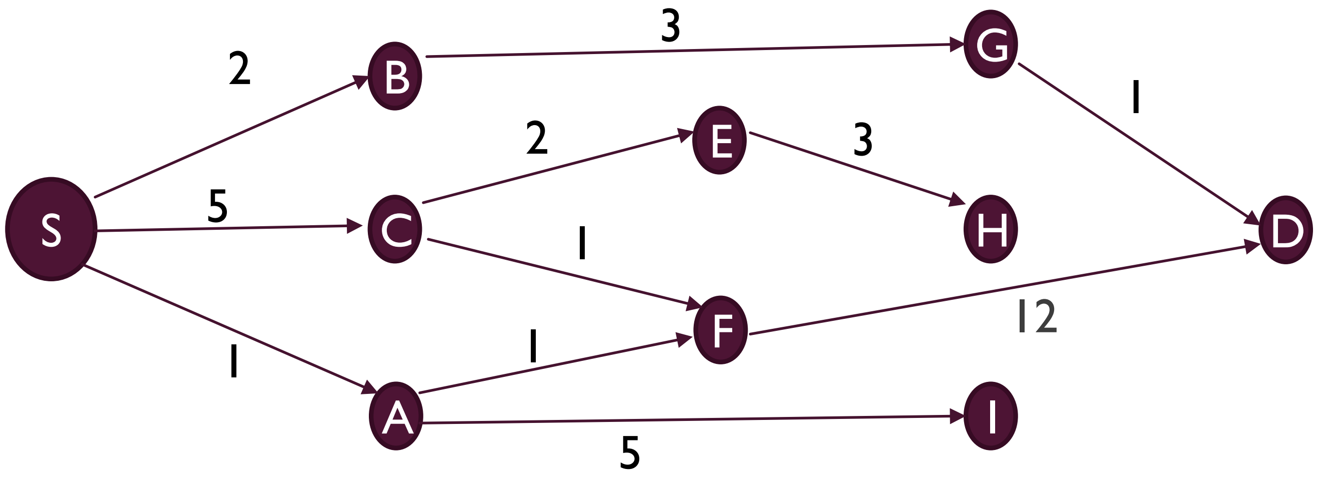
DESIGN & ANALYSIS OF ALGORITHM

PCC-CS501

DESIGN & ANALYSIS OF ALGORITHM
SCHEDULE ----TOPIC WISE

	Topic	Sub Topic
1	INTRODUCTION	DESIGN OF ALGORITHM ,ANALYSIS OF ALGORITHM, ALGORITHM PROPERTIES
2	FRAMEWORK FOR ALGORITHM ANALYSIS	HOW TO COUNT EXECUTION TIME OF ALGORITHM,INPUT INSTANCES
3	ASYMPTOTIC NOTATION	BEST CASE,AVERAGE CASE, WORST CASE
4	SOLVING RECURRENCE RELATION	SUBSTITUTION METHOD, MASTER THEOREM
5	ALGORITHM DESIGN TECHNIQUES	DIVIDE & CONQUER, GREEDY,DYNAMIC PROGRAMMING, BACKTRACKING,
6	DISJOINT SET MANIPULATION	UNION FIND
7	NETWORK FLOW PROBLEM	FORD FULKERSON ALGORITHM
8	NP COMPLETENESS	NP,NP HARD………ALGORITHM
9	APPROXIMATION ALGORITHM	COMPLEXITY ANALYSIS OF NP COMPETE PROBLEM

QUESTION DOES GREEDY ENSURE OPTIMALITY?



GREEDY APPROACH DISADVANTAGE

- GREEDY DOES NOT ENSURE OPTIMAL SOLUTION.

DYNAMIC PROGRAMMING

- Considers All possible solution, then consider the optimal solution.
- Time consuming Method.
- Follows Principle of Optimality: Problem must be solved in sequence of decision.
- Overlapping Sub-problem.
- Memorization
- Tabulation

FIBONACCI SERIES

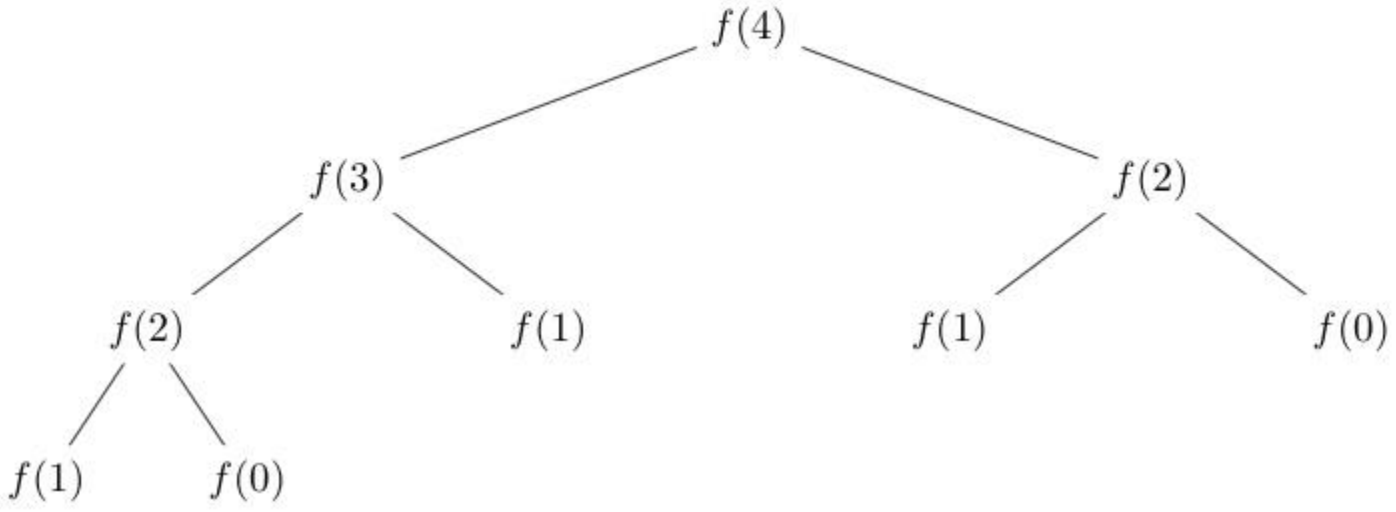
For n>=2, $f_n = f_{n-1} + f_{n-2}$

For n=0, $f_n=0$

For n=1, $f_n=1$

- Int fibo(int n)

```
{
 if(n<=1)
   return n
 return fibo(n-1)+fibo(n-2)
}
```



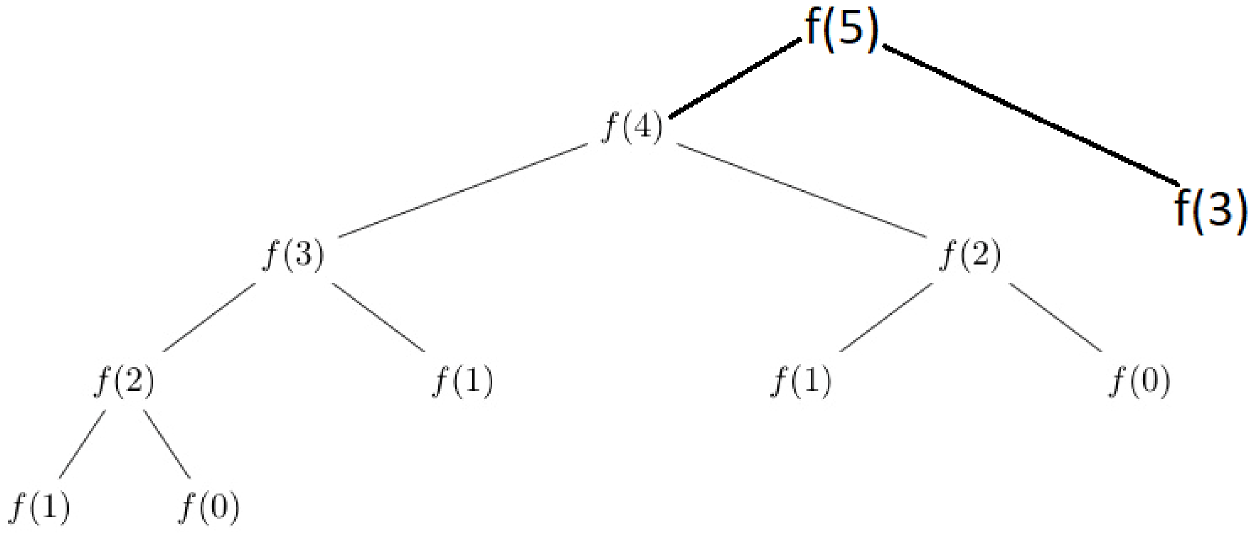
$T(n) = 2T(n-1) + 1$ O(2^n)

MEMORIZATION TOP-DOWN FIBONACCI SERIES

0	1				
0	1	2	3	4	5

- Int fibo(int n) fibo(n)=n+1

```
{
 if(n<=1)
    return n
 return fibo(n-1)+fibo(n-2)
}
```



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

MEMORIZATION

FIBONACCI SERIES

The memoized version of the recursive Fibonacci algorithm looks like this:

- If n is 0 or 1, return n
- Otherwise, if n is in the memo, return the memo's value for n
- Otherwise,
  - Calculate $\text{fibonacci}(n-1) + \text{fibonacci}(n-2)$
  - Store result in memo
  - Return result

n	Original	Memoized
5	15	9
6	25	11
7	41	13
8	67	15
9	109	17
10	177	19

TABULATION METHOD BOTTOM-UP FIBONACCI SERIES

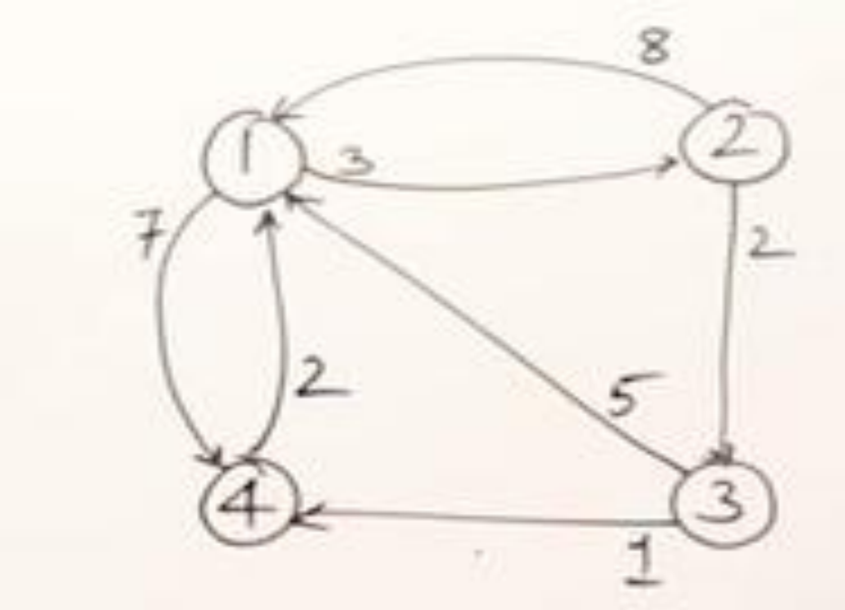
- Int fibo(n)

```
{
  if(n<=1)
    return n
  F[0]=0 F[1]=1
  for(i=2;i<n;i++)
    F[i]=F[i-1]+F[i-2]
}
```

0	1			

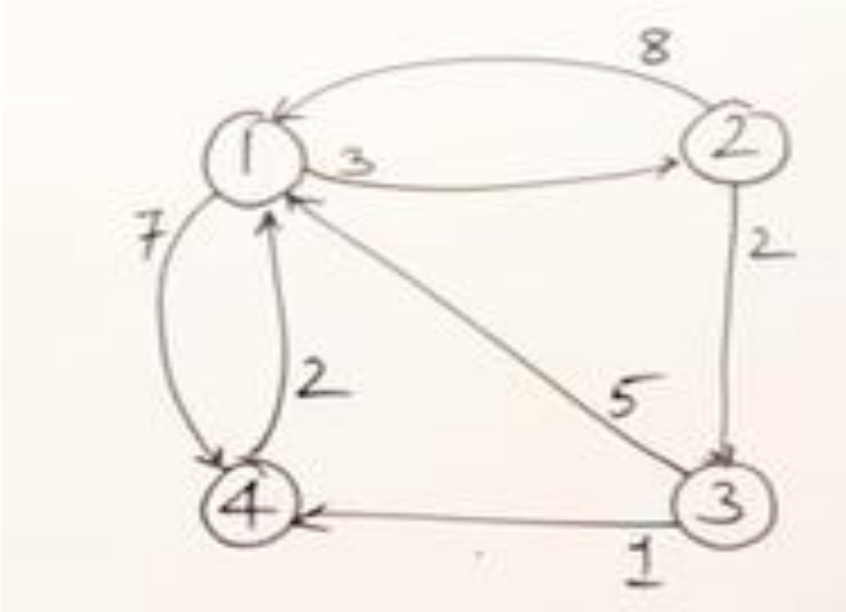
FLOYD WARSHALL'S ALGORITHM

- All Pair Shortest Path Problem.



FLOYD WARSHALL'S ALGORITHM

$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & \infty \\ 5 & \infty & 0 & 1 \\ 2 & \infty & \infty & 0 \end{bmatrix} \end{array}$$



- min(dist[i][k]+dist[k][j],dist[i][j]).

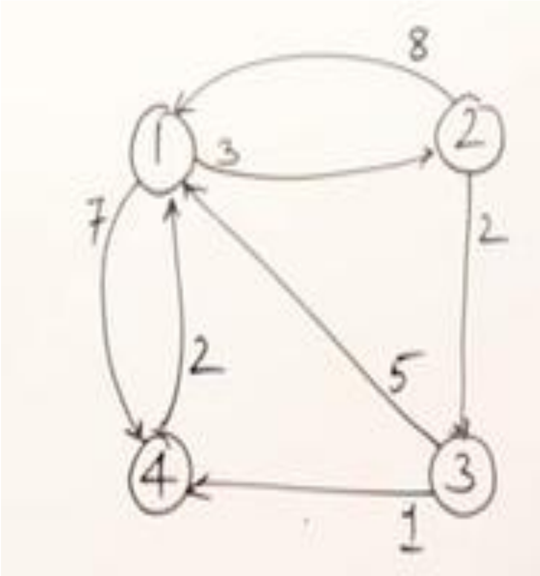
DYNAMIC PROGRAMMING

FLOYD WARSHALL'S ALGORITHM

$$A^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & \infty \\ 3 & 5 & \infty & 0 & 1 \\ 4 & 2 & \infty & \infty & 0 \end{array}$$

$$A^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & & \\ 3 & 5 & & 0 & \\ 4 & 2 & & & 0 \end{array}$$

$$A^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & \infty & 0 \end{array}$$



min(dist[i][k]+dist[k][j],dist[i][j])

DYNAMIC PROGRAMMING

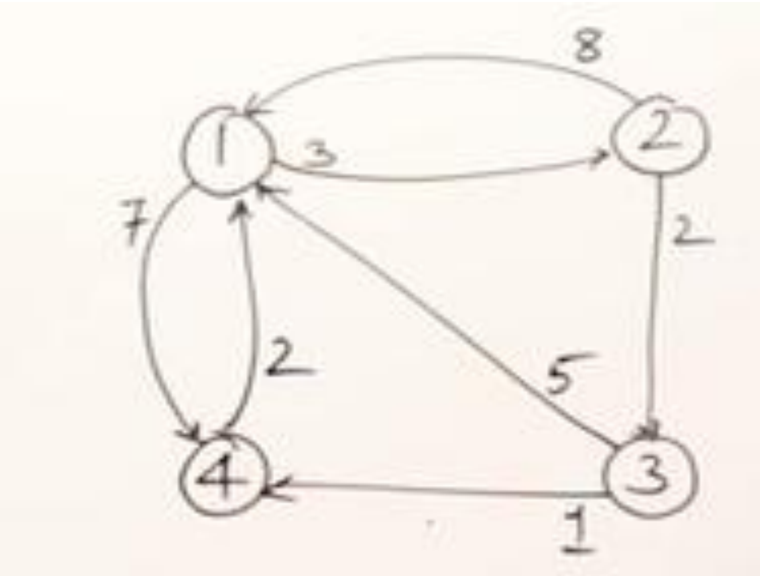
FLOYD WARSHALL'S ALGORITHM

$$A^1 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & \infty & 0 \end{bmatrix} \end{array}$$

$$A^2 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 3 & 5 & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

$$A^3 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 3 & 5 & 6 \\ 7 & 0 & 2 & 3 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

$$A^4 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 3 & 5 & 6 \\ 5 & 0 & 2 & 3 \\ 3 & 6 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$



FLOYD WARSHALL'S ALGORITHM

```
for (k = 0; k < V; k++)
    {
        for (i = 0; i < V; i++)
        {
            for (j = 0; j < V; j++)
            {
                if (dist[i][k] + dist[k][j] < dist[i][j])
                    dist[i][j] = dist[i][k] + dist[k][j];
            }
        }
    }
```

NEXT CLASS

- KNAPSACK USING DYNAMIC PROGRAMMING