GeeksforGeeks

Welcome All

@

Mastering DSA Basic to Advance

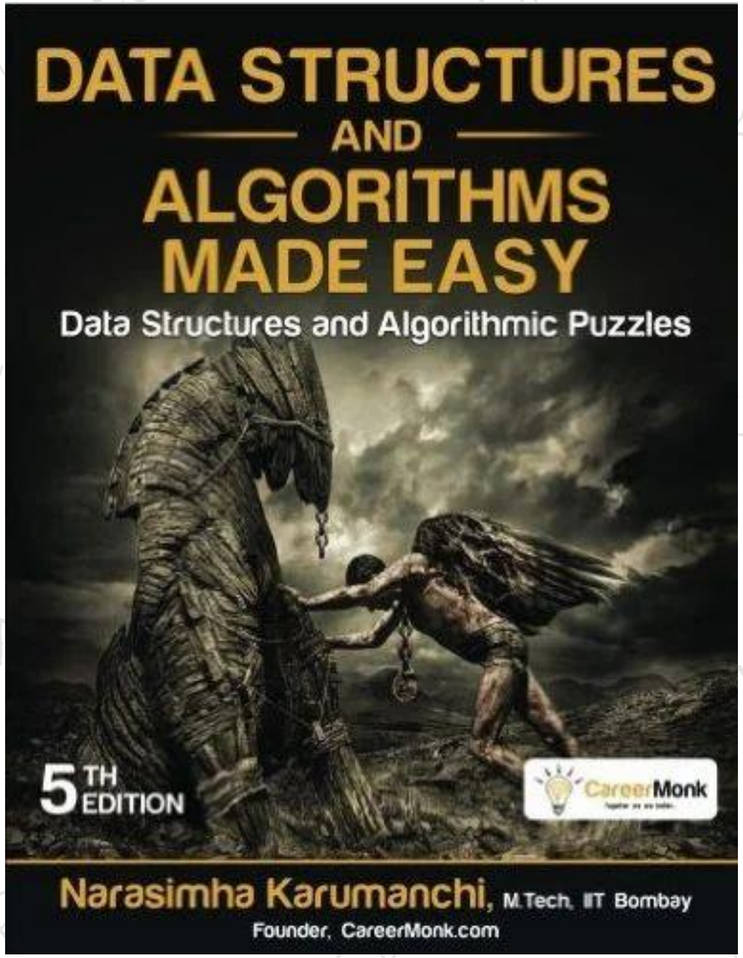
(Course for Product-Based Companies)

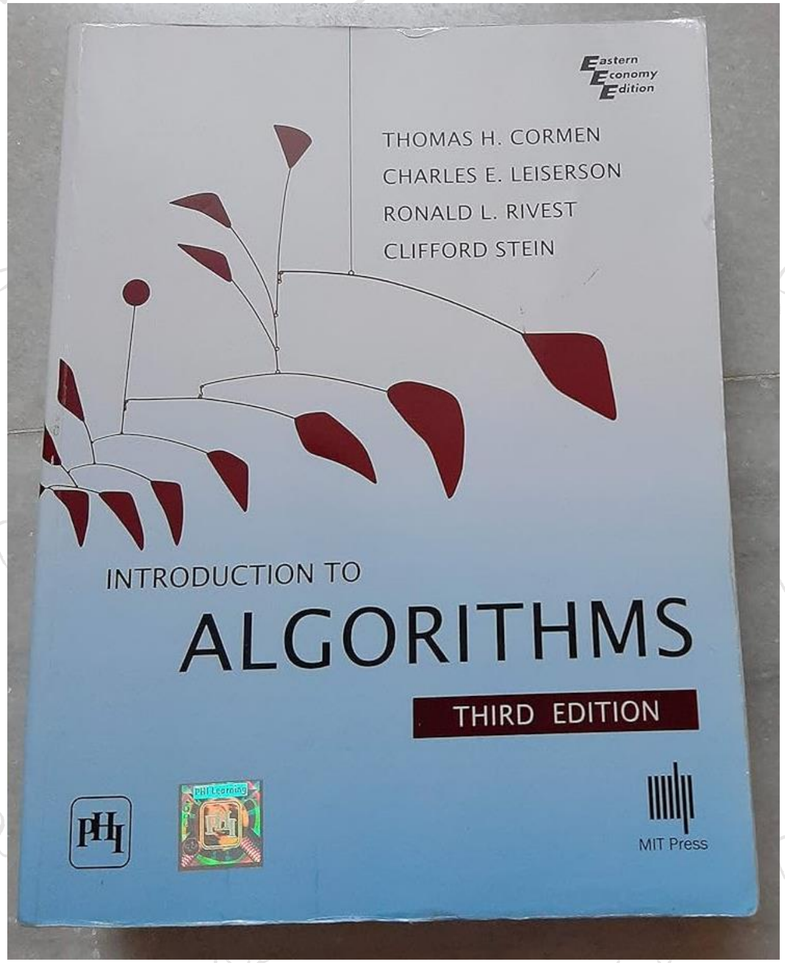
Lect-1&2 (Time/Space Complexity, Asymptotic Notations, Mathematical problems, Bitwise operations)

Presented By: Puneet Kansal

https://www.youtube.com/channel/UCmEBrWO1Z9XDfHZ9s_ZEqxw

BOOKS





Nim Game

You are playing the following Nim Game with your friend:

- Initially, there is a heap of stones on the table.
- You and your friend will alternate taking turns, and you go first.
- On each turn, the person whose turn it is will remove 1 to 3 stones from the heap.
- The one who removes the last stone is the winner.

Given n, the number of stones in the heap, return true if you can win the game assuming both you and your friend play optimally, otherwise return false.

Example1 : Input: n = 4
Output: False

Example2 : Input: n = 7
Output: True

https://leetcode.com/problems/nim-game/description/

Types of Complexity Functions

S. No	Complexity Funtions	Representation
1	Constant	$\Theta(1)$
2	Logarithmic	$\Theta(\log n)$
3	Linear	$\Theta(n)$
4	Quadratic	$\Theta(n^2)$
5	Cubic	$\Theta(n^3)$
6	Polynomial	$\Theta(n^k)$
7	Exponential	$\Theta(c^n)$

Order of Growth

Complexity Funtions	Order of Growth
Exponential - $\Theta(c^n)$	Highest
Polynomial - $\Theta(n^k)$	
Cubic - $\Theta(n^3)$	
Quadratic - $\Theta(n^2)$	
Linear - $\Theta(n)$	
Logarithmic - $\Theta(\log n)$	
Constant - $\Theta(1)$	Lowest

Finding time complexity of codes O(1)

Example 1:

```
main( )
  {
      int i;
      for(i=0;i<10;i++)
          {
           printf("Hello Chacha, how is Chachi");
          }
   }
```

Finding time complexity of codes O(n)

Example 2:

```
main( )
   {
      int  i=1,n;
      scanf(“%d”,&n);
      while(i<=n )
           {
                x =  y + z;
                i = i + 1;
           }
    }
```

Finding time complexity of codes O(n)

Example 2.1:

```
main( )
  {
    int  i=1,n;
    scanf(“%d”,&n);
    while(i<=n )
        {
            x =  y + z;
            i = i + 2;
        }
   }
```

Finding time complexity of codes

O(n)

Example 2.2:

```
main( )
  {
    int  i=1,n;
    scanf(“%d”,&n);
    while(i<=n )
        {
            x =  y + z;
            i = i + 20;
        }
  }
```

Finding time complexity of codes O(1)

Example 3:

```
main( )
  {
      int n=10;
      for(int  i=1; i<=n; i++ )
            printf("%d",i);

      for(int  i=1; i<=n; i++ )
            for(int  j=0; j<n; j++ )
                  printf("%d",j);
  }
```

Finding time complexity of codes

O(logn)

Example 4:

```
main( )
   {
     int n;
     scanf(“%d”,&n);
     for(int  i=1; i<n; i=2*i )
                printf(“%d”,i);
   }
```

Finding time complexity of codes

Example 5:

O(logn)

```
main( )
  {
    int n;
    scanf(“%d”,&n);
    for(int  i=n; i>1; i= i/2 )
          printf(“%d”,i);
  }
```

Finding time complexity of codes

Example 5.1:

O(logn)

```
main( )
   {
     int n;
     scanf(“%d”,&n);
     for(int  i=n; i>1; i= i/20 )
              printf(“%d”,i);
    }
```

Finding time complexity of codes

Example 6:

O(nlogn)

```
main( )
   {
     int n;
     scanf(“%d”,&n);
     for(int  i=1; i<=n; i++ )
            for(int  j=1; j<n; j=2*j)
                   printf(“hi”);
    }
```

Finding time complexity of codes

Example 7:

O(logn)

```
main( )
  {
    int  i=1,n;
    scanf(“%d”,&n);
    while(i<=n )
        {
              i =  i + 2;
              i = i * 2;
        }
  }
```

Finding time complexity of codes

Example 8:

O(log(logn))

```
main( )
  {
    int n;
    scanf(“%d”,&n);
    for(int  i=2; i<=n; i= i^2 )
              printf(“hi”);
  }
```

Finding time complexity of codes

O(log(logn))

Procedure $A(n)$

If $n \leqslant 2$ return (1) else return $(A(\lceil \sqrt{n} \rceil))$;

O(log(log n))

It's Exam Time!

https://www.youtube.com/channel/UCmEBrWO1Z9XDfHZ9s_ZEqxw

Knowledge Check

$O(n^2)$

```
main( )
 {
    for(i=1; i<=n; i++)
        {
         for(j=1; j<=i; j++)
             {
              for(k=1; k<=133; k++)
                   printf("Hi");
             }
         }
 }
```

Time Complexity Summary Table

https://www.youtube.com/channel/UCmEBrWO1Z9XDfHZ9s_ZEqxw

Condition of loop (i)	Complexity
for(i=1; i<=n; i++)	$\Theta(n)$
for(i=n; i>=1; i--)	$\Theta(n)$
for(i=1; i<=n; i=i+2)	$\Theta(n/2) \simeq O(n)$
for(i=n; i>=1; i=i-2)	$\Theta(n/2) \simeq O(n)$
for(i=1; i<=n; i=i+10)	$\Theta(n/10) \simeq O(n)$
for(i=n; i>=1; i=i-10)	$\Theta(n/10) \simeq O(n)$
for(i=1; i<=n; i=i+1) for(j=1; j<=n; j=j+1)	$\Theta(n^2)$
for(i=1; i<=n; i=i+1) for(int j=1; j<n; j=2*j)	$\Theta(n \log_2(n))$
for(i=1; i<=n; i=i*2)	$\Theta(\log_2 n)$
for(i=n; i>=1; i=i/2)	$\Theta(\log_2 n)$
for(i=2; i<=n; i=i^2)	$\Theta(\log_2(\log_2(n))$
for(i=n; i>=1; i = √i)	$\Theta(\log_2(\log_2(n))$

Time complexity of Recursive codes

```
int factorial(int n)
    {
        if (n == 0 || n == 1)
            return 1;

        return n * factorial(n - 1);
    }
```

T(n) = T(n-1) + 1

Time Complexity: O(n)

Master Theorem

- *The Master Method is used for solving the following types of recurrence.*

$$T(n) = a\,T\left(\frac{n}{b}\right) + \theta\left(n^k \log^p n\right)$$

Here, a >= 1, b > 1, k >= 0
and p is a real number.

Case-01:

If $a > b^k$, then $T(n) = \theta\,(n^{\log_b a})$

Case-03:

If $a < b^k$ and

- If $p < 0$, then $T(n) = O\,(n^k)$
- If $p >= 0$, then $T(n) = \theta\,(n^k \log^p n)$

Case-02:

If $a = b^k$ and

- If $p < -1$, then $T(n) = \theta\,(n^{\log_b a})$
- If $p = -1$, then $T(n) = \theta\,(n^{\log_b a}.\log^2 n)$
- If $p > -1$, then $T(n) = \theta\,(n^{\log_b a}.\log^{p+1} n)$

$$T(n) = a\,T\left(\frac{n}{b}\right) + \theta\left(n^k \log^p n\right)$$

Master Theorem Examples

Case-01:

If $a > b^k$, then $T(n) = \theta\,(n^{\log_b a})$

Case-02:

If $a = b^k$ and

- If $p < -1$, then $T(n) = \theta\,(n^{\log_b a})$
- If $p = -1$, then $T(n) = \theta\,(n^{\log_b a}.\log^2 n)$
- If $p > -1$, then $T(n) = \theta\,(n^{\log_b a}.\log^{p+1} n)$

Case-03:

If $a < b^k$ and

- If $p < 0$, then $T(n) = O\,(n^k)$
- If $p >= 0$, then $T(n) = \theta\,(n^k \log^p n)$

Ex1: $T(n) = 3T(n/2) + n^2$

Ex2: $T(n) = 2T(n/2) + n\log n$

Ex3: $T(n) = 8T(n/4) - n^2\log n$

Time complexity of Recursive codes

```
int binarySearch(int arr[], int l, int r, int x)
{
      if (r >= l)
            int mid = (l + r) / 2;

      if (arr[mid] == x)
            return mid;

      if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);
      elif (arr[mid] < x )
            return binarySearch(arr, mid + 1, r, x);
}
```

T(n) = T(n/2) + 1

Time Complexity: O(logn)

Time complexity of Recursive codes

```
int recursive (int n)
{
  if (n == 1)
    return (1);
  else
    return (recursive (n - 1) + recursive (n - 1));
}
```

T(n) = 2T(n-1) + 1

Time Complexity: $O(2^n)$

S = a * (r^n - 1) / (r - 1) //n is number of levels in tree or number of terms in GP

Time complexity of Recursive codes

```
void mergeSort(int array[], int begin, int end)
{
    if (begin >= end)
        return;

    int mid = begin + (end - begin) / 2;
    mergeSort(array, begin, mid);
    mergeSort(array, mid + 1, end);
    merge(array, begin, mid, end);
}
```

$$T(n) = 2T(n/2) + n$$

Time Complexity: O(nlogn)

Space complexity

Definition:

- *Space Complexity of an algorithm is total main memory space taken by the algorithm with respect to the input size. Space complexity includes both Auxiliary space and space used by input.*

- *Auxiliary Space is the extra space or temporary space used by an algorithm*

Space complexity

```
int factorial(int n)
    {
        int res = 1, i;
        for (i = 2; i <= n; i++)
                res *= i;
        return res;
}
```

Time Complexity: O(n)
Auxiliary Space: O(1)

```
int factorial(int n)
    {
        if (n == 0 || n == 1)
            return 1;

        return n * factorial(n - 1);
    }
```

Time Complexity: O(n)
Auxiliary Space: O(n)

Introduction to Asymptotic Notations

- ***Definition:***
  Asymptotic notations are mathematical way of representing an algorithm's complexity for large size input.

- ***Types of asymptotic notations***

1. *Big - Oh (O) Notation*
2. *Big - Omega (Ω) Notation*
3. *Theta (Θ) Notation*
4. *Little - Oh (o)*
5. *Little - Omega (ω)*

It's Exam Time!

My reaction...

When someone asks for an extra sheet during exam!!!

Practice Notations(O, Ω, Θ, o, ω)

True or False ?

1) $n = O(n^2)$

2) $n = O(n)$

3) $n^4 = O(n^3)$

4) $n = O(n^2 - 10)$

5) $n^2 = \Omega(n)$

6) $n = \Omega(n)$

7) $n = \Omega(n^2)$

8) $n^2 = \Omega(n^2 - 10)$

9) $n = \Theta(n)$

10) $n^2 = \Theta(n)$

11) $n^3 = \Theta(n^3)$

12) $n = \Theta(n^2 - 10)$

13) $n = o(n^2)$

14) $n = o(n)$

15) $n^2 = o(n^3)$

16) $n^2 = o(n^2 - 10)$

17) $n^2 = \omega(n)$

18) $n = \omega(n)$

19) $n^3 = \omega(n^2)$

20) $n^2 = \omega(n^2 - 10)$

Mathematical Coding Problems

Size of Problem

MAX value of N	Time complexity
10^9	O(logN) or Sqrt(N)
10^8	O(N) Border case
10^7	O(N) Might be accepted
10^6	O(N) Perfect
10^5	O(N * logN)
10^4	$O(N^2)$
10^2	$O(N^3)$
<= 160	$O(N^4)$
<= 18	$O(2^n * N^2)$
<= 10	$O(N!)$, $O(2^n)$

Note: C++/JAVA Range of int = 10^9 (-2^31 to 2^31-1) and long long int = 10^{18} (-2^63 to 2^63-1).

•Source: Experimental opinion in CP community

Language	Time Limit
Bash	Time Limit: 0.5X
C	Time Limit: X
C++ 17	Time Limit: X
R (Rscript)	Time Limit: 1.5X
C#	Time Limit: 2X
Go	Time Limit: 2X
Java	Time Limit: 2X
Java 8	Time Limit: 2X
Visual Basic	Time Limit: 2.5X
Haskell	Time Limit: 3X
Scala	Time Limit: 3.5X
JavaScript (Rhino)	Time Limit: 5X
JavaScript (Node.js)	Time Limit: 5X
Kotlin	Time Limit: 2X
PHP	Time Limit: 5X
Python 3	Time Limit: 5X

•X - Time Limit mentioned in the problem statement.

Compilation speed of different languages

•Source: Experimental opinion in CP community

Energy Consumption

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

Source: https://greenlab.di.uminho.pt/wp-content/uploads/2017/10/sleFinal.pdf

GCD of Array

Mathematical Problem - 5

Given an array of N positive integers, find GCD of all the array elements.

Example 1: N = 3, arr[] = {2, 4, 6}
Output: 2
Explanation: GCD of 2,4,6 is 2.

Example 2: N = 1, arr[] = {1}
Output: 1
Explanation: Greatest common divisor of all the numbers is 1.

https://practice.geeksforgeeks.org/problems/gcd-of-array0614/1

Mathematical Problem - 6

LCM and GCD

Given two numbers A and B. The task is to find out their LCM and GCD.

Example 1: A = 5 , B = 10
Output: 10 5
Explanation: LCM of 5 and 10 is 10, while their GCD is 5.

Example 2: A = 14 , B = 8
Output: 56 2
Explanation: LCM of 14 and 8 is 56, while their GCD is 2.

A * B = LCM * GCD
LCM = A * B / GCD

https://practice.geeksforgeeks.org/problems/lcm-and-gcd4516/1?page=1&category[]=Mathematical&sortBy=submissions

Mathematical Problem - 7

What is the function of given code?

```
bool fun(int number)
    {
    if (number < 2)
        return false;

    for (int i = 2; i < number; i++)
        {
        if (number % i == 0)
            return false;
        }
    return true;
    }
```

Time Complexity: O(n)

1 is a Prime number or Composite number?

Output: check a number Prime or not.

Ex: N=36

Bitwise operators

Why to use Bitwise operators?

Bitwise operators

What is the output of given code?

```
int main() {

    int x = 3, y = 6;

    cout<<(x & y)<<endl;    // AND
    cout<<(x | y)<<endl;    // OR
    cout<<(x ^ y)<<endl;    // XOR

    return 0;
}
```

If N=5 then what will be value of (N&1) ?

If N=6 then what will be value of (N&1) ?

AND gate

Input A	Input B	Output
0	0	0
1	0	0
0	1	0
1	1	1

OR gate

Input A	Input B	Output
0	0	0
1	0	1
0	1	1
1	1	1

EX-OR gate

Input A	Input B	Output
0	0	0
1	0	1
0	1	1
1	1	0

Odd or Even

Given a positive integer N, determine whether it is odd or even.

(Use Bitwise & operator)

Example:

- ***Input:*** *N = 5*
- ***Output:*** *odd*

https://practice.geeksforgeeks.org/problems/odd-or-even3618/1?page=1&difficulty[]=-2&category[]=Bit%20Magic&sortBy=submissions

Play With OR

You are given an array arr[], you have to re-construct an array arr[]. The values in arr[] are obtained by doing OR(bitwise or) of consecutive elements in the array.

Note: output the last element as it is, because no other element to pair with.

Example:

- ***Input:*** *arr[ ] = {10, 11, 1, 2, 3}*
- ***Output:*** *11 11 3 3 3*
- ***Explanation:***

  At index 0, arr[0] or arr[1] = 11

  At index 1, arr[1] or arr[2] = 11

  ...

  At index 4, No element is left So, it will remain as it is.

https://practice.geeksforgeeks.org/problems/play-with-or5515/1?page=1&difficulty[]=-1&category[]=Bit%20Magic&sortBy=submissions

What is the function of given code?

```
fun2(int x, int y)
    {
        x = x ^ y;
        y = x ^ y;
        x = x ^ y;
    }
```

Output2: Swap the numbers

Single Number

Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1: Input: nums = [2,2,1]
Output: 1

Example 2: Input: nums = [4,1,2,1,2]
Output: 4

Example 3: Input: nums = [1]
Output: 1

$A \oplus A = 0$

$A \oplus A \oplus B = B$

$A \oplus B \oplus A = B$

$A \oplus B \oplus A \oplus C \oplus B = C$

NOTE: $0 \oplus A = A$

https://leetcode.com/problems/single-number/description/

Bitwise operators Continue…

- **Left shift operator (<<)**

```
int main() {

    int x = 3;

    cout<<(x << 1)<<endl;
    cout<<(x << 2)<<endl;

    int y = 4;
    int z = x << y;
    cout << z << endl;

    return 0;
}
```

(x<<y) *equals* **(x * 2^y)**

Right shift operator (>>)

```
int main() {

    int x = 33;

    cout<<(x >> 1)<<endl;
    cout<<(x >> 2)<<endl;

    int y = 4;
    int z = x >> y;
    cout << z << endl;

    return 0;
}
```

(x>>y) *equals* **LB(x / 2^y)**

Hamming Distance

amazon

The Hamming distance between two integers is the number of positions at which the corresponding bits are different.

Given two integers x and y, return the Hamming distance between them.

Example:

- ***Input:*** *A = 10, B = 20*
- ***Output:*** *4*
- ***Explanation:***

  A = 01010

  B = 10100

  As we can see, the bits of A that need to be flipped are 01010. If we flip these bits, we get 10100, which is B.

Constraints:

$0 <= x, y <= 2^{31} - 1$

https://leetcode.com/problems/hamming-distance/description/

Power of Two

Qualcomm

Given an integer n, return true if it is a power of two. Otherwise, return false.

An integer n is a power of two, if there exists an integer x such that $n == 2^x$.

- **Example 1:** Input: n = 1

Output: true Explanation: $2^0 = 1$

- **Example 2:** Input: n = 16

Output: true Explanation: $2^4 = 16$

- **Example 3:**Input: n = 3

Output: false

Expected Time Complexity: O(1)

Hint: Only 1 bit will be set on (in binary format), if a number is divisible by 2.

https://leetcode.com/problems/power-of-two/description/

Sum of two numbers without using arithmetic operators

Given two integers a and b. Find the sum of two numbers without using arithmetic operators.

Example1:

- ***Input:*** *a = 5, b = 2*
- ***Output:*** *7*

Example2:

- ***Input:*** *a = 5, b = 1*
- ***Output:*** *6*

Hint-1: We can use XOR(^) operator to get sum of bits in a and b but without considering any carry

Hint-2: We can use AND(&) operator to calculate positions where a carry is generated

Hint-3: We can use Left-Shift(<<) operator to set positions where a carry need to be added

https://practice.geeksforgeeks.org/problems/sum-of-two-numbers-without-using-arithmetic-operators/1?page=1&difficulty[]=0&category[]=Bit%20Magic&sortBy=submissions

https://www.youtube.com/channel/UCmEBrWO1Z9XDfHZ9s_ZEqxw