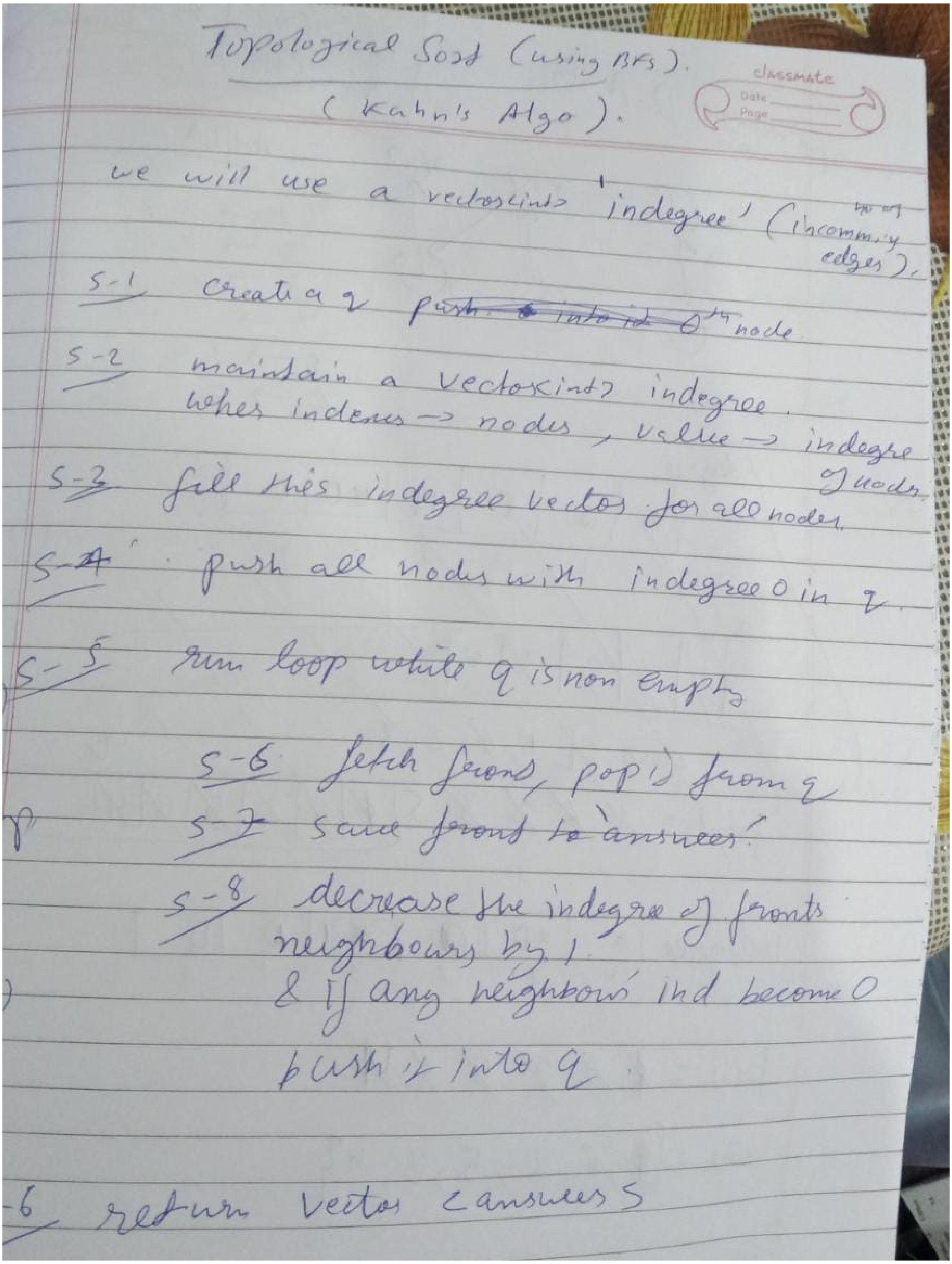
TopoSort using BFS (Kahn's Algo)



Topological Sort (using BFS).
(Kahn's Algo).

we will use a vector<int> 'indegree' (no of incoming edges).

S-1 Create a q ~~push into it 0~~th node

S-2 maintain a vector<int> indegree. where indexes -> nodes, value -> indegree of nodes.

S-3 fill this indegree vector for all nodes.

S-4 push all nodes with indegree 0 in q.

S-5 run loop while q is non empty

S-6 fetch front, pop() from q

S-7 save front to answer.

S-8 decrease the indegree of fronts neighbours by 1.
& if any neighbour ind become 0
push it into q.

-6 return vector<answer>

Why Kahn's Algo works ?

Firstly we don't need a 'visited' data structure here because we know that topologica Sort can be only found for directed, acyclic graph,

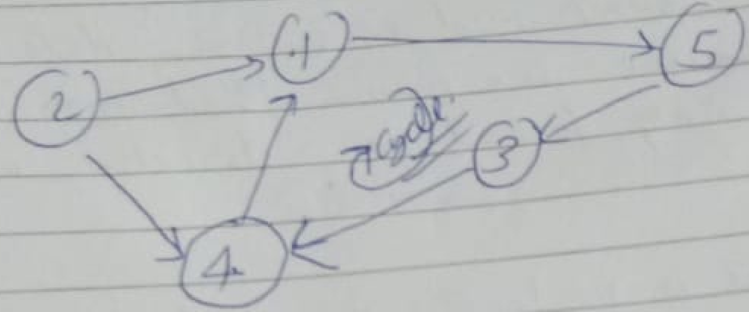
So if there is no loop, directed graph we can not ~~come~~ get into a loop. (so no need of visited)

Q) Why we need to maintain 'indegree' ?

Because in Topological Sort of DAG the very leftiest node can not have any parent or incomming nodes (indegre = 0) so thats why we push node with in0 in the q.

What if we apply Kahn's Algo on Undirected or Cyclic graph

eg.



q

~~2~~

Inorder

1	2	3	4	5
~~1~~	0	1	~~1~~	1

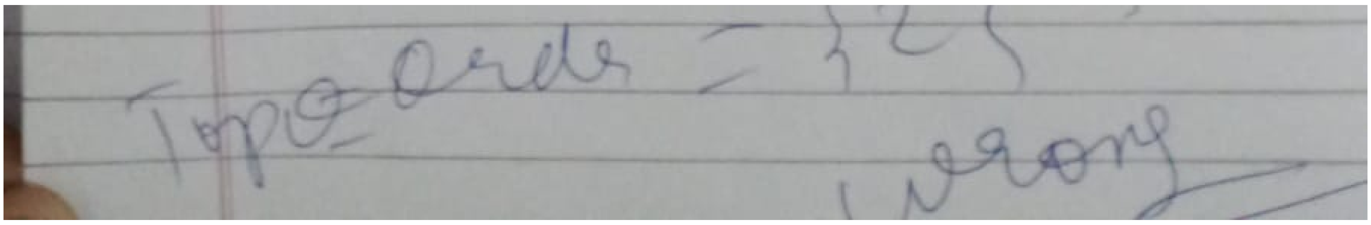
1 -
2 -
3 -

Adj = 1 → 5
2 → 1, 4
3 → 4
4 → 1
5 → 3

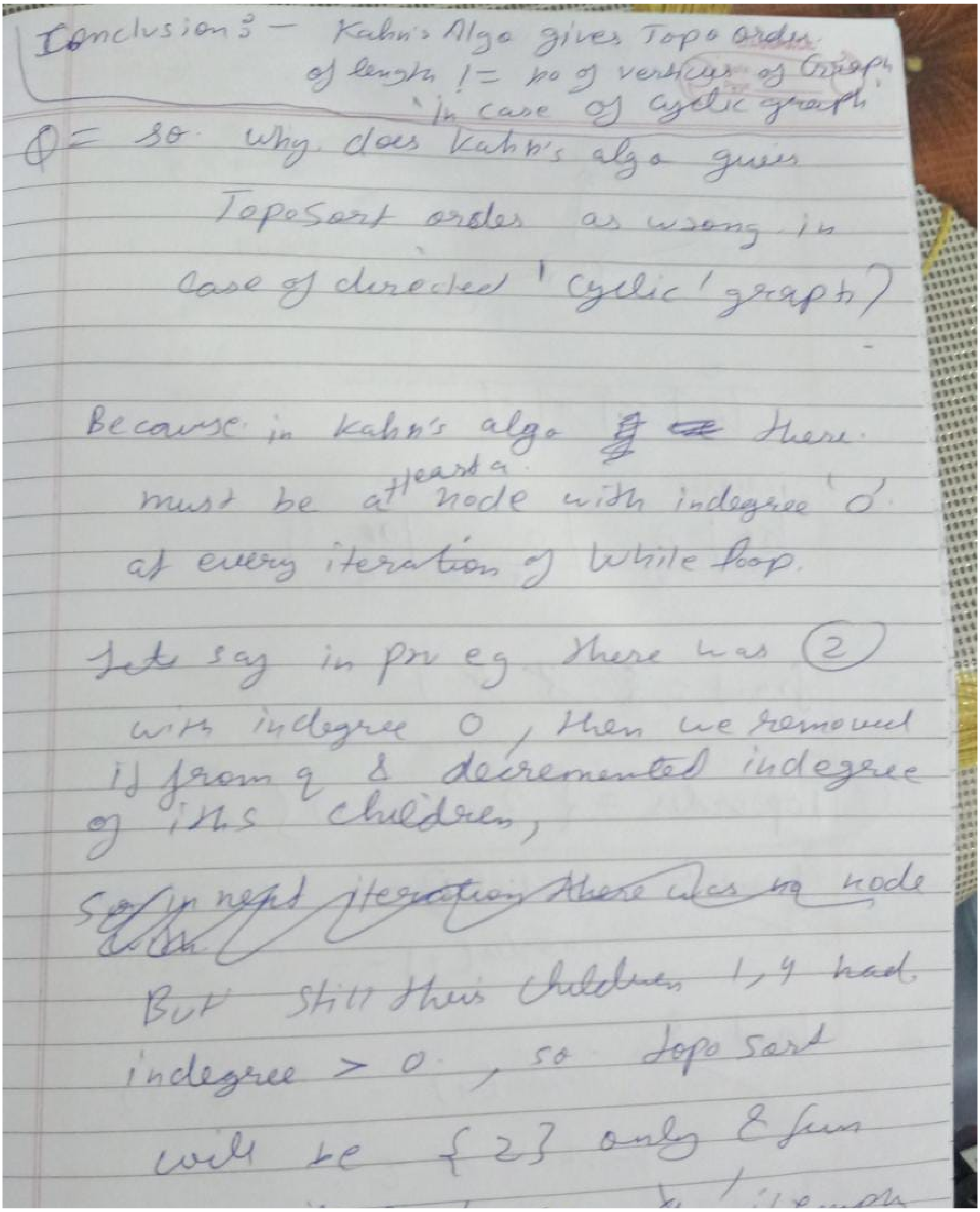
front = 2

Topo Ans : {2,

q is empty: pop


Topo Order = {2{

wrong


Conclusion :- Kahn's Algo gives Topo order of length != no of vertices of Graph 'in case of cyclic graph'

Q = so why does Kahn's algo gives TopoSort order as wrong in case of directed 'cyclic' graph?

Because in Kahn's algo ~~[illegible]~~ there must be atleast a node with indegree '0' at every iteration of while loop.

Lets say in prv eg there was ② with indegree 0, then we removed it from q & decremented indegree of its children,

~~So in next iteration there was no node with~~

But still their children 1, 4 had indegree > 0, so Topo Sort will be {2} only & [illegible]

[illegible]

will end. coz of [illegible]

now

Kahn's Algo (BFS toposort).

important conclusions :—

* if a undirected ~~[illegible]~~/ cyclic graph is provided, the kahn's Algo will return topological order of length not equal to total vertices of graph.

*. This above feature can be used in questions like 'schedule courses I' and 'schedule course II'

* Kahn's algo is a way to find out topological order using BFS.

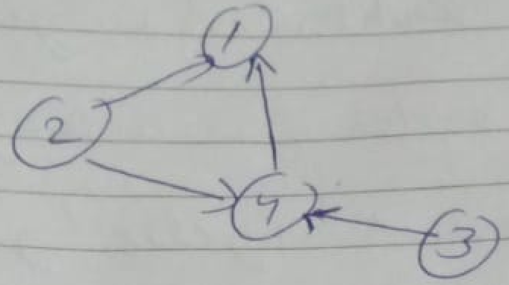
* BFS kahn's algo gives us topo. [illegible]

It BFS Kahn's algo gives us topo length
!= total vertices (in case of cyclic graph)

Whereas DFS topo order method give topo sort of length same as no of vertices (in case of acyclic graph)

which make it harder to know if a 'ToposSort' is correct order or not in case of DFS topo sort

another DAG eg.



1 →
2 → 1, 4
3 → 4
4 → 1

q.

~~2~~	~~3~~	~~4~~	~~1~~

indegree

0 ~~1~~	0	0	0 ~~2~~
1	2	3	4

front = ~~2~~. ~~3~~. ~~4~~ 1 valid

Topoorder = { 2, 3, 4, 1 }

front=2 inorder[1]-- (now=1)

inorder[4]-- (now=1)

front=3

inorder[4]-- (now=0)

push 4 to q.

front=4

inorder[1]-- (now=0)

push 1 to q

front=1

no child

= break