TUTORIAL - 2

Name: Ananya Vishnoi
Course :- B.Tech CSE
Section :- F
Roll no :- 44
Uni Roll no :- 2016635

Ans 1

$j = 1 \quad i = 1$

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

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

fa(x)

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

$\therefore 1+2+3 \ldots m < n$

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

$m \simeq \sqrt{n}$

By Summation Method

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

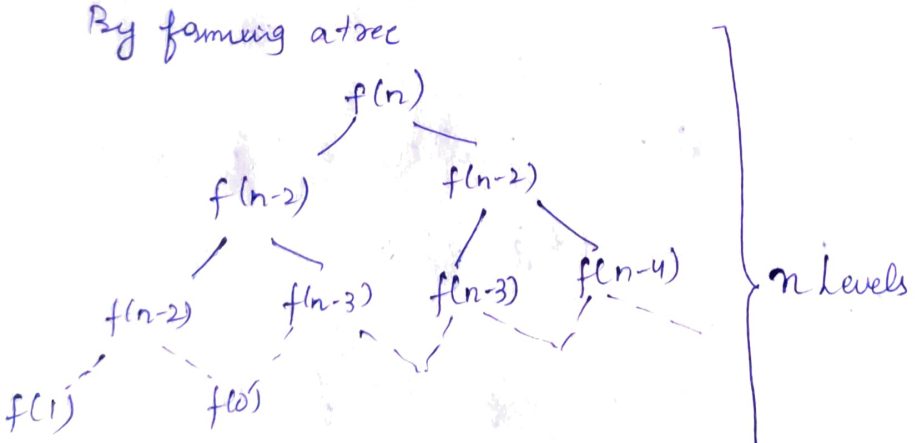
$[T(n) = \sqrt{n}]$

Ans 2

For fibonacci series

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

By forming a tree



At every function call we get two functions calls.

for n levels

we have $= 2 \times 2 \ldots n$ times

$\therefore T(n) = 2^n$

Maximum Space

Considering recursive stack:

no of calls maximum = n

For each call maximum, we have space complexity of O(1)

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

without considering recursive stack each call will have a time complexity of O(1).

$T(n) = O(1)$

Ans 3. $(n \log n)$ → Quicksort

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

```
swap(&arr[i+1] & arr[high]);
return (i+1);
}
```

n^3 → multiplication of a 2 square matrix

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

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

Ans-5

```
for   i      j
      1      1
      2      1+3+5
      3      1+4+7
      .      .
      .      .
      .      .
      n      1+5+7
```

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

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

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

$$T(n) = n\left[1/1 + 1/2 + 1/3 + \cdots + 1/n\right]$$

$$= n \log n \cdot \log n = O(n \log n)$$

Ans 6 for i

2^1

2^k

2^{k^2}

2^{k^3}

⋮

2^{k^m}

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

$k^m = \log_2 n$

$m = \log_k \log_2 n$

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

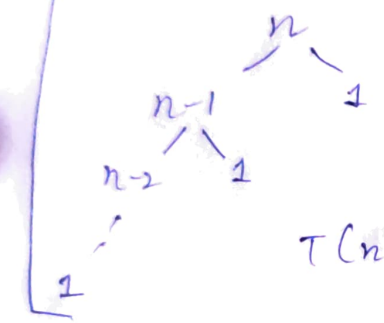
$1+1+1+\cdots m$ times

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

Ans 7 Given algo divides array in 99% and 1% part

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

n-levels



'n' work is done at each level.

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

$= n \times n$

$= n^2$

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

Lowest height = 2

highest height = n

difference = $n-2$ $\quad n>1$

The given algo produces linear Result

Ans 8

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

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

c) $96 < \log_8 n < \log 2n < 5n < n\log_6 n < n\log_3 n < \log(n_b)$
$< 8n^2 < 7n^3 < n_b < 8^{2n}$