Assignment-1

Date / /
Page No. 1

Design and Analysis of Algorithms

Name :- Bharti
Roll No :- 12/2014617
Section :- G1

Q1 What do you understand by Asymptotic notations? Define different Asymptotic notation with examples.

Ans Asymptotic notations are the mathematic notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

Different Asymptotic notations are:-

•) Big-O notation :- Big-O notation represents the upper bound of the running time of an algorithm. Thus, it gives the worst-case complexity of an algorithm

$O(g(n)) = \{f(n)$: there exist positive such that $0 \leq f(n) \leq cg(n)$

•) Omega Notation (Ω-notation) :- Omega notation represents the lower bound

of the running time of an algorithm
$\Omega(g(n)) = \{f(n)$: there exist positive such that $0 \leq cg(n) \leq f(n)$

Theta Notation :- Theta notation encloses the function from above and below. since it represents the upper and the lower bound of the running time of an algorithm, it is used for analyzing the average-case complexity of an algorithm

eg :- $\theta(g(n)) = \{f(n)$: there exist positive such that $0 \leq c_1 g(n) \leq f(n)$

Q2 What should be time complexity of - for(i=1 ton) {i=i*2}

Ans $O(\log n)$

Q3 $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

Ans
$$T(n) = 3T(n-1)$$
$$= 3(3T(n-2))$$
$$= 3^2T(n-2)$$
$$= 3^3T(n-3)$$
$$= 3^nT(n-n)$$
$$= 3^nT(0)$$
$$= 3^n$$
$$= O(3^n)$$

Q4 $T(n) = \{2T(n-1)-1$ if $n>0$, otherwise $1\}$

Ans

$$T(n) = 2T(n-1)-1$$
$$= 2(2T(n-2)-1)-1$$
$$= 2^2(T(n-2))-2-1$$
$$= 2^2(2T(n-3)-1)-2-1$$
$$= 2^3T(n-3)-2^2-2^1-2^0$$
$$\cdots$$
$$= 2^nT(n-n)-2^{n-1}-2^{n-2}-2^{n-3}$$
$$\cdots 2^2-2^1-2^0$$
$$= 2^n-2^{n-1}-2^{n-2}-2^{n-3}\cdots 2^2-2^1-2^0$$
$$= 2^n-(2^n-1)$$
$$T(n) = 1$$

Time complexity is $O(1)$

Q5 What should be the time complexity of

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

Ans $O(\sqrt{n})$

Q6 Time complexity of

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

Ans Log n

Q7 Time complexity of-

```
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++
  }
```

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

Q8 Time complexity of -

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

O(1)

Q9 Time complexity of-

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

Ans $O(n)$

Q10 for the functions, n^k and a^n, what is the asymptotic relationship between these functions? Assume that $k>=1$ & $a>1$ are constant

Ans n^k is $O(a^n)$

Q11 What is the time complexity of below code and why?

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

$O(n)$

Q12 Write recurrence relation for the recursive function that prints Fibonacci Series. Solve the recurrence relation to get time complexity of the program. What will be the space complexity of this program and why?

Ans
```
class Fibonacci {
Public static int fib (int i) {
    if (i<=1)
    return i;
return fib(i-1) + fib(i-2);
}
Public static void main (String args[]) {
  System.out.println (fib (61);
}
};
```

$fib(n) = fib(n-1) + fib(n-2) \rightarrow$ for $n > 1$

$fib(n) = 1 \rightarrow$ for $n = 0, 1$

For the iterative approach, the amount of space required is the same for fib(6) and fib (100) i.e as N changes the space / memory used remains the same Hence it's space complexity is O(1) or constant

Q13 Write programs which have complexity $- n(\log n)$, n^3, $\log(\log n)$

(ii)

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

T.C = nlogn

(iii)

```
function findXYZ(n){
 const Solutions=[];
for (let x=0; x<n; x++){
for (let y=0; y<n; y++){
for (let z=0; z<n; z++){
 if (3*x+9*y+8*z==79){
 Solutions.push ({x,y,z});
  }
  }
  }
  }
   return Solution;
    }
```

T.C = O(n^3)

(iii)

```
class Solution{
 Public int countPrimes(int n){
    if (n<2) return 0;
 boolean[] nonPrime= new boolean[n];
  nonPrime[1]= true;
   int numPrimes=1;
```

```
for (int i=2; i<n; i++)
if (nonPrime[i]) continue;
   int j = i * 2;
While (j<n) {
    if (!nonPrime[j]){
    nonPrime[j] = true;
    NumNonPrime s++;
    }  j += i;
```

Q14 Solve the following recurrence relation
$T(n) = T(n/4) + T(n/2) + cn^2$

Sol
$T(n/2) >= T(n/4)$
$T(n) <= 2T(n/2) + cn^2$
This gives us that
$T(n) <= \Theta(n^2) => T(n) = O(n^2)$
Also
$T(n) >= cn^2 => T(n) >= \Theta(n^2) => T(n) = \Omega(n^2)$
Since, $T(n) = O(n^2)$ and $T(n) = \Omega(n^2)$
$T(n) = \Theta(n^2)$

Q15 What is the time complexity of following function fun()?

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

Sol
$O(n \log n)$

Q What should be the time complexity of

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

Where, k is a constant

Ans $O(\log \log n)$

Q Arrange the following in increasing order of rate of growth:

(a) $n, n!, \log n, \log\log n, \text{root}(n), \log(n!), n \log n, 2^n, 2^{2n}, 4^n, n^2, 100$

Ans $100 < \log\log n < \log n < n^2 < n < n! < n \log n < n^2$
$2^n < 4^n < 2^{2n}$

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

Ans $1 < \log(\log(n)) < \log 2n < \sqrt{\log(n)} < n^2 < 4n < 2n, n^2 < 2(2^n)$

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

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

Q19 Write linear search pseudocode to search an element in a sorted array with minimum comparisons.

Sol

```
int searchArray (int A[], int k)
{
    int l = 0
    int r = 1
    while ( A[r] < k)
    {
        l = r
        r = 2*r
    }
    return binarySearch (A, l, r, k)
}
```

Q20 Write pseudo code for iterative and recursive insertion sort. Insertion sort is called online sorting. Why?

Ans Iterative :-

```
X1 = g(x0)
Step = Step+1
If Step > N
    Print "Not convergent"
    Stop
End If
X0 = X1
While abs f(x1) > e
Print root as X1
    Stop
```

•) Void recursive_insertion_sort(int arr[], int n)

```
    //Base case
    if (n<=1)
      return
recursive_insertion_sort(arr, n-1)
  int val = arr[n-1]
   int pos = n-2
while(pos>=0 && arr[pos] > val){
    arr[pos+1] = arr[pos]
     pos = pos-1
   }
   arr[pos+1] = val
}
```

An online algorithm is one that can process its input piece-by-piece in a serial fashion, ie, in the order that the input is fed to the algorithm without having the entire input available from the beginning. Thus insertion sort is called online sorting

Q21 complexity of all the sorting algorithms that has been discussed in lectures

Ans

Bubble Sort - $O(n^2)$
Selection Sort - $O(n^2)$
Insertion Sort - $O(n^2)$
Merge Sort - $O(n \log n)$

Q23 Write recursive/iteration pseudo code for binary search. What is the time and space complexity of linear and Binary Search?

Ans

```
int binarySearch(int[] A, int x)
{
int low=0, high=A.length-1;
while(low<=high)
{
int mid=(low+high)/
if(x==A[mid]){
return mid;
}
else if(x<A[mid]){
high=mid-1;
}
else{
low=mid+1;
}
}
return -1;
}
```

Space complexity of linear and binary search is $O(1)$.

Time complexity of linear search is $O(N)$ and binary search has $O(\log_2 N)$

Q24 Write recurrence relation for binary recursive search.

```
Public class LinearSearchUsingRecursion
{
Public static int linearSearch(int array[], int i, int n, k)
    {
      if (i==n)
      {
      return -1;
      }
      if(array[i]==k)
      {
        return i;
      }
        i = i+1;
  return linear search (array, i, n, k);
Public static void main(String[] args)
    {
      int array[] = { 5,8,7,9,6,0};
       int n = array.length;
        int k=9;
     int pos= linearSearch (array, 0, n, k);
    System.out.println("Position:" + pos);
      }
      }
```