Solve the following Strassen's Multiplication:

$$A = \begin{bmatrix} \overset{A_{11}}{2} & \overset{A_{12}}{4} \\ \overset{A_{21}}{3} & \overset{A_{22}}{5} \end{bmatrix} \qquad B = \begin{bmatrix} \overset{B_{11}}{1} & \overset{B_{12}}{3} \\ \underset{B_{21}}{4} & \underset{B_{22}}{7} \end{bmatrix}$$

$$P = (A_{11} + A_{22})(B_{11} + B_{22}) = (2+5) * (1+7) = 7 * 8 = 56$$

$$Q = (A_{21} + A_{22}) B_{11} = (3+5) 1 = 8$$

$$R = A_{11} (B_{12} - B_{22}) = 2(3-7) = 2 * (-4) = -8$$

$$S = A_{22} (B_{21} - B_{11}) = 5(4-1) = 5 * 3 = 15$$

$$T = (A_{11} + A_{12}) B_{22} = (2+4) 7 = 6 * 7 = 42$$

$$U = (A_{21} - A_{11})(B_{11} + B_{12}) = (3-2)(1+3) = 1 * 4 = 4$$

$$V = (A_{12} - A_{22})(B_{21} + B_{22}) = (4-5)(4+7) = -1 * 11 = -11$$

$$C_{11} = P + S - T + V = 58 + 15 - 42 - 11 = 18$$

$$C_{12} = R + T = -8 + 42 = 34$$

$$C_{21} = Q + S = 8 + 15 = 23$$

$$C_{22} = P + R - Q + U = 56 - 8 - 8 + 4 = 44$$

$$C = \begin{bmatrix} 18 & 32 \\ 23 & 44 \end{bmatrix}$$

Scanned with CamScanner

Solve the Strassen's Multiplication:

$$A = \begin{array}{c} \begin{array}{cc} A_{11} & A_{12} \end{array} \\ \left[\begin{array}{cc|cc} 1 & 0 & 2 & 1 \\ 4 & 1 & 1 & 0 \\ \hline 0 & 1 & 3 & 0 \\ 5 & 0 & 2 & 1 \end{array}\right] \\ \begin{array}{cc} A_{21} & A_{22} \end{array} \end{array} \qquad B = \begin{array}{c} \begin{array}{cc} B_{11} & B_{12} \end{array} \\ \left[\begin{array}{cc|cc} 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 4 \\ \hline 2 & 0 & 1 & 1 \\ 1 & 3 & 5 & 0 \end{array}\right] \\ \begin{array}{cc} B_{21} & B_{22} \end{array} \end{array}$$

$$P = (A_{11} + A_{22}) * (B_{11} + B_{22})$$

$$= \left[\begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix} + \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}\right] * \left[\begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 5 & 0 \end{bmatrix}\right]$$

$$= \begin{bmatrix} 4 & 0 \\ 6 & 2 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 7 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix}$$

$$Q = (A_{21} + A_{22}) * B_{11}$$

$$= \left[\begin{bmatrix} 0 & 1 \\ 5 & 0 \end{bmatrix} + \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}\right] * \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 1 \\ 7 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix}$$

$$R = A_{11} * [B_{12} - B_{22}]$$

$$= \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 \\ -5 & 4 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix}$$

$$S = A_{22} * (B_{21} - B_{11})$$

$$= \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix}$$

$$T = [A_{11} + A_{12}] B_{22}$$

$$= \begin{bmatrix} 3 & 1 \\ 5 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 5 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix}$$

$$U = (A_{21} - A_{11}) * (B_{11} + B_{12})$$

$$= \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} * \begin{bmatrix} 0 & 2 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix}$$

$$V = (A_{12} - A_{22}) * (B_{21} + B_{22}) =$$

$$= \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix} * \begin{bmatrix} 3 & 1 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix}$$

$$C_{11} = P + S - T + V$$

$$= \underbrace{\begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix} + \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix}} - \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix} + \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix}$$

$$= \begin{bmatrix} 10 & 5 \\ 23 & 14 \end{bmatrix} - \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix} + \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 2 \\ 13 & 9 \end{bmatrix} + \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$$

$$C_{12} = R + T = \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix} + \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix} = \begin{bmatrix} 7 & 3 \\ 1 & 9 \end{bmatrix}$$

$$C_{21} = Q + S = \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix} + \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 1 \\ 5 & 8 \end{bmatrix}$$

$$C_{22} = P + R - Q + U = \underbrace{\begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix}} - \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 8 \\ 17 & 18 \end{bmatrix} - \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 4 \\ 9 & 10 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 7 \\ 7 & 7 \end{bmatrix}$$

$$C = \begin{bmatrix} 5 & 4 & 7 & 3 \\ 4 & 5 & 1 & 9 \\ 8 & 1 & 3 & 7 \\ 5 & 8 & 7 & 7 \end{bmatrix}$$

Scanned with CamScanner

Advantages & Disadvantages of Divide & Conquer Paradigm:

Advantages:

1. Solves Difficult problems
2. It reduces the degree of difficulty, since it divides the problem into sub problems that are easily solvable and usually runs faster than other algorithms would.
3. It also uses memory caches effectively. As when sub problems become simple, it can be solved within cache, without having to access the slower main memory, which saves time and makes the algorithm more efficient.
4. parallelism - It is suitable for multiprocessor machines.

Disadvantages:

1. Divide and conquer strategy uses recursion that makes it a slower and if a small/~~little~~ error occurs in the code, the program may enter into an infinite loop.
2. Usage of explicit stacks make use of extra space.

Decrease and Conquer :

- Decrease and conquer technique is based on exploiting the relationship between a solution to a given instance of a problem and solution to a smaller instance of same problem.

- Once such relationship is established, it can be exploited either top down (recursively) or bottom up (without a recursion).

- There are three major variations of decrease-and-conquer :
  - Decrease by constant
  - Decrease by constant factor
  - Variable size decrease.

Decrease by constant :

The size of instance is reduced by the same constant on each iteration of the algorithm.

- Consider an example, exponential problem of computing a^n for positive integer ~~const~~ exponents.

$$a^n = a^{n-1} * a$$

So $\boxed{f(n) = a^n}$

$$fn = \begin{cases} f(n-1) \cdot a & \text{if } n > 1 \\ a & \text{if } n = 1 \end{cases}$$

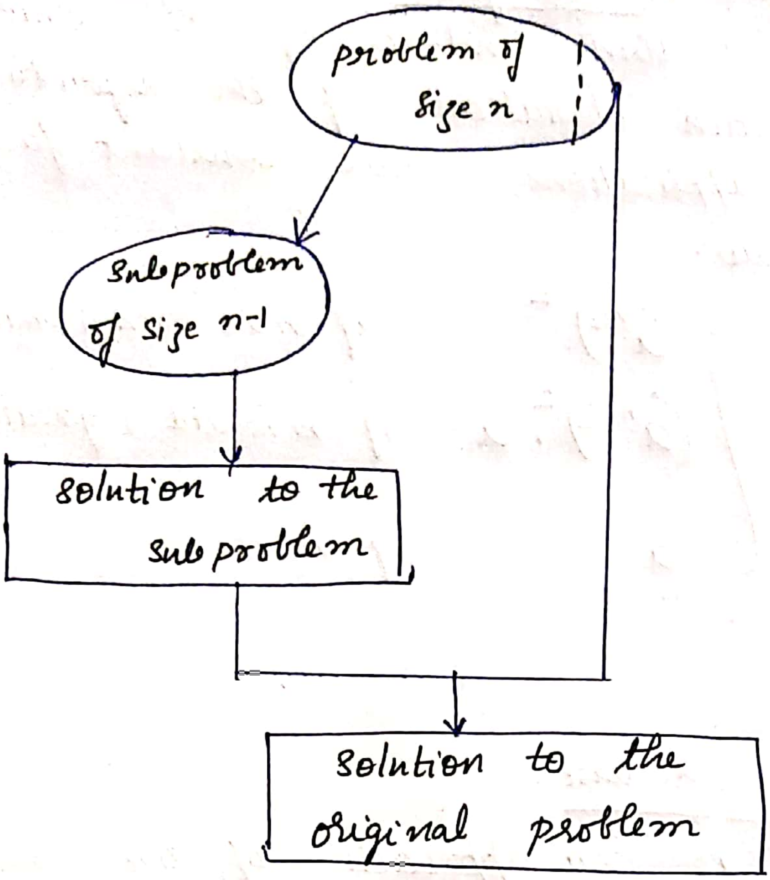
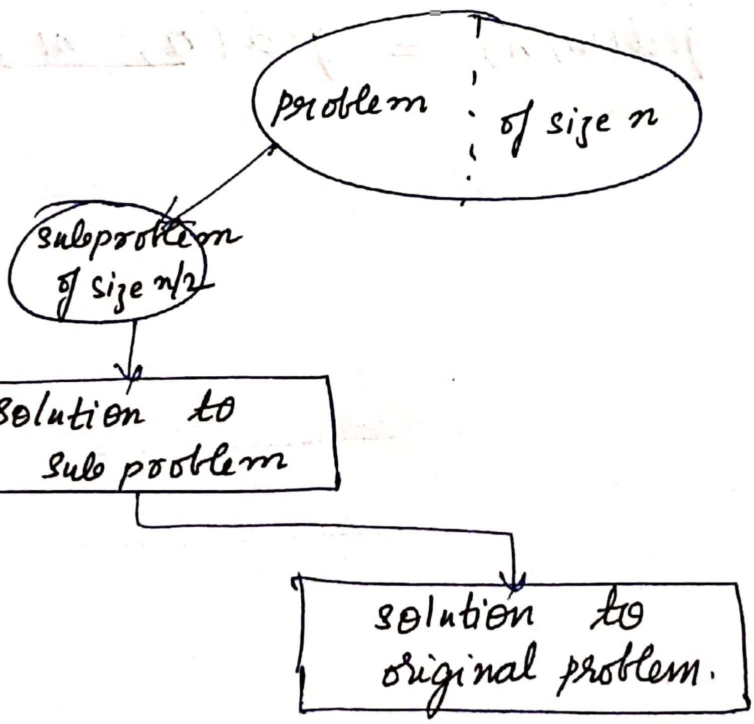


Figure: Decrease & conquer technique.

$f(n) = a^n$ can be computed either "top-down" by using its recursive definition or 'bottom-up' by multiplying a by itself n-1 times.

Decrease by constant factor:



- Decrease by a constant factor technique suggests reducing a problem instance by the same constant factor on each iteration of the algorithm. In most applications, this constant factor is equal to two.

Ex:

$$a^n = \begin{cases} (a^{n/2})^2 & \text{if } n \text{ is even \& positive} \\ (a^{n-1/2})^2 \cdot a & \text{if } n \text{ is odd \& greater than } 1 \\ a & \text{if } n = 1 \end{cases}$$

Variable size decrease:

- It is a approach of size reduction of instance input 'n'.
- Reduction pattern varies from one iteration of an algorithm to another

Ex: Euclid compute of GCD

$$gcd(m, n) = gcd(n, m \bmod n)$$

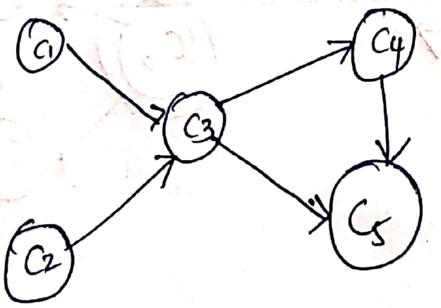
Topological Sorting:

Problem statement:

Given a digraph $G = (V, E)$, the task is to find a linear ordering of vertices such that for any edge (u, v) in E, u precedes v in the ordering.

* For topological sorting, the graph should be a DAG (Directed Acyclic Graph)

- Consider an example of set of five required courses $\{C_1, C_2, C_3, C_4, C_5\}$ a part time student has to take in some degree program. The courses can be taken in any order as long as following course pre-requisites are met: C1, C2 has no prerequistes, C3 requires C1 & C2, C4 requires C3, C5 requires C3 & C4. The student can take only one course per term. In which order should the student take the courses?



The situation can be modeled by a digraph in which vertices represent courses and directed edges indicate prerequisite requirements.

- This problem is solved by topological sorting.
- Two techniques to solve are:
  (i) Source Removal Algorithm
  (ii) DFS

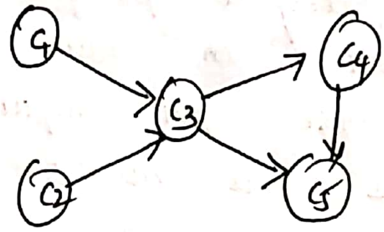
Source- Removal Algorithm:

- In a given digraph, identify a "source", a vertex with no incoming edges and delete it along with all the edges outgoing from it.

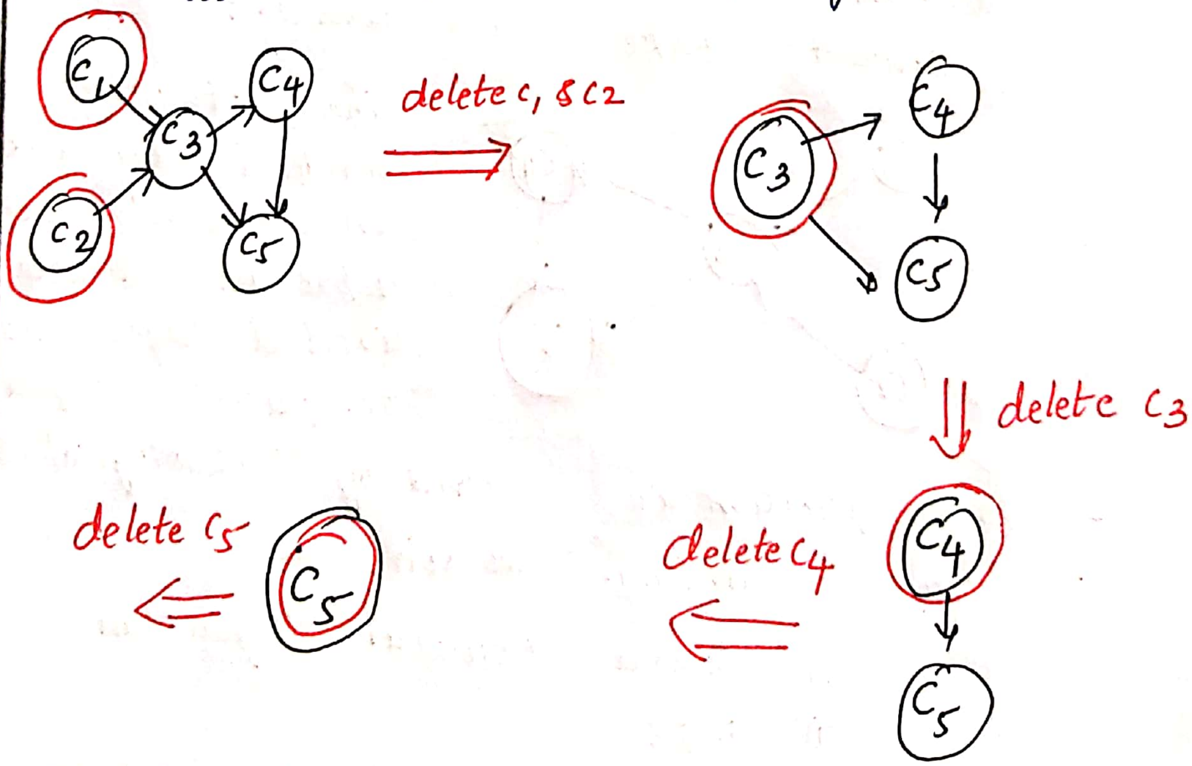
[If there are no source vertex, then problem cannot be solved]

- The order in which the vertices are deleted yields a solution to the topological sorting problem.

Ex:



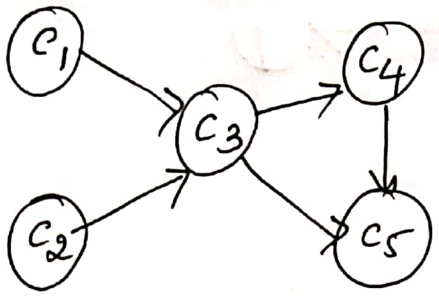
Sol^n In a Graph given, the source is C1, as it has no incoming edge:



Topological order is <c1, c2, C3, C4, C5>

(ii) DFS Based Method: 5

For a given digraph, perform DFS traversal & note the order in which vertices becomes deadends. Reversing this order results a topological ordering.



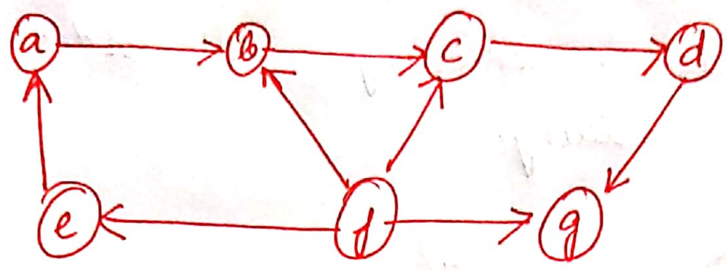
DFS traversal is performed as

C_5	1
C_4	2
C_3	3
C_1	4
C_2	5

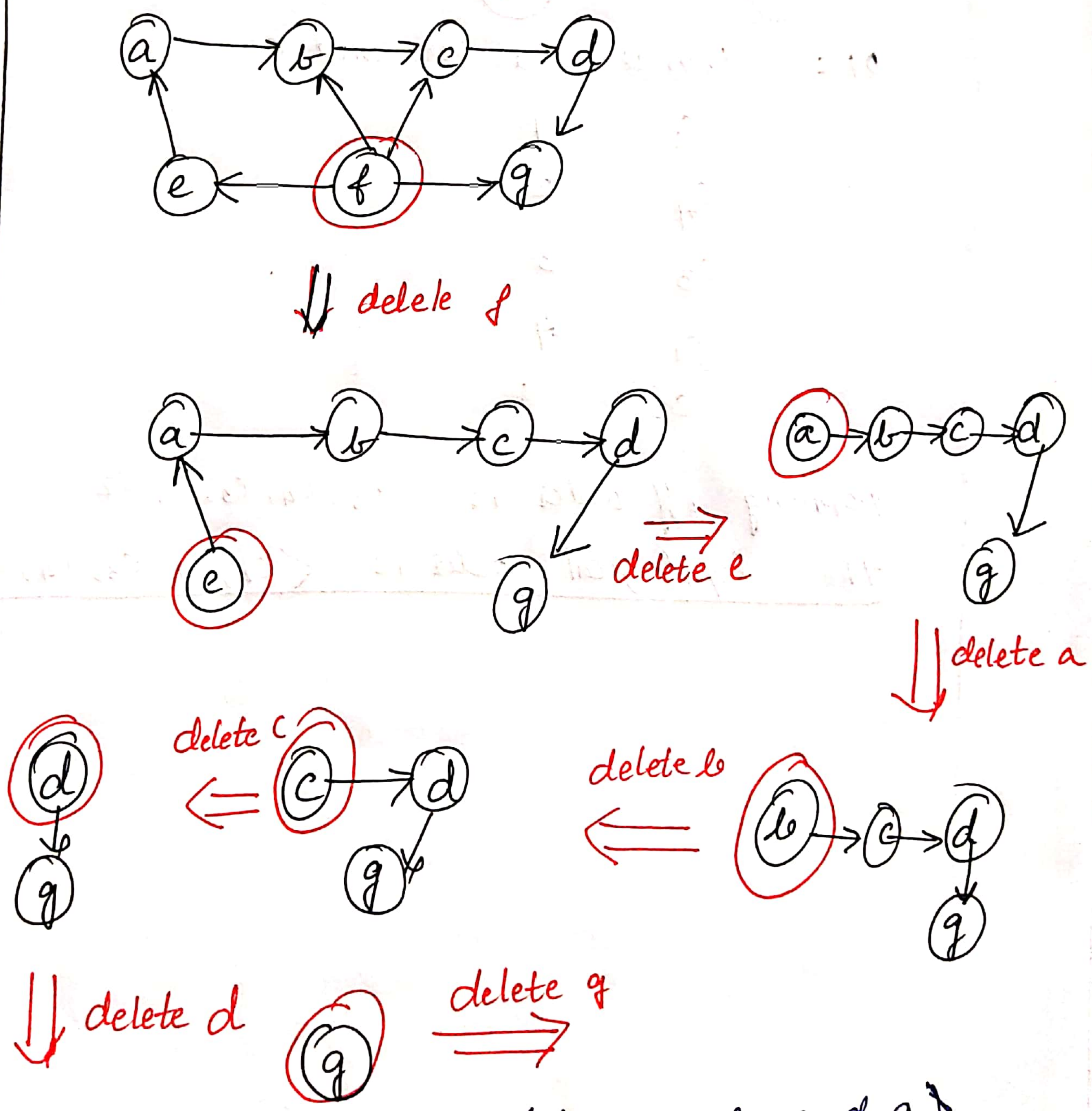
popping off order is C_5, C_4, C_3, C_1, C_2

the topological order is $\langle C_2, C_1, C_3, C_4, C_5 \rangle$

obtain the topological ordering for graph given below:



Source removal Algo:



Topological sorting = ⟨f, e, a, b, c, d, g⟩