DATE
PAGE

Assignment - 1

Design and Analysis of Algorithm.

Q1

Ans1: Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input trends towards a particular value or limiting value.

There are 3 types of Notations:

i) Big-Oh
ii) Big-Omega
iii) Theta

Some examples = $O(n)$, $O(\log n)$ etc.

Q2:

Ans2:
```
for(i=1 to n)
  {
    i=i*2;
  }
```
$= O(\log n)$

Q3:

Ans3: $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

Masters Theorem for decreasing function:

$T(n) = aT(n-b) + f(n)$

Assumption $a>0$, $b>0$ and $f(n) = n^k$ where $k \geq 0$.

if $a = 1$ then $O(n * f(n))$
if $a > 1$ then $O(n^k a^{n/b})$
if $a < 1$ then $O(n^k)$.

$\therefore$ Acc to masters method:

as $a > 1$

$\therefore$

Time required $T(n) = O(3^n)$.

Q 4:

Ans 4: Acc to masters Theorem:

$T(n) = O(1)$

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

$T(n) = 2(2T(n-2) - 1) - 1$
$T(n) = 2^2 T(n-2) - 2 - 1$
$T(n) = 2^2(2T(n-3) - 1) - 2 - 1$
$T(n) = 2^3 T(n-3) - 2^2 - 2 - 1$
Similarly for k steps:

$= 2^k T(n-k) - 2^{k-1} - 2^{k-2} \quad - 2^2 - 2^1 - 2^0$

$T(1) = 1$, $n-k = 1$ $k = n-1$

Substituting $k = n-1$

DATE
PAGE

$T(n) = 2^{n-1} T(1) - [2^0 + 2^1 + \_ \_ \_ + 2^{n-2}]$

$= 2^{n-1} \times 1 - [2^{n-1} - 1]$

$= 2^{n-1} - 2^{n-1} + 1$

$T(n) = 1$

$\therefore T(n) = O(1)$.

Q5:

Ans5: $T(n) = O(\sqrt{n})$.

Q6:

Ans6: $T(n) = O(\sqrt{n})$

Ans7: $T(n) = O(n \log^2 n)$

Ans8: Recurrence Relation:

$T(n) = T(n-3) + n^2$

$T(n) = O(n * n^2)$ Acc to masters

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

Ans9) $T(n) = O(n \log n)$

Ans11) $T(n) = O(\sqrt{n})$

DATE
PAGE

Ans 15 $T(n) = O(n\log(n))$

Ans 16 $T(n) = O(\log\log(n))$

Ans 18 i) $100 < \log\log n < \log n < \log(n!) < \cancel{\text{root } n} < n$
$n\log n < n! < 2^n < 2^{2n} < 4^n$

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

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

Ans 19
```
for (i = 0 to n) {
    if (a[i] == key)
        return true
}
return false;
```

Ans 20 Bubble Sort $= O(n^2)$
Insertion Sort $= O(n^2)$
Selection Sort $= O(n^2)$
Quick Sort $= O(n\log n)$
Merge Sort $= O(n\log n)$

Ans 22 Inplace → Bubble, Quick, Selection, Insertion
Not Inplace → Merge Sort

Stable → Merge, Insertion, Bubble,
Not Stable → Quick, Selection.

DATE
PAGE

Online → Insertion
Offline → Selection, Quick, Merge, Bubble.

Ans23: int mid = (low+high)/2

```
while (low < high)
{
    if (a[mid] == key)
        return true;
    if (a[mid] < key)
        low = mid+1;
    if (a[mid] > key)
        high = mid-1;
}
return false.
```

```
Binary (int a[], int low, int high)
{
  mid = (low+high)/2;
  if (low < high)
  {
     if (a[mid] == key)
        return true;
     if (a[mid] < key)
        Binary(a, mid+1, high);
     if (a[mid] > key)
        Binary (a, low, mid-1);
  }
  return false;
}
```

DATE
PAGE

Ans24: $T(n) = T(n/2) + 1$

Q13

Ans13: ⇒

```
void sum(int n) {
    int i = 0, j = 1;
    intsum = 0;
    while (i < n)
    {
        sum = i+j;
        i = i+j;
        J++;
    }
    cout << "Sum = " <<sum;
```

$O(n \log n)$

⇒

```
void fun(int n)
{   sum = 0;
    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
            for(int k = 0; k < n; k++)
            {
                sum = i+j+k;
            }
    cout << "Sum = " << sum;
}
```

$O(n^3)$

⇒

```
Void fun(n) {
int i, j, k
for(j = 1; j <= n; j=*2;)
  for(k=1; k<=1; k=*2)
      Count++
}
```

$O(\log \log(n))$

DATE
PAGE

Q12

Ans12: $T(n) = T(n-2) + T(n-1)$

Space Complexity = $O(n)$ due to stack.

$$T(n) = T(n-1) + T(n-2) + 4$$

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

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

Acc to Masters Theorem:

$$T(n) = O(4 * 2^{n/2})$$

$$T(n) = O(4 \times 2^{n/2})$$

$$T(n) = O(2^{\frac{n+4}{2}}) = O(2^n).$$

Q14:

Ans14: we can safely Assume

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

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

Case 3: if $\log_b a < K$ if $p \geq 0$ $\Theta(n^k \log^p n)$

if $p < 0$ $O(n^k)$

Here $k = 2$

$a = 2$

$b = 2$

$\therefore \log_2 2 = 1 < 2$

$\therefore O(n^2)$.