DATE
PAGE

Design and Analysis of Algorithms

Assignment - 1

Ques-1 What do you understand by Asymptotic notations. Define different asymptotic notation with examples.

Ans 1 Asymptotic notations are languages that allow us to analyze an algorithm running time by identifying its behaviour as the input size of the algorithm.

Types :-

1. Big O :- It is commonly used for worst case, and gives us upper bound for the growth rate of runtime of algorithm.
   Example :- Big O notation for linear search is $O(n)$

2. Big Omega :- It is notation used for least case complexity, it provides us with an asymptotic lower bound.
   Ex - Big omega of linear search is $\Omega(1)$

3. Theta :- It is used for tight bound on the growth rate of runtime of algo
   Ex - Theta of linear search is $O(n)$

4. Small Omega :- It is used to denote the upper bound (i.e. not asymptotic tight)
   $f(n) = o(g(n)) \Rightarrow f(n) < c(g(n)) \quad c > 0$

Teacher's Signature.....

DATE
PAGE

5. Small Omega - To denote lower bound (that is not asymptotic)

Ques-2 Time complexity of - for (i=1 to n) { i = i*2 }

=) O(log n)

Ques-3 T(n) = { 3 T(n-1) if n>0 otherwise 1 }

Tn = 3T(n-1)
T(1) = ~~3T(1-1)~~ 1
T(2) = 3 T(1) = 3
T(3) = 3T(2) = 9
T(4) = 3T(3) = 27
!

T(n) = (n-1)^3
Time complexity → O(3^n)

Ques-4 T(n) = { 2T(n-1) - 1 if n>0 otherwise }

Ans

T(n) = 2T(n-1) - 1
T(n-1) = 2T(n-2) - 1
T(n) = 4T(n-2) - 2 - 1
T(n-2) = 2T(n-3) - 1
T(n) = 8T(n-3) - 4 - 2 - 1
T(n-3) = 2T(n-4) - 1
T(n) = 16T(n-4) - 8 - 4 - 2 - 1
T(n) = 2^k - ... - 2^3 - 2^2 - 2^1 - 2^0
TC = O(1)

Teacher's Signature

DATE 00 00 00
PAGE 000

Ques-5

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

i	S	n
1	1	10
2	3	
3	6	
4	10	

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

Ques-6

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

Ans $TC = O(\sqrt{n})$

Ques-7

```
void function(int n){
  int i, 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++;
}
```

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

Teacher's Signature.....

DATE
PAGE

Ques 9 void function (int n){
for(i = 0 to n)
for(j = 1; j <= n; j = j+1){
printf("*");
}

Ans → Total time complexity of problem is.
TC = O(n log n)

Ques-10 for the functions, n^k and a^n, what is the asymptotic relation between these functions?
Assume that $k >= 1$, and $a > 1$ are constants.
find out the value of c and n_0 for which relation holds.

Ans n^k is $\Theta(c^n)$ as for example
If we take $n = 2$ $K = 2$, $c = 2$
Then $2^2 \leq 2^2$ so c^n is upper limit of n^k

Ques 1.

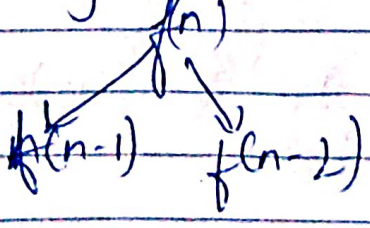
```
void fun (int n){
  int j=1, i=0;
  while (i<n){
    i = i+j;
    j++;
  }
}
```

j = 1	i = 0
	1
2	3
3	6
4	10

$TC = O(2^n)$

Ques-2 Write recurrence relation for the recursive function that prints Faboracci series.
Solve recurrence relation to get time complexity of the program?
What will be the space complexity of this program and why?

Ans-2 Space complexity = $O(n)$ as recursion calls for $f(n-1)$
Time complexity = $O(2^n)$



Teacher's Signature

DATE
PAGE

Ques- 13 Write programs which have complexity
- $n(\log n)$, n^3, $\log(\log n)$

Ans

$n\log n$ →

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j = j + 2)
        printf(" ");
```

n^3 →

```
for (i=0; i<n; i++)
    for (j=0; j<n; j++)
        for (k=0; k<n; k++)
            printf(" ");
```

$\log(\log n)$

```
int fun (int n) {
    if (n == 1)
        return n;
    else
        return fun(√n) + fun(√n);
}
```

Ques- 14 Solve the following recurrence relation
$T(n) = T(n/4) + T(n/2) + cn^2$

Ans- 14 $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + cn^2$

Using master theorem

$a = 2, b = 2$

$= 1$

$f(n) > n^2$ $\quad f n^2 > 1$

$O(n^2)$

Teacher's Signature.....

DATE
PAGE

Ques-15
```
int fun(int n){
  for(int i=1; i<=n; i++){
    for(int j=1; j<n; j+=i){
      //Some task
    }
  }}
```

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

Ques-16
```
for(int i=2; i<=n; i=pow(i,k))
{
  //Some task
}
```
where k is constant

Ans-16 $O(\log \log n)$

Ques-17

$$T(n) = T\left(\frac{99}{100}n\right) + T\left(\frac{n}{100}\right)$$

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

Ques-18 a) $100 < \log\log n < \log n < \sqrt{n} < n\log(n-1) < n\log n < n^2 < 2^n < 2^{2n} < 4^n < n!$

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

c) $96 < \log_2 n < \log_{\sqrt{2}} n < \log 5n < \log n! < n\log_{\frac{1}{2}} n < n\log_2 n < 8n^2 < 7n^5 < 8^{2n} < n!$

Teacher's Signature

Ques-19 Write linear search pseudocode to search an element in a sorted array with minimum comparisons:

```
void linear(arr, key){
    for(int i=0 to n)
        if(arr[i] == key)
            return;
    return -1;
}
```

Ques-20

Insertion Sort :-

→ Iterative :-

```
insertion (int arr[], int n) {
    for(i=1; i<n; i++)
    { int val = a[i]; j=i-1;
      while(j>0 && a[j-1]>val)
      { a[j] = a[j-1]
        j--;
      }
    a[j] = val;
}}
```

Recursive

```
insertion (int a[], int i, int n){
    int val = a[i], j=i-1;
    while (j>0 && a[j-1] > val){
      a[j] = a[j-1]
    } j--;
    a[j] = val
    if (i+1 <= n)
        insertion(a, i+1, n);
}
```

Teacher's Signature

DATE
PAGE

Ques-21

	Stable	Inplace
Bubble Sort	✓	✓
Selection sort	✗	✓
Insertion sort	✓	✓

Ques-22 Time complexity

	Best	Average	Worst	Space complexity
Bubble	$O(n^2)$	$O(n^2)$	$O(n^2)$	1
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$	1
Insertion	$O(n)$	$O(n)$	$O(n^2)$	1

Ques-23 Recursive-

```
Binary (arr, l, r, key){
  if (l < r){
      mid = l+(r-l)/2;
      if (arr[mid] = key)
          return;
  if (key < arr[mid])
      Binary (l, mid-1, key);
  else
      Binary (midt, r, key);
}
```

Iterative

```
while (l < r){
  mid = l+(r-l)/2
  if (arr[mid] == key)
      return;
  if (key < arr[mid])
      r = mid-1;
  else
      l = mid+1;
}
```

Teacher's Signature.....