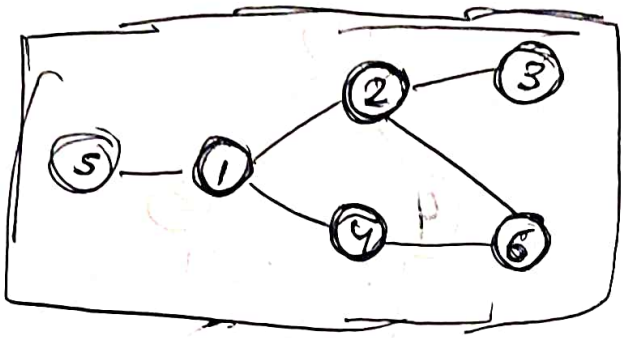
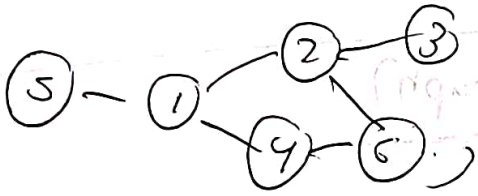
(Explanation)

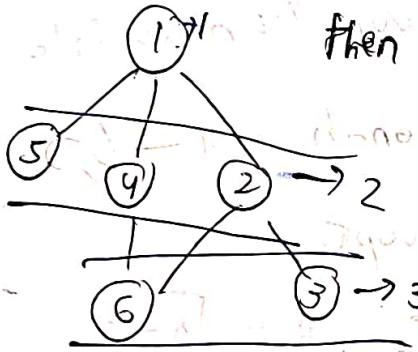
→ Prerequisite { Bipartite graph }



First thought in my mind, I had ~~naive~~ naive idea.

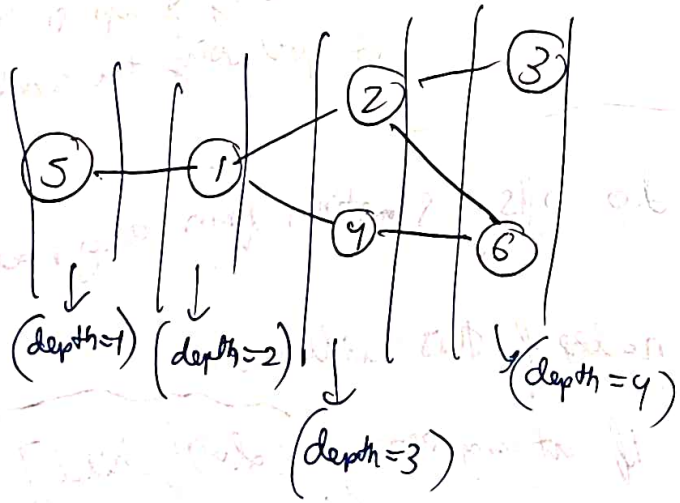


Lets say that we choose node 1 as Root



then we calculate depth. and whatever is the max depth, we consider that as the total number of groups.

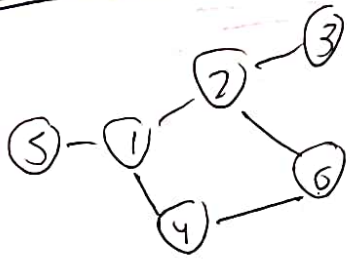
But I saw that depending on the node we choose as root, the max depth changes.



So ideally I could just do bfs starting from each node and get the answer.

(My thought Process)

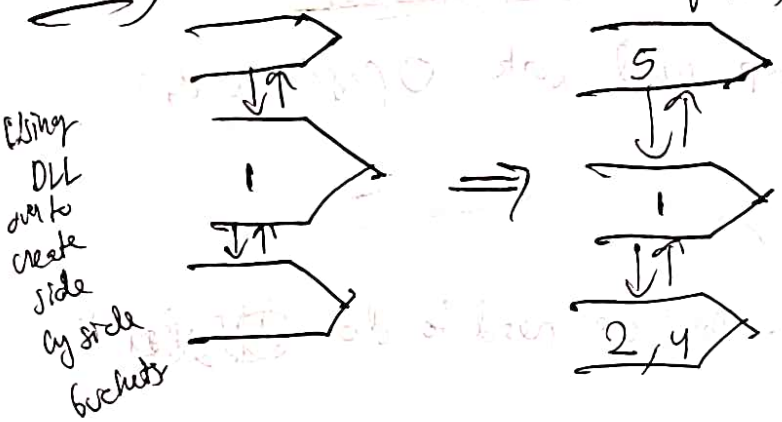
Pick any node at Random.



Step 1) → Picked node 1.

(Placed 1 in a bucket)

Step 2) Picked the neighbours of (1) → {5, 2, 4}.



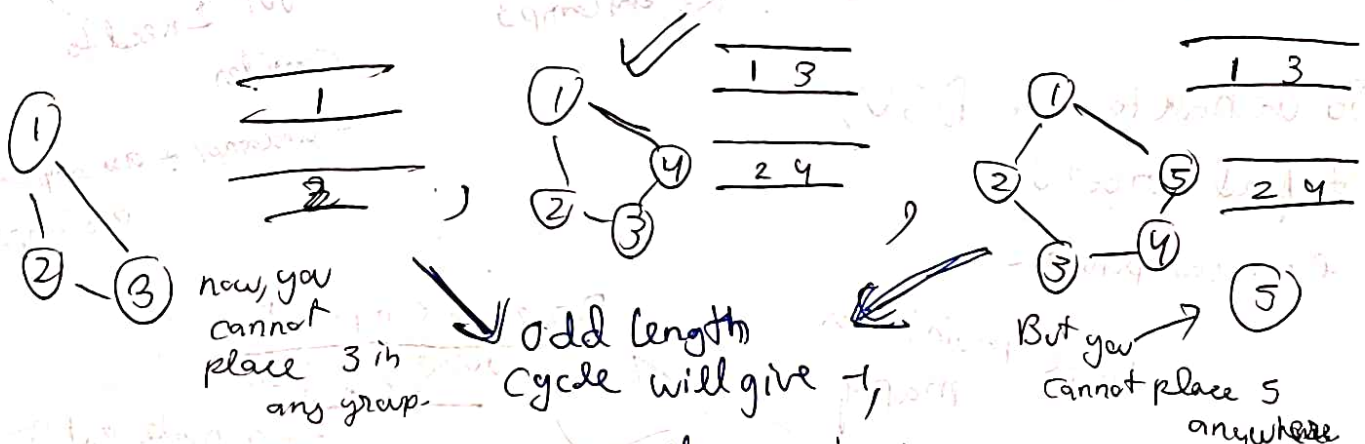
But we are not sure where to place the nodes in which bucket,

Upon placing the nodes on any bucket, in future it might be required to backtrack and place them in new bucket

The code would be very complicated and we are not sure abt the complexity.

Some negated this approach.

Now coming to the part, where this is not possible at all, I mean answer is -1,



now, you cannot place 3 in any group.

Odd length Cycle will give -1,

But you cannot place 5 anywhere

So we need to check if graph is Bipartite or not. If Bipartite, then ok, if not -1.

Now coming back to finding actual soln.:

Since n = 500,

edges = 10000,

$BFS = (n + edges)$

$= (500 + 10000)$

if we do

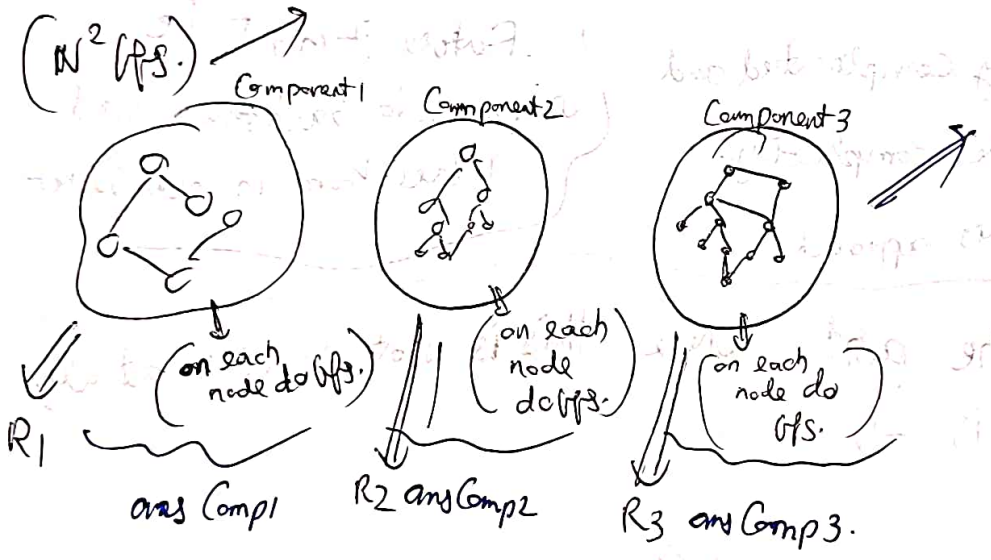
$n * (n + edges) = 500 * (500 + 10000)$

So actually a double for loop will work $O(N^2)$ works :)

---

1 more thing,

the graph can be disconnected. And we need to do the



the mistake I did,

then I will consider

max (ansComp1, ansComp2, ansComp3)

but I need to consider

= ansComp1 + ansComp2 + ansComp3

So we have to use DSU,

to find root of each component.

So we maintain a map of

For each component, doing BFS from each node get the value

{Root → Max depth}

Root (node) (BFS value)