Data Structure Training

Algorithm & Complexity

Department of MCA

GL Bajaj Greater Noida

Data Structure Training(Algo&Complexity)

Algorithm-

invalid symbol

1-Finite set of steps to solve a problem is called algorithm.
2-Here steps means instruction which contain fundamental operators.

i = i ⊥ 1 ;

i = i + 1 ;

Characteristics Of fundamental Instruction-

1-Definiteness-

Every fundamental instruction must be definite without any ambiguity.

i=i 1 ✗
i=i+1; ✓

2-Finiteness-

Every instruction must be terminate within finite amount of time.

```
i=1;
while(1)
{
        i=i+1;//not allowed in algorithm since executing infinitely.
}
```

Infinite loop

break;

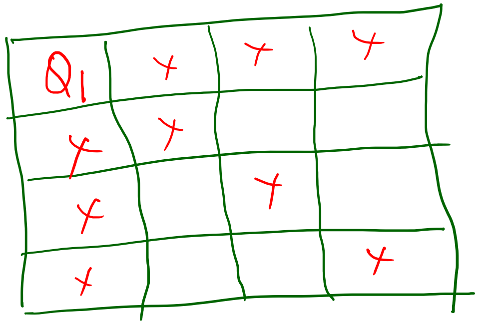
3- Every instruction must accept atleast 0 input and provides atmost 1 output.

Data Structure Training(Algo&Complexity)

Steps For Solving Any Problem-

1-Identifying Problem Statement

4 Queen Problem



Q_1
Q_2
Q_3
Q_4

2-Identifying Constraints

Data Structure Training(Algo&Complexity)

Steps For Solving Any Problem-

3-Design Logic-

Depends on the characteristics of the problem we can choose any one of the following design strategy for design logic.

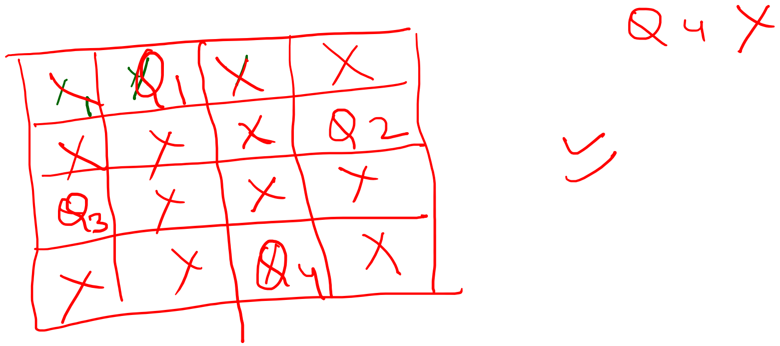
a)Devide & Conquer

b)Greedy Method

c)Dynamic Programming

d)Branch & Bound

e)Backtracking etc……..



4-Validation-

Most of the algorithm validated by mathematical induction.

5-Analysis-

Process of comparing two algorithms with respect to time,space,number of register,network bandwidth etc is called analysis.

Data Structure Training(Algo&Complexity)

Types Of Analysis-

Priory Analysis-

1-Analysis done before executing

X=x+1;

2-Principle-

Frequency count of fundamental instruction.

Since x=x+1 being carried out only one time so its complexity os O(1) order of 1

O(n)

3-It provides estimated values.

4-It provides uniform values.

5-It is independent of CPU,O/S & system architecture

type ⇒ logic

10

12

manually

64

8GB

32

4GB

Data Structure Training(Algo&Complexity)

Types Of Analysis-

Posterior Analysis-

1-Analysis done after executing

X=x+1;

64bit 32bit

8GB 4GB

.003ns .1ns

3-It provides exact values.

4-It provides non uniform values.

5-It is dependent of CPU,O/S & system architecture

6- Implementation

c, c++, java, python

7-Testing And Debugging

Data Structure Training(Algo&Complexity)

Time Complexity

1-Simple for loop

2-Nested for loop

3-if-else

4-Recursive Algorithm

Data Structure Training(Algo&Complexity)

Simple for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
        int sum=0,i;
        for(i=1;i<=n;i++)
        {
                sum=sum+i;
        }
        return 0;
}
```

Data Structure Training(Algo&Complexity)

Simple for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
        int sum=0,i;
        for(i=1;i<=n;i=i+2)
        {
                sum=sum+i;
        }
        return 0;
}
```

$n=10$

1 3 5 7 9 11

1 3 5 7 - - - - - - n

AP

$t_k = a + (k-1)d$

$n = 1 + (k-1)2$

$n = 1 + 2k - 2$

$\frac{n+1}{2} = k$

$k = \frac{10+1}{2} \Rightarrow 5.5$

$\Rightarrow 6$

$\frac{n+1}{2}$

$O(n)$

Data Structure Training(Algo&Complexity)

Simple for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
        int sum=0,i;
        for(i=1;i<=n;i=i*2)
        {
                sum=sum+i;
        }
        return 0;
}
```

$n = 10$

1 2 4 8 16

2^4

1 2 4 8 ---- n

$t_k = a r^{k-1}$

$n = 1 \cdot 2^{k-1}$

$n = 2^{k-1}$

$\log_2 n = \log_2 2^{k-1}$

$\log_2 n = (k-1) \log_2 2$

$\frac{\log 2}{\log 2} = 1$

$1 + \log_2 2$

$1 + 4 \log_2 2 \Rightarrow 5$

$k = 1 + \log_2 n$

$\rightarrow O(\log_2 n)$

Data Structure Training(Algo&Complexity)

Simple for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
        int sum=0,i;
        for(i=n;i>0;i=i/2)
        {
                sum=sum+i;
        }
        return 0;
}
```

$n = 10$

$10 \quad 5 \quad 2 \quad 1 \quad 0$

$n \quad n/2 \quad n/4 \quad n/8 \quad \text{--} \quad 1$

$t_k = a r^{k-1}$

$1 = n\left(\frac{1}{2}\right)^{k-1}$

$1 = \frac{n}{2^{k-1}} \Rightarrow$

$2^{k-1} = n$

$\log_2 2^{k-1} = \log_2 n$

$(k-1) = \log_2 n$

$k = 1 + \log_2 n$

$k = 1 + \log_2 2^4$

$= 5$

Data Structure Training(Algo&Complexity)

Simple for loop

Find total comparison and time complexity of following code and let val(j) denote value stored in variable j.after termination of loop what is the value of val(j)

```
#include <stdio.h>
int main(void)
{
	int sum=0,i,j;
	for(i=n;j=0,i>0;i=i/2,j=j+i)
	{
		sum=sum+i;
	}
	return 0;
}
```

assignment

n n/2 n/4 - - 1

$O(\log_2 n)$

Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```c
#include <stdio.h>
int main(void)
{
        int sum=0,i,j;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        sum=sum+j;

                }
        }
        return 0;
}
```

$n = 10$

$11 <= 10$

$n^2 + n$

$O(n^2 + n)$

110

$110 + 1$

111

$O(n^2)$

100

$110 + 1$

$10(11) + 1$

$n(n+1) + 1$

Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```c
#include <stdio.h>
int main(void)
{
        int sum=0,i,j;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j=j+2)
                {
                        sum=sum+j;
                }
        }
        return 0;
}
```

i 1 2 — — n

j $\frac{n+1}{2}$ $\frac{n+1}{2}$ $\frac{n+1}{2}$

$n(n+1)/2$

$n^2/2 + n/2$

$O(n^2)$

Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
        int sum=0,i,j;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j=j*3)
                {
                        sum=sum+j;
                }
        }
        return 0;
}
```

$n \log n \Rightarrow 10$

$n \log_2 n$ 2 1 2 — n

$n \log_3 n$ J $\log_2 n$ $\log_3 n$ $\log_3 n$

1 3 9 27 — — n

$t_k = a r^{k-1}$

$n = 1 \cdot 3^{k-1}$

$\log_3 n = \log_3 3^{k-1}$

$\Rightarrow k-1 \log_3 3$

$k = 1 + \log_3 n$

Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
        int sum=0,i,j;
        for(i=1;i<=n;i=i*2)
        {
                for(j=1;j<=n;j=j+2)
                {
                        sum=sum+j;
                }
        }
        return 0;
}
```

→ 60sec

$n\log_2 n$

$\log_2 n$

$\frac{n+1}{2}$

1 2

$1+\log_2 n$

$\times$

$\frac{n+1}{2}$

$\frac{n+1}{2}$ $\frac{n+1}{2}$

$O(n\log_2 n)$

$\frac{n+1}{2} + \frac{n+1}{2}\log_2 n$

$\frac{n}{2} + \frac{1}{2} + \frac{n}{2}\log_2 n + \frac{1}{2}\log_2 n$

Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```c
#include <stdio.h>
int main(void)
{
        int sum=0,i,j;
        for(i=1;i<=n;i=i+1)
        {
                for(j=1;j<=i;j=j+2)
                {
                        for(k=1;k<=n;k=k*2)
                        {
                                sum=sum+k;
                        }
                }
        }
        return 0;
}
```

$O(n^2 \log_2 n)$

i	1	2		n
j	1	2	0	$\frac{n+1}{2}$
k	$1+\log_2 n$	$1+\log_2 n$		$1+\log_2 n$

$n^2 \log_2 n + 1$

$n \times \frac{(n+1)}{2}(1+\log_2 n)$

$\frac{n^2}{2} + \frac{n}{2}(1+\log_2 n)$

$\frac{n^2}{2} + \frac{n^2}{2}\log_2 n + \frac{n}{2} + \frac{n}{2}\log_2 n$

Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
        int sum=0,i,j;
        for(i=0;i<=n;i=i+1)
        {
                for(j=1;j<=i;j=j+1)
                {
                        if(j%i==0)
                        {
                                for(k=1;k<=n;k=k+1)
                                {
                                        sum=sum+k;
                                }
                        }
                }
        }
        return 0;
```

Data Structure Training(Algo&Complexity)

Recursive Algorithm-

Find total comparison and time complexity of following code

```
int fact(int n)
{
        if(n==0 || n==1)
                return 1;
        else
                return n*fact(n-1);
}
```

Data Structure Training(Algo&Complexity)

Recursive Algorithm-

Find total comparison and time complexity of following code

```
int fib(int n)
{
	if(n==0)
		return 0;
	if(n==1)
		return 1;
	else
		return fib(n-1) +fib(n-2);
}
```

Data Structure Training(Algo&Complexity)

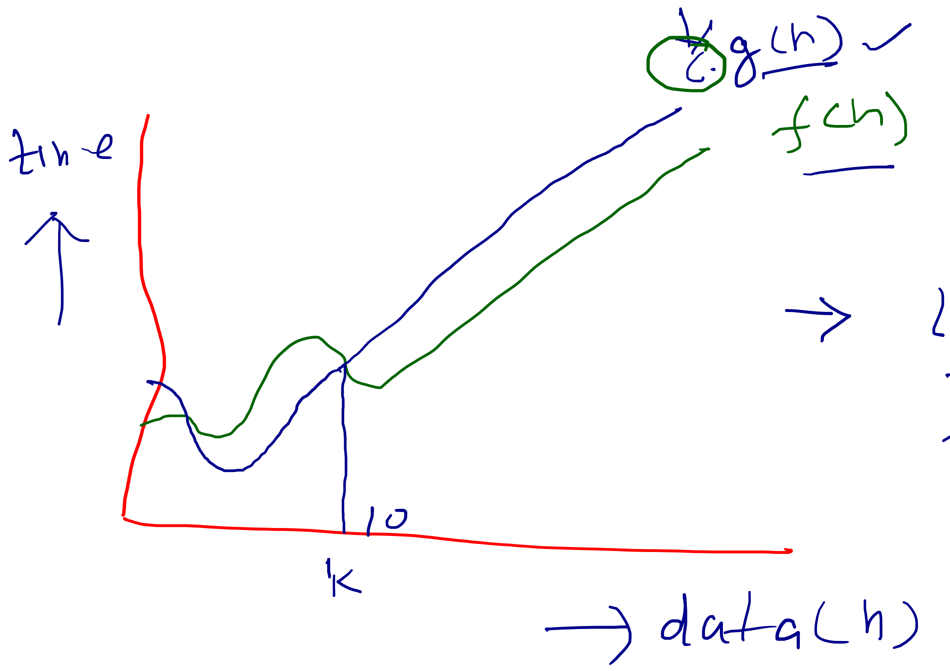
Asymptotic Notation-

to compare two algorithms rate of growth with respect to time and space, need asymptotic notation.

Big-Oh (O)

f(n) is O(g(n)) iff there exist some constants c>0 and k>=0 such that f(n) <= c.g(n) for every n>=k

→ f()



→ f(n) = low

→ g(n) = high

→ low = O(high)

f(n) = O(g(n))

Data Structure Training(Algo&Complexity)

① $\rightarrow f(n) = a_0 + a_1 n + a_2 n^2 + \cdots + a_m n^m$

$g(n)$ x

$O(n^m)$

$f(n) = O(g(n))$
$= O(n^2)$

② low = O(high) $\rightarrow$ $O(g(n))$?

$f(n) = n^2 + n + 1$

$n^2 \leq n^2$

$n^2 + 1 \leq n^2 + n^2$

$n^2 + n + 1 \leq n^2 + n^2 + n^2$

$\underbrace{n^2+n+1}_{f(n)} \leq \underbrace{3}_{c}\, \underbrace{n^2}_{g(n)}$

Data Structure Training(Algo&Complexity)

$n^2 \leq n^3$

$\underline{n^2+n+1} \leq n^3 + n^3 + n^3$

$f(n) \leq 3n^3$

$c \quad g(n)$

$\underline{O(n^3)}$

$f(n) \leq 3n^4$

n^3

$O(n^4)$ ✓

$\vdots$

n^n

$\cancel{n^2} \quad n^3 \quad n^4 \quad - - - \quad n^3$

$\Rightarrow$ lub = least upper bound

Data Structure Training(Algo&Complexity)

$f(n) = n^2 \log n \qquad g(n) = n(\log n)^{10}$

$\Rightarrow \quad f(n) = o(g(n)), \quad g(n) \neq o(f(n))$

$\Rightarrow \quad f(n) = o(g(n)), \quad g(n) = o(f(n))$

$\Rightarrow \quad g(n) = o(f(n)), \quad f(n) \neq o(g(n))$

$g(n) = o(f(n))$

$n > (\log n)^n$

$2 > (\log 2)^2 \to n$

$2 > 1$

$n^2 \log n \qquad n(\log n)^{10}$

$n * n\log n \qquad n\log n \cdot (\log n)^9$

$f(n) > g(n)$

Data Structure Training(Algo&Complexity)

$f(n)$ $\qquad\qquad$ $g(n)$

$n^2 \log n$ $\qquad\qquad$ $n(\log n)^{10}$

$low = O(high)$

$g(n) = O(f(n))$

$n * n \log n$ $\qquad\qquad$ $n\,(\log n)^1 (\log n)^9$

$n * n\log n \; > \; n\log n \; (\log n)^9$

high $\qquad\qquad$ low

n $\qquad\qquad$ $(\log n)^{n}$

$2 \qquad 2 \; > \; (\log_2 2)^2$

$2 \; > \; (1)^2$

.3010

Data Structure Training(Algo&Complexity)

$f(n) = n!$ $\qquad O(?)$

$= n(n-1)(n-2) \cdots 1$

$= n \quad n\left(1-\frac{1}{n}\right) \quad n\left(1-\frac{2}{n}\right) \cdots n\left(\frac{1}{n}\right)$

$= n^n \left[1 \cdot \left(1-\frac{1}{n}\right) \cdot \left(1-\frac{2}{n}\right) \cdots \frac{1}{n}\right]$

$f(n) = a_0 + a_1 n + \cdots a_m n^m$

$= O(n^n)$

Data Structure Training(Algo&Complexity)

$$f(n) = \log(n!)$$

$$= \log(O(n^n))$$

$$= O(\log n^n)$$

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

$$f(n) = O(?)$$

2 3

3 2

Data Structure Training(Algo&Complexity)

$f(n)$ $\qquad\qquad$ $g(n)$

$$2^{2^n} \geq n! \geq 4^n \geq 2^n \geq n^3 \geq n^2 \geq n\log n \geq n$$

$$\geq \log\log n \geq \frac{\log n}{\log\log n} \geq 1$$

Dominance Ranking

$g(n) = O f(n)$

Data Structure Training(Algo&Complexity)

Ω notation

Θ →

$f(n) = O(g(n))$

$c > 0$

$k \geq 0$

$\rightarrow f(n) \leq c\,g(n) \rightarrow O$

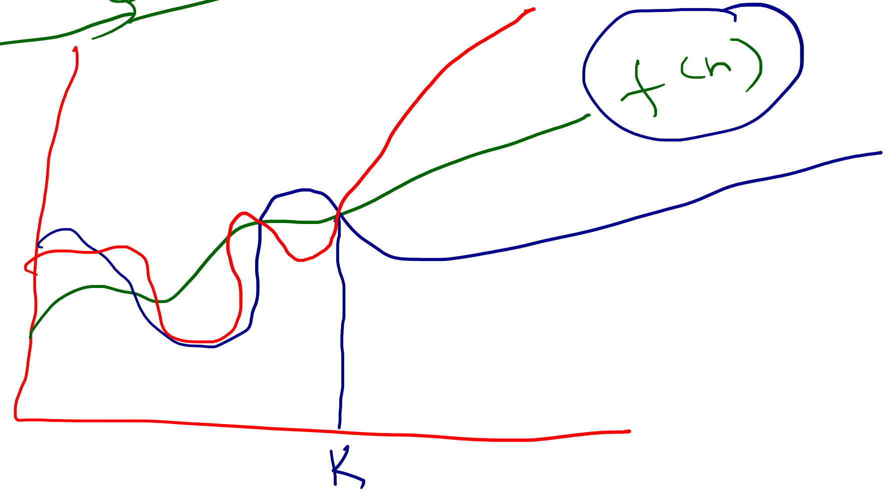
$\rightarrow f(n) \geq c \cdot g(n)$ $\quad \Omega(n)$

$c_2 g(n)$

$1n^2 \leq n^2+n+1 \leq 3n^2$

$c_1 g(n)$ $\quad f(n)$ $\quad g(n)$

$\Theta(n^2)$



$f(n)$

$c_1(g(n))$

$c_1 g(n) \leq f(n) \leq c_2 g(n)$

Data Structure Training(Algo&Complexity)

$f(n) = n^2 + n + 1$

$O(g(n))$

$n^2 \geq n^2$

$n^2 + n \geq n^2$

$n^2 + n + 1 \geq 1\, n^2, n, 1$

$f(n) \geq c\, g(n)$

$\Omega(n)$

$f(n)$

$\Omega(1)$, $\Omega(n)$

Atleast ✓

$\Omega(n^2)$

$n^2, n, 1$

$n^2\ n^3\ n^4 \cdots n^n$

GLB

$\Omega(n^2) \leftrightarrow O(n^2)$

Data Structure Training(Algo&Complexity)

```
int isprime(int n)
{
    int i, n;
    for( i=2 ; i<=√n ; i++)
    {
        if(n%i==0)
        {
            pf("NP");
            return(0);
        }
    }
    return(1);
}
```

2,3,4,5

14

$O(n)$

$\Omega(1)$

n

196

$\sqrt{196}$

14

$\Omega(1)$

sorted

$\Omega(1)$

atmost

atleast

$O(\sqrt{n})$

$\Omega(1)$

$\Omega(5)$

$\Omega(\sqrt{n})$

99%

$O(1) \leftrightarrow O(\sqrt{n})$

$\Omega(1)$

5 L

5 x