HQ 21

To prove 3COLBUTI is NP complete, we ~~need to~~ can prove that 3COLBUTI is NP and NP hard which will thus prove that 3COLBUTI is NP complete.

proof that 3COLBUTI is NP

with the help of verification algorithm we will "proove that 3COLBUTI is NP.

proof P : mapping of vertices ~~to~~ of graph G(V,E) to {R, G, B} ie colour(v) ∈ {R, G, B}

input instance : an undirected graph G(V, E) with v vertices and E edges..

verification algorithm : def verify 3COLBUTI Proof (graph G, Proof P):

```
count = 0
for every edge (u,v) ∈ G:
    if (colour(u) == colour(v))
        count ++
if ( count ≤ 1)
    return True
return False
```

running time :

proof P is linear ie $O(V)$ where V number of vertices in G(V, E).

verification algorithm takes $O(V^2)$ complexity as it has to iterate over each edge ie, $O(E) \approx O(V^2)$. Rest all operations have constant time complexity. Thus overall running time complexity = $O(V^2)$ which is polynomial.

Lemma 1: If G has a yes instance then there must be a proof p, for which verify 3COLBUTI Proof gives true.

If G has a yes instance i.e G can be coloured using atmost 3 colours such that atmost 1 edge violates the colouring constraint, then according to our proof we will have the vertices marked with 3 colours either all different or ~~no~~ all vertices not with different colours. If all vertices are marked with different colour then the count variable in our verification algorithm will have a value 0 and thus return true satisfying the definition of the algorithm 3COLBUTI.

If all vertices are not of ~~sam~~ different colours then there must be maximum one edge whose 2 vertices have same colours. This is so because graph G ~~is maxi~~ can be coloured with maximum 3 colours with maximum one vertex repeating one of the 3 colours thus satisfying the 2nd part of the problem: ~~each~~ 1 edge violates colouring constraint. Thus this edge will cause count to increment to 1 in our verification algorithm which will again return True after satisfying the if condition.

Lemma 2: If G has a NO instance then for every P, verify 3COLBUT1 Proof will False.

If G has a NO instance i.e, G is not colorable with atmost 3 colours such that atmost 1 edge violates the colouring constraint then ~~according to our proof we have only 3 inst colours to map to the vertices. Now if G is not colorable with atmost 3 colours ie~~ there must be more than one edge violating the colouring constraint thus in our algorithm count will increment to greater than 1 (ie no such proof exist such that ~~only~~ maximum one edge exist with same colour of vertices) and thus will ultimately return a false.

proof that 3COL BUT1 is NP Hard

To proove that 3COLBUT1 is NP Hard, we will reduce a known NP Hard problem to 3COLBUT1.

We will reduce 3COLOUR problem (which is a known NP Hard problem) to 3COLBUT1

reduction algorithm

```
def reduce( graph G):
    G' = new empty graph
    for all vertices V in G:          — O(V)
        copy V in G'
    for all edges E in G:             — O(E)
        copy E in G'
    add 4 new vertices in G' and connect all of
    them to each other.               — O(c)
    return G'
```

explanation

we are copying the entire graph G to a new graph G' and in G' we are creating a disconnecting component of 4 ~~...~~ vertices which are connected to each others (to each of the 4 vertices).

running time

running time complexity of the reduce ( ) algorithm as shown above is — O(V) + O(E) + O(c). which is polynomial in time.

lemma

we will get Yes instance of 3COLOUR iff we get a Yes instance of 3COLBUT1.

If 3COLOR returns a YES instance ie graph G is colourable with maximum 3 colours, then in my reduced graph G' we have 2 graphs, the graph which is similar to G is colourable with 3 colours but the graph with 4 vertices all connected to each other will have ~~...~~ exactly 1 edge whose vertices have same colour. Thus overall G' has 1 edge whose constraint is not satisfied. Thus 3COLBUT1 also returns ~~It~~ a Yes instance as it satisfies the definition of the problem.

If 3COLOR returns a NO instance ie graph G ~~has~~ cannot be coloured with 3 vertices then in my reduced graph the disconnected component of 4 vertices already has an edge whose colour constraint is not satisfied. Now another disconnected component of G' which is the same graph as G is now not colourable ~~to~~ using atmost 3 colours ie in this disconnected component we have atleast one edge not satisfying the colouring constraint. Thus G' has more than 1 edge not satisfying the colouring constraint. Hence 3COLOROUT1 will also give a NO instance