Design and Analysis of Algorithm

Tutorial - 1.

PIYUSH CHAUHAN
CST SPL 1
34

(11/3/22)

Ans 1) Asymptotic notations are used to find the complexity of an algorithm when input is very large.

- Big oh (O): $f(n) = O(g(n))$

  iff $f(n) \leq c \cdot g(n)$
  for all $n > k$ and
  for some constant $c > 0$
  $g(n)$ is "tight upper bound" of $f(n)$

- Big omega (Ω): $f(n) = \Omega(g(n))$

  iff $f(n) \geq c \cdot g(n)$
  $\forall\ n \geq k$
  for some constant $c > 0$
  $g(n)$ is "tight lower bound" of $f(n)$

- Big theta (Θ): $f(n) = \Theta(g(n))$

  iff $c_1 g(n) \leq f(n) \leq c_2 g(n)$
  $\forall\ n \geq \max(n_1, n_2)$
  for some constant $c_1 > 0$ and $c_2 > 0$
  $g(n)$ is both "tight upper bound" and tight lower bound" of $f(n)$.

Answer 2: `for (i=1 to n) { i = i*2; }`

1, 2, 4, 8 ... n.

Let k^{th} term $= n$

$n = 1 \cdot (2^{k-1})$

taking log on both sides.

$\log n = (k-1)\log_2 2$

$k = \log n + 1$

$O(\log n + 1)$

$O(\log n)$ = Ans.

Ans 3) $T(n) = 3T(n-1)$ — ①

putting $n = n-1$ in eq^n (1)

$T(n-1) = 3T(n-2)$ — ②

put (2) in (1)

$T(n) = 9T(n-2)$

putting $n = n-2$ in eq^n (1)

$T(n-2) = 3T(n-3)$ — ③

$T(n) = 27T(n-3)$

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

$n-k = 0$

$n = k$

$T(n) = 3^n T(n-n)$

$= 3^n T(0)$

$= 3^n$

$O(3^n)$ Ans

Ans 4) $T(n) = 2T(n-1)$ — ①

$n = n-1$ in eq ①

$T(n-1) = 2T(n-2)$ — ②

$T(n) = 4T(n-2)$ — ③

putting $n = n-2$ in eq^n ①

$T(n-2) = 2T(n-3)$ — ④

$T(n) = 8T(n-3)$

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

$n-k = 0$

$k = n$

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

$= 2^n T(0)$

$= 2^n$

$O(2^n)$ = Ans

Ans 6)

```
void function (int n)
{ int i count = 0;
  for (i = 1; i*i <= n; i++)
      count++;
}
```

(annotations: $i = 1$ → ①; $i*i <= n$ → $\sqrt{n}$; $i++$ → $\sqrt{n}$; count++ → $\sqrt{n}$)

complexity : $O(1 + \sqrt{n} + \sqrt{n} + \sqrt{n})$

$O(1 + 3\sqrt{n})$

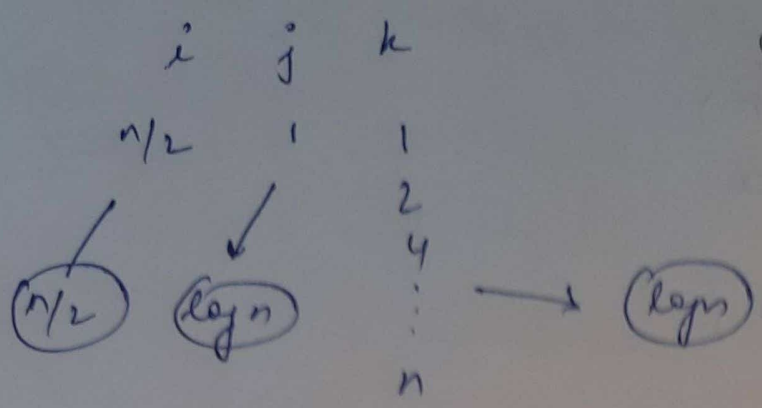
$O(3\sqrt{n})$

$O(\sqrt{n})$

$O(n^{1/2})$ Ans

Ans 7:

```
void function (int n)
{ int i, j, k, count = 0;
  for (i = n/2 ; i <= n ; i++)
  for (j = 1; j <= n ; j = j * 2)
  for (k = 1 ; k <= n; k = k * 2)
      count++;
}
```

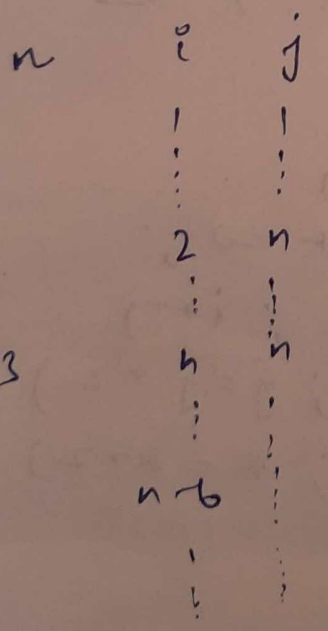


$O(n/2 * \log n * \log n)$

$O(n (\log n)^2)$ Ans

Ans 8:

```
function (int n)
{ if (n==1)
    return;
  for (i=1 to n)
  { for (j=1 to n)
    { printf ("*");
    }
    function (n-3);
  }
```



complexity

$1+4+7+\dots n$

$n = 1+3(k+1)$

no. of terms $= 3k-2$

$K = n+n/2$

$= n+2/6\,[\,2+(n-1/3)\times 3\,]$

$= [\,n+2/6\,(n+1)\,] \times n^2$

$= O[(n^2+3n+2)/6 \times n^2]$

$= O(n^4)$ Ans

Ans 4:

```
void function (int n)
{ for (i=1 to n)                          (n)
  { for (j=1; j<=n; j=j+1)                (n^2)
      printf ("*");                       (n^2)
  }
}
```

$O(n+n^2+n^2+n^2)$

$O(3n^2+n)$

$O(n^2)$ Ans

Ans 5:

```
int i=1; s=1;
while (s<=n)
{ i++; s=s+i;
  printf ("#");
}
```

$i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \ldots$

$S = 1 + 3 + 6 + 10 + 15 + \ldots + n$

sum of $S = 1 + 3 + 6 + 10 + \ldots + n$ — ①

also $S = 1 + 3 + 6 + 10 + \ldots + n-1 + n$ — ②

from ① − ②

$0 = 1 + 2 + 3 + 4 + \ldots \; n - n$

$T_k = 1 + 2 + 3 + 4 + \ldots k$

$T_k = 1/2 \; k(k+1)$

for k

$1 + 2 + 3 + \ldots + k <= n$

$k(k+1)/2 \leq n$

$(k^2 + k)/2 \leq n$

$O(k^2) <= n$

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

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

$O(n^{1/2})$ Ans