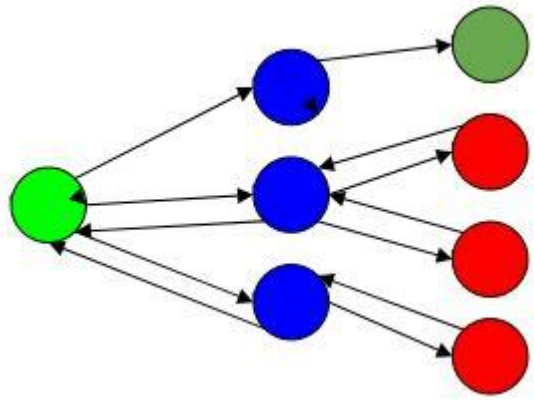
Recursion

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]];
```