Nour Mohamed Raafat

PAGE lab3
DATE

Algorithm 1

```
int t=0;
For (int i=1; i<=n; i++)
    For (int j=0; j*j<4*n; j++)
        For (int k=1; k*k<=9*n; k++)
            t++;
```

Solution: the Loop will end when $i=n$

1.A) For(int i=1; i<=n; i++)

$i=0$ $i=3$

$i=2$ $i=n \Rightarrow$ will stop

$\therefore$ Complexity 1 = (n)

1.B) For(int j=0; j*j<4*n; j++)

$j=0$ $j=2$ $j*j=4n \rightarrow$ will stop

$j=1$ $j=4$ $\therefore j^2=4n$

$j=2\sqrt{n}$

$\therefore$ Complexity 2 = $2\sqrt{n}$

1.C) For (int k=1; k*k<=9*n; k++)

$k=1$ $k=3$ $k^2=9n$ will stop

$k=2$ $k=4$ $k=3\sqrt{n}$

$\therefore$ Complexity 3 = $\sqrt{n}$

because there are three For loop inside each other

$\therefore$ Totally Complexity = $n * \sqrt{n} * \sqrt{n} = n * n = n^2$

Algoritha 2

```
int z=0;
int x=0;
For (int i=1; i<=n; i=i*3)
    { z=z+5;
      z++;
      x=2*x;
    }
```

Solution: the loop will end when $i=n$ But $i=3^k$

$i=1*3=3$ & $i=3*3=6$ & $i=3*3=9$

3^k

$\therefore 3^k=n \rightarrow \therefore k=\log_3 n$

$\therefore$ Complexity $o(\log n)$

Algorithm 3

```
int x=0;
For(int i=1; i<=n; i=i*3) → when stop when i=n & i=3^k (log_3 n)
    { iF (i%2 !=0)
        For (int j=0; j<i; j++) → when stop when j=i (n)
            x++;
    }
```

Solution: $(n) * (\log_3 n) = n\log n$

Complexity $= o(n \log(n))$

Algorithm 4

```
int Fun (int n)                                  → T(n)
{   int Count = 0;
    For (int i = n; i > 0; i /= 2)               → i = i/2 (or) i = i * 1/2
        For (int j = 0; j < i; j++)
            Count += 1;
    return count;
}
```

Solution : outer loop (will stop when $i = n$) $< i = 2^k$

initial $i_1 = 8 = 2^3$

$i_2 = 8/2 = 4 = 2^2$

$i_3 = 4/2 = 2 = 2^1$

$i_4 = 2/2 = 1 = 2^0$

$\therefore 2^k = n \quad \therefore k = \log_2 n \quad \therefore \Rightarrow$ Complexity $(\log n)$

inner loop : (will stop when $j =$ [i] variable)

$\therefore$ Complexity $= O(n)$

Totally Complexity $n * \log n =$ **$n(\log n)$**

Algorithm 5

```
int n, rev;
rev = 0;
while n > 0
{   rev = rev * 10 + n % 10;   (variable not intializing in condition)
    n = n / 10;   → important
}
```

Solution : $n = n/10$ Assume $n = 1000$

$n_1 = 1000/10 = 100 = 10^2$

$n_2 = 100/10 = 10 = 10^1$

$n_3 = 10/10 = 1$

$\therefore 10^k = n$

$k = \log_{10}^n$

$\log n$

Complexity = $O(\log n)$

Algorithm 6

```
int Fun1 (int n)
{ int i, j, p, q = 0;
  For(i=1; i<n ; ++i)  ------> (n)
  {
      p=0;
      For(j=n; j>1; j = j/2) -> (log2 n)
          ++p;
      For(k=1 ; k<p; k=k*2) -> log2 n
          ++q;
  }
  return q+i;
}
```

Solution: Complexity For outer loop = o(n)

Complexity For inner loop = o(log n)

$n(\log(n) + \log(n) = n(\log(n)) + n(\log(n)$

Totally complexity = $o(n(\log(n))$

Question 2 : Solve recurrence relation (use Recurrence Tree)

[1] $T(n) = T(n-1) + 5$ $\quad T(1) = 1$

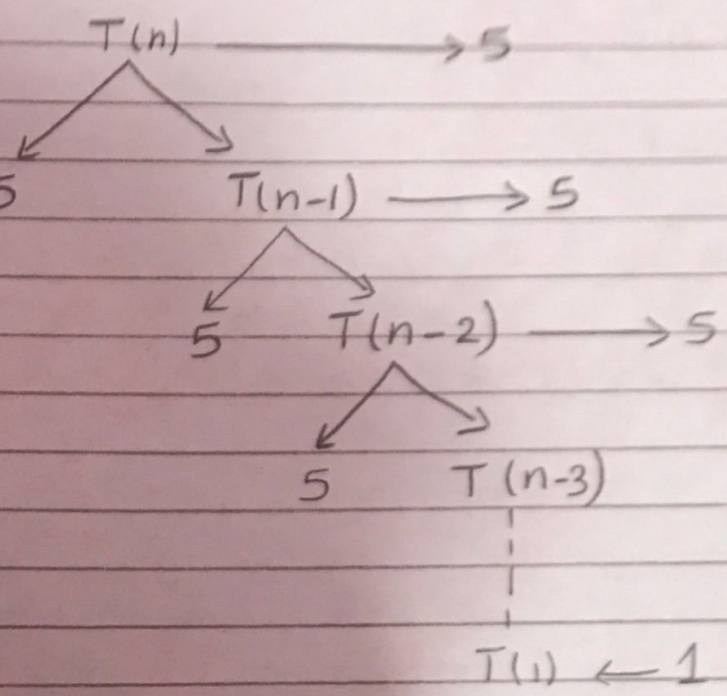
بنعوض الأصلية للفوق

$T(n-1) = T(n-1-1) + 5$

$T(n-1) = T(n-2) + 5$

$T(n-2) = (n-2-1) + 5$

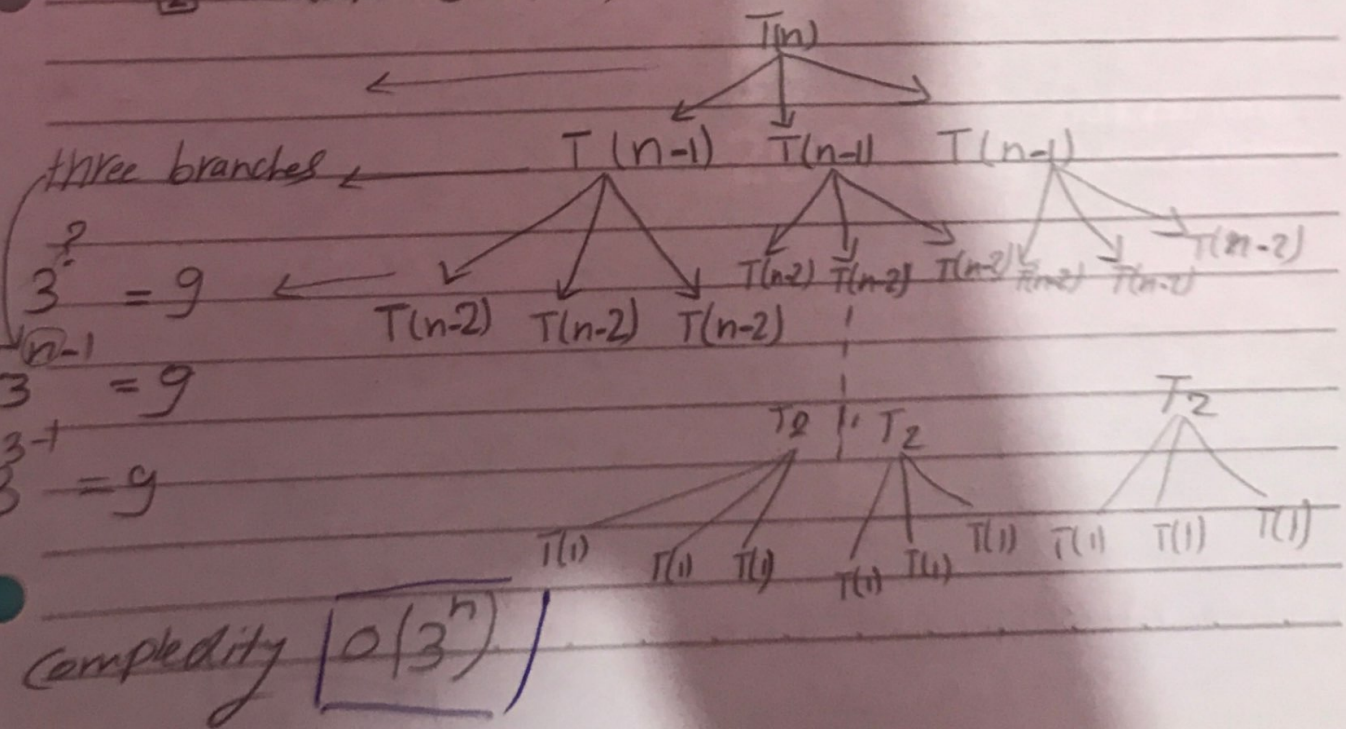
$T(n-2) = (n-3) + 5$



$T(n) = 5 * n + 1$

$\therefore O(n)$

[2] $T(n) = 3T(n-1)$



complexity $O(3^n)$

Scanned with CamScanner

PAGE
DATE

③ $T(n) = T(n/2) + n$

$a = 1$, $b = 2$, $F(n) = n$

$F(n) = n^k (\log n)^P = n$

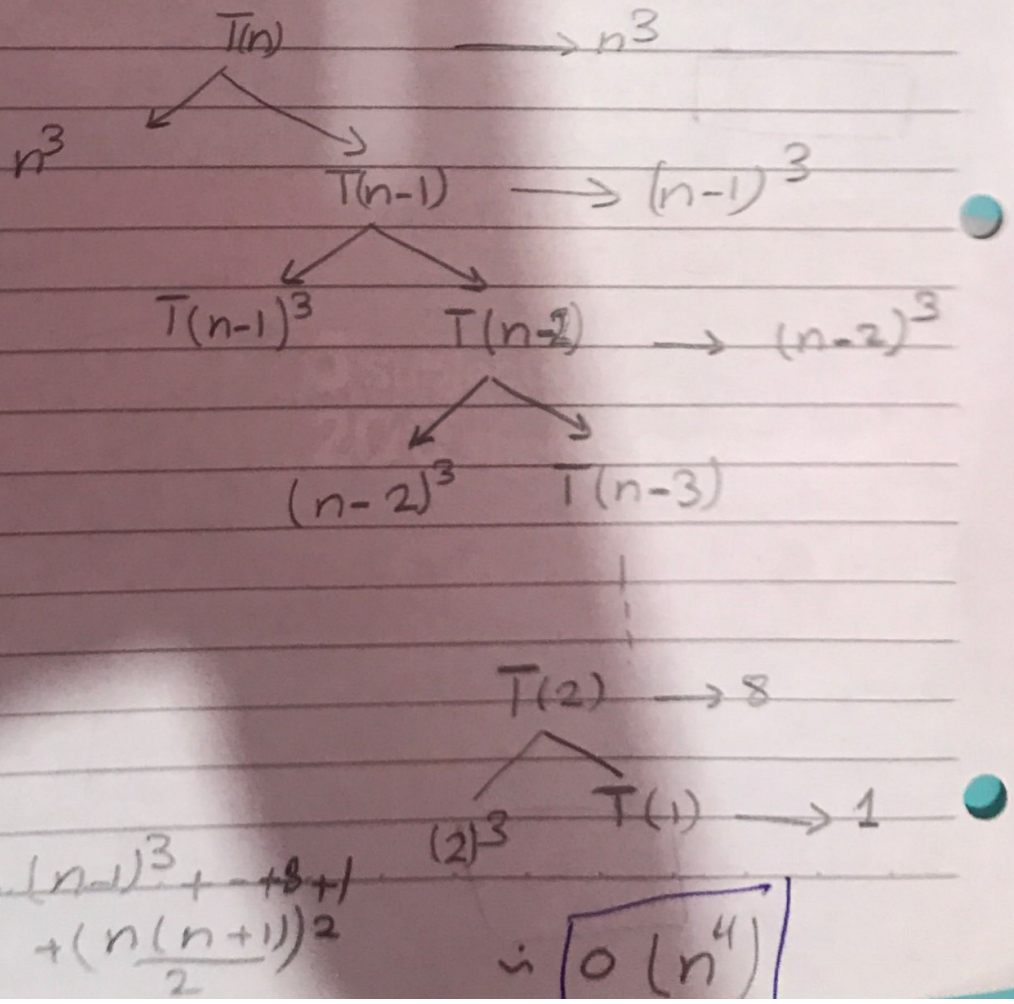
$\therefore k = 1$, $P = 0$

$\log_b^a = \log_2^1 = 0$, $k = 1$

$\therefore \log_b^a = 0 < k$ $\therefore O(n^k \log n^P) = O(n^1 (\log n)^0)$

$\therefore O(n)$

④ $T(n) = T(n-1) + n^3$



$T(n) = n^3 + (n-1)^3 + \cdots + 8 + 1$

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

$\therefore \boxed{O(n^4)}$

PAGE
DATE

5. $T(n) = 4T(n/3) + n$

$a = 4 \quad b = 3 \quad F(n) = n$

$F(n) = n^k (\log n)^p = n \quad \therefore k = 1 \text{ & } p = 0$

$\log_b^a = \log_3^4 = 1.26 > k$

$\therefore O(n^{(\log_b^a)}) = \boxed{O(n^{\log_3^4})}$

6. $T(n) = 2T(n/2) + \log n$

$a = 2 \quad b = 2 \quad F(n) = \log n$

$F(n) = (n^k (\log n)^p) = \log n \quad \therefore k = 0 \text{ & } p = 1$

$\log_b^a = \log_2^2 = 1 > k$ ∴

$\therefore O(n^{\log_b^a}) = O(n^{(\log_2^2)}) \therefore \boxed{O(n)}$

7. $T(n) = 2T(n/2) + n^2$

$a = 2 \text{ & } b = 2 \text{ & } F(n) = n^2$

$F(n) = (n^k (\log n)^p) = n^2 \quad \therefore k = 2 \text{ & } p = 0$

$\log_b^a = \log_2^2 = 1 < 2 \quad \therefore O(n^k \log n^p)$

$O(n^2 \log n^0) = O(n^2)$

8) $T(n) = 2T(n/4) + \sqrt{n}$

$a=2$, $b=4$, $F(n) = \sqrt{n}$

$F(n) = n^k \log_n^p = \sqrt{n}$ $\quad \therefore k = \frac{1}{2}$, $p=0$

$\log_b^a = \log_4^2 = 0.5 = k$ $\quad \therefore O(n^k \log n^{p+1})$

$O(n^{0.5} \log n^{0+1}) = \boxed{O(n^{0.5} (\log n))}$

Question 3 ⇒ find time complexity use Back-substitution

Algorithm 1

```
Public Static int F (int n)  ------> T(n)
    iF (n = =1)
    { return 1;}
    else
    { return f(n-1) + f(n-1) ; }  ->T(n-1)+T(n-1)
                  |
                  -> 1
}
```

using Back substitution

$T(n) = 2T(n-1) + 1 \longrightarrow I$

$T(n-1) = 2T(n-1-1) + 1 = 2T(n-2)+1 \longrightarrow II$

$T(n-2) = 2T(n-2-1) + 1 = 2T(n-3)+1 \longrightarrow III$

Substitute II in (I)

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

Substitute III in (IV)

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

⑧ Continue

$T(n) = 2^k T(n-k) + 2^{k-1} + 2^{k-2} + \cdots + 2^2 + 2^1 + 2^0$

$T(0)$: $n-k=0$

$\therefore n=k$

$T(n) = 2^n T(n-n) + 2^{n-1} + 2^{n-2} \cdots + 2^2 + 2^1 + 2^0$

$T(n) = 2^n \cdot T(0) + 2^{n-1} + 2^{n-2} + \cdots + 2^2 + 2^1 + 2^0$

$T(n) = 2^{n+1} - 1$ $\quad \therefore O(2^n)$

Algorithm 2

```
Public Static int F(int n)  ⟶ T(n)
{   if (n==1)
    { return 1; }
    else
    {
        int y = F(n/2) + F(n/2); ⟶ (n/2)+(n/2)
        For (int i=0; i<n; i++) ⟶ n
        { For (int j=0; j*j<n; j++) ⟶ j²=n
                                      j=√n
            { y = y+1; }
        }
        return y;
    }
}
```

Solution: $\therefore T(n) = T(n/2) + T(n/2) + n\sqrt{n}$

$\therefore T(n) = 2T(n/2) + n\sqrt{n}$

$a=2$, $b=2$, $F(n)=n\sqrt{n}$

$\therefore F(n)=O(n^k(\log n)^p)=n\sqrt{n}$ $\quad \therefore k=\frac{3}{2}, P=0$

$\log_b^a=\log_2^2=1<k$ $\therefore O(n^k \log n^p)$

$O(n^{3/2}\log n^0)=O(n^{\frac{3}{2}})=\boxed{O(n\sqrt{n})}$

Algorithm 3

```
Public static int F(int n) ---> T(n)
    int result;
    if (n==1)
    { result = 1; }
    else
    { result = 0;
      For (int i=0 ; i<n ; i++) ---> (n)
      { result = result + f(n-1) ---> (n-1)
      }
    }
}
```

Solution : $T(n)= nT(n-1)$

$T(n)= nT(n-1) \longrightarrow I$

$T(n-1)=(n-1)T(n-2) \rightarrow II$

$T(n-2)=n-2)T(n-3) \rightarrow III$

Subsitute (II) in(I)

$T(n)= n[(n-1)T(n-2)] \rightarrow IV$

Subsitute (III) in(IV)

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

$T(n)= n(n-1)(n-2)(n-3) \cdots T(n-(n-1))$

$n(n-1)(n-2)(n-3) \cdots T(n-n+1) \cdots T(1)$

$\therefore T(n)=\boxed{O(n!)}$