DATE .....

Ayush Dhiman
(CSE) Section H
Roll No. 28

PAGE No .....

DAA Tutorial - 2

Ans 1

$j = 1 \quad i = 0 \quad \Rightarrow i = 1$

$j = 1 \quad i = 1$

$i = 1 + 1 = 2$

$j = 2 \quad i = 3$

$j = 3 \quad i = 3 + 3 = 6$

$j = k \quad i = 1 + 2 + 3 + 6 + \ldots + k$

as $i < n$

sum of k consecutive integers $= k\frac{(k+1)}{2}$

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

$$k^2 < n$$

$$k < \sqrt{n}$$

$$O(\sqrt{n})$$

Ans 2

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

Let $T(n-2) \approx T(n-1)$

$T(n) = 2T(n-2) + C$

$= 2[2T(n-4) + C] + C$

$= 4T(n-4) + 3C$

$= 8T(n-6) + 7C$

$T(n) = 2^k T(n-2k) + (2^k - 1)C$

$n - 2k = 0 \Rightarrow k = n/2$

$T(n) = 2^{n/2} T(0) + (2^{n/2} - 1)C$

$T(n) \propto 2^{n/2}$ (lower bound)

TEACHER'S SIGNATURE .....

DATE ..... PAGE No .....

Q1

```
void fun(int n) {                    Complexity
int j=1, i=0;                        1+1 = 2
while (i<n) {                        n
i = i+j;                             n
j++; } }                             n
```

Total complexity = $2+n+n+n = 3+n$

$= O(n)$

Q2 Recurrence relation

$$T_n = T_{n-1} + T_{n-2} + C$$

$$T_{n-1} \approx T_{n-2}$$

$$T_n = 2T_{n-2} + C$$

$$T_n = 2(2T_{n-4} + C) + C$$

$$T_n = 2.2T_{n-4} + C(1+2)$$

$$T_n = 2.2(2T_{n-6} + C) + C(1+2)$$

$$T_n = 2.2.2T_{n-8} + C(1+2+2^2)$$

$$T_n = 2^k T_{n-2k} + C(2^k - 1)$$

TEACHER'S SIGNATURE .....

DATE .....

PAGE No .....

Q1

```
void fun(int n) {                    Complexity
int j=1, i=0;                        1+1 = 2
while (i<n) {                        n
  i = i+j;                           n
  j++;                               n
} }
```

Total complexity $= 2+n+n+n = 3+n$

$= O(n)$

Q2 Recurrence relation

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

Let

$T(n-2) \simeq T(n-1)$

$T(n) = 2T(n-2) + C$

$= 2[2T(n-4) + C] + C$

$= 4T(n-4) + 3C$

$= 8T(n-6) + 7C$

$= 16T(n-8) + 15C$

$T(n) = 2^k T(n-2k) + (2^k - 1)C$

$n - 2k = 0 \Rightarrow k = n/2$

$T(n) = 2^{n/2} T(0) + (2^{n/2} - 1)C$

$T(n) \Omega 2^{n/2}$ (Lower bound)

TEACHER'S SIGNATURE .....

DATE .....

PAGE No .....

$T(n-1) \approx T(n-2)$

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

$= 4T(n-2) + 3c$

$= 8T(n-3) + 7c$

$= 2^k T(n-k) + (2^k - 1)c$

$n - k = 0 \Rightarrow k = n$

$T(n) = 2^n T(0) + (2^n - 1)c$

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

$T(n) \propto 2^n$ (upper bound)

Time Complexity = $O(2^n)$

Q3 =) $O(\log n)$

→ Binary search

→ Finding largest / smallest number in binary search tree.

=) $O(n \log n)$

→ Merge sort

→ heap sort

→ quick sort

TEACHER'S SIGNATURE .....

DATE .....

PAGE No .....

=) log(log n)

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

=) n^3

```
for(int i=0 ; i<n ; i++)
{ for(int j=0 ; j<n ; j++)
  { for(int k=0 ; j<k ; k++)
    {
        // O(1) expression
    }
  }
}
```

TEACHER'S SIGNATURE .....

DATE .....

PAGE No .....

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

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

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

$a = 2 \quad b = 2$

$n^{\log_2 2} \Rightarrow n$

$f(n) = c^2$

$f(n) \geq n$

$\therefore O(n^2)$

Q5

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

Time complexity $\sum_{i=1}^{n} \left(\sum_{j=1} \right.$

For $i=1$ the inner loop is executed n times

For $i=2$ the inner loop is executed $n/2$ times

For $i=3$ the inner loop is executed $n/3$ times

For $i=n$ the inner loop is executed n/n times

TEACHER'S SIGNATURE .....

DATE .....

PAGE No .....

Total time complexity $= n + n/2 + n/3 + \cdots + n/n$

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

$= O(n \log n)$

Q6

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

For $i = 2$

For $i = 2^k$

For $i = 2^{k^2}$

For $i = 2^{k^3}$

$\vdots$

For $i < n$

2^k

$(2^k)^k \Rightarrow 2^{k^2}$

2^{k^3}

2^{k^4}

2^{k^x}

$2^{k^x} < n$

Taking log

$k^x \log_2 2 < \log n$

Taking log again

$\log k^x < \log(\log n)$

$x < \frac{\log(\log n)}{\log k}$

TEACHER'S SIGNATURE .....

DATE .....

PAGE No .....

Removing constants

$x < \log(\log(n))$

Time complexity = $O(\log(\log n))$

Ans 7 When the given array is already sorted and you are taking the 1st or last element as key element then it is the worst case of quicksort

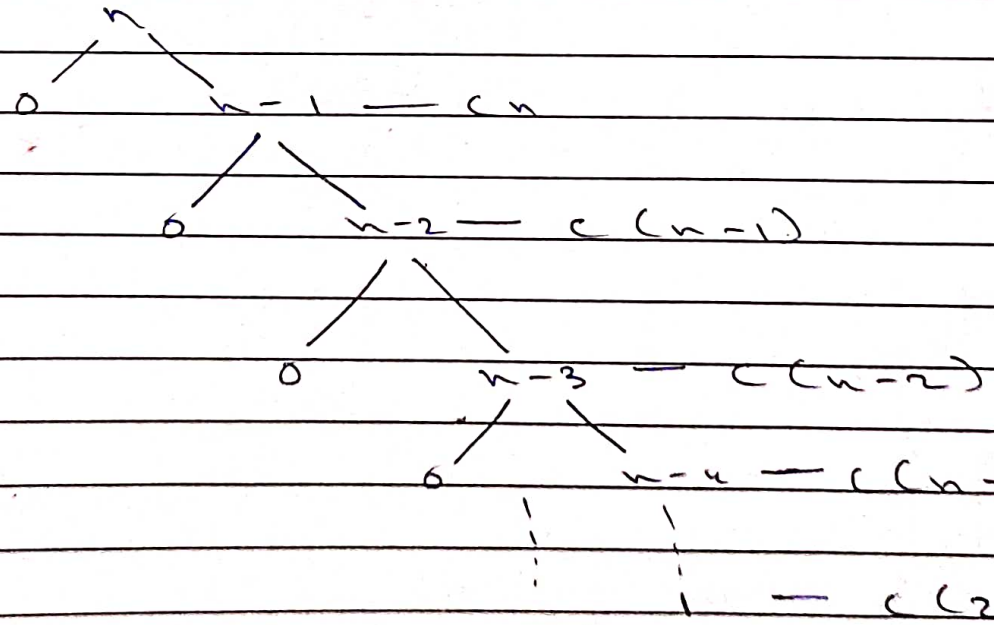
$T(n) = T(n_1) + T(n_2) + cn$

For worst case

$n_1 = 0 \quad n_2 = n-1$

$T(n) = T(0) + T(n-1) + cn$

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



TEACHER'S SIGNATURE .....

DATE ..... PAGE No .....

c → constant

$c\left[n+(n-1)+(n-2)+(n-3)+\ldots+1\right)$

$c\ \frac{n(n+1)}{2}$

Time complexity = $O(n^2)$

Ans 8 a) $1 < \log(\log n) < \log n < \log^2 n < \sqrt{n} < n < n\log n < n^2 < 2^n < 4^n < 2^{2n} < \log(n^2) < n!$

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

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

TEACHER'S SIGNATURE .....