1

Name - Suryansh Makhloga
2017563
Roll No - 22
Sec - CST SPL 1

Tutorial - 2

DAA

Q1)
```
void fun (int n)
{
  int j=1, i=0;
  while (i<n)
  {
    i+= j;
    j++;
  }
```

Sol) for

$j=1 \quad i=1$

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

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

for (i)

$1+2+3+ \ldots \propto n$

$1+2+3+ \ldots m \propto n$

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

$m \simeq \sqrt{n}$

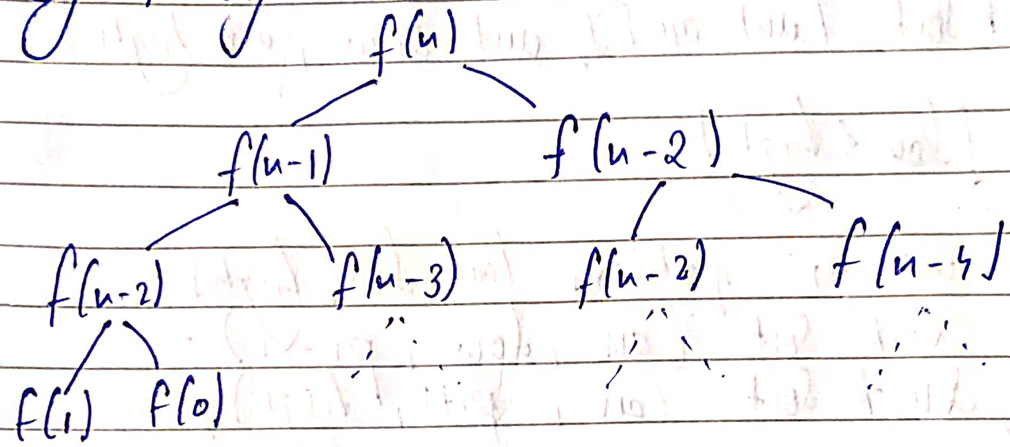
$\because$ By summation

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

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

Q2) for fibonacci series
$f(n) = f(n-1) + f(n-2)$

Sol) By forming tree




$\because$ At every func call we get 2 func call for n levels

$= 2 \times 2 \times \cdots n$ times

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

Max space:
Considering recursive stack
No of calls max $= n$.
for each call
space complexity $= O(1)$
$T(n) = O(n)$

without considering recursive stack:
Space complexity $= O(1)$
$T(n) = O(1)$

Q3] i] nlogn, n^3 log(logn)

2) Quicksort

```
void QuickSort (int arr [], int low, int high)
{
    if (low < high)
    {
        int pi = partition (arr, low, high);
        Quick Sort (arr, low, pi-1);
        Quick Sort (arr, pi+1, high);
    }
}
int partition (int arr [], int low, int high)
{
    int part = arr [high];
    int i = (low -1);

    for ( int j = low ; j <= high -1 ; j++)
    {
        if (arr [i] < part)
        {
            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 < r1; i++)
{
  for (j=0; j<c2; j++)
   for (k=0; k<c1; k++)
   {
     res[i][j] += a[i][k] * b[k] b[j];
   }
```

3) log (log n)

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

Q4) $T(n) = T(n/4) + T(n/2) + cn^2$

$cn^2$

$T(n/4)$ $T(n/2)$

further breaking $T(n/4)$ & $T(n/2)$

$cn^2$

$c(n^2)/16$ $c(n^2)/4$

$T(n/16)$ $T(n/8)$ $T(n/8)$ $T(n/4)$

Summation level by level

T(n) = c (n^2 + 5(n^2) /16 + 25 (n^2) /256)

GP ⇒ $r = \frac{5}{16}$

To get upper bound we can sum above series for infinite term

$$Sum = \frac{(n^2)}{1 - 5/16}$$

T(n) = O(n^2)

Q5) int fun (int n)

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

Sol) for i = 1        Inner loop (i)

| | |
|---|---|
| 1 | n |
| 2 | n/2 |
| ⋮ | |
| n | n/n |

Total time complexity = (n + n/2 + ... n/n)

$= n * (1 + \frac{1}{2} + \frac{1}{3} \ldots \frac{1}{n})$

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

Q6| Time complexity

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

Sol]

$i$
$\downarrow$

$2$

$2^{k}$

$2^{k^2}$

$\vdots$

$2^{k^{\log_k(\log(n))}}$

where

$2^{k^{\log_k(\log n)}} \leq n$ $[m = \log_k \log_2 n]$

$2^{\log n} \leq n$

$n \leq n$ Agree

So there are in total $\log_k(\log(n))$ many iterations & each iteration take constant amt of time.

$T(n) = 1 + 1 + \dots \log_k \log n$ times

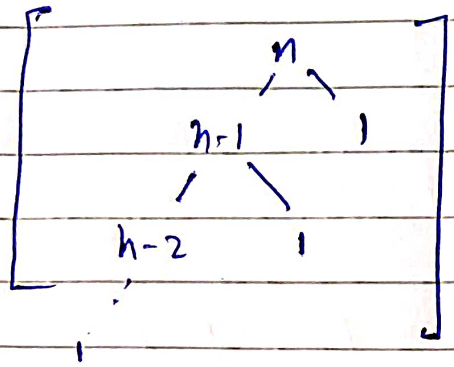
$T(n) = O(\log_k(\log(n)))$

Q7)

Sol) given algo divides array in 99 % & 1 %

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

n level



n work is done at each level for merging

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

$= n \times n$

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

Lowest Height = 2

Highest " = n

$\therefore$ Difference = n-2 $\quad n > 1$

The given algo provides linear result.

Q) Arrange following in increasing order of rate of growth

Sol) for large values of n:

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

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

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