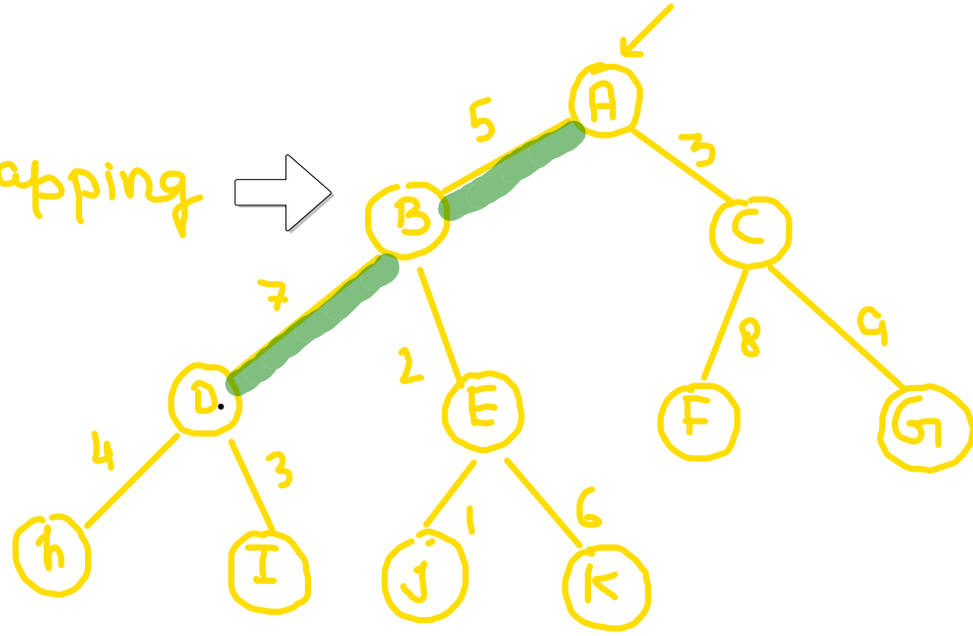
Optimization Problems

Dynamic Programming :-

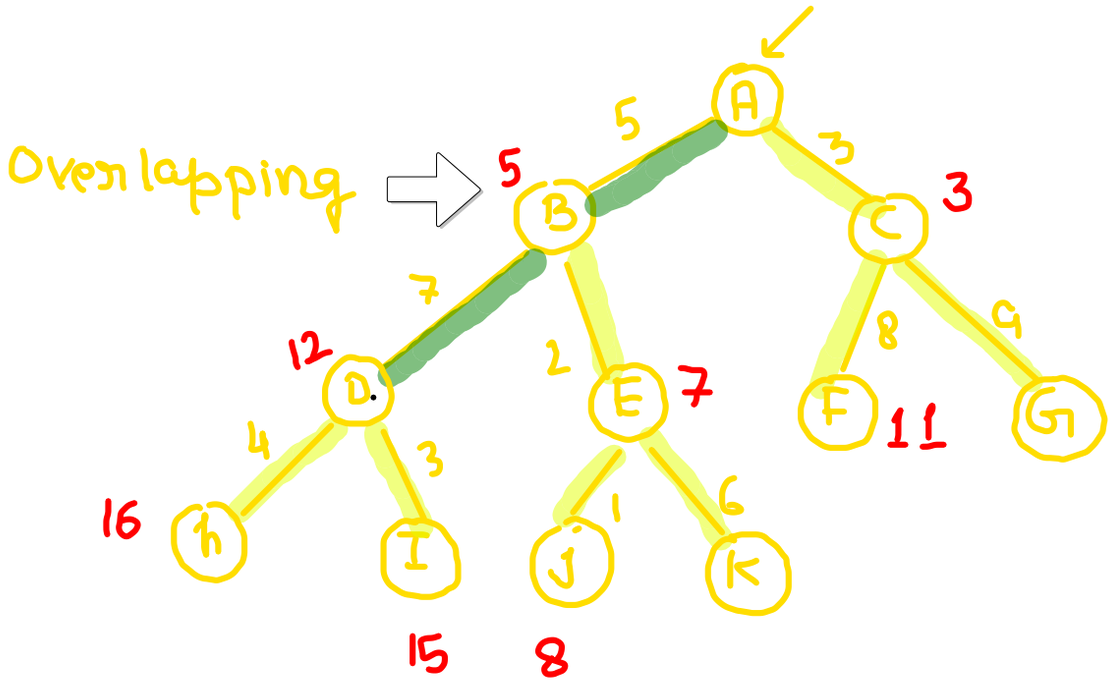
Greedy Technique

Overlapping Substructures

Overlapping ⇨



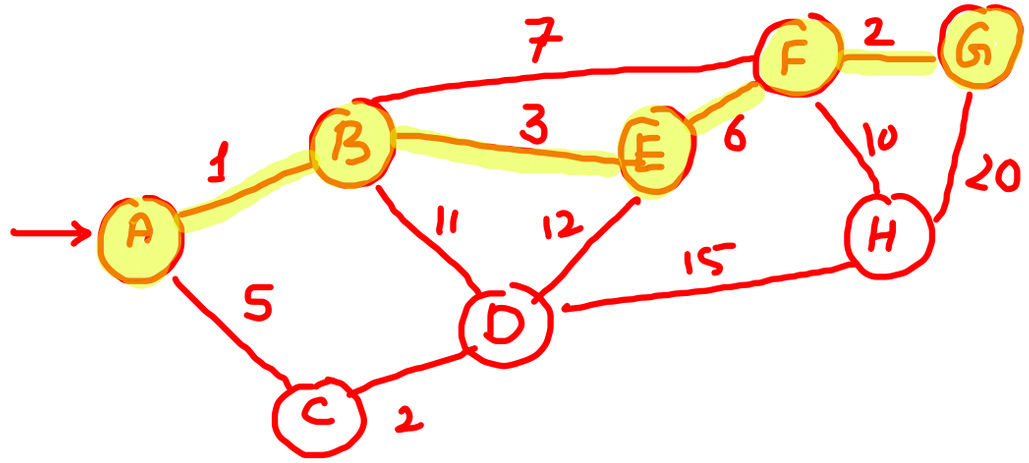
* Find the distance from Root to every Leaf node

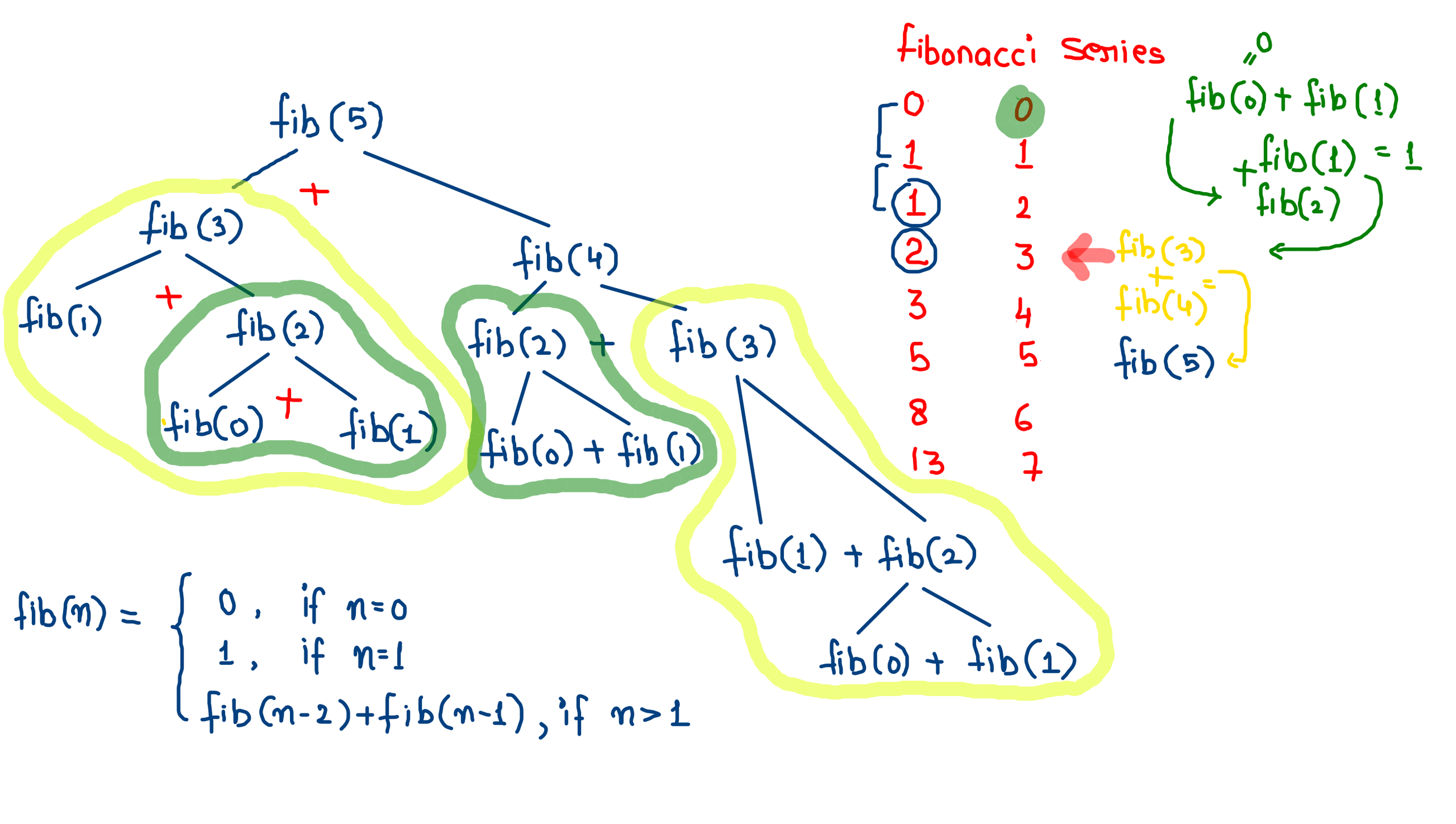


	A	B	C	D	E	F	G	H	I	J	K
A	0	5	3	12	7	11	12	16	15	8	13
B		0		7	2						
C			0			8	9				
D				0				4	3		
E					0					1	6
F						0					
G							0				
H								0			
I									0		
J										0	
K											0

Greedy Technique

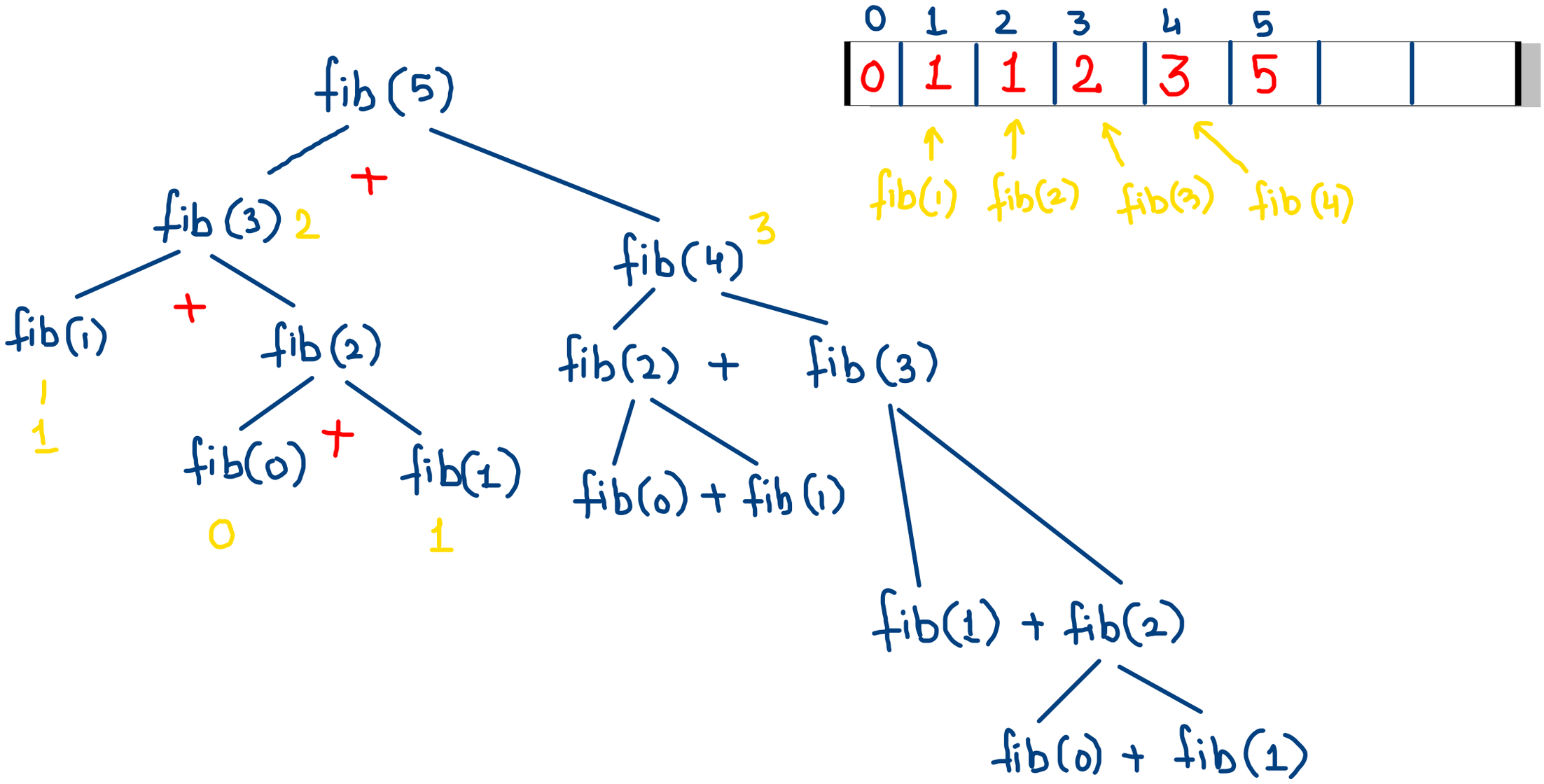
Immediate Profit




fib(5)
+
fib(3)
fib(4)
fib(1)
+
fib(2)
fib(2) +
fib(3)
fib(0)
+
fib(1)
fib(0) + fib(1)
fib(1) + fib(2)
fib(0) + fib(1)
Fibonacci Series
0
1
1
2
3
5
8
13
0
1
2
3
4
5
6
7
fib(0) + fib(1) = 1
+ fib(1)
fib(2)
fib(3)
+
fib(4) =
fib(5)
fib(n) = { 0, if n=0
1, if n=1
fib(n-2)+fib(n-1), if n>1

$$fib(n) = \begin{cases} 0, & \text{if } n=0 \\ 1, & \text{if } n=1 \\ fib(n-2)+fib(n-1), & \text{if } n>1 \end{cases}$$

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


0 1 2 3 4 5
0 1 1 2 3 5
fib(1) fib(2) fib(3) fib(4)
fib(5)
+
fib(3) 2
fib(4) 3
fib(1)
+
fib(2)
1
1
fib(2) + fib(3)
fib(0)
+
fib(1)
fib(0) + fib(1)
0
1
fib(1) + fib(2)
fib(0) + fib(1)

Binomial Coefficient

Dynamic Programming

$(a+b)^2 = 1 * a^2 + 2 * ab + 1 * b^2$

$(a+b)^3 = 1 * a^3 + 3 * a^2b + 3 * ab^2 + 1 * b^3$

⋮

$$(a+b)^n = {}_nC_0 * a^n b^0 + {}_nC_1 * a^{n-1} b^1 + {}_nC_2 * a^{n-2} b^2 + \cdots + {}_nC_n * a^0 b^n$$

$${}_nC_K = C(n,k) = \frac{n!}{K! * (n-K)!}$$

if k == n then

C(3,3) means that

n = 3 and K = 3

$$\frac{3!}{3! * (3-3)!} = 1$$

if k == 0 then

C(3,0) means that n = 3 k = 0

$$\frac{3!}{0! * (3-0)!} = 1$$

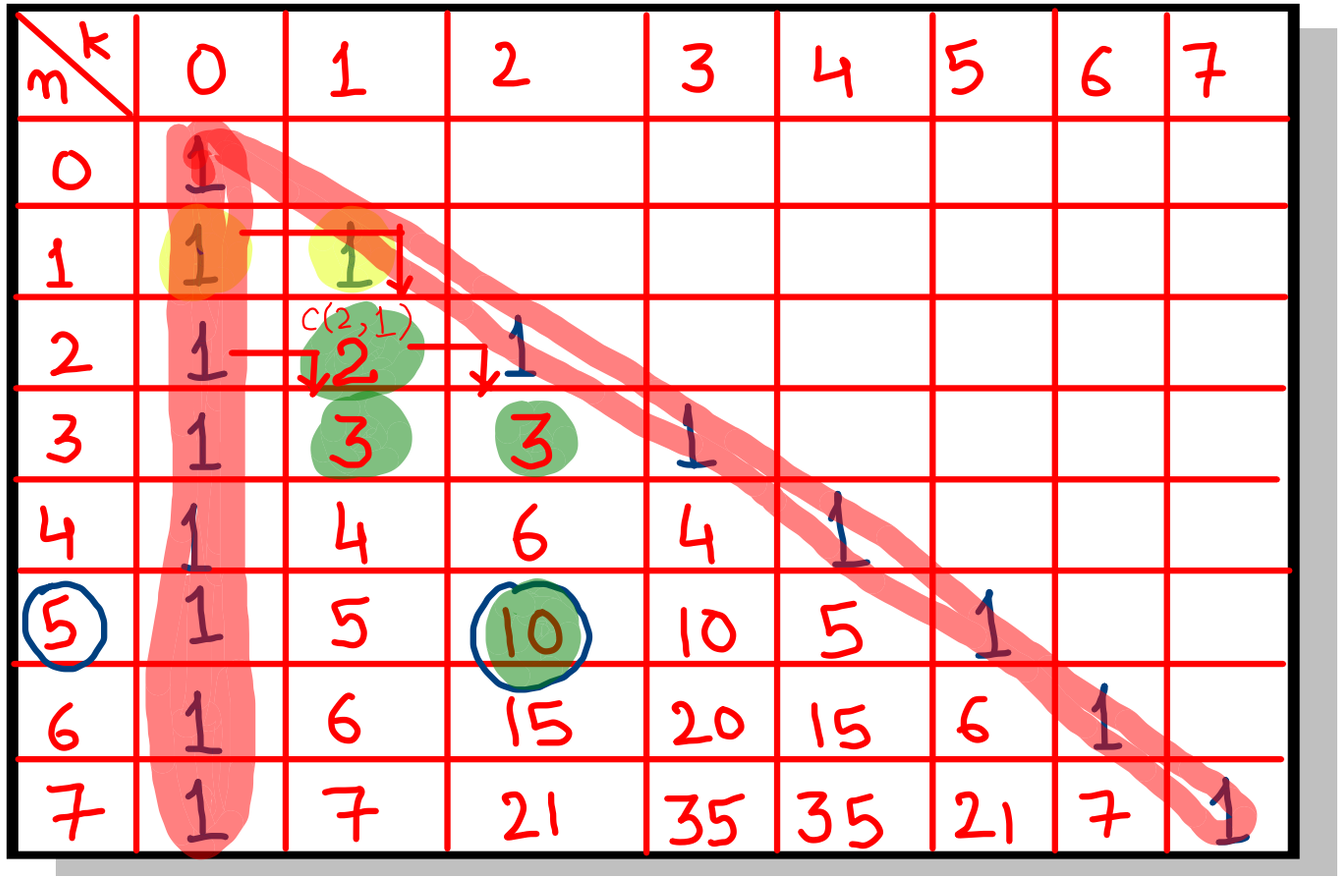
using recursion

$$C(n,k) = \begin{cases} 1 & \text{, if } k=0 \\ 1 & \text{, if } k==n \\ C(n-1,k-1)+C(n-1,k) & \text{if } n>k>0 \end{cases}$$

n \ k	0	1	2	3	4	5	6	7
0	1							
1	1	1						
2	1	2 C(2,1)	1					
3	1	3	3	1				
4	1	4	6	4	1			
5	1	5	10	10	5	1		
6	1	6	15	20	15	6	1	
7	1	7	21	35	35	21	7	1

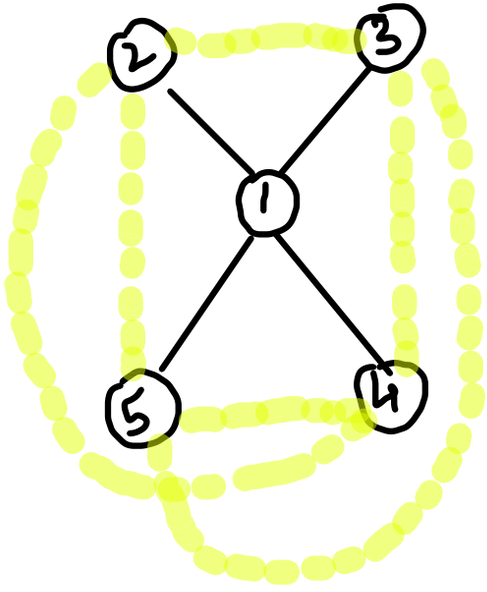
$C(2,1) = C(2-1,1-1) + C(2-1,1)$

$= C(1,0) + C(1,1)$




```
Algorithm
arr[8][8]
for i=0 to 7
    arr[i][0] = 1
    arr[i][i] = 1
for i=2 to 7
  for j = 1 to (i-1)
    arr[i][j] =
    arr[i-1][j-1]+arr[i-1][j]
```

ALL PAIR SHORTEST PATH



no. of edges = 4

Density factor = $\frac{\text{No. of Edges}}{\text{Max. No of Edges}}$

Max. no. of edges = VC_2 (where V is no. of Vertices)

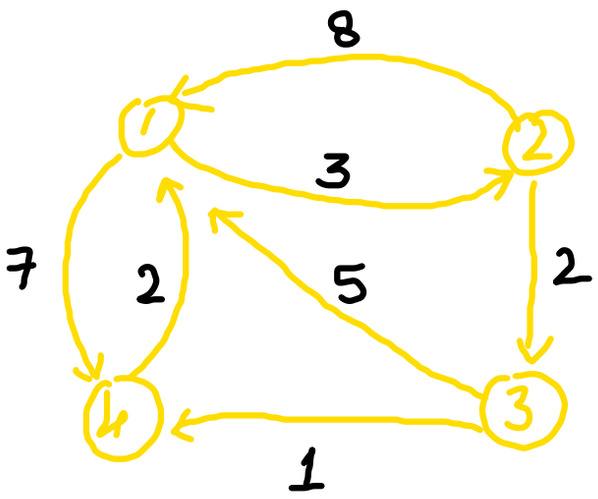
$= 5C_2 = 10$

Density factor = $\frac{4}{10} = \frac{2}{5}$

if density factor $> \frac{1}{2}$ then dense

if density factor is $< \frac{1}{2}$ then sparse

FLOYD - WARSHALL - ALGORITHM



$A_0 =$

	1	2	3	4
1	0	3	∞	7
2	8	0	2	∞
3	5	∞	0	1
4	2	∞	∞	0

Adjacency Matrix showing the direct path between nodes.

$A_1 =$

	1	2	3	4
1	0	3	∞	7
2	8	0	2	15
3	5	8	0	1
4	2	5	∞	0

$A_1[2,3] = \min\{ A_0[2,3] \text{ OR } A_0[2,1] + A_0[1,3] \}$

$\min\{ 2 \text{ OR } 8 + \infty \}$

$A_1[2,4] = \min\{ \infty \text{ OR } 8 + 7 \}$

$A_1[3,2] = \min\{ \infty \text{ OR } 5 + 3 \}$

$A_1[3,4] = \min\{ 1 \text{ OR } 5 + 7 \}$

$A_1[4,2] = \min\{ \infty \text{ OR } 2 + 3 \}$

$A_1[4,3] = \min\{ \infty \text{ OR } 2 + \infty \}$

$A_1 =$

	1	2	3	4
1	0	3	∞	7
2	8	0	2	15
3	5	8	0	1
4	2	5	∞	0

$A_2 =$

	1	2	3	4
1	0	3	5	7
2	8	0	2	15
3	5	8	0	1
4	2	5	7	0

$A_3 =$

	1	2	3	4
1	0	3	5	6
2	7	0	2	3
3	5	8	0	1
4	2	5	7	0

$A_2[1,3] = \min\{\infty \text{ OR } 3+2$

$A_2[1,4] = \min\{7 \text{ or } 3+15$

$A_2[3,1] = \min\{5 \text{ or } 8+8$

$A_2[3,4] = \min\{1 \text{ or } 8+15$

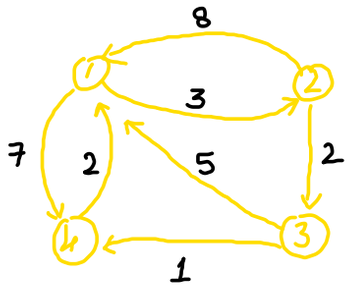
$A_2[4,1] = \min\{2 \text{ or } 5+8$

$A_2[4,3] = \min\{\infty \text{ or } 5+2$

$A_3[1,2] = \min\{3 \text{ or } 5+8\}$

$A_4 =$

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



Arr=

	1	2	3	4
1	0	3	∞	7
2	8	0	2	∞
3	5	∞	0	1
4	2	∞	∞	0

Algorithm:- FLOYD WARSHALL

```
for k= 1 to 4
{
  for i = 0 to 4
  { for j=0 to 4
    {
      if (i==k or j==k)
        Continue;
      if (arr[i][j] > arr[i][k] + arr[k][j])
       { arr[i][j] = arr[i][k] + arr[k][j]; }
    }
  }
}
```

$O(n^3)$

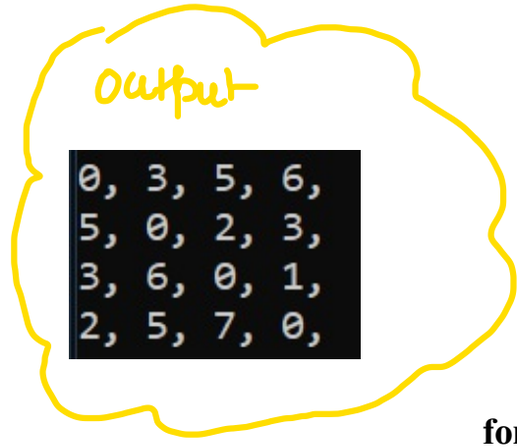
Output

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

Output

```
0, 3, 5, 6,
5, 0, 2, 3,
3, 6, 0, 1,
2, 5, 7, 0,
```

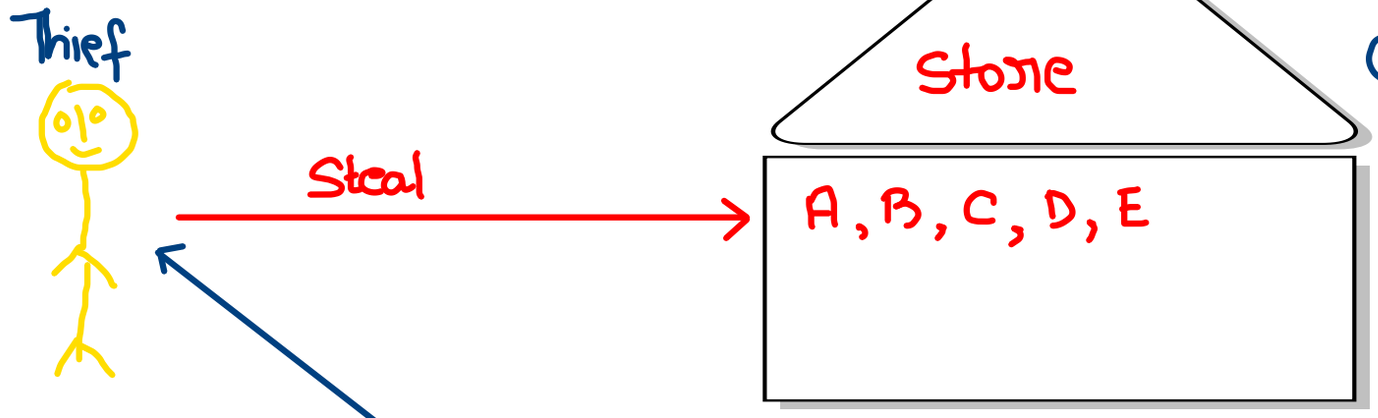
```
class Floyd
{
	public static void main(String args[])
	{
		int arr[][]={{0,3,99,7},{8,0,2,99},{5,99,0,1},{2,99,99,0}};

		for(int k=0;k<4;k++)
		{
			for(int i=0;i<4;i++)
			{
				for(int j=0;j<4;j++)
				{
					if(i==k || j==k)
					{continue;}
				  if(arr[i][j]>arr[i][k]+arr[k][j])
					{
						arr[i][j]=arr[i][k]+arr[k][j];
					}}
			}}
		for(int i=0;i<4;i++)
		{
			for(int j=0;j<4;j++)
			{
				System.out.print(arr[i][j]+", ");
			}
			System.out.println();
		}
	}
}
```



```
#include<bits/stdc++.h>
using namespace std;
int main()
{
int k,i,j,n,m;
 // int arr[][]={{0,3,99,7},{8, 0, 2, 99},{5,99,0,1},{2,99, 99, 0}};
cin>>n>>m;
int arr[n][m];
for(i=0;i<n;i++)
{
        for(j=0;j<m;j++)
        {
                cin>>arr[i][j];
        }
}
for(int k=0;k<4;k++)
{
 for(i=0;i<n;i++)
 {
        for(j=0;j<m;j++)
        {
                if(i==k || j==k)
                {
                        continue;
                }
                if(arr[i][j]>arr[i][k]+arr[k][j])
                {
                        arr[i][j]=arr[i][k]+arr[k][j];
                }
        }
}
}
for(i=0;i<n;i++)
{
        for(j=0;j<m;j++)
        {
                cout<<arr[i][j]<<", ";
        } cout<<endl;
} return 0;
}
```

Knapsack Problem

1. Simple Knapsack Problem
2. 0/1 Knapsack.



item	weight	Price	P/W
A	4	8	8/4 = 2
B	2	4	4/2 = 2
C	3	9	9/3 = 3
D	4	2	2/4 = 0.5
E	5	5	5/5 = 1

} 9 kg

} 2 kg