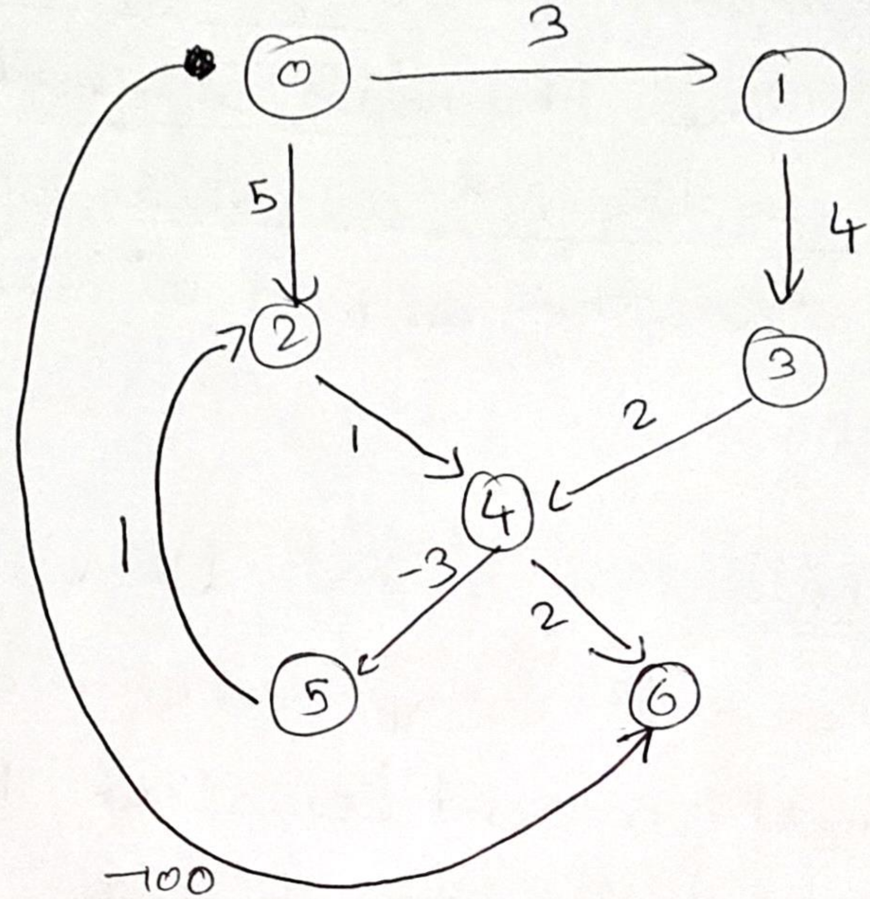
CORRECTNESS

Q1) The idea is to implement Bellman Ford Algorithm, with vertices being employees & edges being the weight of the conversation b/w them, & then in the nth iteration (n = no. of vertices) check for a negative cycle, which if exists, then do a depth first search from the source/node/vertex/employee ~~to~~ on the conversation graph & which ever nodes are reachable in that *single implementation of DFS are the nodes (employees) involved in a -ve weight cycle.

Reason (why DFS) : In a cycle, all ~~edges~~ vertices can be reached from 1 point to another. ∴, performing DFS from any 1 point in the cycle can lead us to discovering every other node that is a part of the cycle => which is what we ~~want~~ want.
(or the count)

So any conversation that could cause some employees that are either part of this cycle or downstream from this cycle to spiral endlessly into fouler & fouler moods, in other words => nodes that are part of a -ve cycle can be found this way.

For eg.:



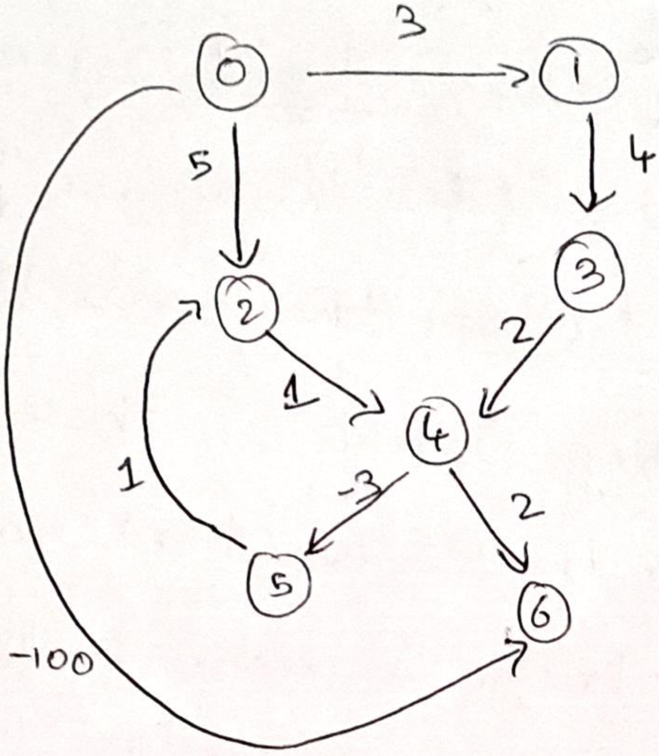
For the graph above, after running bellman fordalgorith & checking for -ve cycles, we find changes ~~in distance~~ in node ②, ④ & ⑤'s value.

Note that in the n^{th} iteration (or even few iterations after that), there will be ~~no~~ change in node ②, ④ & ⑤' value (-ve cycle) but not in node ⑥'s value.

∴, one might argue that why is DFS necessary ? Just the nodes in which change occurs in the nth iteration of bellman ford algorithm must be our required answer.

★ The thing to note here is that there might be some nodes in the graph which might not change in the current iteration, but might change in ~~futu~~ some future iteration.

~~Exam~~ For our example,



Notice that node ②, ④ & ⑤, which form the -ve cycle, ~~do~~ have their values decremented by 1 in every iteration.

So, node ② value goes from = 5 → 4 → 3 → ----

node ④ value goes from = 6 → 5 → 4 → 3 ...

node ⑤ value goes from = 3 → 2 → 1 → 0 ...

Now, node ⑥ value remains the same i.e. = -100, until the point node ④ value reaches = -103, & that's when node ⑥ value also changes from value = -100 to -101.

DFS from ② or ④ or ⑤ ⇒ [②, ④, ⑤, ⑥] : nodes involved in -ve cycle

∴ performing a DFS from any 1 node that is a part a -ve cycle, not only gives us the rest of the nodes that are part of the -ve cycle, but also the nodes that might not have any change in their distance value in the current iteration, but they might change in some future iteration.

Note that there might be multiples -ve cycles in the graph, which might be disjointed cycles. ∴, we need to perform DFS on every unvisited node in the n^{th} iteration of the bellman ford algorithm - whenever we detect a -ve cycle.

Q1) Complexity

Bellman ford algorithm $\Rightarrow O(mn)$ — ①

where m = edges

n = vertices

But in the n^{th} iteration, when checking for -ve cycles, we run a loop for all edges & whenever there is a change in node's distance value (-ve cycle), on that node we perform a DFS - if it's unvisited. If it's visited, then that means ~~cycle~~ ~~then~~ ~~th~~ cycle involving that node has already been ~~visit~~ discovered.

O(m) — ① for all edges

O(1) — ② if change in a node's value

O(1) — ③ if node is unvisited

④ perform DFS & find all nodes in cycl

The complexity of doing a DFS $= \underline{O(m+n)}$.

$\therefore$, one might think that the complexity of this part of the algorithm is

$$= O(\underbrace{n}_{\text{edges}} \times \underbrace{(n+m)}_{\text{DFS}})$$

$$= \underline{O(n^2+nm)}$$

But on careful analysis, it is very clear that DFS is performed only on those nodes which are unvisited i.e., if visited once, there cycle is already discovered & hence we do not need to revisit that cycle.

$\therefore$ complexity actual $= O(n \times m)$

$$= \underline{\underline{O(nm)}}$$