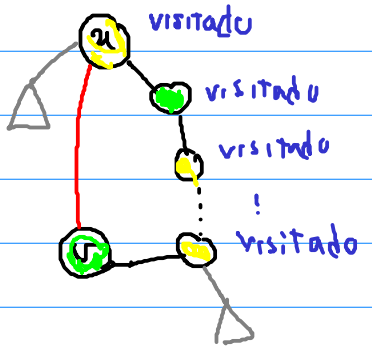
Ex. 2 Probar la correctitud del algoritmo greedy que corre un DFS para determinar si un grafo es bicoloreable.

Prueba: Sabemos que un grafo es bicoloreable si y solo si no contiene ciclos de longitud impar. Por lo tanto, separaré la prueba en dos caso, cuando el input es un grafo bicoloreable y cuando no lo es.

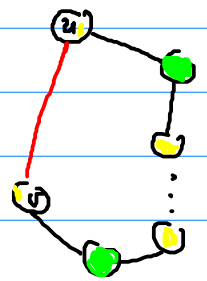
Caso 1: G es un grafo bicoloreable. Tras correr el DFS, en la única línea en la cual el algoritmo puede decidir si el grafo es o no bicoloreable es cuando hemos encontrado un nodo ya visitado; es decir, la arista actual es un back edge. Asimismo, nos encontramos en un ciclo y, al ser bicoloreable, sabemos que la longitud de este es par, esto quiere decir que tenemos el path $\langle a_1, a_2, \dots, a_{2k-1}, a_{2k}, a_1 \rangle$ ($a_1 = u$, $a_{2k} = v$, $a_1 = u$)



y como el algoritmo pinta a los nodos en el orden de descubrimiento, si a_i es ● , i impar / ● , i par

tenemos que no hay conflicto con el color de a_{2k} y a_1.

Caso 2: G es un grafo no bicoloreable. Entonces existe al menos un ciclo de longitud impar. Como el DFS recorre cada arista del grafo, analicemos cuando se pasa a través de la arista que cierra uno de estos ciclos. Sabemos que el DFS genera un árbol, el cual es siempre bicoloreable, así que tenemos el path $\langle a_1, a_2, \dots, a_{2k}, a_{2k+1}, a_1 \rangle$ ($a_1 = u$, $a_{2k+1} = v$, $a_1 = u$)



con a_i ● , i impar / ● , i par, donde tendríamos los nodos a_{2k+1} y a_1 del mismo color unidos por una arista y, por lo tanto, el algoritmo inmediatamente retornaría que G no es bicoloreable. ∎