Lab04: SAT y reducciones entre problemas

1. **Sobre el problema SAT.**

   Escribe una función `is_satisfied` que recibe como entrada el número de variables, una fórmula Booleana y una asignación a las variables y devuelve True (cierto) si la fórmula evalúa a cierto con dicha asignación y False (falso) en caso contrario.

   La fórmula Booleana es expresada como sigue:

   - `num_variables` contiene el número de variables que pueden aparecer en la fórmula. Las variables siempre están numeradas de 1 a `num_variables`.
   - La fórmula Booleana está representada como una lista de listas (cláusulas). Cada lista interna correponde con una única cláusula de manera que cada literal x_i de la cláusula se representa con un i y cada literal $\neg x_i$ se represeta con un $-i$.
   - Por ejemplo, la fórmula Booleana

     $$\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4)$$

     estaría representada por

     `num_variables` $= 4$
     `clauses` $=$ [[1,2,-3],[2,-4],[-1,3,4]].
   - Para facilitar el proceso, las asignaciones `A` serán listas con `len(A) = num_variables + 1`. Por convenio, siempre `A[0] = 0`, ya que no hay ninguna variable x_0. Una asignación posible para las variables de φ podría ser `A=[0,1,0,1,0]`, de forma que `A`(x_1)`=1, A`(x_2)`=0, A`(x_3)`=1, A`(x_4)`=0`.

   Para realizar el ejercicio disponéis del fichero `lab04_satisfied.py`.

   ¿Cuál es el tiempo de ejecución de tu algoritmo?

2. **Reducción de grafo 3-coloreable a grafo 4-coloreable.**

   Escribe una función `graph_is_3colorable` que devuelve True si un grafo de entrada se puede colorear con 3 colores de forma que todos los nodos adyacentes tengan diferente color y False en caso contrario.

   El propósito de este ejercicio es implementar la función `graph_is_3colorable` a partir de la función `graph_is_4colorable`, que decide si un grafo se puede colorear con 4 colores de forma que los nodos adyacentes tengan siempre diferente color.

Para realizar el ejercicio disponéis de la carpeta `Three-color-maps`, que contiene el código de la función `graph_is_4colorable` y el fichero `lab04_threecolor.py`, que importa la función anterior en su cabecera: (from fourcolor import graph_is_4colorable).

Es imprescindible usar `graph_is_4colorable` en la implementación de la función `graph_is_3colorable`.

3. **Reducciones entre Vertex-Cover, Independent-Set y Clique.**

   En este ejercicio os damos la función `solve_vc` que resuelve Vertex-Cover para un grafo dado representado con su matriz de adyacencia. La función `solve_vc` devuelve un vertex mínimo del grafo, representado por una lista de 0s y 1s, donde un 0 significa que el correspondiente nodo no se encuentra en el vertex y un 1 que sí se encuentra.

   El propósito de este ejercicio es usar `solve_vc` para implementar una función `multisolve` que recibe como entrada un grafo y un string identificando un problema. El string recibido será "VERTEX COVER", "INDEPENDENT SET" o "CLIQUE". Cuando el string sea "VERTEX COVER" debe devolver un vertex mínimo del grafo de entrada. Cuando el string sea "INDEPENDENT SET" debe devolver un independent set máximo del grafo de entrada. Cuando el string sea "CLIQUE" debe devolver un clique máximo del grafo de entrada.

   Para realizar el ejercicio disponéis de la carpeta `Reductions-3problems`, que contiene el código de la función `solve_vc` y el fichero `lab04_reduction_3problems.py`, que importa la función anterior en su cabecera (from vertex_cover import solve_vc).

   Puedes usar funciones adicionales pero es necesario usar `solve_vc` para resolver cada uno de los problemas.