4. Either prove that the following problem is NP-complete or prove that it belongs to P by giving a polynomial time algorithm.
Input: An undirected graph H.
Question: Does H have a proper colouring with three colours R, G, B that assigns the colour B to at most 10 vertices?
Solution:
This problem is in P.
Since we know that bipartite graphs are 2-colorable, this question is equivalent to asking if there are 10 vertices or less in the graph such that the removal of them makes the rest of the graph bipartite. This means that we can color those 10 (or less) vertices B and the rest of the graph can be colored using R and G since it's bipartite.
Now in the question since we have "at most 10 vertices", we have to check if this is possible for all integers from 0 to 10.
The polynomial time algorithm I propose:
We will use nested for loops to examine all the combinations of vertices to be removed and DFS or BFS to check if the rest of the graph is bipartite.
To check for 0 vertices colored B, we can just run BFS or DFS to check the original graph is bipartite.
For the rest, so 1 to 10 vertices, we will have 10 individual for loops, where for examining for n vertices we have n nested loops. So for example, if we want to check if we can color 1 vertex B and the rest with G and R then our loop only need to iterate through all the vertices of the graph and check if the graph is bipartite with that vertex removed this loop has O(n) iterations. But if we want to check the same thing for 10 vertices we will have 10 nested for loops, to examine all combinations of 10 vertices in the graph, so worst case (n^{10}) iterations. Here n is the number of vertices we have in the graph H.
If at any point of the loops we reach a bipartite graph we will return TRUE, and if we finish all the loops and don't get a bipartite graph for any combination of vertices removes, then we return FALSE.
Depending on how we're representing the graph, the complexity of checking if it is bipartite changes but with BFS or DFS it'll always be polynomial.
Since we are using BFS or DFS to check if the graph is bipartite and we have a polynomial time calls to it, we will have the sum of polynomial time algorithm which makes our algorithm polynomial time.

Example Algorithm:
for 0 vertecies colored B:
BFS(G)

for 1 vertices colored B:
loop for all vertices v in G
BFS(G\v)

for 2 vertices colored B:
loop for all vertices v1 in G

loop for all vertices v2 in (G\v1)
BFS(G\{v1,v2})

...
for 10 vertices colored B:
loop for all vertices v1 in G
...
loop for all vertices v10 in G\{v1,v2,..,v9}
BFS(G\{v1,v2,..,v10})