07 February 2021 18:33

Re ① → Factorial using Recursion ?

sol →
```
int factorial ( int n)
{
  if ( n == 0 )          // Base condition
     return 1;
  else                   // Recursive part
  { return fac(n-1) * n;
```

n = 3 → fac(2) * n

fac(3) = fac(2) * 3 ✓
fac(2) = fac(1) * 2
fac(1) = fac(0) * 1
fac(0) = 1 → Base condition ✓

O(n!) → values
→ O(n) for time

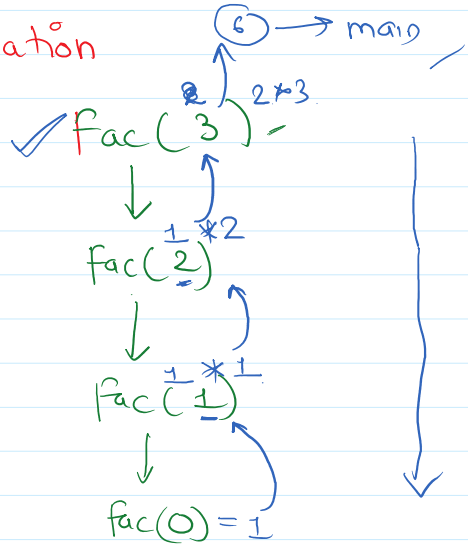
$$fac(n) = \begin{cases} fac(n-1) * n & n > 0 \\ 1 & n == 0 \end{cases}$$

→ for value ✓
→ Base condition ✓

→ O(n!)
= O(n^n) ✓

2^n
a^n
n^n

Recurrence Relation



② → Fibbonacci Number using Recursion ?
for n^{th} term.
0 1 1 2 3 5 .....
1st 2nd 3rd .....

Base condition n == 1 return 0 ✓
n == 2 return 1
Recursive condition n > 2 return f(n-2) + f(n-1) ✓

Recurrence Relation

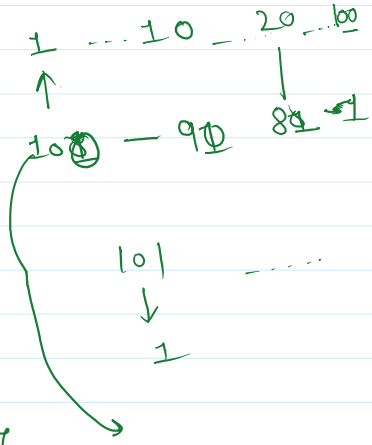
$$Fib(n) = \begin{cases} f(n-2)+f(n-1) & n>2 \\ 1 & n=2 \\ 0 & n=1 \end{cases}$$

Time complexity :-

```
void print(int n)
{
    if (n > 100) { return; }   O(1)
    else
    {
        cout << n; }   O(1)
        print(n+1);
    }
}
```

print(1)

1 100



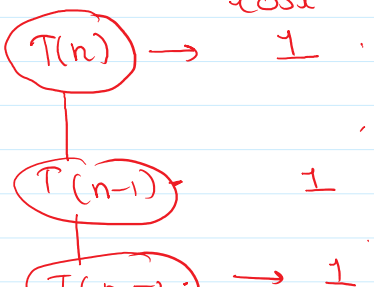
TC ? = Recurrence Relation for Time

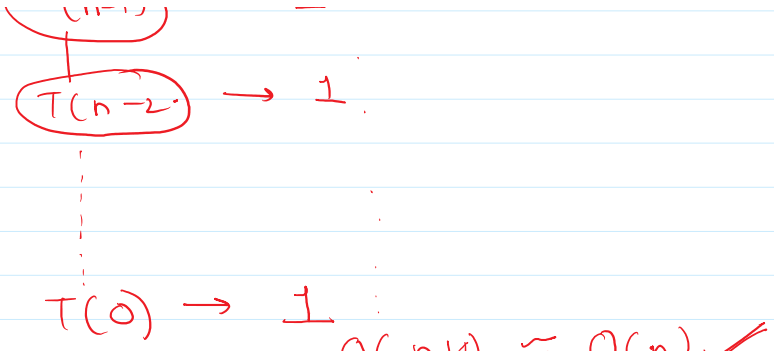
$$T(n) = \begin{cases} 1 & n>100 \text{ Base case} \\ 101-n & \text{otherwise} \end{cases}$$

T(n=50) = 51 loop ✓

```
void print(int n)   T(n)
{
    if (n == 0)
        return;   } O(1)
    else
    {
        cout << n;   } O(1)
        print(n-1);   } T(n-1)
    }
}
```

$$T(n) = \begin{cases} T(n-1) + 1 & n>0 \\ 1 & n==0 \end{cases}$$





Question for TC?

①
```
int fun(int n)
{
    if (n==0)
        return 1
    else
    return fun(n-1) + fun(n-1)
}
```

for val

$$fun(n) = \begin{cases} 1 & n==1 \\ 2\,fun(n-1); & n>1 \end{cases}$$

For Time ? ✓

$$f(n) = \begin{cases} 1 & n==1 \\ 2\,fun(n-1) & n>1 \end{cases}$$

②
```
int fun(int n)
{
    if (n==1) {
        return 1;
    }
    else
    return 2*fun(n-1);
}
```

For value

$$fun(n) = \begin{cases} 1 & n==1 \\ 2*fun(n-1) & n>1 \end{cases}$$

For time

$$T(n) = \begin{cases} 1 & n==1 \\ T(n-1)+1 & n>1 \end{cases}$$

For 2nd

T(n) — 1
|
T(n-1) — 1 → (n+1) * 1
|
T(n-2) — 1
:

= O(n+1) ≃ O(n)

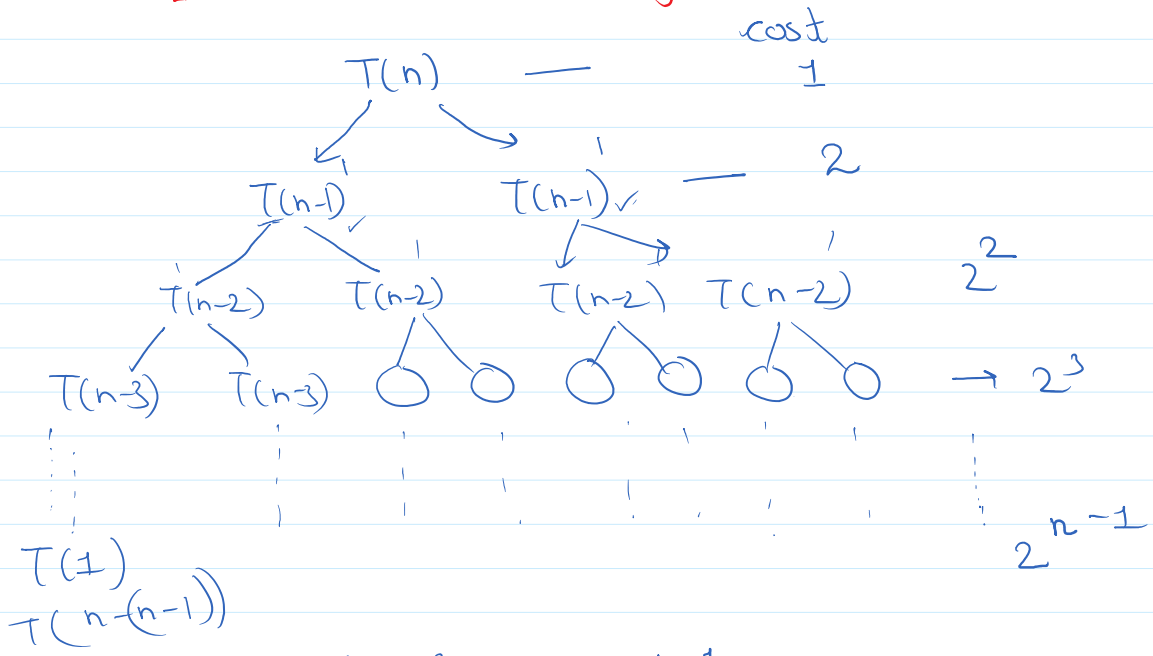
$T(n-1) - 1 \rightarrow (n+1) * 1$

$T(n-2) - 1 = O(n+1) \simeq O(n)$

$T(1) \rightarrow 1$

For 1st

$$\begin{cases} T(n) = 2T(n-1) + 1 & n > 1 \\ 1 & n == 1 \end{cases}$$



Sum $= 1 + 2 + 2^2 + 2^3 \ldots\ldots 2^{n-1}$ No of terms n

$= $ sum of GP $\quad a = 1 \quad r = 2$

$$\frac{a(r^{\text{no of term}} - 1)}{r-1} = \frac{1(2)^n - 1}{2-1} = \frac{2^n - 1}{1}$$

$= O(2^n - 1)$

$= O(2^n) \simeq \exp$ for (a^n)

Approach

Divide & conquer → Power fun → $\log(n)$ $\quad O(n)$

→ Min Max →

→ Quick Sort

← Merge Sort

Sorting

Greedy

DP

→ Generate all permutation of string (Recursion) (Backtracking)

→ Generate all permutation of string (Recursion) (Backtracking) (Stack)

Input → ABC

Output →
ABC
ACB
BAC
BCA
CAB
CBA
} 6

("ABC", curr 0)