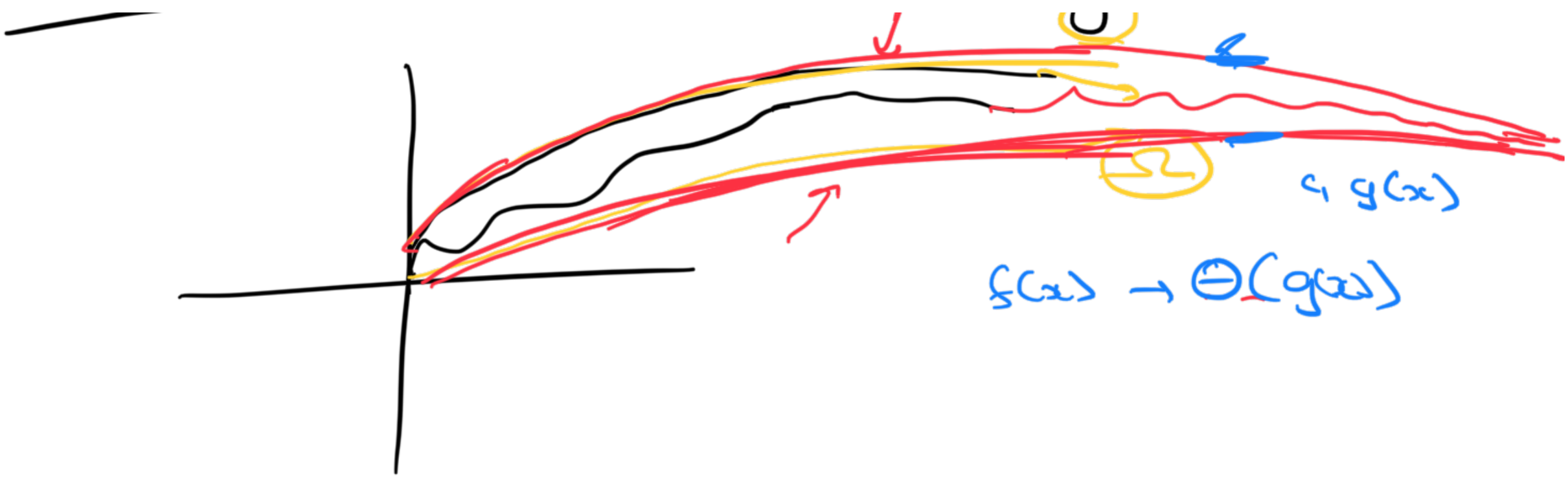
$\rightarrow \frac{i(i+1)}{2} > n \rightarrow i(i+1) = 2n$

$i^2 + i = 2n$

$\propto \quad i \propto \sqrt{n}$

Extra

$c_2 g(z)$

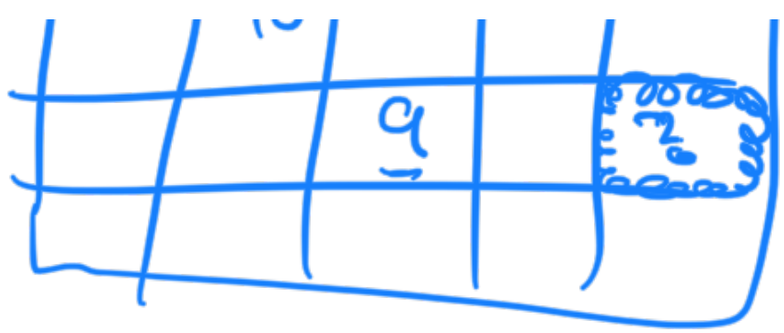


$c_1\, g(x) \le f(x) \le c_2\, g(x)$ . . . No .

Puzzle

x, y → 10

3, 2

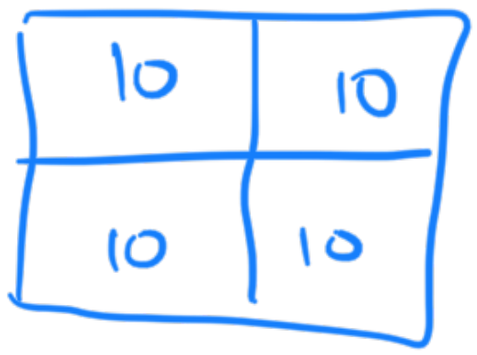


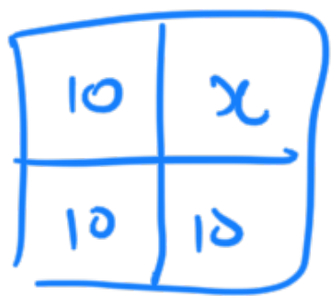
(x2, y2) ?

↓

cin >> i;
cout << i;

I





x > 10 ⟶ x is max

x < 10 → x is min