Recursion

- Function calling itself is called as recursive function.
- For each function call stack frame is created on the stack.
- Thus it needs more space as well as more time for execution.
- However recursive functions are easy to program.
- Typical divide and conquer problems are solved using recursion.
- For recursive functions two things are must
    - Recursive call (Explain process in terms of itself)
    `- Terminating or base condition (Where to stop)

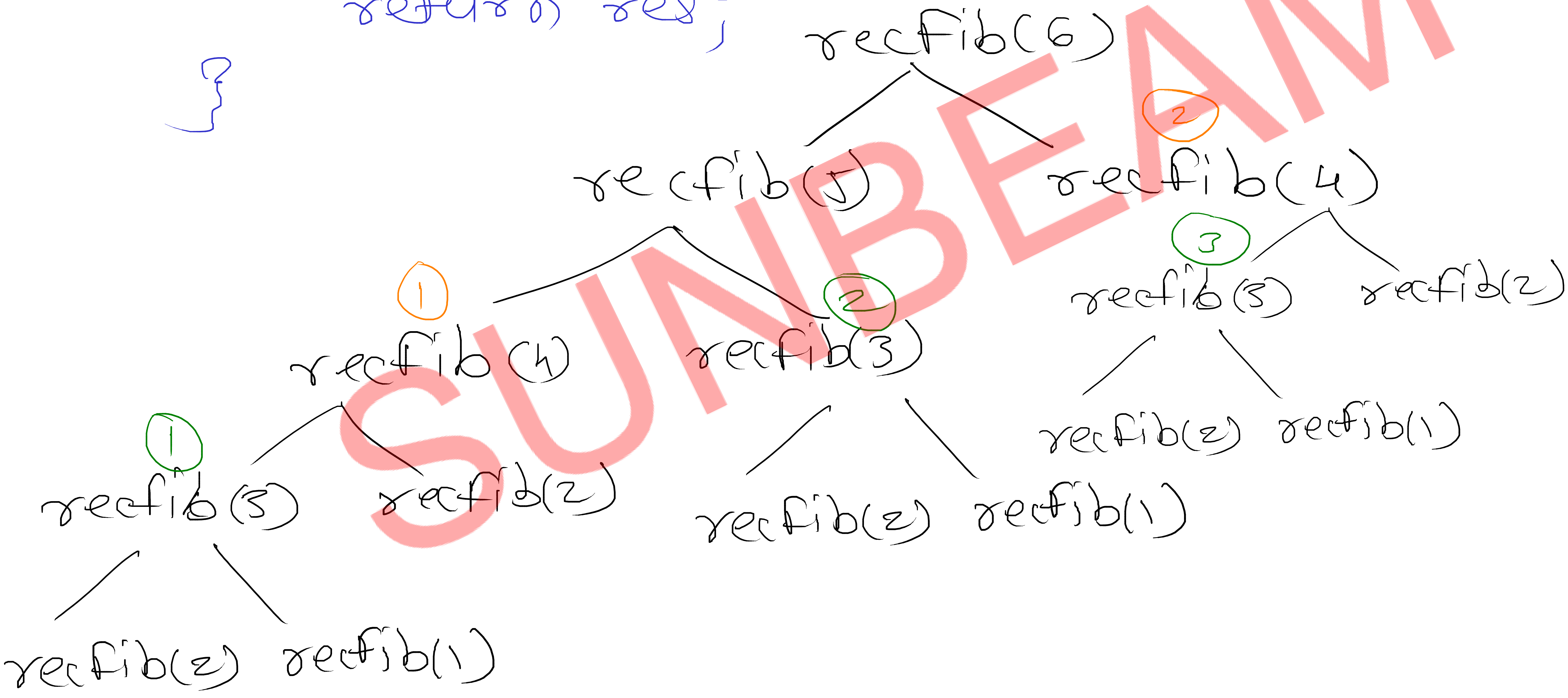
eg. Fibonacci Series

- Recursive formula
    - $T_n = T_{n-1} + T_{n-2}$
- Terminating condition
    - $T_1 = T_2 = 1$

1 1 2 3 5 8 13

- Overlapping sub-problem

```
int  recFib (int n){
    if (n==1 || n==2)
        return 1;
    int res= recFib(n-1)+recFib(n-2)
    return res;
}
```



Dynamic Programming

memoisation

recursion

+

1D or 2D array

- bottom up approach

Tabulation

Loops

+

1D or 2D array

- top down approach

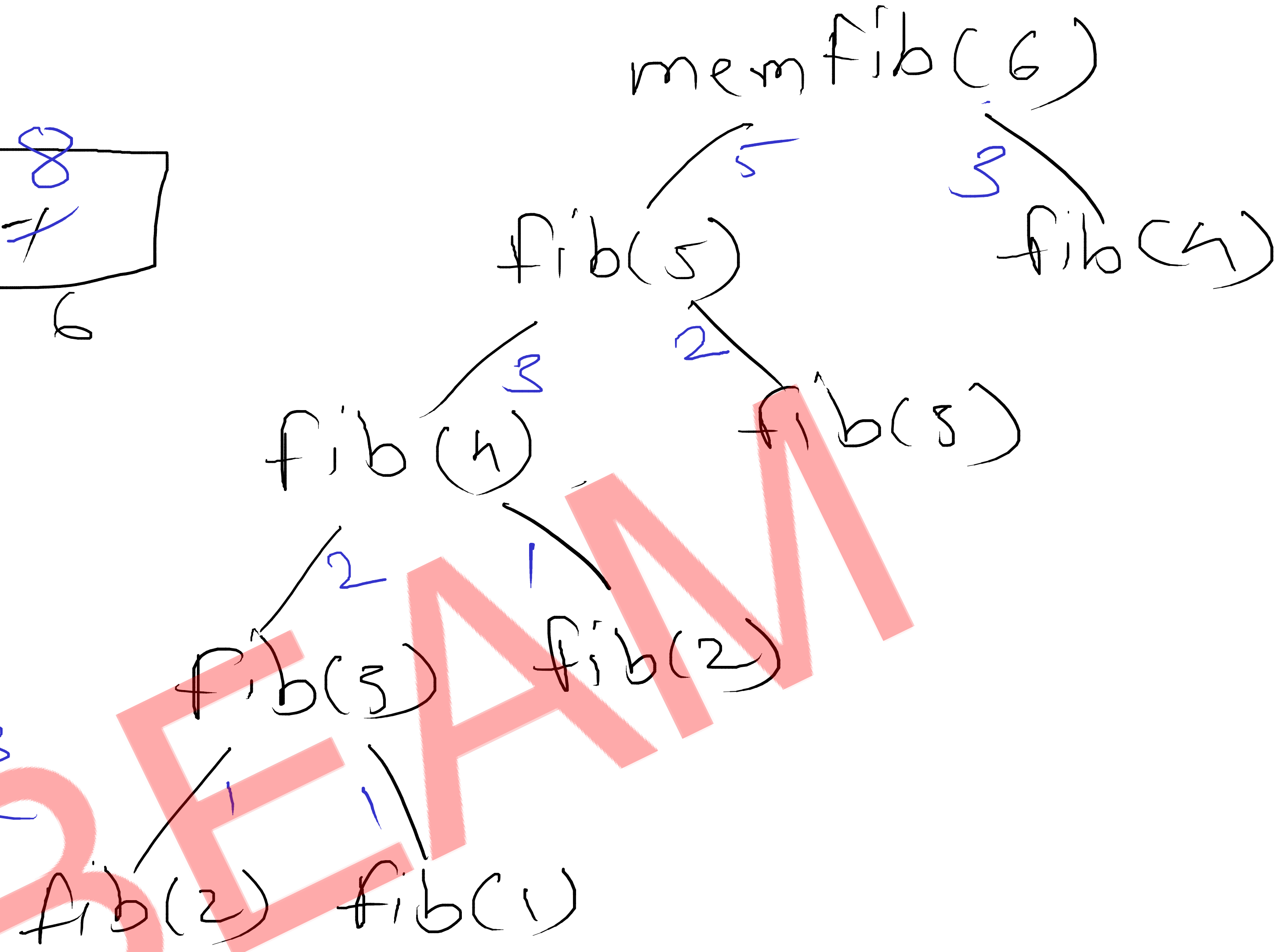
DP arrays

Memoization

- It's based on the Latin word memorandum, meaning "to be remembered".
- Memoization is a technique used in computing to speed up programs.
- This is accomplished by memorizing the calculation results of processed input such as the results of function calls.
- If the same input or a function call with the same parameters is used, the previously stored results can be used again and unnecessary calculation are avoided.
- Need to rewrite recursive algorithm. Using simple arrays or map/dictionary.

dp[] =

-1	~~-1~~ 1	~~-1~~ 1	~~-1~~ 2	~~-1~~ 3	~~-1~~ 5	~~-1~~ 8
0	1	2	3	4	5	6



```
int memFib(int n){
    if(dp[n] != -1)
        return dp[n];
    if(n==1 || n==2){
        dp[n]=1;
        return dp[n];
    }
    dp[n] = memfib(n-1)+memfib(n-2)
    return dp[n];
}
```

dp[] =

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

tabFib(6)

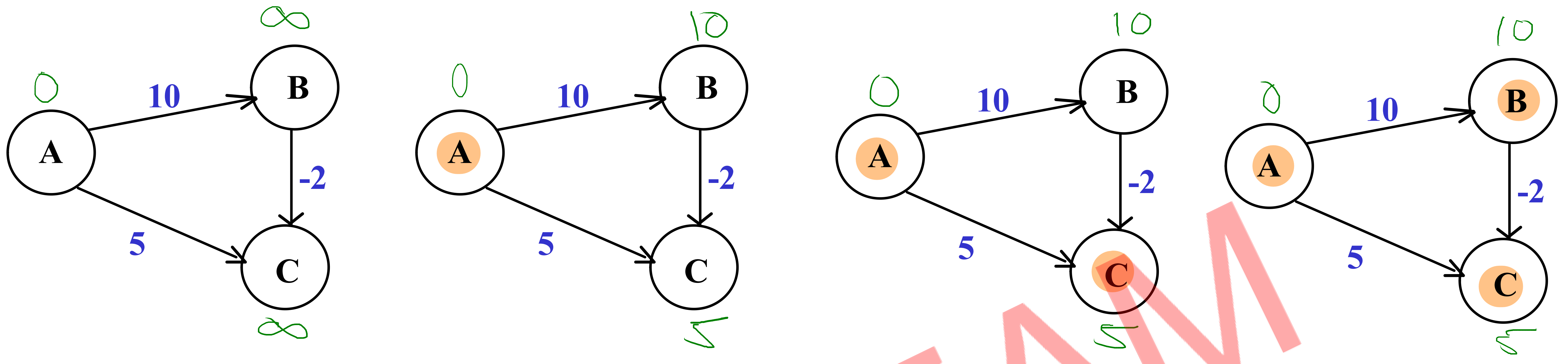
```
int tabFib(int n) {
    dp[1] = dp[2] = 1;
    for(int i=3; i<=n; i++)
        dp[i] = dp[i-1] + dp[i-2];
    return dp[n];
}
```

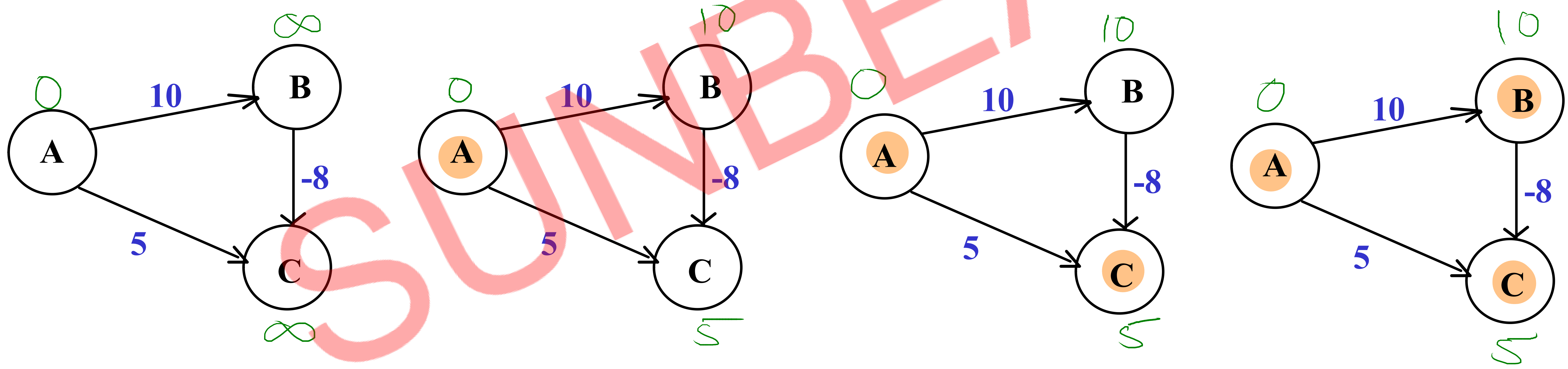
Dynamic Programming

- Dynamic programming is another optimization over recursion.
- Typical DP problem give choices (to select from) and ask for optimal result (maximum or minimum).
- Technically it can be used for the problems having two properties
  - Overlapping sub-problems
  - Optimal sub-structure
- To solve problem, we need to solve its sub-problems multiple times.
- Optimal solution of problem can be obtained using optimal solutions of its sub-problems.

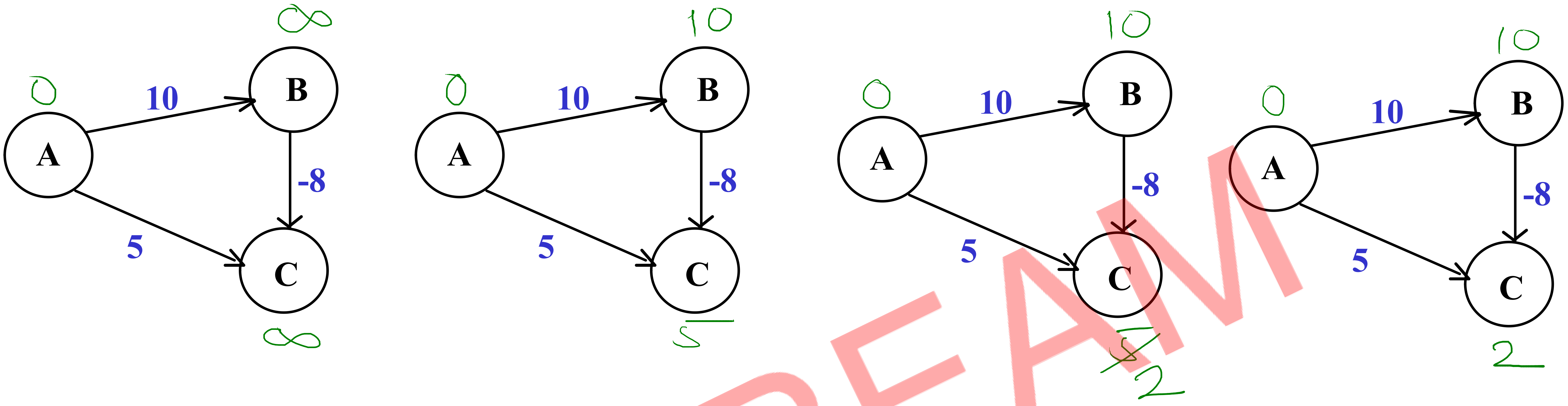
- Greedy algorithms pick optimal solution to local problem and never reconsider the choice done.
- DP algorithms solve the sub-problem in a iteration and improves upon it in subsequent iterations.

Dijkstra's SPT



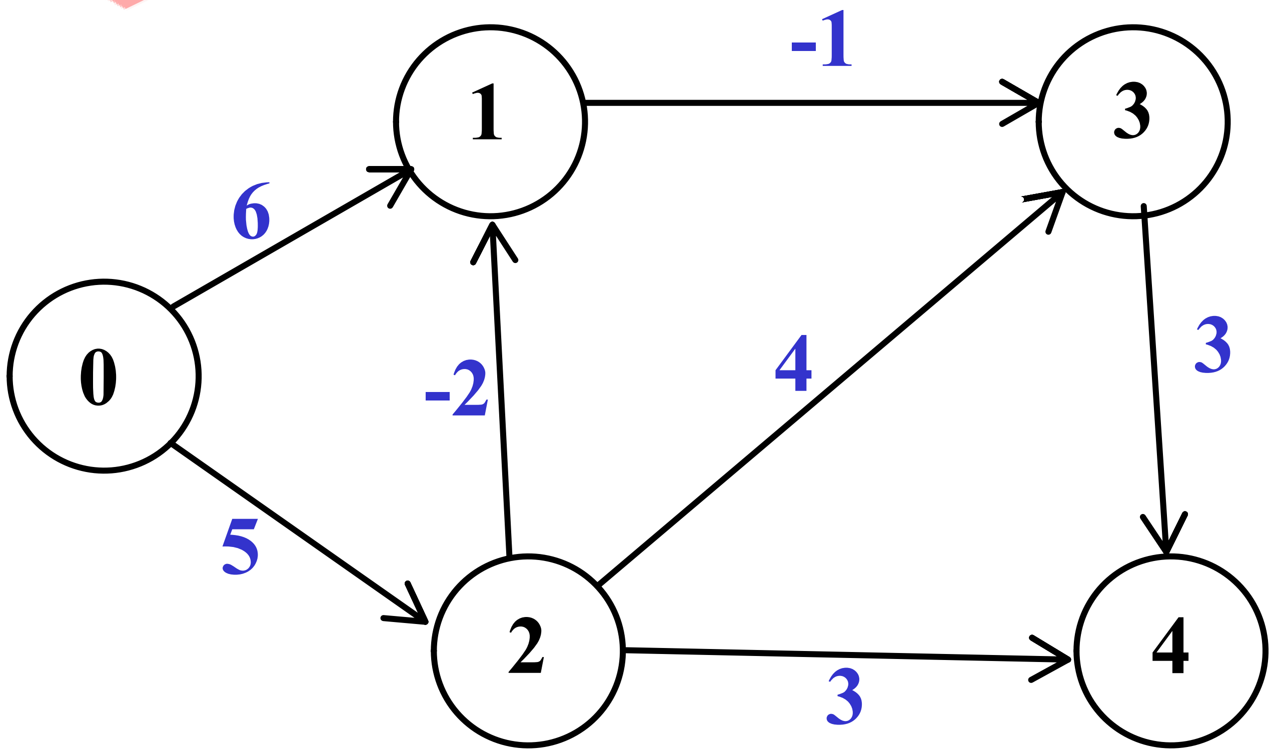


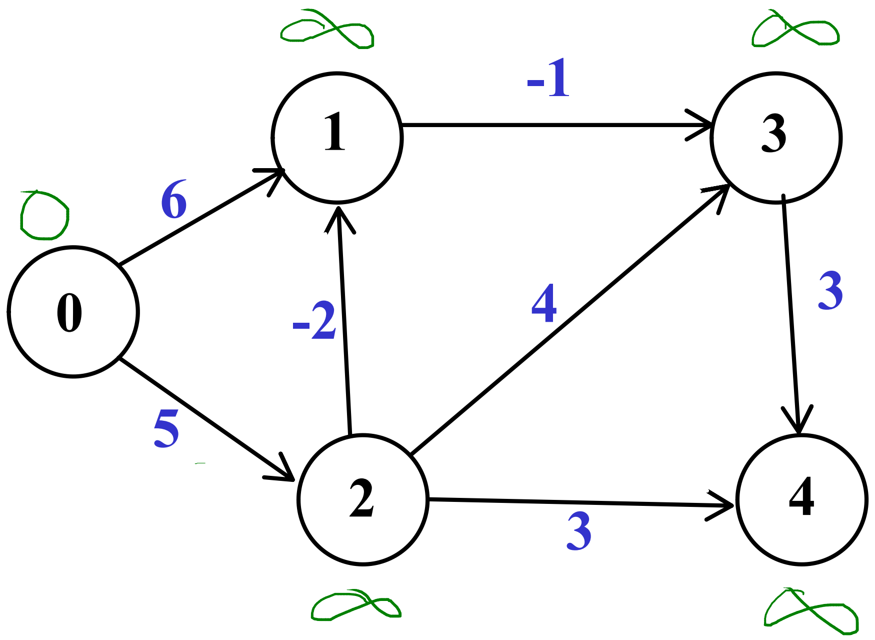
- Dijkstra's algorithm may fail if graph is having -ve edges

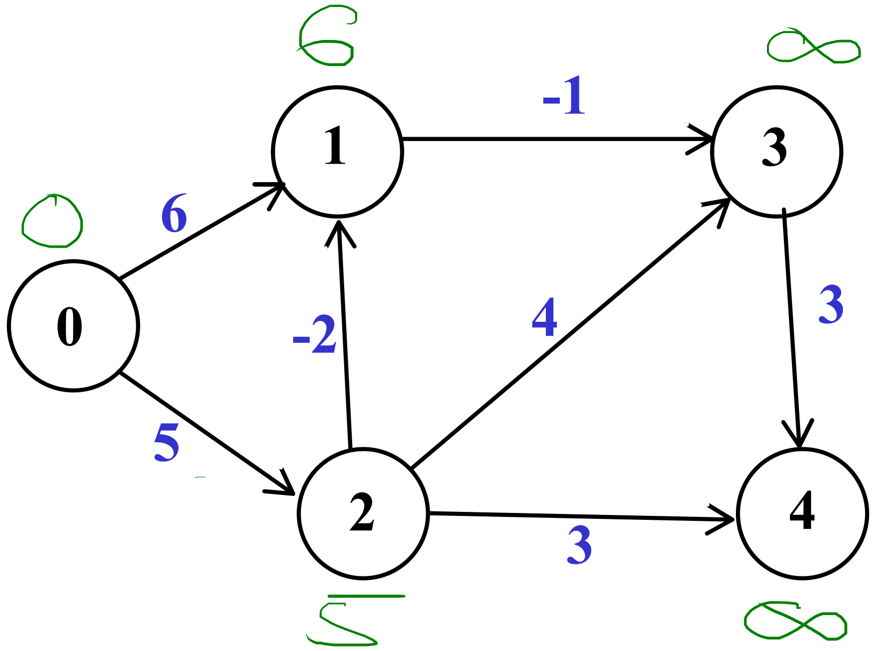

0
A
10
B
∞
-8
5
C
∞
0
A
10
B
10
-8
5
C
5
0
A
10
B
10
-8
5
C
~~5~~
2
0
A
10
B
10
-8
5
C
2

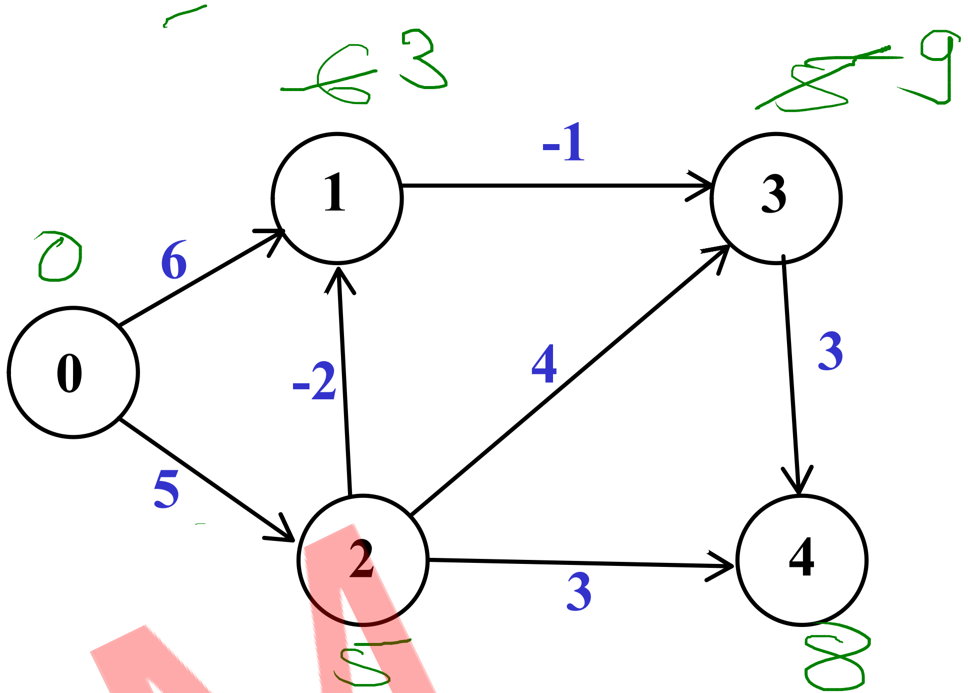
Bellman Ford Algorithm

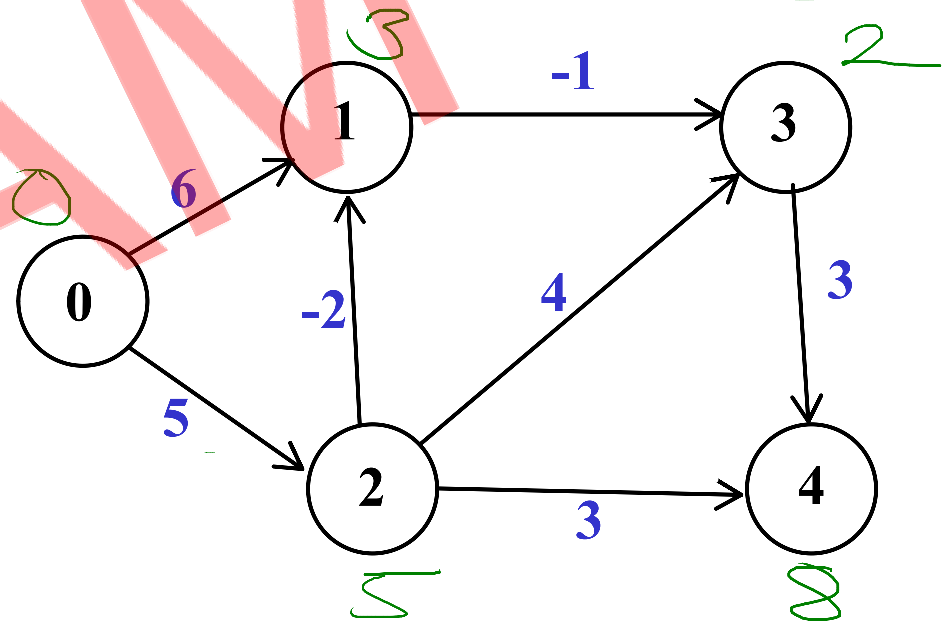
```
1. Initializes distances from the source to all vertices as infinite and
        distance to the source itself as 0.
2. Calculate shortest distance V-1 times:
        For each edge u-v,
        if dist[v] > dist[u] + weight of edge u-v,
        then update dist[v], so that
                dist[v] = dist[u] + weight of edge u-v.
3. Check if negative edge cycle in the graph:
        For each edge u-v,
        if dist[v] > dist[u] + weight of edge (u,v),
        then graph has -ve weight cycle.
```



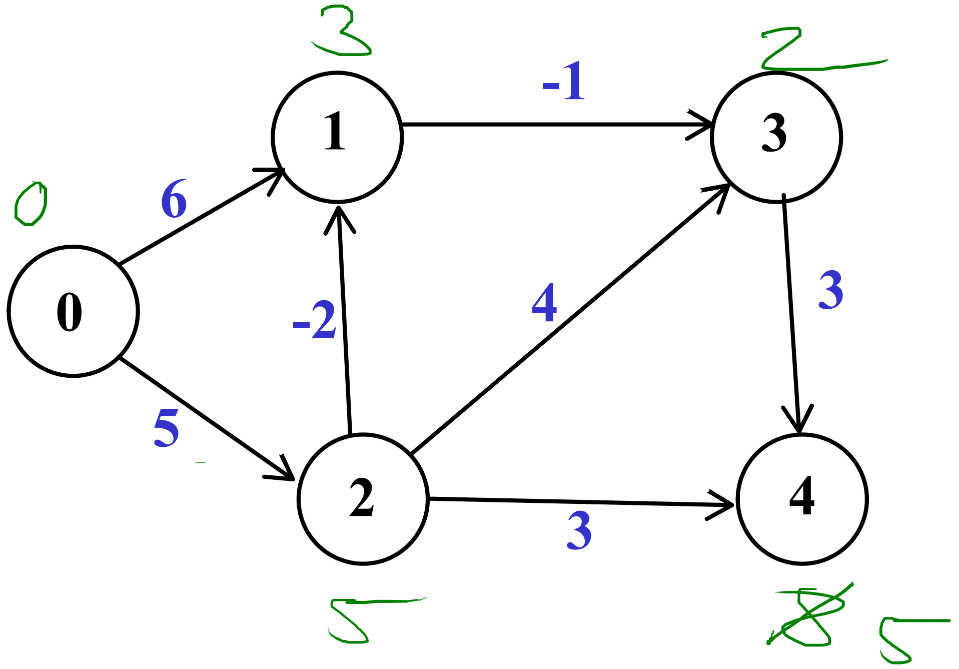




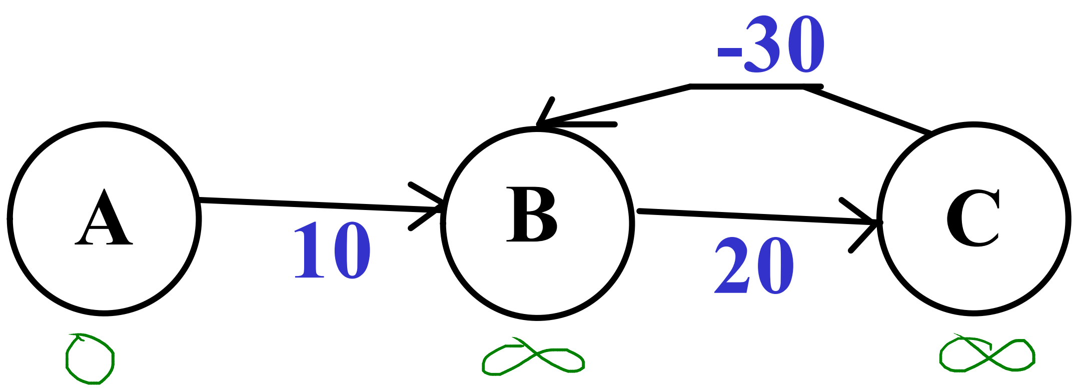


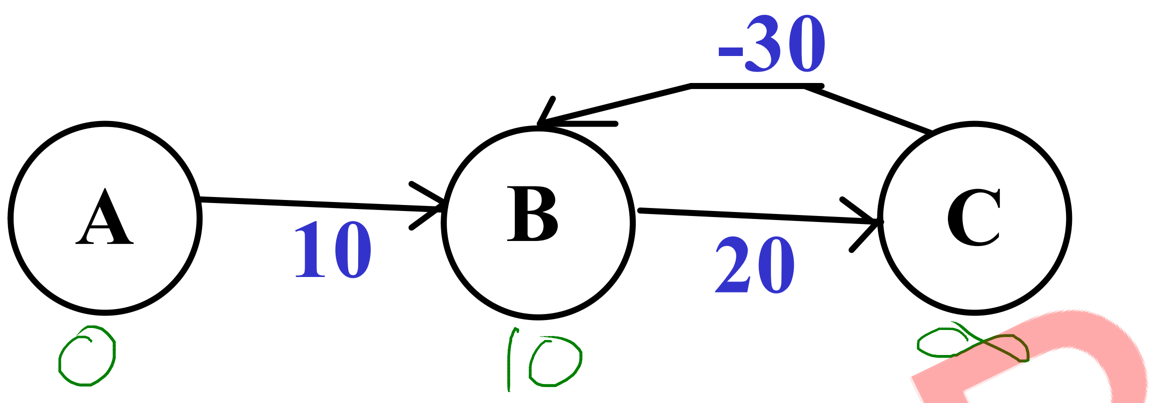


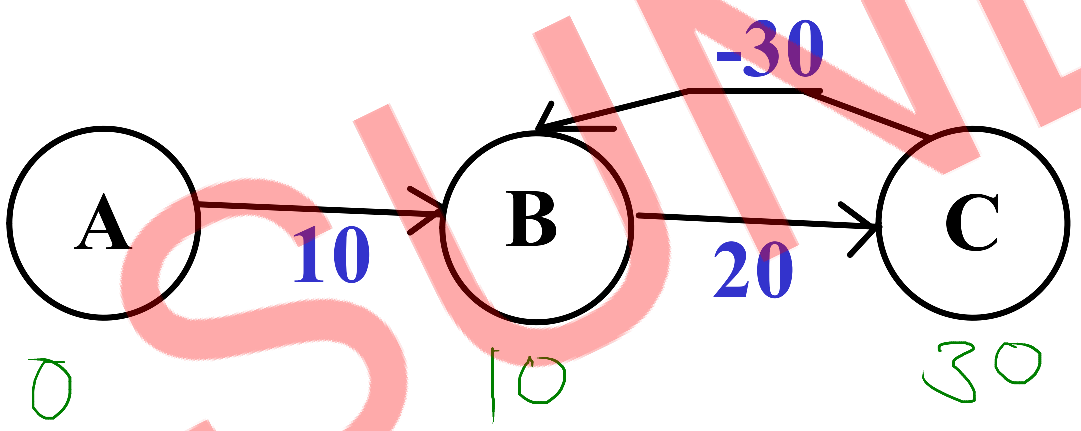
	0	1	2	3	4
	0	∞	∞	∞	∞
Pass 1	0	6	5	∞	∞
Pass 2	0	3	5	9	8
Pass 3	0	3	5	2	8
Pass 4	0	3	5	2	5

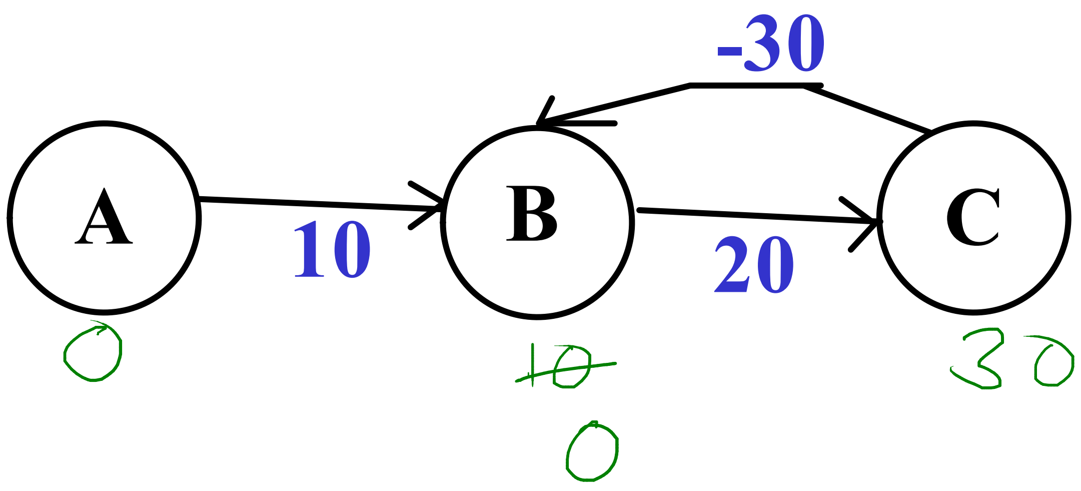


cycle wt = 20 + -30
= -10





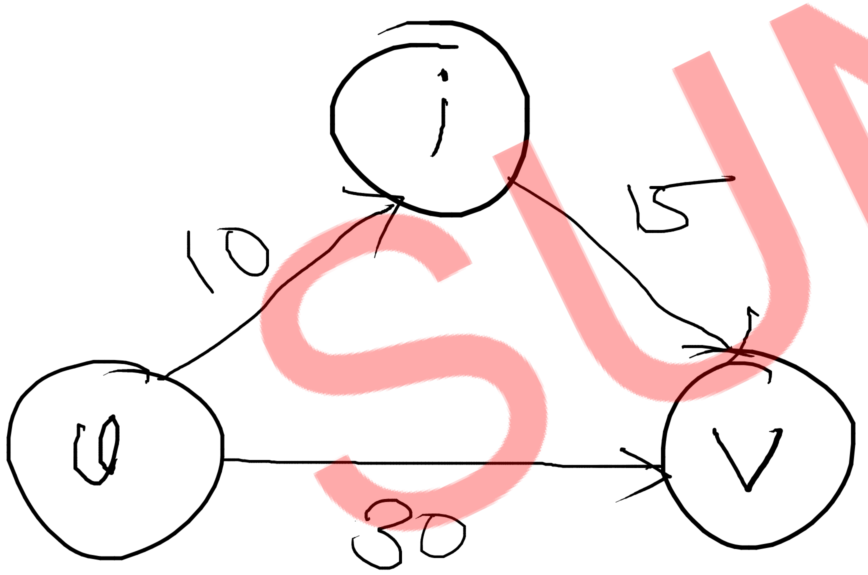




Floyd Warshall Algorithm

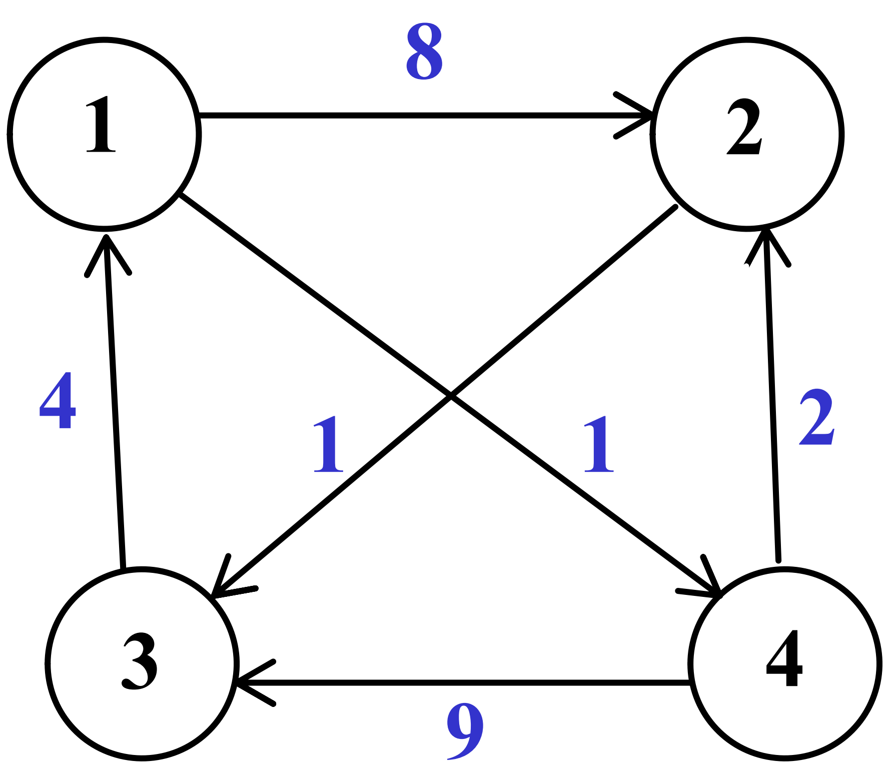
1. **Create distance matrix to keep distance of every vertex from each vertex.**
   Initially assign it with weights of all edges among vertices
   (i.e. adjacency matrix).

2. **Consider each vertex (i) in between pair of any two vertices (s, d) and**
   find the optimal distance between s & d considering intermediate vertex
   i.e. dist(s,d) = dist(s,i) + dist(i,d),
   if dist(s,i) + dist(i,d) < dist(s,d).



if(dist(u,i)+dis(i,v)<dist(u,v))

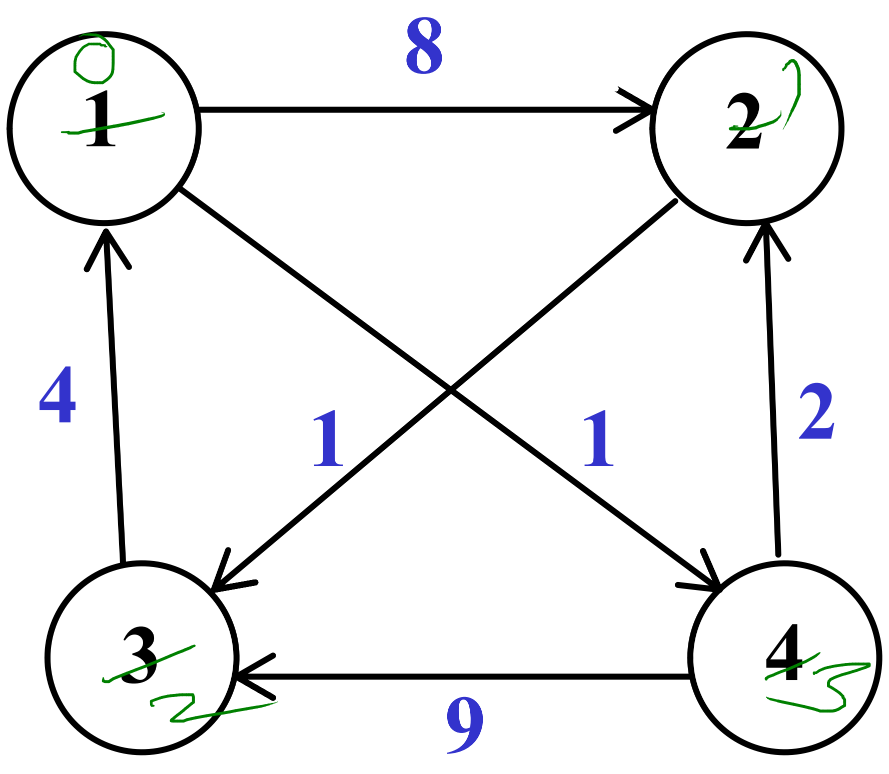
dist(u,v)=dist(u,i)+dist(i,v)



$$G = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & \infty & 8 & \infty & 1 \\ 2 & \infty & \infty & 1 & \infty \\ 3 & 4 & \infty & \infty & \infty \\ 4 & \infty & 2 & 9 & \infty \end{array}$$

$$A^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & \infty & 1 \\ 2 & \infty & 0 & 1 & \infty \\ 3 & 4 & \infty & 0 & \infty \\ 4 & \infty & 2 & 9 & 0 \end{array}$$

$$A^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & \infty & 1 \\ 2 & \infty & 0 & 1 & \infty \\ 3 & 4 & 12 & 0 & 5 \\ 4 & \infty & 2 & 9 & 0 \end{array}$$



$$A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & 9 & 1 \\ 2 & \infty & 0 & 1 & \infty \\ 3 & 4 & 12 & 0 & 5 \\ 4 & \infty & 2 & 3 & 0 \end{array}$$

$$A^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & 9 & 1 \\ 2 & 5 & 0 & 1 & 6 \\ 3 & 4 & 12 & 0 & 5 \\ 4 & 7 & 2 & 3 & 0 \end{array}$$

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

Graph applications

- Graph represents flow of computation/tasks. It is used for resource planning and scheduling. MST algorithms are used for resource conservation. DAG are used for scheduling in Spark or Tez.

- In OS, process and resources are treated as vertices and their usage is treated as edges. This resource allocation algorithm is used to detect deadlock.

- In social networking sites, each person is a vertex and their connection is an edge. In Facebook person search or friend suggestion algorithms use graph concepts.

- In world wide web, web pages are like vertices; while links represents edges. This concept can be used at multiple places.
    - Making sitemap
    - Downloading website or resources
    - Developing web crawlers
    - Google page-rank algorithm

- Maps uses graphs for showing routes and finding shortest paths. Intersection of two (or more) roads is considered as vertex and the road connecting two vertices is considered to be an edge.