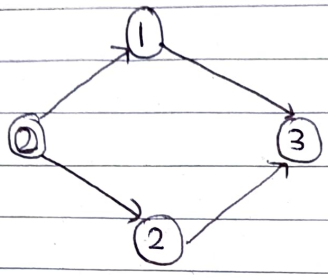
Date _____ Page _____

Day - 201

Graph-5

* Topological Sort:



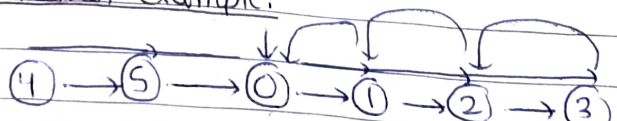

0 → 1 2
1 → 3
2 → 3
3 →

⇉ For topological Sort, graph should be Directed Acyclic Graph (DAG).

⇉ TS for a DAG is a linear ordering ~~of~~ of vertices such that every directed edge U → V, the vertex U comes before V.

⇉ So, TS of the above graph is —
0 2 1 3
0 1 2 3

⇉ Another example:




DFS of the above graph:
0 1 2 3 4 5

T.S. : 4 5 0 1 2 3

Date _____ Page _____

⇒ So, we can see that if we put 4 5 in the starting then the DFS will become our answer for T.S.

⇒ So, if we shift 4 5 then it will increase our time complexity.

⇒ So, we will use stack here for this task.

⇒ Now, when we are returning after visiting all the neighbours of that vertex then we will add that vertex to the stack.

⇒

| |
|---|
| 4 |
| 5 |
| 0 |
| 1 |
| 2 |
| 3 |

| ∅1 | ∅1 | ∅1 | ∅1 | ∅1 | ∅1 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

⇒ So, here we start from 0 then visit 1 → 2 → 3.

⇒ Now, there is no vertex remaining, now we backtrack.

⇒ And pushing the vertex into the stack.

⇒ So, T.S. → 4 5 0 1 2 3

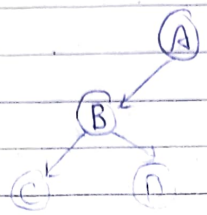
⇒ T.S. is used in the processors.

⇒ Suppose, a process B is dependent on A. That means, A → B.

⇒ Also, B → C D

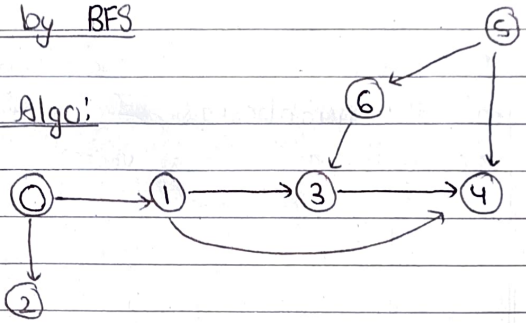
Date _____ Page _____

⇉ So, the graph will looks like —




⇉ Solving by BFS

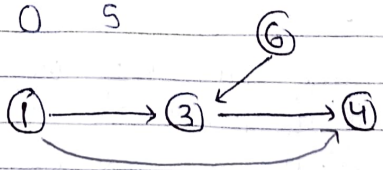
Kahn's Algo:




⇉ So, if we take above graph, the which process will execute first (if we are considering vertex as a process).

⇉ That process will execute first that is not dependant on any other process i.e 0 & 5.

⇉ So, 0 5




⇉ So, how → 0 5 2 1 6

(2)

Date _____ Page _____

③ → ④

0 5 2 1 6 3

④

0 5 2 1 6 3

So, this is Kahn's Algo.

So, we have to find those vertex that don't have any incoming edges.

| 0 | 1 | 1 | 2 | 3 | 0 | 1 |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

Indegree of every vertex.

Now, we will use a queue.

| 0 | 5 |
|---|---|

we will start from 0 indegree vertex.

Now, pop from queue then we decrease the outdegree of 5 that are on the other vertices.

| 0 | 0 | 0 | 2 | 3 | 0 | 1 |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

queue → | 5 |