26 Time Complexity Of Recursive Algorithm & OOPs - I

Time Complexity   Total time required as function of input.

Factorial

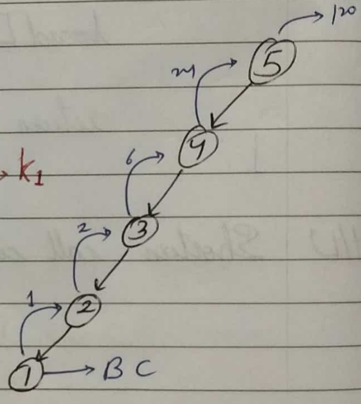
$f(n) \supseteq T(n)$

```
int factorial (int n) {
    if (n <= 1)
        return 1;          } O(1) → k1

    return factorial (n-1) * n;
}
```

call: $f(n-1) \longrightarrow T(n-1)$

multiply: $O(1) \downarrow k_2$



Recursive Relation

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

$T(n) = ?$

$\boxed{T(n) = \underbrace{k_1 + k_2}_{k} + T(n-1)} \Rightarrow T(n) = k + T(n-1)$

$T(n) = k + T(n-1)$ Adding all the eq together

$\cancel{T(n-1)} = k + \cancel{T(n-2)}$

$\cancel{T(n-2)} = k + \cancel{T(n-3)}$

$\vdots$

$\cancel{T(2)} = k + \cancel{T(1)}$

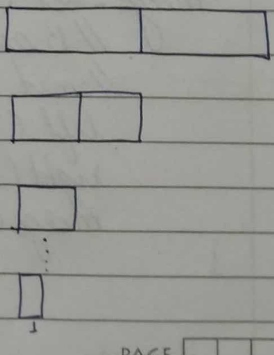
$\cancel{T(1)} = k_1$ $\rightsquigarrow (n-1)k + k_1 = n*k - k + k_1$

$\underbrace{\qquad}_{n \text{ times}}$ $- k_3 \rightarrow nk$

$T(n) = n * k$

Binary Search

```
int fun (int arr [], int s, int e, int target) {
     //BC
  ~~if~~ while (s < e) {                                    Time
        int mid = (s+e)/2;                                   1
        if (arr [mid] == target)                             1
              return true;                                   1
        if ( )                                               1
              return fun (arr, s, mid, target)          + (n/2)
        else                                                 1  cutting
              return fun (arr, mid, e, target)          T(n/2)
     }
}
```



$f(n) = k + f(n/2)$

$T(n) = \underbrace{k_1 + k_2 + k_3}_{k} + T(n/2)$

$= k + T(n/2)$

$T(n) = k + \cancel{T(n/2)}$
$\cancel{T(n/2)} = k + \cancel{T(n/4)}$
$\cancel{T(n/4)} = k + \cancel{T(n/8)}$
$\vdots$
$\cancel{T(2)} = k + \cancel{T(1)}$
$\cancel{T(1)} = k$

$$\begin{tikzcd} n \arrow[d] \\ \frac{n}{2} \arrow[d] \\ n/4 \arrow[d] \\ n/8 \arrow[d] \\ \vdots \arrow[d] \\ n/2^a = 1 \end{tikzcd}$$

$$\begin{tikzcd} n/2^0 \to n \arrow[d] \\ n/2^1 \to n/2 \arrow[d] \\ n/2^2 \to n/4 \arrow[d] \\ n/2^3 \to n/8 \arrow[d] \\ \vdots \arrow[d] \\ n/2^a \to 1 \end{tikzcd}$$

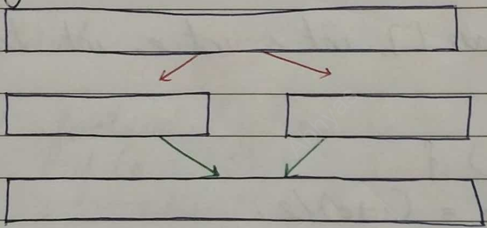
$\frac{n}{2^a} = 1$

$n = 2^a$

$\log_2 n = a$

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

Merge Sort



Steps --→ Left / Right split
→ Recursion sorts left & right
→ merge

```
mergeSort ( ) {                Time  T(n)
    // Base case                1
    // mid                      1
    left ( )                    n/2
    right ( )                   n/2
    merge ( )                   k_3 n
}
```

~~$T(n) = k + T(\frac{n}{2}) + T(\frac{n}{2}) + k_3 \cancel{N} n$~~

~~$2\times [T(n) = 2T(n/2) + k_5 n] \rightsquigarrow 2T(n) = 4T(n/2) + 2k_5 n$~~

~~$4\times [T(n/2) = 2T(n/4) + k_5 (n/2)] \rightsquigarrow 4T(\frac{n}{2}) = 8T(n/4) + 4k_5(n/2)$~~

~~$8\times [T(n/4) = 2T(n/8) + k_5 (n/4)] \rightsquigarrow 8T(n/4) = 16T(n/8) + 8k_5(n/4)$~~

~~$\vdots \qquad \vdots$~~

~~$T(1) = k$~~

$T(n) = k + T(\frac{n}{2}) + T(\frac{n}{2}) + kn$

$n \quad T(n) = 2T(\frac{n}{2}) + k(n)$

$n/2 \quad 2[T(\frac{n}{2}) = 2T(\frac{n}{4}) + k(n/2)] \rightsquigarrow 2T(\frac{n}{2}) = 4T(\frac{n}{4}) + \cancel{2}k(\frac{n}{2})$

$n/4 \quad 4[T(\frac{n}{4}) = 2T(\frac{n}{8}) + k(\frac{n}{4})] \rightsquigarrow 4T(\frac{n}{4}) + 8T(\frac{n}{8}) + kn$

$\vdots \qquad \vdots$

$\underbrace{1}_{\log_2 n \text{ times}} \quad T(1)$

$$T(n) = a * k * n - kn + k$$
$$= akn - k(n+1)$$
$$= \cancel{k}n\log n - \cancel{k(n+1)}$$
$$= n\log n$$

Fibonacci Series

```
int fib (int n) {
    //BC
    if (n == 0 || n == 1)
        return n;
    return fib(n-1) + fib(n-2);
}
```

DATE

Recurrance Relation

$f(n) = f(n-1) + f(n-2)$

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

$T(n-1) = T(n-2) + T(n-3)$

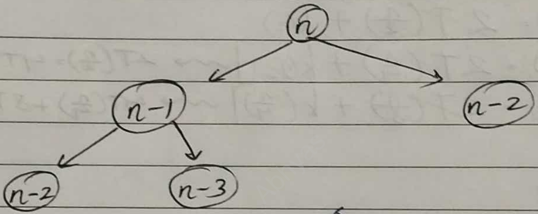
$T(n-2) = T(n-3) + T(n-4)$

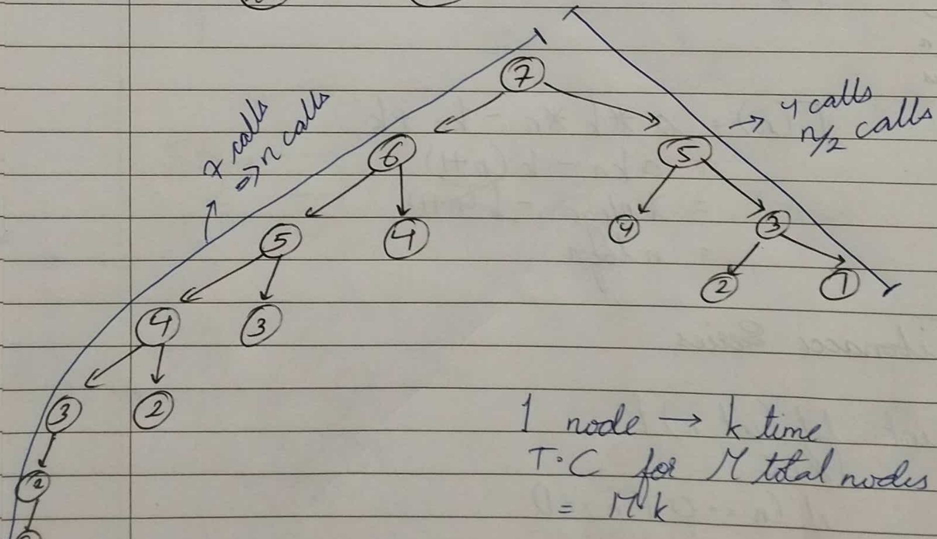
$\vdots$

$T(2) = T(1) + T(0)$

$T(1) = k$

$T(0) = k$





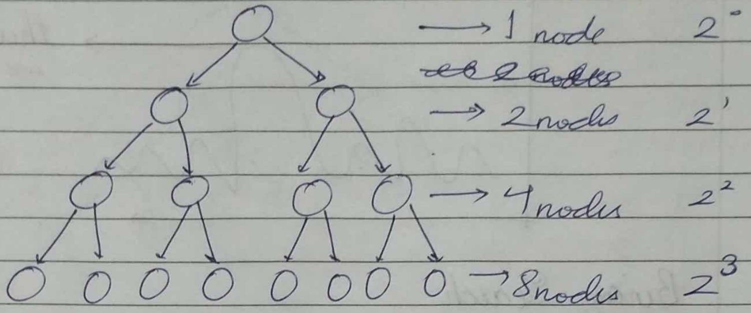
1 node → k time

T.C for M total nodes

$= M \cdot k$

classmate

PAGE



Total nodes $= 2^0 + 2^1 + 2^2 + \cdots + 2^n$
$\hookrightarrow$ GP

$= 2^{n+1} - \cancel{1}$
$= \cancel{2}\, 2^n$
$= 2^n$

T.C = $O(2^n)$

Space Complexity

Factorial

```
int fact (n) {
    if (n <= 1)
        return 1;

    return n * fact (n-1);
}
```

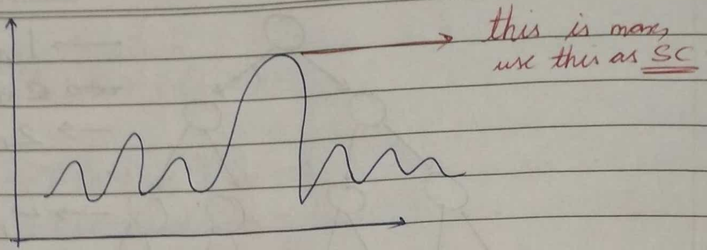
one call
$\Rightarrow$ constant space

Call stack

f(1)	→ O(1)
f(2)	→ O(1)
f(3)	→ O(1)
f(4)	→ O(1)
f(5)	→ O(1)

$n \times O(1) \Rightarrow O(n)$

Space Complexity → $O(n)$



Binary Search

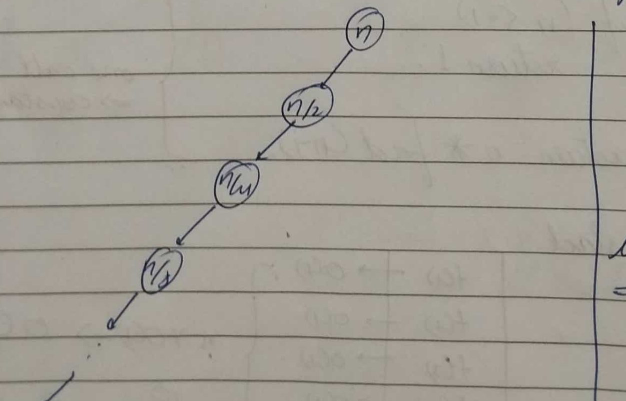
```
fun () {
    //BC
  //mid
  // compare
  // if ()
        left()
    else
        right()
}
```

k space

$n \downarrow n/2 \downarrow n/4 \downarrow \vdots \downarrow 1$

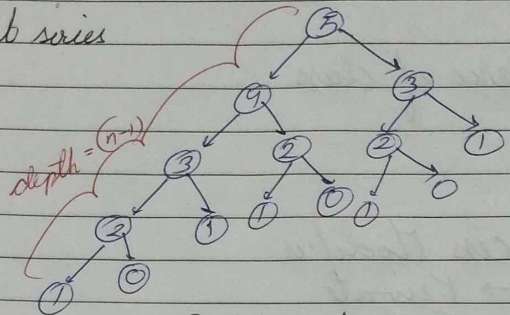
$k \log(n)$

S.C $\rightarrow \log n$)



depth logn
=> S.C logn

Fib series



$(n-1)k = k$

$S.C \rightarrow nk \longrightarrow \underline{\underline{O(n)}}$

OOPs

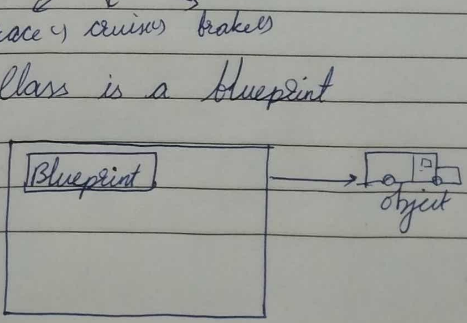
→ Programming using objects

class

$$\begin{tikzcd}
\text{Human} \arrow[r] \arrow[d] & \text{Properties} \arrow[r] \arrow[r, shift left] \arrow[dr] & \text{eyes} \\
\text{behaviour} \arrow[d] \arrow[dl] \arrow[dr] & & \text{hands} \\
\text{sleep()} & \text{code()} & \text{legs} \\
\text{eat()} & &
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Car} \arrow[r] \arrow[d] & \text{Props} \arrow[r] \arrow[r, shift left] \arrow[r, shift right] \arrow[dr] & \text{engine, color, type} \\
\text{behaviour} \arrow[d] \arrow[dl] \arrow[dr] & & \text{tyre} \\
\text{cruise()} & \text{brake()} & \\
\text{race()} & &
\end{tikzcd}$$

→ Class is a blueprint



DATE

Object
$\longrightarrow$ Instance of class

C++
$\hookrightarrow$ Access Modifier
- $\hookrightarrow$ Private
- $\hookrightarrow$ Public
- $\hookrightarrow$ Protected

H/W Padding & Alignment
Does class take space?
Deep copy vs Shallow

classmate

PAGE