Lab09: 3SAT-solver

En este laboratorio vas a construir un 3SAT-solver. Con este laboratorio y el siguiente podrás obtener el 10% de la nota final de la asignatura.

1. **Formato de las fórmulas**

   Recuerda la representación de las 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 $-i$.
   - 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`.

2. **Juego de pruebas y MINISAT**

   La carpeta `instances` en la carpeta `Code for Students` contiene un juego de pruebas (`https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`).

   Cada instancia está en un fichero de texto en formato DIMACS (éstandar para la mayoría de los SAT-solvers).

   MINISAT

   Vamos a ver cómo funciona **minisat** con la primera fórmula del fichero `instancias`: La fórmula `1-unsat.cnf` es insatisfactible. Teclea en un terminal:

   ```
   minisat 1-unsat.cnf
   ```

   En caso de que la fórmula sea satisfactible, puedes obtener una asignación a las variables que la haga cierta. Si escribes

   ```
   minisat 2-sat.cnf asig-2.txt,
   ```

   se creará un fichero `asig-2.txt` donde se encuentra la asignación. Ten en cuenta que el cero final no significa nada.

   Uso del juego de pruebas

   Puedes usar el juego de pruebas para probar este laboratorio.

   Para añadir este juego a tus pruebas hay que transformar el formato DIMACS en nuestro formato. Para ello hemos creado una función

   ```
   list_minisat2list_our_sat()
   ```

Esta función, dado un fichero en formato DIMACS, devuelve una tupla (t1, t2) donde t1 = número de variables y t2 = fórmula en nuestro formato. Para usarla hay que copiar el fichero `tools.pyc` que se encuentra en la carpeta `Code For Students` e incluir from tools import list_minisat2list_our_sat.

Para ver cómo se preprocesa la fórmula `1-unsat.cnf` puedes usar la función de esta manera: tupla = list_minisat2list_our_sat ('instancias/1-unsat.cnf')

y ver el resultado de tu 3SAT-solver: print **solve_3SAT**(tupla[0], tupla[1]).

3. **Implementación del 3SAT-solver**

   Implementa la función **solve_3SAT**(num_variables, φ) que se aplica a una fórmula 3CNF, φ, de forma que, tras realizar un pre-processing, usa un árbol de búsqueda para construir una asignacíon que haga cierta a φ o asegurar que φ es UNSATISFIABLE. Por tanto tu función debe devolver o bien una asignación o UNSATISFIABLE. Para ello, abre el fichero `Lab09_3SAT_solver.py` que se encuentra en la carpeta `Code for Students`.

   - Elige una cláusula que tendrá literales sin asignar. Ten en cuenta que tras el pre-processing puede haber cláusulas con menos de 3 literales, pero no más.
   - Abre tres posibles ramas del árbol (puede haber menos). Una por cada variable que aparece en la cláusula. La primera rama corresponderá a la primera variable que está sin asignar. Asigna el valor de verdad a esa primera variable, de forma que la cláusula se haga cierta. Continúa con esta opción hasta conseguir hacer cierta a φ. Si lo consigues, has terminado. Si no, debes abrir la segunda rama. En esta rama debes asignar a la primera variable el valor contrario a lo que asignaste previamente (por tanto este literal ahora será falso) y asignar a la segunda variable el valor que haga cierta a la cláusula. Continúa con esta opción hasta conseguir hacer cierta a φ. Si lo consigues has terminado. Si no, debes abrir la tercera y última rama (si es posible), donde los dos primeros literales serán falsos y la tercera variable llevará el valor que haga cierta a la cláusula. Continúa con esta opción hasta decidir si φ es satisfactible o no.
   - Por ejemplo, si tienes una cláusula como $[-1, 3]$, la primera rama corresponderá a asignar a la variable x_1 el valor 0. En caso de que φ no se satisfaga, la segunda rama corresponderá a asignar a x_1 el valor 1 y a x_3 el valor 1. No podrás abrir más ramas.
   - Valora si es conveniente hacer pre-processing sólo una vez y al principio o cada vez que asignas un nuevo valor a una variable.

4. **Comprobación del tiempo de ejecución de tu 3SAT-solver**

   Se valorará el tiempo de ejecución de tu 3SAT-solver. Para que te hagas una idea de cuánto debería tardar tu solución, hemos dejado un documento, en la carpeta `instances`, con los tiempos que tarda un 3SAT-solver hecho por nosotros. Ten en cuenta que estos tiempos dependen del ordenador que estés usando.