# Recursive and Non-recursive Functions

Wednesday, February 3, 2021 11:56 AM

Recursive and non-recursive equation
- Time and space complexity

Non-recursive
- Constant time

Int I; //0 time units
For(i=1; i<=3; i++) // 1 + 3 + (2*(3-1)) = 8
    Sum = sum + I; // 3

T(n) = O(n)

Recursive equations do not have a concrete value

$$T(n) = a \ * T\left(\frac{n}{b}\right) + c$$

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

```
If n=1
    Return 1
Else
    Fact(n-1)*n
```

T(1) = 1

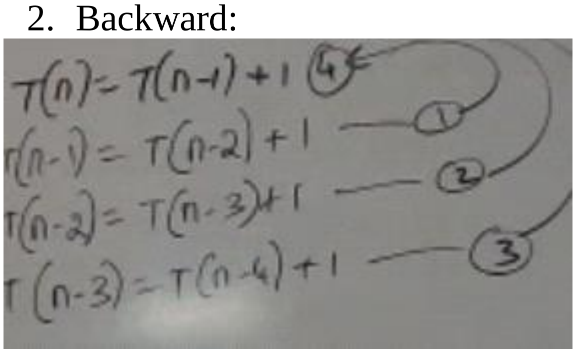
Finding time complexity of recursive functions using:

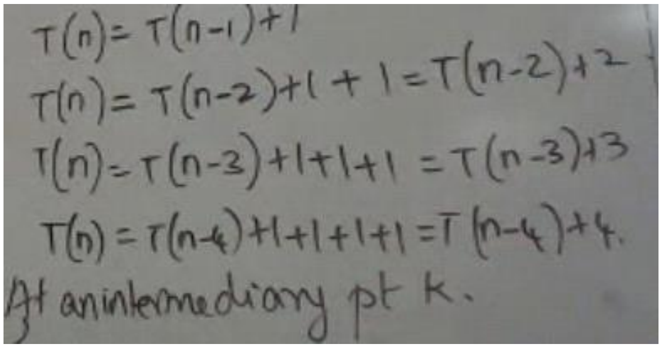
a. **Induction**
b. **Substitution**

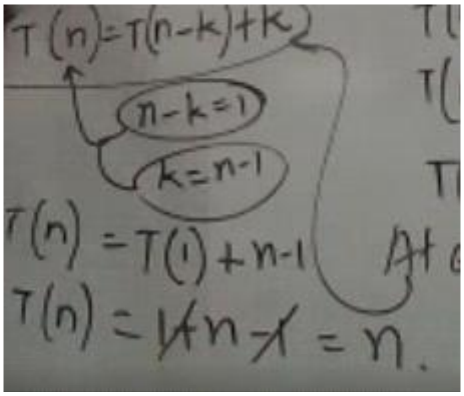
1. Forward

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

2. Backward:






**c. Recursion tree**

T(n) = 2 T(n/2) + n
T(1) = 1
T(n/2) = 2T(n/4) + (n/2)
T(n/4) = 2 T(n/8) + (n/4)

| Tree element | Level | Nodes | Value |
|---|---|---|---|
| n | 0 | 1 = 2^0 | 1*n=n |
| n/2 | 1 | 2 = 2^1 | 2*(n/2) = n |
| n/4 | 2 | 4 = 2^2 | |
| n/8 | 3 | 8 | |
| | | | |
| n/(h-1) | h-1 | 2^(h-1) | 2^(h-1) * n/(h-1) = n |
| n/h | h | 2^h | 2^h * n/h = n |

$$T(n) = \sum_{i=0}^{h-1} n + [2^h * T(1)]$$

$$T(n) = n\sum_{i=0}^{h-1} 1 + \left[2^h * T(1)\right]$$

T(n) = $n[h - 1 - 0 + 1] + \ [2^h * T(1)]$

$$n = 2^h$$

$h = \log(2)n$
$T(n) = n * \log(2)\, n + (n * 1)$
$T(n) = n * \left(\log(2)\, n + 1\right)$
$\boldsymbol{T(n) = O(nlogn)}$

**d. Master theorem**