2-April-2022

Khushboo Shakya

# RECURSION

$$n! = n*(n-1)*(n-2)* \cdots *1$$

$$n! = n*(n-1)!$$

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

→ Recursion is used when we want the sol. of a thing which depends on the same thing but in smaller input size.

$$\boxed{fact(n-1) = (n-1) * fact(n-2)}$$

fact(4) → fact(3) → fact(2) → fact(1) → $\boxed{fact(0)}$

Stop the base cond$^n$

---

4 x 3 x 2 x 1

main
↓
fact(4) n=4
↓
fact(3) n=3
↓
fact(2) n=2
↓
fact(1) n=1 → Small O/P = 1
↓
fact(0) → return 1

n*SO = 1

SO = 1 n = 2

return 2

2x3 = 6

return 6

$\boxed{6\times4}$ — 24

# Recursion and PMI (Principal of mathematics Index)

PMI

Prove
1. Base case :- Prove $f(0)$ or $f(1)$ is true
2. Induction hypothesis: Assume that $f(K)$ is true
3. Indn step: Using 2 prove that $f(K+1)$ is true.

Ex: $E_n = n\frac{(n+1)}{2}$

BC: Base Case $f(0) = \Sigma 0 = 0$ L.H.S.

$\frac{n(n+1)}{2} = 0$ R.H.S

$f(1) = \Sigma 1 = 1$

$1\frac{(1+1)}{2} = 1$ same

I.H.: I.H $f(K) = \Sigma K = \frac{K(K+1)}{2}$

I.S.: T.P. → $\Sigma(K+1) = \frac{(K+1)(K+2)}{2}$

$K+1+\Sigma K = K+1 + \frac{K(K+1)}{2}$ from

$= \frac{(K+1)2}{2} + \frac{K(K+1)}{2}$

$= \left(\frac{K+1}{2}\right)(K+2)$

Date:
Page No.

[if K true then K+1 is also true]

Now I can solve rec. question acc to PMI

```
int factorial (int n)
{ if(n==0)
      return 1;

      [int SO = factorial (n-1)]
```

I.H
(n-1) se hi
to n
se hi

```
      int Output = n * SO
      return Output
```

Q Fibonacci Number
$fib(n) = fib(n-1) + fib(n-2)$

Q Extended form of PMI

I.H $f(i)$ is true $\forall$
(Assum) $i \leq K$

[making 2 jumps
to 2 B.C.]

Q Check if array is sorted

| 3 | 4 | 5 | 6 | 7 |

n=0 — 0
n=1 — return n

ans a[0] = a[0] + a[1]

1 Base case
2 Recursive call
3 small calculation

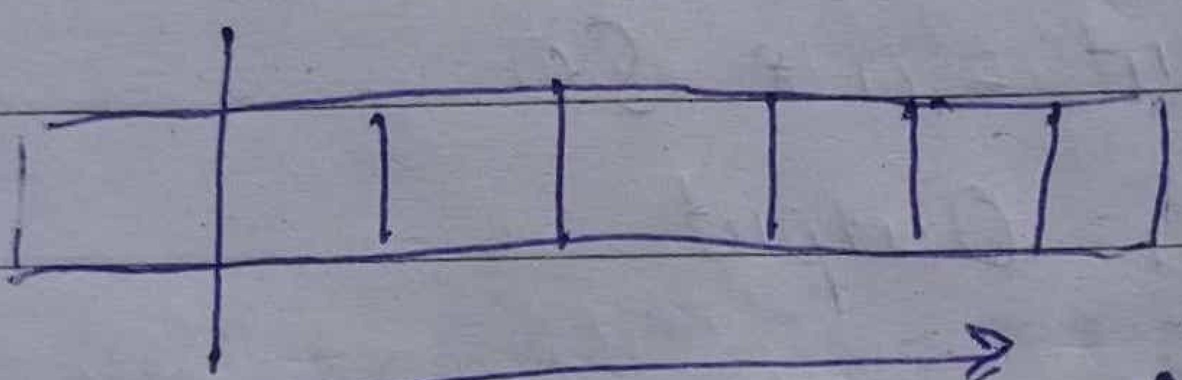
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 5 | 6 | 5 | 1 |

a[5-1] = = n
return 5-1 → 4

a-1 size-1

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

Q First index



→ first step:- break

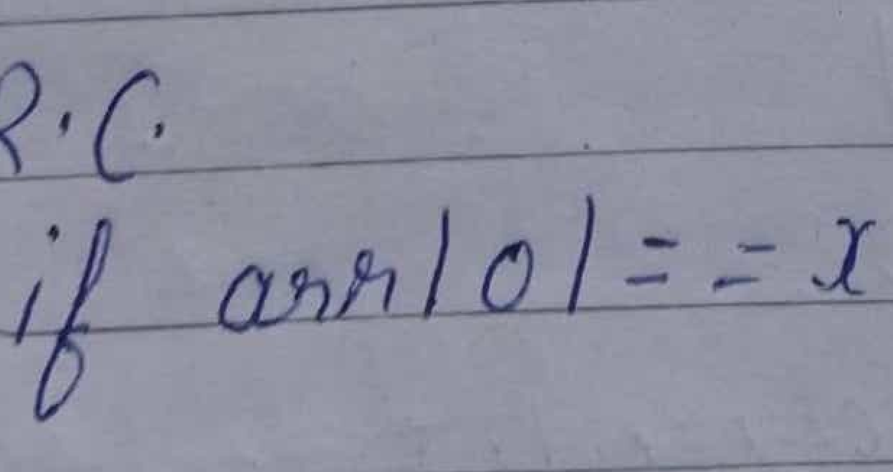
Find first index in which x is present

1) Base case → -1
2) Recursive case → a[0] = = x
→ (a+1, size-1, x) = ans
if x = -1 → -1
if x != -1 → ans+1

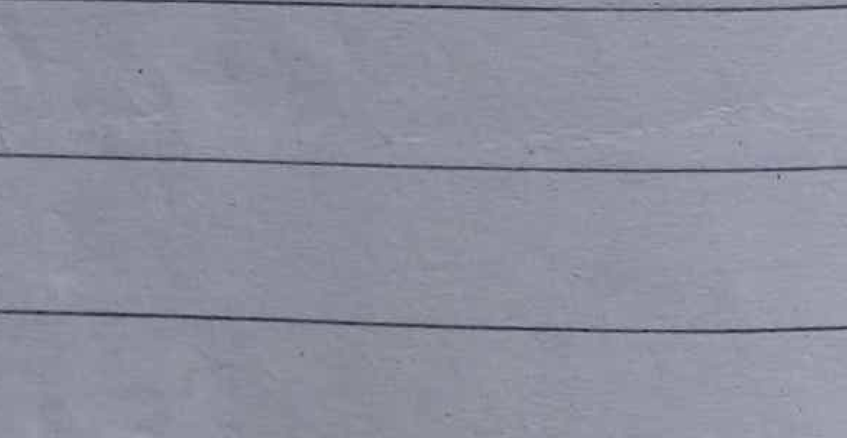
3) Small calculation

Q. Last index

i) R.C
ii) if arr[0] = = x



or



we can do in this way also.

First index
i) if arr[0] == x
ii) R.C

Last index
i) R.C
ii) if arr[0] == x

Q. All indices in Array

going - arr. count

2 3 6 9 8 3 3

returning - no filling in array

| 1 | 5 | 6 |
|---|---|---|
| 0 | 1 | 2 |

Q. Count zeros

n = 708000

| 70800 | 0 |
|---|---|

If 708000 % 10 == 0

4 — return 1 + (70800)

70800 % 10 == 0

3 — return 1 + (7080)

If (7080 % 10 == 0)

2 — return 1 + (708)

708 % 10 == 0 X

return (70)

1 — 70 % 10 == 0
return 1 + (7)

B.C - n == 0

# Recursion and String

Q length of string

$\boxed{a|b\ c\ d}$

abcd \0

3 bcd \0

2 cd

1 d \0 → " \0 "

0

1+3=4

1+ length(s+1);

Q Remove x

abcxdx → abcd

base cases
s[0] = '\0'
return 0

Two cases
s[0] != 'x'
remove x(s+1);
s[0] == 'x'
for (int i=1; s[i] != '\0'; i++){
s[i-1] = s[i];
}
s[i-1] = s[i];

x abcx
abx

x abc x \0

a bcxx

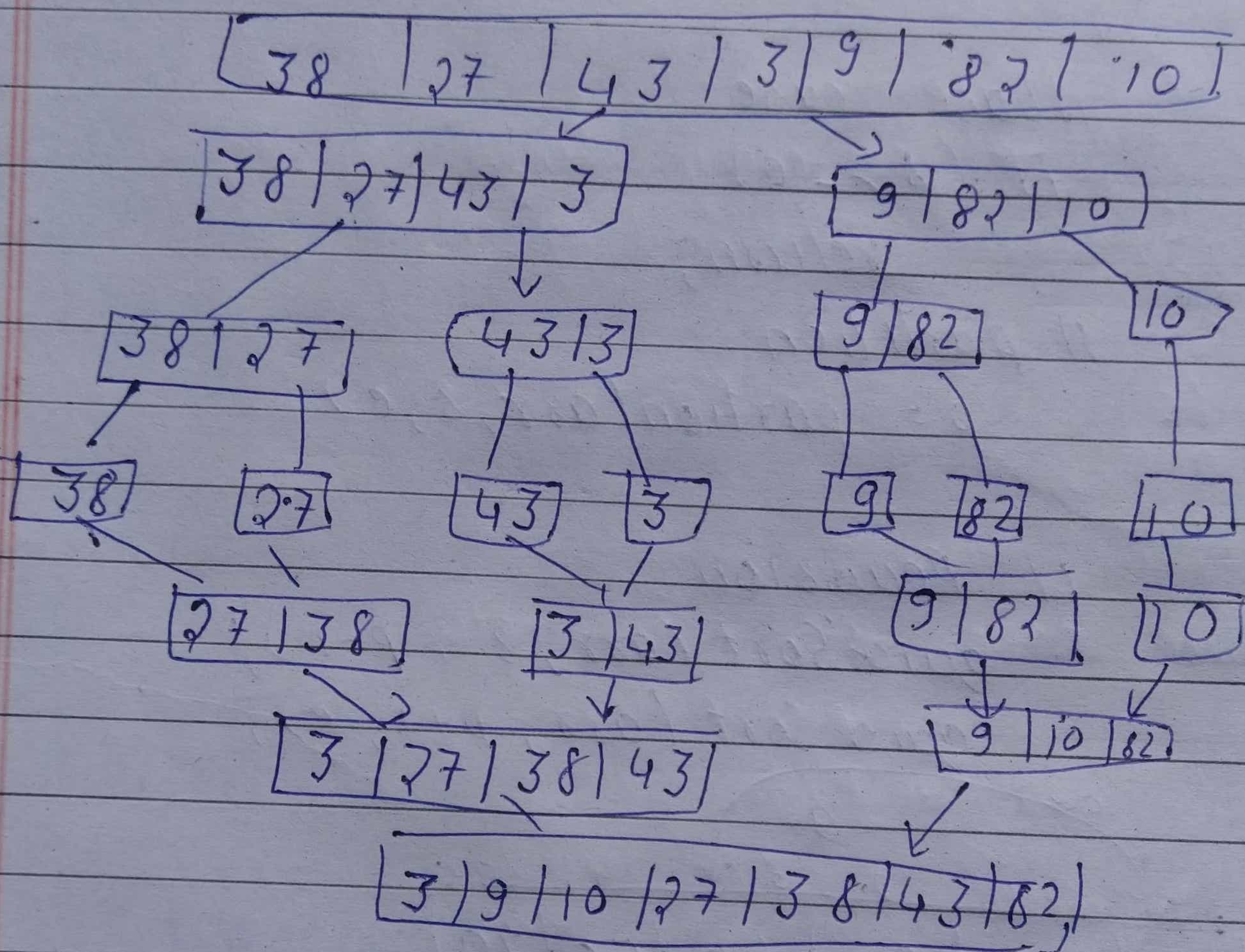
# Merge Sort

Base case
- Size = 0
- Size 1

Si = ei

return/already sorted

Si >= ei or empty

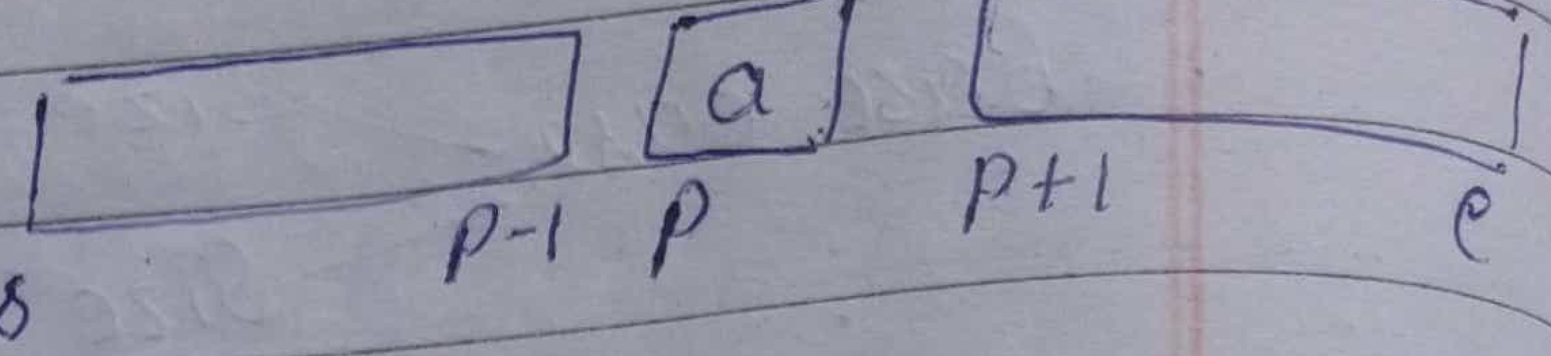



Pseudocode

- Declare left variable to 0 and right variable to n-1
- Find mid. (mid = left + right)/2
- Call merge sort on (left, mid)
- Call merge sort on (mid+1, rear)
- Continue till left is less than right
- Then call merge function to perform merge sort.

# Quick Sort

Partition
Recursion




approch

```
void quickSort(int arr[], int s, int e)
{
    //base case
    if (s >= e)
        return;
    // partition
    p = partion(arr, s, e)

    // Recursion
    quickSort(arr, s, p-1);
    quickSort(arr, p+1, e);
```

swap

4.

Dry run

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 80 | 30 | 90 | 40 | 50 | 70 |

Step1 take pivot
pivot = 80

Step 2 Count all elemen < pivot

count = 0 → 4

Step3 pivot → s + count
= 0 + 4 = 4. (swap with 4th element)

Step 4) [ <a | a | >a ]

After swap

~~50 | 30 | 90 | 90 | 80 | 90~~

| 40 | 30 | 90 | 80 | 50 | 70 |
|---|---|---|---|---|---|
| i | i | i | | | |

| 50 | 30 | 70 | 40 | 80 | 90 |
|---|---|---|---|---|---|
| | | i | | j | |

sort <80 >80 right

## Strings

s + "abc" → concatenation like this for string

s.size()

s.substr(3) → def    abcdef (0 1 2 3 4 5)
s.substr(3,2) → de

→ for char array we strlen or find null character

→ In char array we use concatenation

s.find ("def") → will give index
→ 3

$\boxed{\text{string *s = new string;}}$ dynamically making string

# Subsequence of string

'abc'

$\boxed{ac}$ subsequence but not substring
↓
contious

" "
a
b
c
ab
ac
bc
abc

→ for every length n it will have $2^n$ subsequence

Base case — " " — return 1

a bc

Ye mera kaan

Ye recursion ka