Tutorial-2
DAA

Name - Ayushman Bhatt
Section - AI&DS
Class Roll No. - 21
University Roll No. - 2017640

Answer 1

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

$j=1 \rightarrow i=0+1$

$j=2 \rightarrow i=0+1+2$

$j=3 \rightarrow i=0+1+2+3$

⋮ (n or k times)

Loop ends when $i>=n$

$\Rightarrow 0+1+2+3+\ldots+k > n$

$\Rightarrow \frac{k(k+1)}{2} > n \Rightarrow k^2 > n$

$\Rightarrow k > \sqrt{n} \quad \therefore \boxed{T(n) = O(\sqrt{n})}$

$\therefore$ Time Complexity $= O(\sqrt{n})$

Answer 2 Recurrence Relation for Fibonacci Series

$$T(n) = T(n-1) + T(n-2) \quad \text{also } T(0) = T(1) = 1$$

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

(Lower Bound) $T(n) = 2T(n-2) = 2[2T(n-4)] = 4T(n-4)$

$= 4(2T(n-6)) = 8T(n-6)$

$= 8(2T(n-8)) = 16(n-8)$

⋮

$T(n) = 2^k T(n-2k)$

$\Rightarrow n - 2k = 0 \Rightarrow n = 2k \Rightarrow \boxed{k = n/2}$

$\therefore T(n) = 2^{n/2} T(0) = 2^{n/2} \Rightarrow \boxed{T(n) = \Omega(2^{n/2})}$ ← Lower Bound

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

(Upper Bound) $T(n) = 2T(n-1) = 2(2T(n-2)) = 4T(n-2) = 4T(2T(n-3))$

$= 8T(n-3) = 2^k T(n-k)$

$\therefore n - k = 0 \Rightarrow \boxed{k = n}$

$T(n) = 2^k \times T(0) = 2^n$

$\Rightarrow \boxed{T(n) = O(2^n)}$ ← Upper Bound

Answer 3

- **$O(n(\log n))$** ⇒

```
for(int i=0; i<n; i++)
{ for (int j= 1; j<n; j=j*2)
  {
      // same O(1)
  }
}
```

- **$O(n^3)$** ⇒

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

- **$O(\log(\log n))$** ⇒

```
for (int i=1; i<=n; i=i*2)
{   for (int j=1; j<=n; j=j*2)
    {
        // some O(1)
    }
}
```

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

→ Lets assume $T(n/2) >= T(n/4)$

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

Applying Master's Theorem → $\boxed{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$

$a = 2, b = 2 ; f(n) = n^2$

$C = \log_b a = \log_2 2 = 1$

$n^c = n$

Comparing n^c and $f(n) = n^2$

$f(n) > n^c$ so, $\boxed{T(n) = \Theta(n^2)}$

Time Complexity $= \Theta(n^2)$

Answer 5

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

$i=1$ — $j=1, j=2, j=3, \ldots, j=n$ — n times

$i=2$ — $j=1, j=3, j=5, j=7$ — Loop ends when $j>n$
$1+3+5+7+\ldots k>n$
$k>n/2$
— n times

$i=3$ — $j=1, j=4, j=7$ — $1+4+7>n$
$k>n/3$

$i=4$ — $k>n/4$

$\vdots$

$i=n$ — $k>1$

$\therefore$ So total Time Complexity
$= O(n^2+n^2+n^2+\ldots)$

Total Time Complexity $= O(n^2)$

Answer 6

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

Complexity of Pow (i,k) — $O(\log N) = \log(k)$

Sequencing $\rightarrow 2, 2^k, (2^k)^2, (2^k)^3, (2^k)^4 \ldots\ldots (2^k)^M$ (no. till the statement is going to be executed)

Generalising $\rightarrow 2^{k^0}, 2^{k^1}, 2^{k^2}, 2^{k^3}, 2^{k^4}, \ldots\ldots 2^{k^M}$

Assumption $\rightarrow$ Let $(M+1)$ be the total no. of terms. $\therefore$ the statement in the loop will be executed M times.

Loop ends when ~~i<n~~ $i>n$:-

$\Rightarrow 2^{k^M} > n \Rightarrow \log(2^{k^M}) > \log n$

$\Rightarrow k^M(\log 2) > \log n \Rightarrow k^M > \log n$

$\Rightarrow \log(k^M) > \log(\log n) \Rightarrow M \log k > \log(\log(n))$

$\Rightarrow M > \dfrac{\log(\log n)}{\log(k)}$ $\therefore$ $\boxed{T(n) = O(\log(\log n))}$

Answer 8

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

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

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