Lab07: Pre-processing SAT

1. **Sobre preprocesar instancias de SAT**

   Para este problema usaremos el mismo formato del laboratorio 4 para representar fórmulas booleanas en CNF.

   - `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 representa 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 `lab07_pre_processing.py`.

   Debes implementar la función `sat_preprocessing`, que dada una fórmula booleana en el formato anterior, el número de variables sobre el que la fórmula está definida y una asignación de las variables que hay que ir calculando (al principio todas las variables tienen asignado el valor None), aplica varias reglas para reducir la fórmula y, a la vez, va asignando valores a las variables según convenga.

   Las reglas son las siguientes:

   (a) Si hay alguna cláusula formada por una única variable (negada o no), asigna a dicha variable el valor de verdad para que la cláusula se haga cierta.

   (b) Si una variable aparece sólo una vez, asigna a dicha variable el valor de verdad para que la cláusula se haga cierta.

   (c) Elimina todas las cláusulas que se hayan hecho ciertas

   (d) Elimina todos los literales que evalúen a `False`. Es decir, todas las variables x que tengan asignado un 0 y todas las $\neg x$ que tengan asignado un `True`. Como resultado a este proceso, si una cláusula se reduce a [ ], la fórmula de entrada es insatisfactible y se debe devolver la fórmula [[1],[-1]]

Tu implementación debe recoger el hecho de que las reglas se deben aplicar exhaustivamente. Esto es, hasta que no se pueda aplicar más. Ten en cuenta que después de aplicar las reglas (a), (b), (c), y (d) puede ocurrir que haya nuevas variables que aparezcan sólo una vez y que haya cláusulas formadas por una única variable, por lo que se pueden volver a aplicar las reglas.

Cuando el pre-proceso haya acabado, puede ser que en la fórmula resultante no queden cláusulas y la fórmula sea igual a la lista vacía. Esto quiere decir que la fórmula resultante es satisfactible trivialmente. En este caso debes devolver la lista vacía y la asignación que hayas calculado. Si la fórmula resultante es insatisfactible debes devolver [[1],[-1]] y cualquier asignación. En otro caso devolverás la lista de cláusulas resultante y la asignación que hayas calculado.

2. **Para probar la solución**

   Se han incluido dos tipos de tests `test1` y `test2`. Hay que descomentar las llamadas a `test1` y `test2` para probar vuestro programa. El primer test contiene aserciones como en los demás laboratorios. El segundo test llama a un generador aleatorio de fórmulas CNFs con las que se prueba vuestro programa mostrando la solución.