NAME: Aastha Verma

SECTION: I

ROLL NO: 12

[Tutorial - 2]

Q1 What is the time complexity of below code and how?

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

⇒ $j = 1 \quad i = 1$

$j = 2 \quad i = 1 + 2$

$j = 3 \quad i = 1 + 2 + 3$

] m - level

for (i)

$\because 1 + 2 + 3 + \ldots + < n$

$\because 1 + 2 + 3 + m < n$

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

$m \approx \sqrt{n}$

By summation method

$$\sum_{i=1}^{m} 1 \Rightarrow 1 + 1 + \ldots + \sqrt{n} \text{ terms}$$

$$\boxed{T(n) = \sqrt{n}}$$ Ans

Q2: Write recurrence relation for function that prints Fibonnacci series. Solve it to get the time complexity. What will be the space complexity & why?

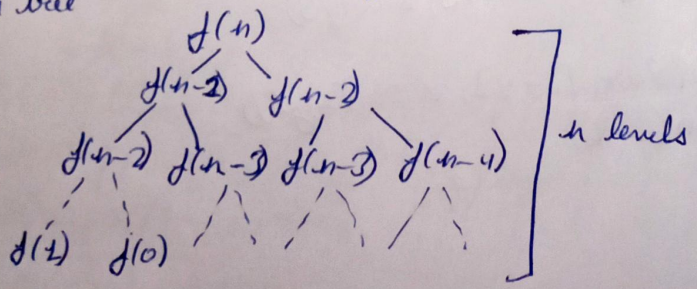
Ans: For Fibonnacci series

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

$f(0) = 0$

$f(1) = 1$

By forming a tree



∵ At every function call we get 2 function calls.

∴ for n levels

We have = 2×2 ..... n times

$\therefore \boxed{T(n) = 2^n}$

MAXIMUM SPACE Considers Recursive

stack: no. of calls maximum = n

For each call we have space complexity O(1)

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

Without considering Recursive stack:

each call we have time complexity O(1)

$\therefore \boxed{T(n) = O(1)}$

Q3: Write Programs which have complexity: n(log n), n^3, log(log n)

Ans: n log n → Quick sort

```
void quicksort (int arr[], int low, int high)
{ if (low<high)
   { int p = partition (arr, low, high)
     quicksort (arr, low, p-1);
     quicksort (arr, p+1, high);
   }
}
int partition (int arr[], int low, int high)
{ int pivot = arr[high];
  int i = (low-1);
  for (int j = low; j <= high-1; j++)
   { if (arr[j] < pivot)
      { i++;
        swap (&arr[i], &arr[j]);
      }
   }
  swap (&arr[i+1], &arr[high]);
  return (i+1);
}
```

2) n^3 → Multiplication of 2 square Matrix

```
for (i=0; i<n1; i++) {
    for (j=0; j<n2; j++) {
        for (k=0; k<c1; k++)
        {
            res[i][j] += a[i][k] * b[k][j];
        }
    }
}
```

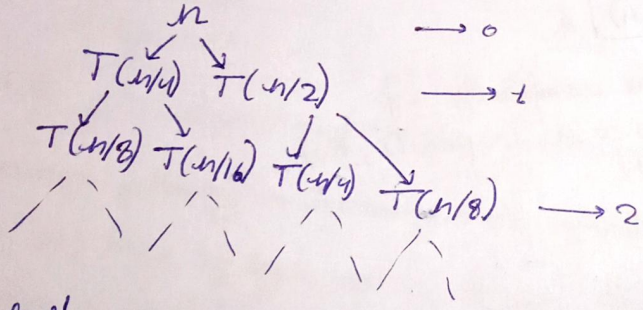
3) log(log n)

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

Q4: Solve the following recurrence relation

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

⇒



At level

$0 \rightarrow Cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = C\frac{5n^2}{16}$

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

⋮

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

$= k = \log_2 n$

$T(n) = C(n^2 + (5/16)n^2 + (5/16)^2 n^2 + \dots + (5/16)^{\log_2 n} n^2)$

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

$T(n) = Cn^2 \times \frac{11}{5} \times \left(1 - \left(\frac{5}{16}\right)^{\log n}\right)$.

$\boxed{T = O(Cn^2) = O(n^2)}$ ∎

Q5: What is the time complexity of following fun()?

```
int fun(int n){
  for(int i=1; i<=n; i++){
    for(int j=1; j<n; j+=i){
      // Some O(1) task
    }
  }
}
```

⇒ for

i	j
1	1
2	1+3+5
3	1+4+7
:	1+5+9.
n	

$j = (n-1)/i$ times

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

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

$$T(n) = n[1 + 1/2 + 1/3 + \ldots + 1/n] \Rightarrow 1\times[1 + 1/2 + 1/3 + \ldots + 1/n]$$

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

$$\boxed{T(n) = O(n\log n)}$$ A.

Q6: What should be time complexity of

```
for(int i=2; i<=n; i=pow(i,k))
  { // Some O(1)
  }
```

where k is a constant

⇒ for

i
2^{k^0}
2^{k^1}
2^{k^2}
2^{k^3}
:
2^{k^m}

where $2^{k^m} <= n$

$k^m = \log_2 n$

$m = \log_k \log_2 n$

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

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

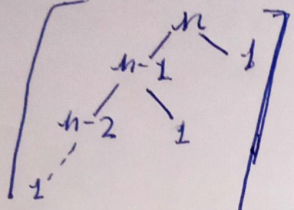
$$\boxed{T(n) = O(\log_k \log n)}$$ A

Q.7 Write a recurrence relation when quick sort repeatedly divides array into 2 parts of 99% and 1%. Derive time complexity in this case. Draw the recurrence tree while deriving time complexity & find complexity difference in heights of both extreme parts. What do you understand by this analysis?

⇒ Given algorithm divides array in 99% & 1% part.

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

n levels:



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

$\leq n \times n$

$\therefore \boxed{T(n) = O(n^2)}$

Lowest height → 2

Highest height = n

$\therefore \boxed{\text{difference} = n-2}$ $n > 1$

∴ The given algorithm produces linear result.

Q.8 Arrange following in increasing order of rate of growth:

a) $n, n!, \log n, \log\log n, \text{root}(n), \log(n!), n \log n, \log^2(n), 2^n, 2^{2^n}, 4^n, n^{100}$

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

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

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

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

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