Recurrence relations

By:
Dr. Rimjhim Singh
Asst. Professor (Sr. Gr)
CSE, Coimbatore

Recurrence relations

- ***A recurrence equation*** defines mathematical statements that the running time of a recursive algorithm must satisfy

- ***Recurrence relation*** is a mathematical model that captures the underlying time-complexity of an algorithm

- Writing recurrence relations ✓

- Solving recurrence relations ✓
  - Iterative substitution method
  - Recurrence tree method
  - Master's theorem

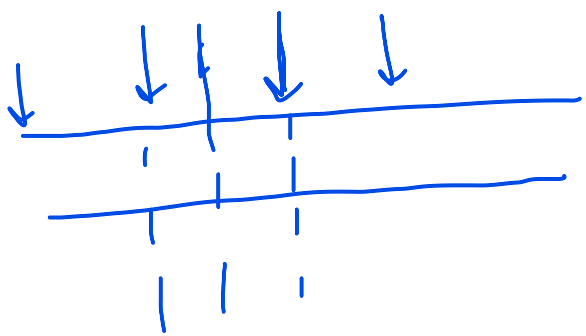
$fact(n) = f(n-1) \cdot n$

$\Rightarrow f(n-2) \cdot (n-1) \cdot n$

$\vdots$

$n = 1$

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



$n/2$ = Div fn..

$T(n) = T(n-2)$ = Dec.

$\vdots$

Writing recurrences: Ex-1

```
Void fun1(int n) ---- T(n)
{
    if (n>0) ---- 1
    {
        Print(n) ---- 1
        fun1 (n-1) ---- T(n-1)
    }
    else
        return --- 1
}
```

$T(n) =$

$\underbrace{1 + 1}_{} + T(n-1) \quad n > 0$

$O, O \Rightarrow O$

$T(n) = 1 \quad ; \quad n = 0$

(Base)

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

Writing recurrences : Ex-2

```
Void fun2 (int n)          ---- T(n)
{
    if (n>0)               ---- 1
    {
        for( i=0; i<n; i++)  ----- n+1
        {
            Print(n)       ---- n
        }
        fun2 (n-1)         ---- T(n-1)
    }
    else
        return             --- 1
}
```

for $n>0$

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

$\Rightarrow T(n-1)+\underbrace{2n+2}_{O(n)}$

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

Writing recurrences : Ex-3

```
Void fun3 (int n)
{
    if (n>0)
    {
        for( i=1; i<n; i=i*2)
        {
            Print(n)
        }
        fun3 (n-1)
    }
    else
        return
}
```

Void fun3 (int n) - - - - - $T(n)$

if (n>0) - - - - 1

for loop: $2 \log n$

fun3 (n-1) - - - $T(n-1)$

return - - - 1

For $n>0$

$$T(n-1) + \underbrace{2 \log n + 1}_{O(\log n)}$$

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

Writing recurrences: Ex-4

Void fun4 (int n) ---- $T(n)$
{
    if (n>0) ----------- 1
    {
        Print(n) ------ 1
        fun4 (n/2) ---- $T(n/2)$
    }
    else
        return ---- 1
}

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n/2) + 2 & n > 0 \end{cases}$$

Writing recurrences: Ex - 5

```
Void fun5 (int n)        ---- T(n)
{
   if (n>0)
   {
        Print(n)         ---- 1
        fun5 (n-1)       ---- T(n-1)
        fun5 (n-1)       ---- T(n-1)
   }
   else
        return
}
```

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

Iterative substitution method/ Induction method

- Also known as the “plug-and-chug” method.
- Assumes that the problem size ***n is fairly large***
- ***Substitutes the general form*** of the recurrence for each occurrence of the function T on the right-hand side.

Recurrence tree method

- It is a ***visual approach***.

- ***Draw a tree R*** where each node represents a different substitution of the recurrence equation.

- Thus, each node in R has a value of the argument ‘n’ of the function T(n) associated with it.

- In addition, *associate the overhead* with each node v in R,
  - **Overhead** is defined as the **value of the non-recursive** part of the recurrence equation for v.

Solving recurrences: Ex-1

```
Void fun1(int n)
{
    if (n>0)
    {
        Print(n)
        fun1 (n-1)
    }
    else
        return
}
```

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

Substitution

$$T(n) = T(n-1) + 1 \quad \text{----} \ 1$$

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

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

$$T(n) = \left(T(n-2) + 1\right) + 1$$

$$T(n) = \left[\left(T(n-3) + 1\right) + 1\right] + 1$$

$$\Rightarrow T(n-3) + 3$$

$\vdots$ k times

$$T(n) = T(n-k) + k$$

$$n - k = 0$$

$$\Rightarrow k = n$$

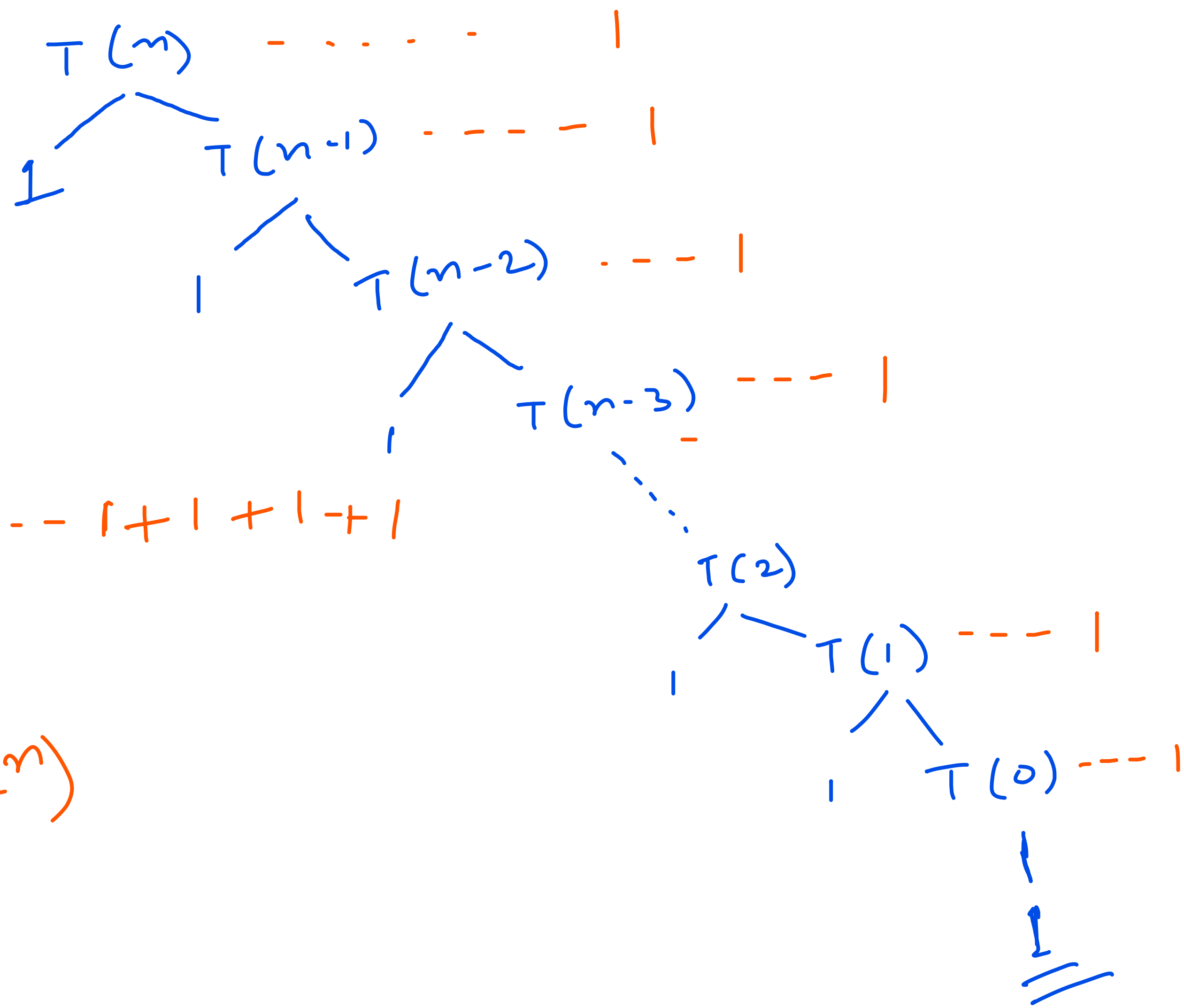
$$T(n) = T(n-n) + n$$

$$= T(0) + n$$

$$1 + n$$

$$\Rightarrow O(n)$$

Recurrence Tree



$1 + 1 + 1 + \text{-----} \; 1 + 1 + 1 + 1$

$= n$

$\rightarrow O(n)$

Solving recurrences : Ex-2

```
Void fun2 (int n)
{
    if (n>0)
    {
        for( i=0; i<n; i++)
        {
            Print(n)
        }
        fun2 (n-1)
    }
    else
        return
}
```

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

Sub

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

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

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

$T(n) = [T(n-2) + n] + n$

$= [[T(n-3) + n] + n] + n$

$\Rightarrow T(n-3) + 3 \cdot n$

k time

$T(n) = T(n-k) + k \cdot n$

$n-k=0$

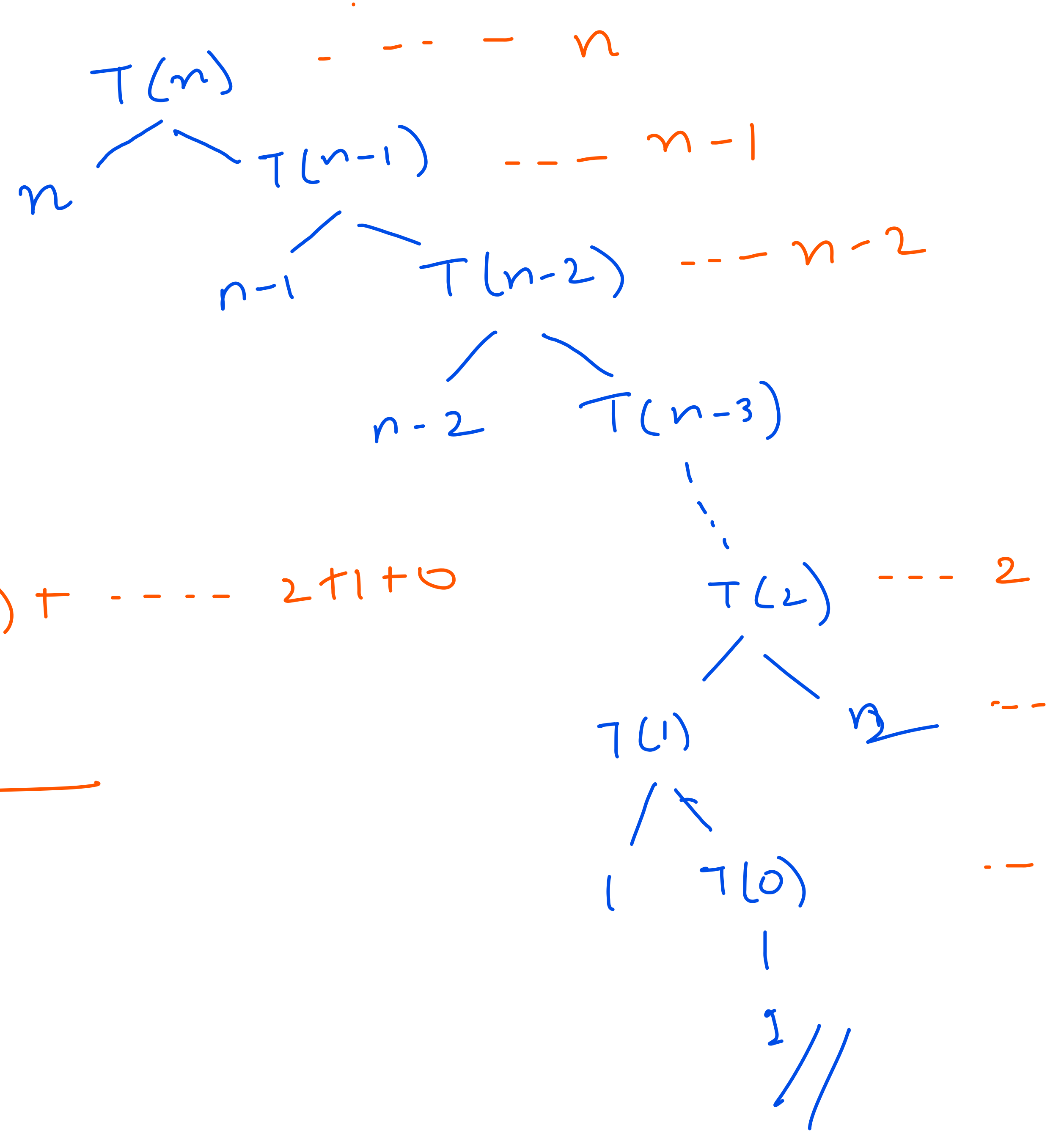
$n=k$

$T(n) = T(n-n) + n * n$

$= T(0) + n \times n$

$= O(n^2)$

Recur



$$n + (n-1) + (n-2) + \cdots 2 + 1 + 0$$

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

$$= O(n^2)$$

Solving recurrences : Ex-3

```
Void fun3 (int n)
{
    if (n>0)
    {
        for( i=0; i<n; i=i*2)
        {
            Print(n)
        }
        fun3 (n-1)
    }
    else
        return
}
```

Solving recurrences: Ex-4

```
Void fun4 (int n)
{
    if (n>0)
    {
        Print(n)
        fun4 (n/2)
    }
    else
        return
}
```

Solving recurrences: Ex - 5

```
Void fun5 (int n)
{
    if (n>0)
    {
        Print(n)
        fun5 (n-1)
        fun5 (n-1)
    }
    else
        return
}
```