Wootech DSA Mentorship

Github Link - https://github.com/WooTechnology/chitra-ds-algo

Resources
Videos - pepcoding course, yt channels - codealittle, takeuforward, apna college, mycodeschool
Questions - leetcode, gfg, hackerearth, hackerrank, striver sde sheet, love babbar sheet, (interview bit) (codeforces, codechef)
Books - ctci, narsimha, (clrs)

Compiler - online(codeforces, coding blocks, geeksforgeeks) , offline(vs code, codeblocks, sublime)
Language- gcc(gnu) C++ 14 or 17

Practice at least 5 ques on each topic
And do at least 7-8 ques daily along with studying at least one new topic

Some tips -
- Read the question first properly and try to dry run with example (to avoid any misunderstanding)
- Try to think of basic brute force solution (generally greater than n*2, n^3 or exponential time)
- Then try to optimize space and time by using the techniques studied till now
- If you are not getting the approach, then read a hint for the question and try again
- Again if not getting anywhere, then read the editorial and code it yourself
- If there are errors in code, try to fix them yourself(don't use debuggers tools) (you can fix them by taking different examples and checking if its working or not), if not see the solution
- If thinked of the solution yourself and the code gets accepted, after that also read the editorial because sometimes it happens that you have solved it in a different way and there are 3-4 approaches available. It is necessary to try them all.
- **It is important to know solution from brute force to most optimised because in interviews you have to explain all the approaches**

All this takes time and practice, just be consistent and keep practicing!

Week 1 - Basics of Programming

How to take infinite input

If we don't know how many integers are given for input

```
int x;
while(cin>>x)
{

}
```

If it is given that -1 is present at the last of input

```
while(true)
{
    int x;
    cin>>x;

    if(x==-1)
        break;

}
```

Fast I/O

```
ios_base::sync_with_stdio(0);
cin.tie(0);
cout.tie(0);
```

Prime number

Segmented sieve

Given l, r as ranges and we have to find the prime number that lie between l and r

max value of r = 10^12

r-l <= 10^6

```
n= r-l+1;
a[n];
l, l+1, l+2, .. , r
0, 1, 2,      , n-1
```

Code-

```
for(int i=0;i<n;i++)
{
        x = i+l;
        for(int j=0;j<v.size();j++)
        {
                if(x%v[j] ==0)
```

```
            {
                f=0;
                break;
            }
        }
}
```

Bitwise operators

Consider 2 integers a and b

a= 9 1001

B = 14 1110

or (|) A|b = 1111 15

And (&) a&b = 1000 8

Not (~) A = 0110 6

Xor (^) a^b = 0111 7

<< (*2) a<<1 10010

>> (/2) a>>1 0100

Some properties of these operators

1 | n = 1

0 | n = n

n | n = n

0 & n = 0

1 & n = n

n & n = n

1 ^ n = ~n

0 ^ n = ~n

n ^ n = 0

How to find whether a number is power of 2 or not

```
while(n%2 ==0)
{
        N = n/2;
}

if(n==1)
        Return true;
Else
        Return false;
```

If we have to find this in O(1) constant time

N-1 0111111
N 1000000

N&(N-1) 0000000
N^(N-1) 1111111 problem is that we also have to find this number

n&(n-1) ==0
n^(n-1) == ()

To count number of set bits
cout<<__builtin_popcount(n);
Read more builtin function from gfg

Parity - count number of set bits in a number and find it's parity accordingly
Even = 0
Odd = 1

Some STL functions used in number theory
min(a,b)
min(a,min(b,c))
max(a,b)
__gcd(a,b)
LCM = a*b/__gcd(a,b)

- Print pascal triangle till n rows. For n=5 given below

```
   1
  11
 121
1331
14641
```

0C0 1
1C0 1C1 1 1
2C0 2C1 2C2 1 2 1

Code:

```
vector<int> a;
a.push_back( 1);
a.push_back(1);
```

```
for(int i=2;i<=n;i++)
{
        // print spaces according to row number

        vector<int> b;
        b.push_back(1);
        cout<<1<<" ";
        for(int j=0;j<a.size()-1;j++)
        {
                b.push_back(a[i]+a[i+1]);
                cout<<a[i]+a[i+1]<<" ";
        }
        b.push_back(1);
        cou<<1<<" \n";
        A = b;
}
```

Modulus operator : %
a%b = 5%3 = 2
1%5 = 1
(-a)%b = (b-a)%b = -2%5 = 3%5 = 3

(a+b)%m = (a%m + b%m)%m
(a-b)%m = (a%m - b%m +m)%m
(a*b)%m = ((a%m)*(b%m))%m
(a/b)%m = ((a%m)*(b^-1 %m))%m

Read about modulo inverse (euclid - extended euclid)

Fibonacci - 0 1 1 2 3 5 8 13
f(n) = f(n-1) + f(n-2)

4 methods of finding nth fibonacci number
Recursive - O(2^n) O(n)(stack space)
Iterative - O(n) DP - O(n), else O(1)
Matrix - O(log n) O(log n)
Binet's formula - O(1) O(1)

Matrix
1 1 1 1 = 2 1 1 1 = 3 2
1 0 1 0 1 1 1 0 2 1

Try to code it (refer gfg)

Binet's formula -
Fn = {[(√5 + 1)/2] ^ n} / √5

Tribonacci -- 0 0 1 1 2 4 7

Catalan number = 2nCn / (n+1)
(n+1) = (0)(n) + (1)(n-1) + (2)(n-2) + … (n)(0)

N , 5
floor()
(n/5) + (n/25) + (n/125) ---

Fast expo
a^b
B even a^(b/2) * a^(b/2)
B odd a^(b/2) * a^(b/2) * a

a=5, b=7
5^7 = 5^(3) * 5^(3) * 5
5^3 = 5^(1) * 5^(1) * 5

A = a%m
B = b%m

5^7 = 5^(3) * 5^(3) * 5
5^3 = (5^(1)%m * 5^(1)%m * 5)%m

Week 2 - Arrays

Arrays -
Int a[n]; // array declaration of size n
vector<int> v; // vector declaration of size 0
vector<int> v(n); // vector declaration of size n
vector<int> v(n,0); // vector declaration of size n and initialize all elements with 0

Repeating and missing number in array in range 1-n
5 1 4 2 3 4

// try to map element to index
i=0

```
while(i<n){
        if (a[i]==i+1 && a[i]!=a[a[i]-1])
                swap(a[i],a[a[i]-1]);
        Else
                I++;
}
```

3 1 4 2 5 4
4 1 3 2 5 4
2 1 3 4 5 4
1 2 3 4 5 4
Repeating - 4
Missing - 6

Ternary search -

Binary search -
0 - n l,r
M = (l+r)/2
R = m-1
L = m+1
O(log 2 n)

Ternary search -
0 - n l,r
M1 = l+ (r-l)/3
M2 = r- (r-l)/3

L - m1, m1+1 - m2, m2+1 - r
Time complexity - O(log 3 n)

Maximum size array
10^ 8 - boolean - bitset
10^7 - global
10^ 6 - function(main also)

Count sort -
0 n 0 - 6
1 3 4 5 6 2 3 4 5 3 2 4 5

A[7] = {0, 1, 2, 3, 3, 3, 1};

Merge overlapping intervals
Que - [1,3], [1,4], [2,3],[4,5]
Ans - [1,4],[4,5]

```
vector<pair<int,int>> v;
for(int i=0;i<n;i++)
{
        //Take input
}

sort(v.begin(),v.end());
// complete rest code
```

Largest consecutive subsequence
2 3 6 1 4 5 8 9 10

1 2 3 4 5 6 8 9 10

Try to find the sol in best complexity -
Best - O(n)
Min
Max
Repeating
Max-min+1 = length

Kadane's algorithm
Array[n] = 3 0 -2 3 -4 2 1 -2 4
Find the maximum subarray sum

Brute force - try all subarrays(n*n) - calculate their sum and take max
O(n*n*n)

Prefix sum = 0 -2 1 -3 3 4 2 6
Sum[l,r] = p[r]- p[l-1]
O(n*n)

Code :

```
Int csum=0,msum=0;
for(int i=0;i<n;i++)
{
        Csum += a[i];
        if(csum<0)
```

```
            Csum = 0;
        msum= max(msum,csum);
}

// if all negative or 0
if(msum==0)
{
        Int d=a[0];
        for(int i=0;i<n;i++)
        {
            d = max(d,a[i]);
        }
        msum=d;
}
cout<<msum;
```

Stock buy sell problem
One time buy and one time sell

Array = 2 4 6 3 5 3 5 1
Find max profit
profit= sell - buy

Suppose you buy on ith day and sell on jth day then i<j

2 4 6 3 5 3 5 1
Min till now (i-1)
And if we sell on ith day then take max of profit

Do its 6 variations from pepcoding videos after you study DP
(solution includes DP, that's why)
Do only this one now - https://www.youtube.com/watch?v=4YjEHmw1MX0

Links for rest variations
https://www.youtube.com/watch?v=3YILP-PdEJA
https://www.youtube.com/watch?v=HWJ9kIPpzXs
https://www.youtube.com/watch?v=wuzTpONbd-0
https://www.youtube.com/watch?v=pTQB9wbIpfU
https://www.youtube.com/watch?v=GY0O57llkKQ

Array - 0 2 4 6 3 5 1
Diff arr - 0

Diff - A[i]-a[i-1]

Prefix sum = p[i] = p[i-1]+a[i]

Sum in range[l,r] = p[r]-p[l-1]

Update [l,r] by k
L r k
2 4 5
1 6 -2

Q queries and n length
q*n not valid

Array -a 0 2 4 6 3 5 1
query
L r k
2 5 3
Final array after updation - 0 2 7 9 6 8 1

Diff arr -d 0 2 5 2 -3 2 -7
B = final elements after updation

```
while(q--)
{
        Int l,r,k;
        cin>>l>>r>>k;
        d[l]+=k;
        if(r+1<n)
                D[r+1] -=k;
}
Int s=0;
for(int i=0;i<n;i++)
{
        b[i] += s+d[i];
        S += d[i];

}
```

(code written in meeting was correct)

Matrix - 90 deg rotation clockwise
1 2 3 7 4 1
4 5 6 8 5 2

7 8 9 9 6 3
transpose
1 4 7
2 5 8
3 6 9

Print the matrix in spiral form
1 2 3 6 9 8 7 4 5

Staircase search
Matrix n*n rows and columns sorted and we have to find an element k

Linear traverse O(n*n)
Binary search on every row/column - O(n*logn)

1 2 6
3 4 8
5 7 9

K = 8

Set matrix zero
Set every row and column zero if that element is zero

1 0 3 5 0
2 4 0 6 4
1 5 1 2 5

Actual ans
0 0 0 0 0
0 0 0 0 0
1 0 0 2 0

Brute force - O(n*m*(n+m)) time, O(n*m) space
Time optimized - O(n*m), O(n+m)
Space optimized - O(n*m), O(1)

(Try to solve in above both complexities)

Median of row wise sorted matrix
Time = O(n*m*(_)
Space O(1)

Kth smallest element
Let k=3 => ans = 3
1 2 6
3 4 8
5 7 9

Sliding window

0 1 2 3 4 5 6 7 8 9 10
Size j-i+1
I =1, j=4 => i=2,j=5

K - size
N total

N-k+1 = total subarrays of size k
I j
0 k s
1 k+1 s + a[j] - a[i-1]
2 k+2
3 k+3
…
n-k n

Array -only 0 and 1
Find Max subarray size all 1
0 0 1 0 1 1 1 0 1 1 ans=3

---

k th root of number n
k=3 n=7
(Ans)^k = n

Square root of n
STL functions - sqrt(n), cbrt(n)

0 - n
k*k <= n
K should be max

5
2

We use Binary search
0 1 2 3 4 5 6 7 8

Code -

```
while(L<=r)
{
        M = (l+r)/2

if(pow(m,k) <=n)
        ans=m, l = m+1;
Else
        R = m-1;
}
```

Unique number 1 = All elements appear twice except 1 element and find that element
Unique number 2 = All elements appear twice except 2 element and find those element
Unique number 3 = All elements appear thrice except 1 element and find that element

Try these questions
Required space O(1) and time O(n) (can be O(n*(log (base 10)n)))

Armstrong -
23 = 2*2 + 3*3
1234 = pow(1,4) + pow(2,4) + pow(3,4)+ pow(4,4)

Find min of more than 2 numbers - min({a,b,c)}

Lower bound - lower_bound(a,a+n,k) - a
Upper bound - upper_bound(a,a+n,k) - a

1 2 4 4 4 5 6 7
K = 4

Array find pairs which leads to sum k - 2 pointers

Sorting
Stl - sort(a,a+n)
sort(v.begin(),v.end())

sort(a,a+n, greater<int>)

Suppose vector of string , you have to sort according to length

```
Bool comp(string a, string b)
{
        Return a.length()<b.length();
}

main()
{
        vector<string> v(n);

        sort(v.begin(),v.end(), comp);

}
```

---

priority_queue<int> pq; - min heap

priority_queue<int,vector<int>, greater<int>> pq; -max heap

```
Class comp
{
Public:
        Bool operator()(int a, int b)
        {

        }
};
```

priority_queue<int,vector<int>, comp> pq; -custom heap

Product array puzzle

N = 5
A[] = {10, 3, 5, 6, 2}
Output: 180 600 360 300 900

Left = 10 , 30, 150, 900, 1800
Start iterating from right
0, 0, 0, 300, 900

```
Int r=1;
```

```
for(int i=n-1;i>=0;i--)
{
        if(i>0)
                ans [i] = r*left[i-1];
        Else
                ans[i]=r*1;

        r = r*a[i];
}
```

Inplace sort
Stable and unstable sort

1 3 2 2 4 6 2*

1 2 2 2* 3 4 6 - stable
1 2* 2 2 3 4 6- unstable

Week 3 - Strings and Greedy

Rotation of string
Abcdef

cdefab

Subsequence -
Abcdefghij
Cgh
Afi

{}	1	nC0
A b c	n	nC1
Ab ac ad ae ,,... bc bd be bj..	_ _	nC2
Abc,		
….		
Abcdefghij	1	nCn
	total	2^n

Subsequence - 2^n

A b c d

b, b d , a d, a b c d, ()

_ _ _ _
0 0 0 0 0 {}

0 0 0 1	1	a
0 0 1 0	2	b
0 0 1 1	3	a b
0 1 0 0		c
0 1 0 1		a c
0 1 1 0		b c
…		
1 1 1 0		b c d
1 1 1 1	2^n -1	a b c d

Code -
Array n elements

```
Int m = ( 1<<n);
for(int i=0;i<m;i++)
{
        int x = i;
        for(int j=0;j<n;j++)
        {
                if((x&(1<<j)) >0)
                {
                        cout<<a[j]<<" ";
                }
        }
        cout<<endl;
}
```

Time complexity - O(n* 2^n)

0 1 0 0		
j=0	0 0 0 1	0
j=1	0 0 1 0	0
j=2	0 1 0 0	1
j=3	1 0 0 0	0

Substring
Abcdefghij

Cfg not a substring

Cde

Abc
Abcdef

A b c d e f		n
Ab bc cd de ef		n-1
Abc bcd cde def		n-2
Abcdefghi	bcdefghij	2
Abcdefghij		1
	Total =	n(n+1)/2

Permutations

Abcdabcd

Aabbccdd
Aabbcddc
Aabbcdcd

Find next permutation of this number
124631

Anagrams of string

Longest Palindrome substring in a string
Brute force - check all substrings time- O(n*n*n) space- O(1)

abcdcfgh
Go to every index -
Odd length - consider it as center and j=i-1,k=i+1 and j--,k++ check s[j]==s[k]
Even length - j=i,k=i+1, and j--,k++, check s[j]==s[k]
Time - O(n*n)

Longest common prefix
Abcdef
Abcfde
Abc
Ab

Ans = ab
Time - O(n*k)

Pattern matching
String - aaaabcdaaddccbfdf
Pattern - abcd

Basic brute - O(n*k)
Kmp
Rabin karp
Boyer Moore Algorithm
Just read and understand for now

GREEDY

Fractional knapsack

Bag - 11	W
Weights - 2, 5, 7, 8	w1, w2, w3, w4
Values - 1, 4, 2, 3	v1, v2, v3, v4

Find max value

Find value/weight

½ ⅘ 2/7 ⅜ - value for 1 unit of weight
Take max

11

⅘	11-5=6	4
½	6-2=4	4+1=5
2/7	4 - 4 = 0	5 + (4*2/7)

Min number of flips
`0001010111`

0001010111
0101010101
1010101010

WEEK - 4 LINKED LIST

Rotation of linked list in groups of k
Rotate a ll by k nodes to left = rotate a ll by n-k nodes to right
10->20->30->40->50->60 k=2
Right - 50->60->10->20->30->40
Left - 30->40->50->60->10->20

Delete a given node when pointer to node is given (O(1) time)

10->20->30->40->50->60

Find intersection point of y linked list

10 -> 20 -> 30 -> 40 -> 50 -> 60

5 -> 25 -> 40 -> 50 -> 60

STACKS AND QUEUES

Infix a+b

Prefix +ab

Postfix ab+

(a+b)*c - d*e/f

Postfix = ab+c*de*f/-

Prefix = -*+abc/*def

ab+c*de*f/-

54+3*21*1/-

Stack - 25

---

2 stacks in array

M stacks in array

Next greater element in right

1 4 2 6 7 4 5 6

2 6 6 7 -1 5 6 -1

a[i]<a[j] i<j and j should be min

And A[j] should be min

WEEK 5 - RECURSION

Recursion

```
Void func(int n)
{
        Base condition

        Return func(n-1);
}

int fib(int n)
```

```
{
        if(n<=1)
                Return n;
        Return fib(n-1) + fib(n-2);
}
```

Time - O(2^n)

Memory - O(1) without call stack

O(n) with call stack

```
                                        fib(n)
                        fib(n-1)                fib(n-2)
                fib(n-2)        fib(n-3)
        fib(n-3)        fib(n-4)

        fib(2)
fib(0)  fib(1)                                          n levels - 2^n-1
```

Memory -

3 type

Data

Heap

Stack

..

Fast expo

Gcd -

```
Int gcd(int a, int b)
{
        if(a==0)
                Return b;

        Return gcd(b,a%b);
}
```

Sorting algo

Binary search

Tower of hanoi

Src = 1

Des = 3

Aux =2

N =4

```
Void hanoi(int n, int src, int des, int aux)
{
        if(n==1)
        {
                cout<<"Move one disc from "<<src <<" to "<<des;
                return;
        }

        Hanoi (n-1, src, aux, des);
        cout<<"Move one disc from "<<src <<" to "<<des;
        Hanoi (n-1, aux, des, src);
}
```

WEEK 6 - HASHMAPS

Hashing

Int -

Table - a[n][2]

1 2 3 2 4 1 4 5 count sort -- c

C[3] = 1

C[key] = value

cout<<c[key];

keys	values
1	2
2	2
3	1
4	2
5	1

Stl - unordered_map< string , vector<int> > arr;

Arr[1] = 2;
Arr[3] =2;
…
arr.insert(make_pair(3,2));

Keys - int, long int, char, string

Values - int, long int, char, string, vector, pair

Size - table_size

Abc bcd abcd abcd asd

Keys must be unique

Hashmaps -
Search
Insert
Delete -
O(1) best , avg
Worst - O(n) O(1)

Insertion
H[key] = value
h.insert(make_pair(key,value))

Deletion
h.erase(key)

Search
H[key]

h.count(key) 1 or 0
h.find(key) h.end() - if not present

How to find the index of key
Hash function - input key - return index

Table size = n = 10
Int,int
19,20
23,56
34,54
73, 76

key%n = 19%10 = 9
23 == 3
34 == 4
73 == 3

Collision -
Open hashing, chaining, open addressing, linear probing, double hashing

Hash function
Modulo with prime number - 11

2 primes - p1, p2
P2 = nearest to table size
P1 = nearest to data/input size

Basic implementation
Hash function, double hashing example

Hashmaps - unordered_map array O(1)
Map - bst O(logn)

2 sum
Find the pairs which has their sum as target

Basic = O(n2)
Sorting = O(nlogn)
Hashmaps = O(n)

Sum = 8
1 3 5 4 2 1 6 7

```
Int c=0;
unordered_map<int,int> h;
for(int i=0;i<n;i++)
{
        Int x = sum - a[i];
        if(h.count(x) >0)
                C += h[x];
        H[a[i]]  += 1;
}
```

C = 4 i = 7

1	2
3	1
5	1
4	1
2	1

Count subarrays with 0 sum
2 -4 2 4 -6 -3 2

2 -2 0 4 -2 -5 -3

Intersection of 2 arrays
Basic = O(n2)
Sorting = O(nlogn) 2 pointers = i and j
Hashmap = O(n)

Heaps -
Complete binary tree

0 based indexing
Curr = i
Parent = floor((i-1)/2)
Child = 2*i+1, 2*i+2

1 based indexing
Curr = i
Parent = floor(i/2)
Child = 2*i, 2*i+1

10 15 30 40 50 100 40
0 1 2 3 4 5 6

priority_queue = priority - max element max-heap
priority_queue = priority - min element min-heap

Heaps
Insert - add new node to end and up heapify O(logn)
Delete - delete top node and down heapify O(logn)
Space - O(n)

```
Priority-queue <int> pq;                    // max heap
pq.push(0);
pq.pop();
Pq.top;

Class comp
{
Public:
```

```
        Bool operator()(int a,int b)
        {
                //comp
        }
};
Priority-queue <int, vector<int>, greater<int>> pq;          // min heap
Priority-queue <int, vector<int>, comp> pq;                  // custom heap
Priority-queue <pair<int,int>, vector<pair<int,int>>, comp> pq; // custom heap

sort(a,a+n, greater<int>())
```

Heap sort
Push all elements in min heap
Pop one by one and add to array

K th max min element in array
Time - O(nlogn + klogn)
Space - O(n)

Time - O(nlogk + (n-k+1)logk)
Space - O(k)
Max element - min heap

K elements
1 pop - kth max
2nd element - k-1 th max

1, 2, 3, …, n-k-1,n-k,n-k+1,... n

Running stream
Input size - very large , you can't store it anywhere

Kth Max element in running stream

Merge k arrays in sorted arrays of same size
Brute force - add all to single array and sort
Min heap of k size

Value, ind, array ind
pair<int,pair<int,int>>
Struct

WEEK 7 - TREES

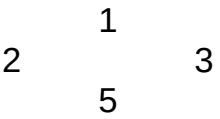
Trees

Non - linear data structure
Hierarchical nature
Binary and n-ary tree
Bst
Different types - full, perfect, complete
Traversals - in, pre, post, level dfs, bfs
A tree is a graph
Balanced binary tree

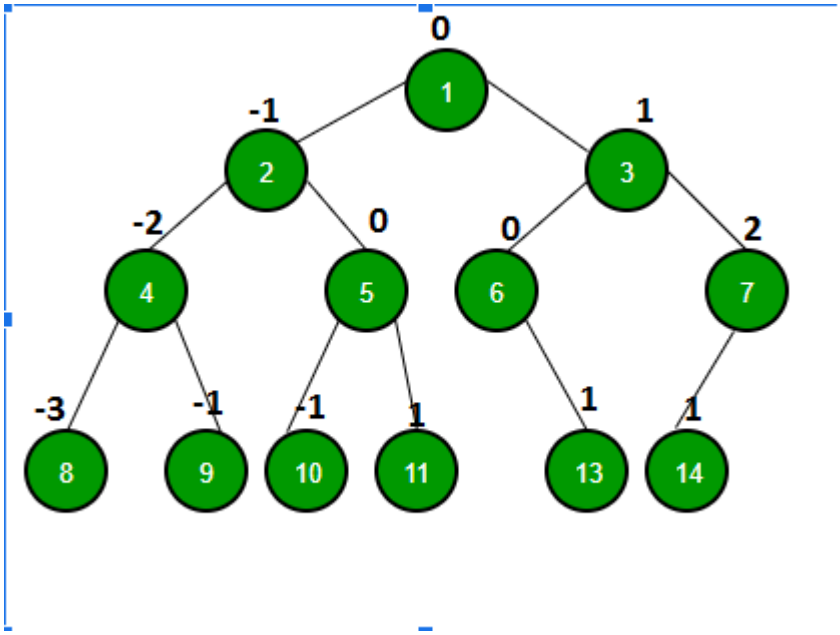
N - nodes n-1 edges => tree (directed)
N - nodes x edges => graph

```
Class node{
Public:
        Int val;
        node* left;
        node* right;
        node(int d)
        {
                Val = d;
                Left  right = NULL;
        }
};
Class tree{
public:
        node* root;
        // FUNCTIONS
};
```

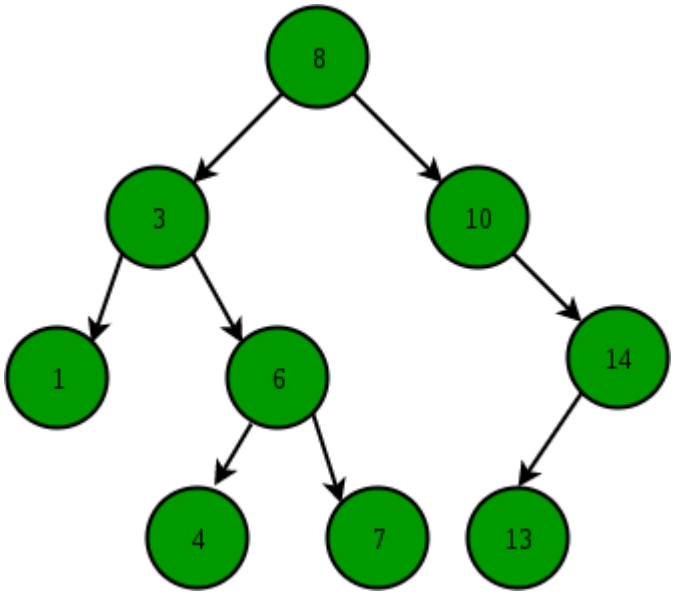
(a) Inorder (Left, Root, Right) : 4 2 5 1 3
(b) Preorder (Root, Left, Right) : 1 2 4 5 3
(c) Postorder (Left, Right, Root) : 4 5 2 3 1



Que. level order and pre order are given, can you construct a unique binary tree



Left view: 1, 2, 4, 8
Right view: 1, 3, 7, 14
Top view: 8, 4, 2, 1, 3, 7
Bottom view: 8, 4, 10, 6, 14, 7



Left: 8, 3, 1, 4
Right: 8, 10, 14, 13
Top: 1, 3, 8, 10, 14
Bottom: 1, 4, 6, 13, 14

Level - 8, 3, 10, 1, 6, 14, 4, 7, 13
Level spiral - 8, 10, 3, 1, 6, 14, 13, 7, 4

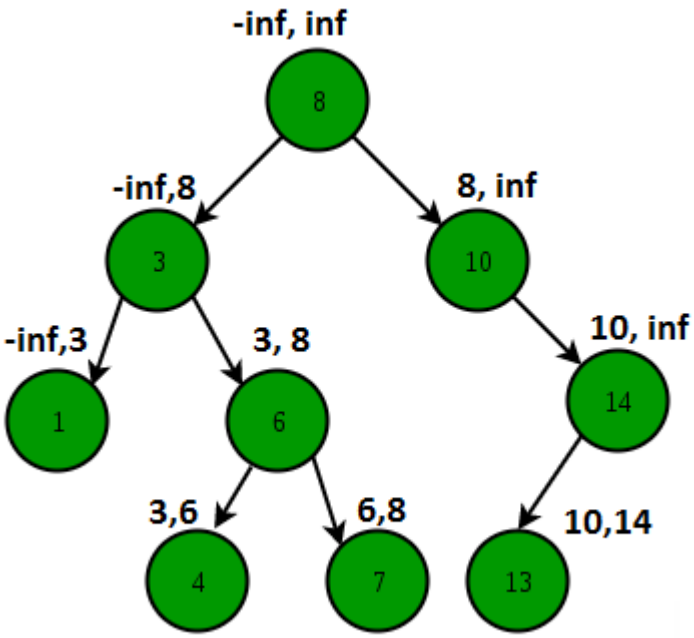
Lca - lowest common ancestor
3 cases -

Diameter of tree - 3 cases
Identical or mirror trees

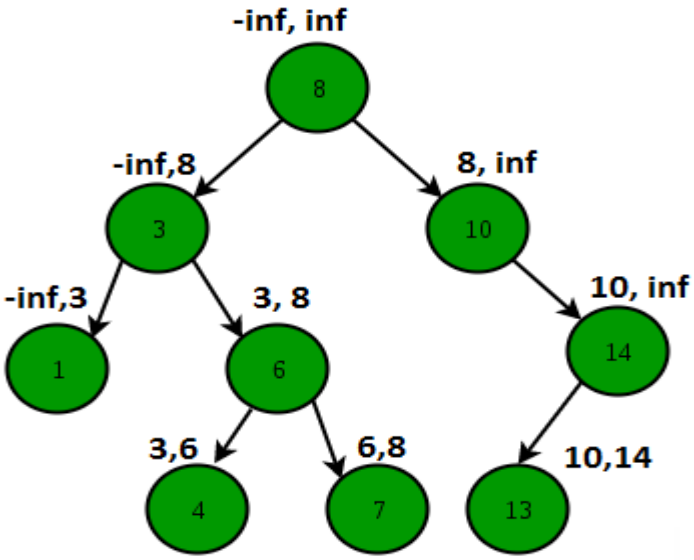
Construct tree from inorder and preorder -
Find root from preorder and then check in inorder left and right subtree and do same for them also

BST - binary search tree
Search
Insert
Delete
Inorder successor
Inorder predecessor

Check if binary tree is bst or not
By using preorder



MEETING - TREES & GRAPHS

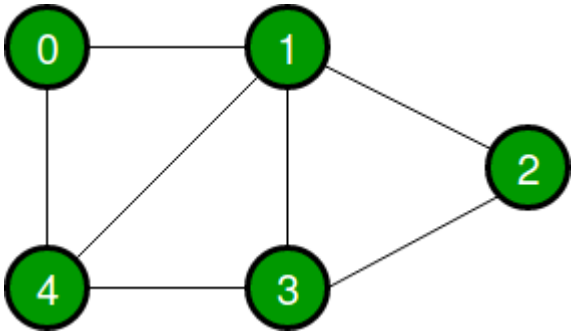


Tree construction
Level order
Pre order 8 3 1 N N 6 4 N N 7 N N 10 N 14 13 N N N

node* root
root->left root->right root->val

N nodes/vertices
Tree - n-1
Graph - (n-1) - (n(n-1)/2)

Matrix = O(v*v)
List = O(e)



Bfs - 0, 4, 1, 3, 2
Dfs - 0, 4, 1, 3, 2

Adj list

0 -> 1,4
1 -> 0,2,3,4
2 -> 1,3
3 -> 1,2,4
4 -> 0,1,3

N - vertices
M - edges

```
vector<vector<int>> v(n+1)
for(int i=0;i<m;i++)
{
        Int x,y;
        cin>>x>>y;
        v[x].push_back(y);
        v[y].push_back(x);
}
```

GRAPHS

Fig. 7.7 A directed graph

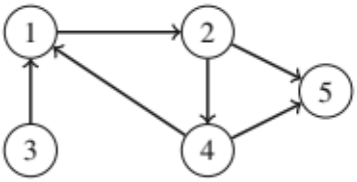


Fig. 7.8 A weighted graph

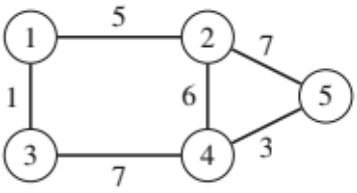


Fig. 7.9 Degrees of nodes

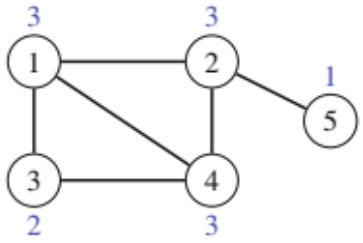


Fig. 7.10 Indegrees and outdegrees

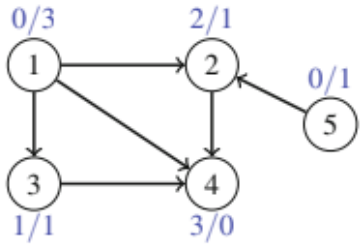
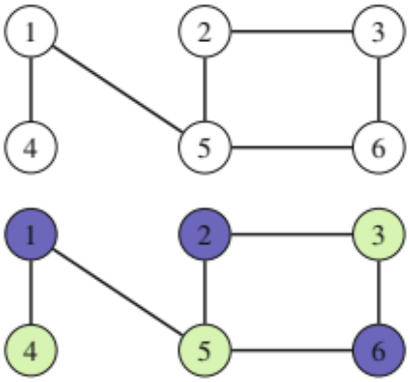


Fig. 7.11 A bipartite graph and its coloring



Connectivity Check A graph is connected if there is a path between any two nodes of the graph. Thus, we can check if a graph is connected by starting at an arbitrary node and finding out if we can reach all other nodes.

Cycle Detection A graph contains a cycle if during a graph traversal, we find a node whose neighbor (other than the previous node in the current path) has already been visited.

Bipartiteness Check The idea is to pick two colors X and Y , color the starting node X, all its neighbors Y , all their neighbors X, and so on. If at some point of the search we notice that
two adjacent nodes have the same color, this means that the graph is not bipartite.
Otherwise the graph is bipartite and one coloring has been found.

Fig. 7.3 A cycle of three nodes

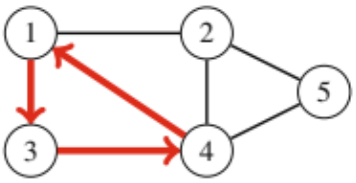


Fig. 7.4 The left graph is connected, the right graph is not

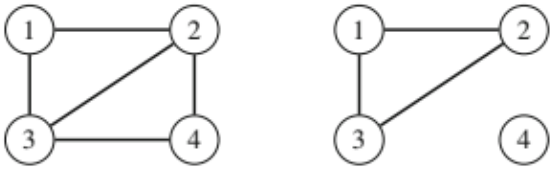


Fig. 7.5 A graph with three components

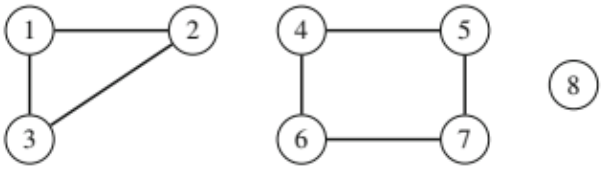
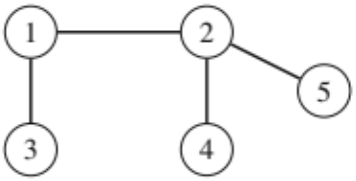


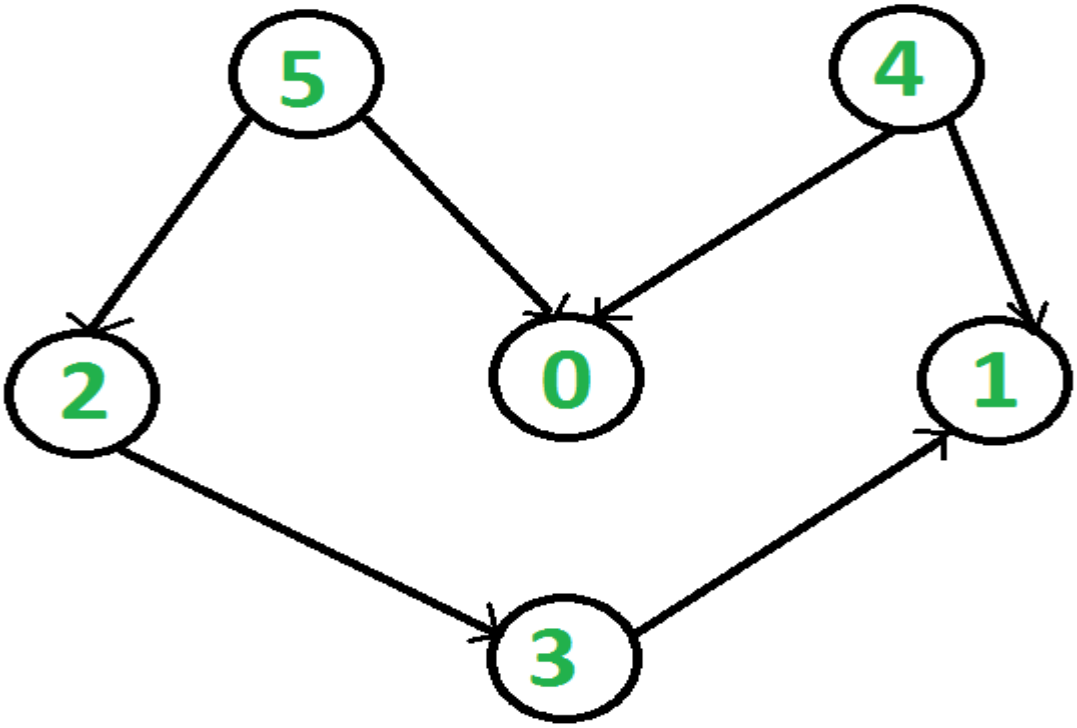
Fig. 7.6 A tree



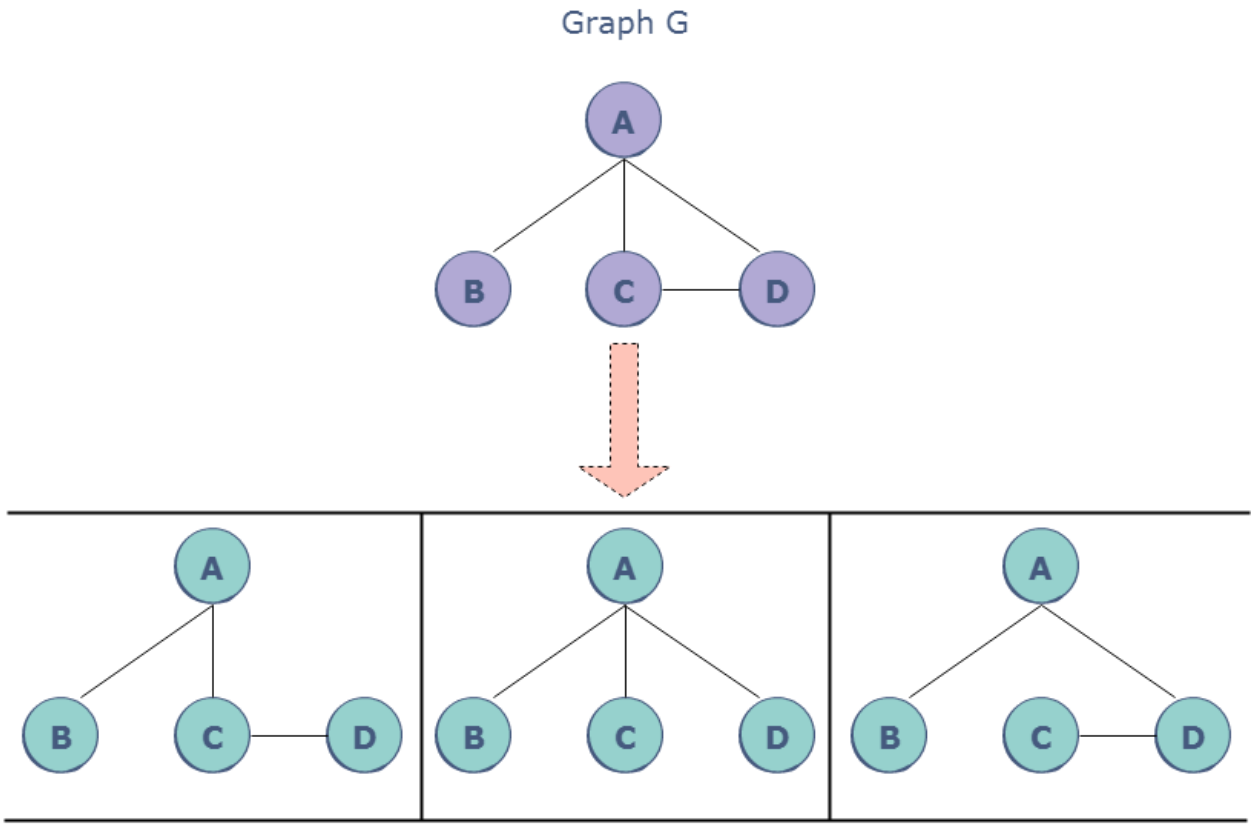
Complete graph - edges = n*(n-1)/2
Indegree, outdegree

https://practice.geeksforgeeks.org/problems/steps-by-knight5927/1
https://practice.geeksforgeeks.org/problems/rat-in-a-maze-problem/1

Topological sort - https://www.geeksforgeeks.org/topological-sorting/



5 4 2 0 3 1
4 5 0 2 3 1
4 5 2 0 3 1

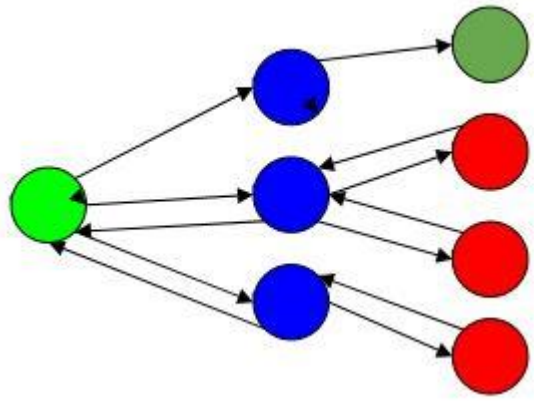


Kruskal - https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

Prim - https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/

Recursion
https://www.geeksforgeeks.org/recursion/

Backtracking
Consider a situation where you have three boxes in front of you and only one of them has a gold coin in it but you do not know which one. So, in order to get the coin, you will have to open all of the boxes one by one. You will first check the first box, if it does not contain the coin, you will have to close it and check the second box and so on until you find the coin. This is what backtracking is, that is solving all sub-problems one by one in order to reach the best possible solution.



Sudoku

Check every empty position , consider every no 1-9 - can place or not
Can place - then place and continue
Not - return

N - queen
Place n queens on a chessboard of size n*n such that they don't attack each other

---

WEEK - 8 DYNAMIC PROGRAMMING

Factorial -

```
Int a[n+1] = {};
In fact(int n)
{
```

```
        if(n<1)
                Return 1;
        if(a[n]>0)
                Return a[n];
        A[n] = fact(n-1)*n;
        Return a[n];
}
```

1	1	2	6	24	120

T test cases - **10^5**
N factorial **10^6**

```
Int a[n+1] = {};
memset(a,a+n+1,-1);                          // check syntax
int func(int n)
{
        if(n<=1)
                Return n;
        if(a[n-1] ==-1)
                A[n-1] = func(n-1);
        if(a[n-2]==-1)
                A[n-2] = func(n-2);

        Return a[n-1] + a[n-2];
}
```

Time- 2^n

0 1 1 2

func(n)

func(n-1) func(n-2)

func(n-2) func(n-3)

f(0) f(1)

Memoization - storing the output of function calls so that it will not be called again in future.

Dp - bottom up = recursion + memoization
Top down = iterative filling of that array

Knapsack -

Values and weights total weight

Fractional - we can divide the weight
Bounded - 0/1
Unbounded -

0/1 knapsack
Bag - 11 W

Weights - 2, 5, 7, 8 w1, w2, w3, w4
Values - 1, 4, 2, 3 v1, v2, v3, v4
Find max value

Make all possible subsets , take max of those which have weight <= 7
{}

2	1
5	4
7	2
8	3
2,5	5
2,7	3
2,8	4
5,7	6
5,8	7
7,8	5
2,5,7	7
2,7,8	6
5,7,8	9
2,5,7,8	10

```
Int m=0;
Int n,W;
Int w[n];
Int v[n];
Void fun(int i, int cw, int cv)
{
        if(cw>W)
                return;
        if(i==n)
        {
                M = max(m,cv);
```

```
            return;
        }
        m= max(m,cv);
        fun(i+1,cw,cv);                 // current item excluded
        fun(i+1,cw+w[i],cv+v[i]);       // current item included
}
fun(0,0,0);
cout<<m;
```

Time- O(2^n)

```
Int a[n+1][W+1] = {};
Void fun(int i, int cw, int cv)
{
        if(cw>W)
            return;
        if(i==n)
        {
            M = max(m,cv);
            return;
        }
        if(a[i][cw] >0)
            Return a[i][cw];

        A[i][cw] = fun(i+1,cw,cv);              // current item excluded
        A[i][cw] = max(a[i][cw], fun(i+1,cw+w[i],cv+v[i]));     // current item
included
}
```

0 1 2

0	0	0	0				

https://www.youtube.com/playlist?list=PL_z_8CaSLPWekqhdCPmFohncHwz8TY2Go

Must watch 5 videos (1-5)

Watch this before the next meeting , friday -

GREEDY

Min platforms needed

pair<int,int>

Dept time, platform

K - 3

9:00 9:10
9:40 12:00
9:50 11:20
11:00 11:30
15:00 19:00
18:00 20:00

heap
20:00 3
12:00 1
19:00 2

Job sequencing

N = 4

Jobs = (1,4,20)(2,3,10)(3,4,40)(4,1,30)

Profit desc sorted - deadline asc sorted 3,4,1,2

1-2	2-3	3-4	4-5
4	2	1	3

Max product subsequence

Array of n integers

All positive = product of all except 0

All negative = take product of all if n is even else leave smallest one

+ve and -ve = above all

Longest increasing subsequence

Array n integers -

2 5 **3** 6 8 **4 6**

You can't change order of elements

1	2	2	3	4	3	4
2	5	3	6	8	4	6

Initialize the array with 1

```
for(int i=0;i<n;i++)
        for(int j=i-1;j>=0;j--)
                if(a[i]>a[j])
                        A[i] = max(a[i], a[j]+1)
```

5	3	1	4	6	8	2	7
1	1	1	2	3	4	2	4

0	1	2	3	4	5	6	7
0	1	2	2	3	4	2	4

8 6 4 1

Time - O(n*n)

Space O(n)

Longest common subsequence

5 3 6 8 4 6

5 3 1 4 **6 8**

5 3 6 8

5 3 4 6

Edit distance

2 strings given

3 operations -

Insert =

Delete =

Update =

0	1	2	3	4	5

Coin change

Sum given, Coins given

Sum can be formed by using those coins

Coins unlimited

Min coins

28 1,2,5,10,20
20, 5, 2, 1

10 1,2,3,5,6
6, 3, 1

Number of ways to form m using n coins

1,2,5,6 a
1, 2,5

1	1	2	2	3	4	6

0 1 2 3 4 5 6

1	1					
2	1 +1	2				
3	1 + 1 +1	1+2				
4	1 + 1+ 1+1	1+1+2	2+2			
5	1 +1+1+1+1	1+1+1+2	1+2+2	5		
6	1+1+1+1+1+1	1+1+1+1+2	1+1+2+2	2+2+2	1+5	6

```
for(int i=0;i<n;i++)
        for(int j=a[i],j<=m;j++)
                dp[j]+=dp[j-a[i]];
```