***topological sorting***

**Scr code:**

from collections import defaultdict

class Graph:

    def \_\_init\_\_(self, vertices):

        self.graph = defaultdict(list)

        self.vertices = vertices

    def add\_edge(self, u, v):

        self.graph[u].append(v)

    def topological\_sort\_util(self, v, visited, stack):

        visited[v] = True

        for neighbor in self.graph[v]:

            if not visited[neighbor]:

                self.topological\_sort\_util(neighbor, visited, stack)

        stack.append(v)

    def topological\_sort(self):

        visited = [False] \* self.vertices

        stack = []

        for i in range(self.vertices):

            if not visited[i]:

                self.topological\_sort\_util(i, visited, stack)

        return stack[::-1]

graph = Graph(5)

graph.add\_edge(0, 2)

graph.add\_edge(1, 0)

graph.add\_edge(1, 3)

graph.add\_edge(3, 2)

graph.add\_edge(4, 1)

graph.add\_edge(4, 3)

result = graph.topological\_sort()

print("Topological Sorting Order:")

print (result)

**The output:**

Topological Sorting Order: [4, 1, 3, 0, 2]

**time complexity:**

**The topological sorting involves visiting each vertex once and each edge once (in the worst case).**

**The for loop iterates over each vertex, and within the loop, the topological\_sort\_util method may visit each edge once.**

**The time complexity of topological\_sort\_util is O(V + E), where V is the number of vertices and E is the number of edges.**

**Visualization tool:t**:[https://visualgo.net/en](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbjUxWUVjOUF4eE8zNjh5YVdOWkY3U2x0R2RBd3xBQ3Jtc0tsVWJSTFNpTUZrbmhkOGpwZkpGeG0yVU5ab2hFOVpTc0Q4bXo3dEd5NVRYWjZnUk5nSEVwbTdPUlRDVjBOUVZJMGpBaW81M2JUYmxDOHVpeUh6UlpuNWZKMTFmMFo3dEFHbHpaQ0lFSV9ONjMwOWJMQQ&q=https://visualgo.net/en&v=04imlyZIqgQ" \t "https://www.youtube.com/_blank)

**#Note :I have used chatGPT to help me in my code and The analysis**