$1 \to N$

freq

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 4 | 5 | 1 | 3 | 2 | 6 |
| | | $l$ | | $r$ | |

if $(r-l+1 == i)$
→ found
else
→ not found

✓ $i=1$ ✓

✗ $i=2$ $\quad \boxed{r-l+1} = 4-2+1 = \boxed{3} > 2$ ⇒ there are some extra values b/w $l$ and $r$

✓ $i=3$ $\quad \boxed{r-l+1} = 4-2+1 = 3 == 3$

✗ $i=4$ $\quad \boxed{r-l+1} = 4-0+1 = 5 > 4$

✓ $i=5$ $\quad \boxed{r-l+1} = 4-0+1 = 5 == 5$

✓ $i=6$ $\quad r-l+1 = 5-0+1 = 6 == 6$

$(l, r) \to \{1, 2, 3, 4, 5\}$

$l = 2$

$r = 4$

$4 \rightarrow$ at index 0

which one to shift ??


1 $a < l$ : shift $l$ to $a$

② $b >= l$ and $b <= r$ : no shift

③ $c > r$ : shift $r$ to $c$

freq

| | | | | | | |
|---|---|---|---|---|---|---|
| X | 2 | 4 | 3 | 0 | 1 | 5 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

ary =

| | | | | | |
|---|---|---|---|---|---|
| 4 | 5 | 1 | 3 | 2 | 6 |
| 0 | 1 | 2 | 3 | 4 | 5 |

freq[7] = { | | 2 | | | 0 | 1 | |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| | | 2 | | | 0 | 1 |

i=0 freq[4] = 0

i=1 freq[5] = 1

i=2 freq[1] = 2

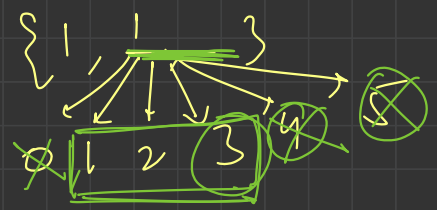
i=3 ⋮

freq[arr[i]] = i

1 as the min. element

other possible elements {1, 2, 3}

| value | freq |
|---|---|
| 1 → | 3 |
| 2 → | 4 |
| 3 → | 2 |
| 4 → | 5 |
| 6 → | 7 |



fix 1 as the min. element.

$1 \to 3$

$2 \to 4$ $\quad 3 \to 2$

- triplet with 3 ones $\Rightarrow {}^3C_3$ triplets $\{1, 1, 1\}$
- triplet with 2 ones $\Rightarrow {}^3C_2 \times \underbrace{(1+4+2)}_{x}$ triplet

$\{1, 1, x\}$ $\quad 1, 2, 3 \to 1, 4, 2$

- triplet with 1 one $\Rightarrow {}^3C_1 \times {}^{(2+4+2)}C_2$ triplet

$\{1, -, -\}$

$1 \quad 2 \quad 3$

$2 \quad 4 \quad 2$

$4C_2$ $3C_2$ $3C_2$ $4C_2$

arr 1, 1, 2, 3, 3, 4, 5, 5, 7, 7, 9, 9, 9, 11

{3, _, _}

→ iterate from left to right in sorted array:

→ and find the last index of element

arr[i] + 2

using binary search

$n \cdot \log n$

$$^{n}C_{2} = \frac{n!}{(n-2)!\,2!} = \frac{n(n-1)}{2}$$

# Recursion

- function calling itself
- solving a problem by solving its sub-problem

factorial(5) → use a loop and get the product of 1 to 5.

get me the factorial (4) and multiply it with 5 to get the answer

$fact(5) = \boxed{5} \times fact(4)$ → it is a smaller problem

original prob.

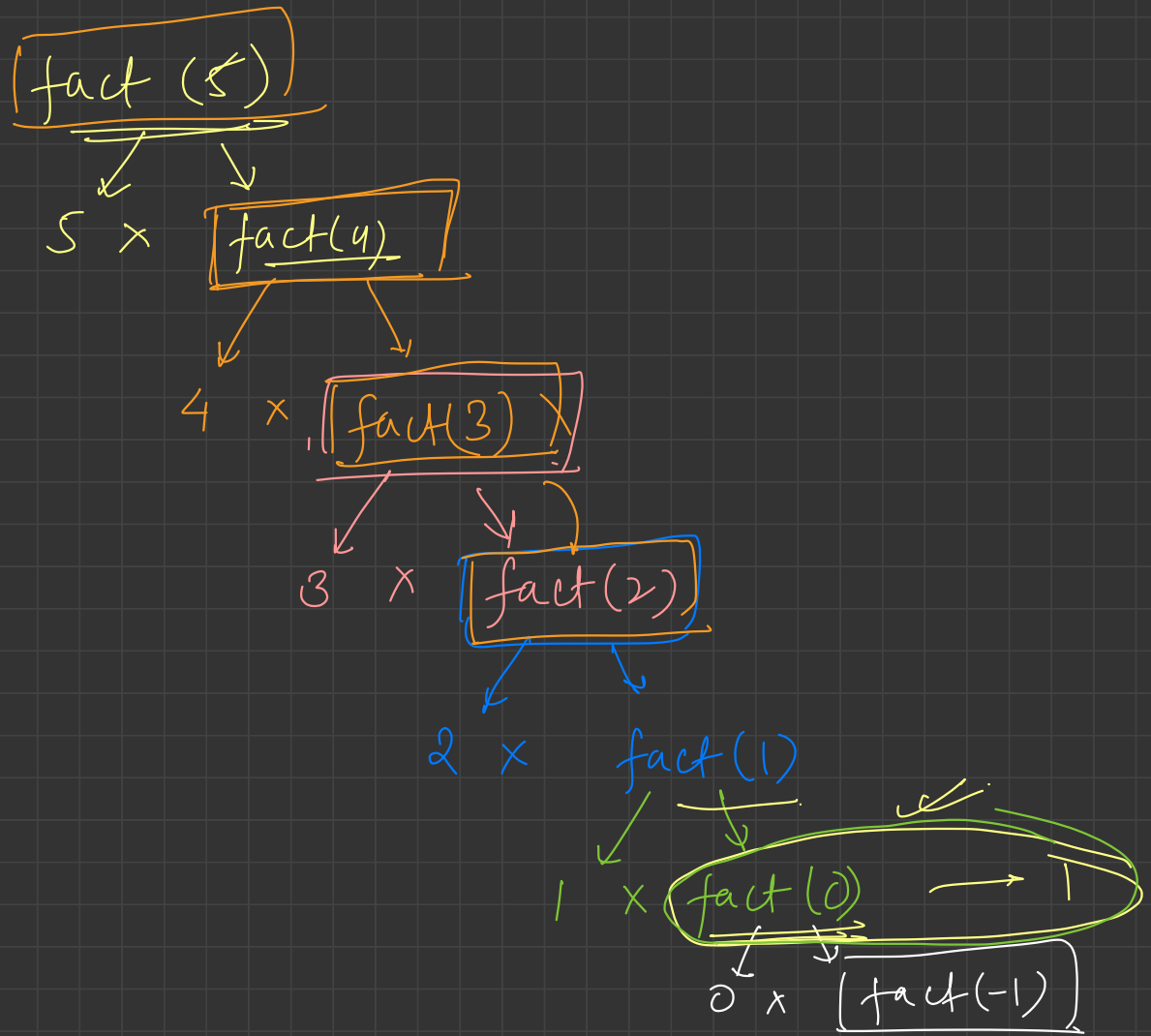
some work

$4 \times fact(3)$

$3 \times fact(2)$

⋮

$$\boxed{fact(n) = n \times fact(n-1)}$$


fact (5)
5 x
fact(4)
4 x
fact(3)
3 x
fact(2)
2 x
fact(1)
1 x
fact(0)
1
0 x
fact(-1)

loops ⇒ end-condition

similarly recursion has a base_case

↳ stopping condⁿ

$-1 \times \boxed{fact(-2)}$

$-2 \times \boxed{fact(-3)}$

⋮

infinity

recursion

1. base case / otherwise goes to ∞
2. work.
3. recursive call

- we create a function to solve a problem

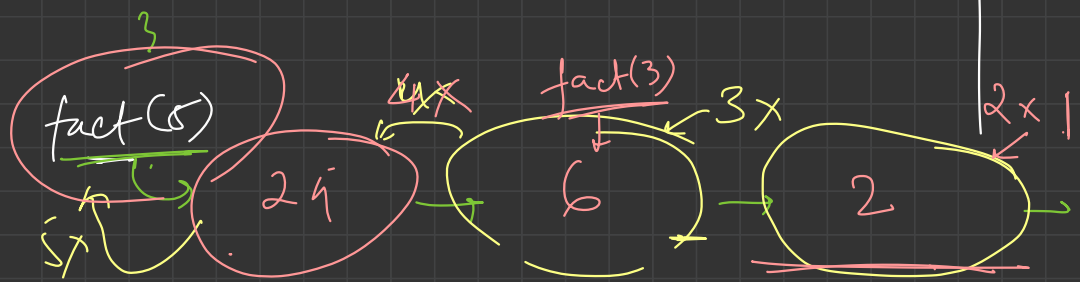
⇓

recursively solves some sub-problem

```
int factorial (int n) {
int ans = n x factorial (n-1);
    return ans;
```

it takes an input an integer(n) and returns an integer (n!)

is the factorial of input



```
int fact (int n) {
    if (n==0)
        return 1;
    int ans = n x fact(n-1);
    return ans;
}

main() {
    cout<<fact(4);
}
```

base case

24

fact(4): n = 4.
ans = 4 x 6 = 24

call stack

$$fact(n) = n \times fact(n-1)$$

Fibbonacci Series : 0th 1st 2nd 3rd 4th 5th

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Factorial Series : $\boxed{1}$, 1, 2, 6, 24, 120, 720, ...

×1 ×2 ×3 ×4 ×5 ×6

$$\boxed{fibb(n) = fibb(n-1) + fibb(n-2)}$$

nth term

$$\boxed{\begin{aligned} fibb(0) &= 0 \\ fibb(1) &= 1 \end{aligned}}$$

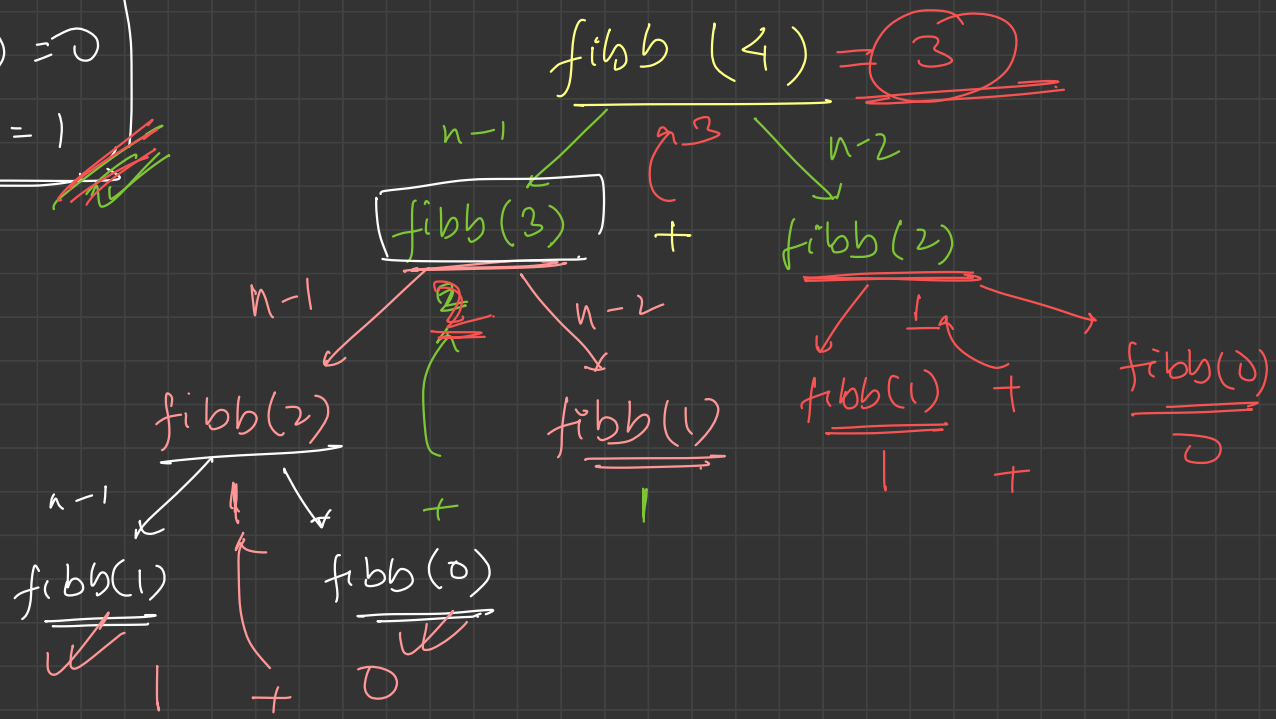



fig: fibb() recursive tree

fib (n)

fib (n)

fib(n-1)

fib(n-2)

fibb(4): n=4

ans: 2 + 1 = 3

1. identify
2. problem / input
3. how to break into smaller problem
4. solve sub-problem (recursively)
5. combine the result to get answer
6. handle the base cases

Q. print numbers from 1 to N, recursively

Q. print N to 1 recursively.

int printSeries (int start, int end)

global

printSeries ( 1, N )

cout << 1;

printSeries (2, N)

cout -

printSeries (3, N)

base case

```
main() {
    int n;
    cin>>n;
    printSeries(1,N);
}
```

Q. Given a number, print it in words recursively.

163 : One Six Three

Q. Check if an array is sorted recursively.