Data Structure	Week (3)

Section Content:

- Asymptotic Analysis

Algorithms Efficiency

Program efficiency is measured by how much of various types of resources it consumes.
The main measures are: -

- Speed or running time
- Space (memory)
- Power consumption

Big O notation

This Big(O) examines the rate of growth of a function by comparing it with some standard functions whose rate of growth is known.

O (1) < O (log n) < O(n) < O (n log n) < O (n2) < O (n3) < O (nk) < O (Kn) < O (n!)

Finding Big O: -

- Keep the fastest growing term and discard the lower terms and constants
- Ignore coefficients

EX: -

- $f(n)=3n^2+4n+7$

  $g(n)=n2=\mathrm{O}(n2)$

- $f(n)=3n+5n2+7n3+2n$

  $\mathrm{O}(2n)$

- $f(n)=4n+6n+9n5$

  $\mathrm{O}(6n)$

- $f(n)=7n+6n+n!$

  $\mathrm{O}(n!)$

Big O analysis of Algorithms Calculating Time complexity

Running time is proportional to the number of primitive operations executed during run time.

1. Int x=2; constant time c1
For (int i=0; i<=n; i++) n
Sum=sum+I; constant time c2

T(n)=c1+n+c2
Is O(n)

2. For (int i=0; i<=n; i++) n
   For (int j=0; j<=n; j++) n
   Print i+j;

   T(n)=n*n
   Is O(n2)

3. For (int i=0; i<=n; i++) n
   Print I;
   For (int j=0; j<=n; j++) n
   For (int k=0; k<=n; k++) n
   Print j+k;

   T(n)=n+n*n
   Is O(n2)

4. For (int i=0; i<=n; i++) n
   Print I;
   For (int j=0; j<=n; j++) n
   For (int k=0; k<=n; k++) n
   For (int l=0; l<=n; l++) n
   Print j+k+l;

   T(n)=n+n*n*n
   Is O(n3)

5. Int i;
   I=1 constant time c1
   For (i; i<=n; i*2) log2 n
   Print I; constant time c2

   T(n)=c1+ log2 n +c2
   Is O(log n)

6. For (int i=n/2; i<=n; i++) n/2
   For (int j=0; j<=n; j=j*2) log n
   For (int k=0; k<=n; k=k*2) log n
   Print I+j+k; constant time c1

$T(n)=n/2 * \log n * \log n + c1$
Is $O(n \log n)2$