DATA STRUCTURES AND ALGORITHMS

Question Bank

Mathematical notations- Big O, Omega and theta, Complexity- Time space trade off

(Objective Type Question -8 Nos. , Understanding Questions (Blooms level Two) - 4 Nos. , Scenario Based Question-2 Nos. with ANSWER KEY

Objective Type Question -8 Nos. ,

1. Select the main measures of efficiency of an algorithms are?
   A. Time and space complexity
   B. Data and space
   C. Processor and memory capacity
   D. Complexity and Capacity

Ans: A

2. Which case indicate the minimum time required for program execution?
   A. best case
   B. average case
   C. worst case
   D. Complexity case

Ans: A

3. Choose the following functions provided the maximum asymptotic complexity?
   A. f1(n)=n^(3/2)
   B. f2(n)=n^(log n)
   C. f3(n)=n log n
   D. f4(n)=2^n

Ans: D

4. Which of the following is linear asymptotic notations?
   A. O(1)
   B. O(log n)
   C. O(n)
   D. O(n log n)

Ans: C

5. __________ is the formal way to express the upper bound of an algorithm's running time.

   A. Omega Notation
   B. Theta Notation
   C. Big Oh Notation
   D. Small Omega Notation

Ans: C

6. Let us consider a given function, f(n) = 4.n3+10.n2+5.n+1. Considering g(n) = n3, f(n) < 5.g(n) for all the values of n > 2. Hence, the complexity of f(n) can be represented as ______.

A. $O(n^3)$
B. $\Omega(n^3)$
C. $\Theta(n^3)$
D. $\omega(n^3)$

Ans: A

7. When we say an algorithm has a time complexity of O(n), what does it mean?

A. The algorithm has ‘n’ nested loops.
B. The computation time taken by the algorithm is proportional to n
C. The algorithm is ‘n’ times slower than a standard algorithm
D. There are ‘n’ number of statements in the algorithm

Ans: B

8. Asymptotic analysis is _______ bound.

A. output
B. input
C. outer
D. inner

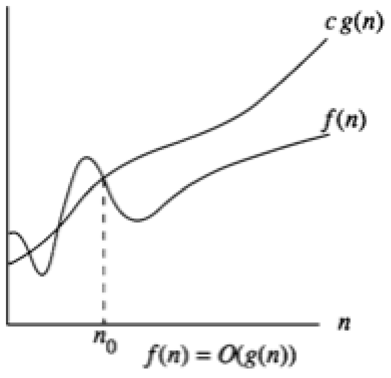
Ans : B

Understanding Questions (Blooms level Two) - 4 Nos. ,

1. Differentiate the Big oh and Omega notations. (5 marks)

Big 'Oh' Notation (O): O(g(n)) = { f(n) : there exist positive constants c and n0 such that 0 ≤ f(n) ≤ cg(n) for all n ≥n0 }

It is the upper bound of any function. Hence it denotes the worst case complexity of any algorithm. We can represent it graphically as

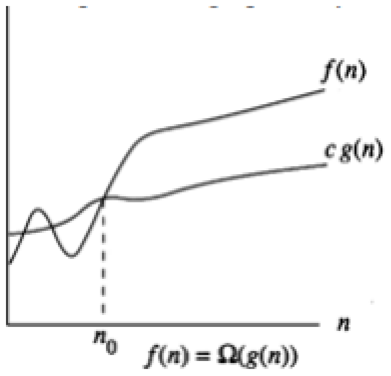


Omega Notation (Ω): Ω (g(n)) = { f(n) : there exist positive constants c and n0 such that 0 ≤ cg(n) ≤ f(n) for all n ≥ n0 }

It is the lower bound of any function. Hence it denotes the best case complexity of any

algorithm.

We can represent it graphically as



2. Find the Big Oh for the following functions (i) f(n) = 3n + 2 (5 marks)

General form is f(n) ≤ cg(n)

When n ≥ 2, 3n + 2 ≤ 3n + n = 4n

Hence f(n) = O(n), here c = 4 and n0 = 2

When n ≥ 1, 3n + 2 ≤ 3n + 2n = 5n

Hence f(n) = O(n), here c = 5 and n0 = 1

(ii) $f(n) = 100n + 6$

When $n \geq 6$, $100n + 6 \leq 100n + n = 101n$

Hence $f(n) = O(n)$, here $c = 101$ and $n0 = 6$

3. Analysis the following Function and find the worst case running time: (5 marks)

```
int linearSearch(int data[], int length, int val) {
for (int i = 0; i <= length; i++) {
if (val == data[i]) {
return i;
}}
return -1;
}
```

Worst Case:

The worse case for Linear Search is achieved if the element to be found is not in the list at all. This would entail the algorithm to traverse the entire list and return nothing. Thus the worst case running time is: O(N).

4. Solve the exponential function and find the Big Oh for the following function $f(n) = 6*2^n + n^2$ (5 marks)

When $n \geq 4$, $n^2 \leq 2^n$

So $f(n) \leq 6*2n + 2^n = 7*2^n$

Hence $f(n) = O(2^n)$, here $c = 7$ and $n0 = 4$

Scenario Based Question-2 Nos

1. Mani is a shop owner he wants to count the number of balls available in the box. The balls are numbered, we need to create a software to accept n number of balls and we need to display the count in each number. Write the C program to solve the problem and find the time complexity function of the program.

```
#include <stdio.h>
Int main()
{
        int n,a[50],t[50]=0,i;
        scanf("%d".&n);
        for(i=0;j<n;i++){
                scanf("%d",&a[i]);
                t[a[i]]++;}
        for(i=0;i<n;i++)
        {
                int j,flag=0;
                for(j=1;j>=0;j--)
                {
                        if(a[j]==a[i]{ flag=1;}
                        if(flag==0){printf("%d occurs %d times \n", a[i],t[a[i]]);
                }
        } return 0;}
```

Ans: Since it is nested loop, the major part will run for n x n times. So Big oh(n^2)

2. Study the following code, find the time complexity of the source code and give the conclusions.

```
#include <iostream>
using namespace std;


int main()
{
int N;
cout<<"Enter a number: ";
cin>>N;
while(n>0)
{
   cout<<N<<" ";
   N=N/2;
}
return 0;
}
```

Answer:

In the code above, there is a while loop in addition to initialisation, input and output statements, an arithmetic operation and a return statement.
Let us examine mathematically how this while loop works here.

Suppose N=8; then we are getting series of 8 4 2 1
Likewise, if we try for different inputs like 10, we will get 10 5 2 1
We can see that in each iteration, the number will be reduced to the previous half until the whole condition is violated.
Such algorithms of breaking a number or set of numbers into halves fall under logarithmic complexity O(log N).