Kunal Dhami
CST SPL I

Tutorial 2

(1)

```
void func (int n)
{ int j = 1, i = 0;
  while (i<n)
  {
     i += j;
     j++;
  }
}
```

→ for $j=1$ $i=1;$
$j=2$ $i=1+2;$
$j=3$ $i=1+2+3;$

$\Bigg]$ m levels

$\therefore$ $1+2+3+ \dots < n$

$1+2+3+m < n$

$\frac{m(m+1)}{2} < n$

$m \approx \sqrt{n}$

by summation method

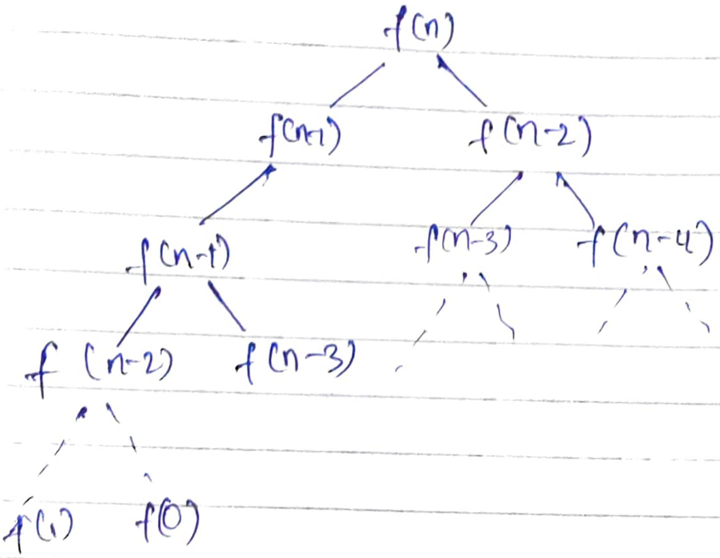
$\Rightarrow \sum_{i=1}^{m} 1 \Rightarrow 1+1+ \dots \sqrt{n}$ times

$\therefore T(n) = \sqrt{n}$

(2) for fibonacci series :-

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

$f(0)=0$ $f(1)=1$



∵ at every function call we get two function calls
for n levels:.

we have ⇒ 2×2 - - - - n times

∴ $T(n) = 2^n$

Maximum space :- considering recursive stack ∴ no of calls max. = n

For each call we have space complexity $O(1)$

∴ $T(n) = O(n)$

(3) n logn :-

quick sort

```
void func (int arr[], int l, int h)
 {
   if (l<h)
   { int pi= portion (arr, l, h)
   func (arr, l, pi-1);
   func (arr, pi+1, h);
   }
 }
```

```
int paotion (int arr[], int l , int h)
  { int pi = arr [h];
    int i = (l-1);

for (int j = l; j<=h; j++)
  { if (arr [i] <pi)
     { i++;
       swap (arr [i] , arr[j]);
     }
  }.
   swap (arr [i+1], arr [h]);
   return (i+1);
}
```

(b) n^3 -

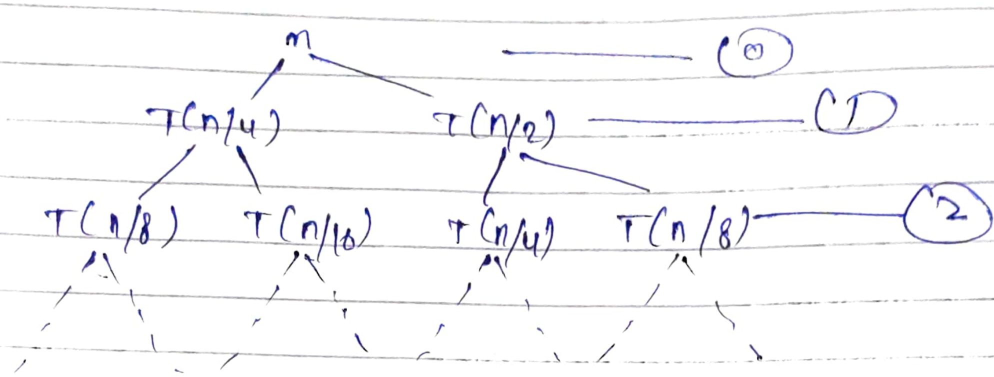
Multiplication of Two Square Matrix

```
for (i=0; i<n; i++)
   { for (j=0; j<c2; j++)
      { for (k=0; k<c1; k++)
       { res [i] [j]+= a[i] [k] * b[k] [j];
```

(c) Log (log n)

```
for (i=2; i<n; i= i*i)
  { c++
  }
```

(4) $T(n) = T(n/4) + T(n/2) + Cn^2$



At level → 0 → Cn^2

1 → $\frac{n^2}{4} + \frac{n^2}{2^2} = \frac{C5n^2}{16}$

2 → $\frac{n^2}{8^2} + \frac{n^2}{16^2} + \frac{n^2}{4^2} + \frac{n^2}{8^2} = \left(\frac{5}{16}\right)^2 n^2 C$

⋮

max level $= \frac{n}{2^K} = 1$

$\Rightarrow K = \log_2 n$

$\because T(n) = \left(Cn^2 + \left(\frac{5}{16}\right)n^2 + \left(\frac{5}{16}\right)^2 \dots + \left(\frac{5}{N}\right)^{\log n} n^2\right)$

$T(n) = Cn^2 \left[1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \dots \left(\frac{5}{16}\right)^{\log n}\right]$

$T(n) = O(n^2 C) \Rightarrow O(Cn^2)$

(5) int fun (int n)

```
{
for (i=1; i<=n; i++)
{ for (j=1; j<n; j+=i)
   {
   }
}
}
```

for →

i	j
1	1
2	1+3+5
3	1+4+7
4	1+5+9
⋮	⋮
n	⋮

$$\sum_{i=1}^{n} \frac{(n-1)}{i}$$

$$\therefore T(n) = \frac{(n-1)}{1} + \frac{(n-1)}{2} + \frac{(n-1)}{3} + \cdots + \frac{(n-1)}{n}$$

$$T(n) = n\left[1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}\right] - 1\times n\left[1+\frac{1}{2}+\frac{1}{3}+\cdots\frac{1}{n}\right]$$

$$= n\log n - \log n$$

$$\therefore T(n) = O(n\log n)$$

(6) for (i=2; i<=n; i=pow(i,k))

```
{ O(1)
}
```

for ⇒ i

2^1

2^K

$2K^2$

$2K^3$

|

$2K^m$

where, $2^{K^m} <= n$

$K^m = \log_2 n$

$m = \log_K \log_2 n$

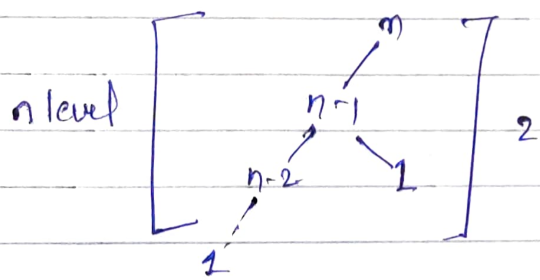
$\therefore \sum_{i=1}^{m} 1$

$\Rightarrow 1 + + + \ldots$ m times

$\Rightarrow T(n) = O(\log_k \log n)$

(7) Given algo divides array in 99% & 1% part

$\therefore T(n) = T(n-1) + O(1)$



'n' work is done at each level for merging

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

$= n \times n$

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

Lowest higher $= 2$

height higher $= n$

$\therefore$ diff $= n-2$ $\because (n>1)$

(8) considering for large values of 'n'

a) $100 < \log\log n < \log_2 n < (\log n)^2 < \sqrt{n} < n < n\log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2}$

b) $1 < \log\log n < \sqrt{\log n} < \log n < \log 2n < 2\log n < n < n\log n < 2n < 4n < \log(n!) < n^2 < n! < 2\log 2n < 5n$

c) $96 < \log_8 n < \log_2 n < 5n < n\log_6 n < n\log_2 n < \log(n!) < 8n^2 < 7n^3 < n! < 8^{2n}$