Laboratorio de Algoritmos y Estructuras III
Dpto. de Computación y Tecnología de Computación.

2-Satisfacibilidad

Proyecto 2 (Semana 4 - Miércoles Semana 6)

El problema de satisfabilidad se define como: dado una fórmula lógica, determinar si existe una asignación de sus variables en constantes, tal que dicha fórmula evalúe verdadero. Por ejemplo, si tenemos la fórmula lógica

$$p \wedge (\neg q \vee r)$$

La asignación $[p, q, r := true, false, false]$ la hace verdadera. Este problema pertenece a la clase de complejidad *NP-Completo*. Informalmente, estos problemas tienen la cualidad de que no se conoce un algoritmo polinomial que los resuelva, de hecho, se cree que no existe. La definición formal de este concepto no es de interés en el presente proyecto dado que estaremos trabajando con una reducción del problema.

Se dice que una fórmula lógica está en Forma Normal Conjuntiva (CNF por sus siglas en inglés) si es una conjunción de cláusulas, donde cada cláusula es a la vez una disyunción de literales (variables o negaciones de variables).

La siguiente es una fórmula válida en CNF con tres cláusulas.

$$(p \vee w \vee \neg r) \wedge (q \vee \neg p \vee w) \wedge (q \vee \neg r \vee p)$$

Decimos que una fórmula lógica se encuentra en $k - CNF$ si el número de literales por cláusula es k. La fórmula anterior se encuentra en $3 - CNF$. Cualquier fórmula $k - CNF$ se puede reducir a $3 - CNF$ si $k \geq 3$. Sin embargo, no es siempre posible transformar una fórmula en $3 - CNF$ a $2 - CNF$.

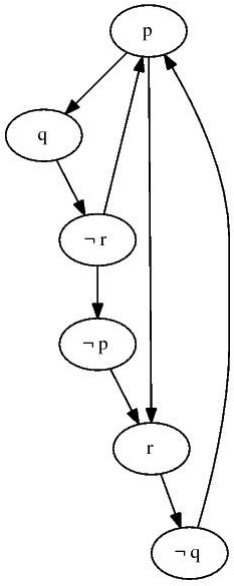
De hecho existe un algoritmo que resuelve el problema de satisfacibilidad para fórmulas lógicas en $2 - CNF$. Este algoritmo se apoya en la teoría de grafos que estamos estudiando. Veamos cómo modelar el problema utilizando grafos.

Sea la fórmula:

$$
\begin{array}{cc}
 & (\neg p \vee q) \\
\wedge & (\neg q \vee \neg r) \\
\wedge & (p \vee r) \\
\wedge & (\neg p \vee r)
\end{array}
$$

Por cada literal (positivo y negativo) generamos un nodo del grafo, por lo tanto si la fórmula tiene n variables tendremos $2n$ nodos en el grafo.

Sabemos por definición de la implicación que $p \Rightarrow q \equiv \neg p \vee q$. Los arcos del grafo vienen dados por la relación de implicación entre cada par de nodos. Para la fórmula dada, el grafo final sería:



El algoritmo simplemente verifica si algún nodo x alcanza a su análogo $\neg x$ y a su vez $\neg x$ alcanza a x. En dicho caso responde que la fórmula no es satisfacible.

Nótese que en el ejemplo anterior p alcanza a $\neg p$ y $\neg p$ alcanza a p. Dado que el grafo viene dado por el grafo de implicaciones de la fórmula, esto significa que $p \equiv \neg p$. Por lo cual dicha fórmula no se puede satisfacer.

Una primera implementación de este algoritmo podría considerar explorar los nodos alcanzables desde cada literal y verificar la condición. Sin embargo, para una fórmula con n variables, este algoritmo sería $\mathcal{O}(n^2)$.

Estudiando la condición a verificar, podemos ver que si x y $\neg x$ se alcanzan mutuamente, deben pertenecer a la misma componente fuertemente conexa, por lo que es posible lograr una implementación $\mathcal{O}(n)$.

Su labor en este proyecto es determinar, si una fórmula en $MAX-2CNF$ es satisfacible. La variante $MAX-2CNF$ permite cláusulas de tamaño menor o igual a dos.

Piense cómo debe modificar su algoritmo con las bases que conoce de lógica para que sea válido con cláusulas de tamaño cero y uno.

1. Entrada

Su programa recibirá como argumento en la línea de comandos un nombre de archivo. Dicho archivo contendrá una fórmula lógica. La primera línea del archivo tendrá dos naturales `n` y `m`. `n` es el número de variables y `m` el número de cláusulas. Los identificadores de las variables son naturales positivos. Un literal negativo indica que es una ocurrencia negada de la variable. Es decir, cada literal está identificado por un número.

Cada cláusula termina con un cero (0). Nótese que una cláusula puede estar en más de una línea y que cada línea puede tener más de una cláusula.

La sintáxis del ejemplo brindado sería:

```
3 4
-1 2 0
-2 -3 0
 1 3 0
-1 3 0
```

2. Salida

Debe imprimir por salida estandar "Si" en caso de ser satisfacible la fórmula dada o "No" en caso contrario. Las comillas se utilizan por claridad, no deben ser mostradas en la salida. Adicionalmente debe colocar un fin de línea posterior a su respuesta.

La salida para el ejemplo brindado sería:

```
No
```

3. Ejemplo Entrada

```
4 3
1 0
2 0
3 -4 0
```

4. Ejemplo Salida

```
Si
```

5. Restricciones

$1 \leq$ `n` $\leq 10^4$
$0 \leq$ `m` $\leq 6\ 10^6$

Cada cláusula puede tener entre 0 y 2 literales.

Entrega

La entrega se realizará en dos partes:

- El día miércoles de la semana 6 hasta las 23:59, Ud. colocará en la sección de documentos de su grupo en AV en el directorio nombrado Proy2, el archivo P2_GXX.tar.gz. Debe incluir todos los archivos necesarios para correr su aplicación: los viejos y los nuevos.
- El día lunes de semana 7 Ud. entregará a la hora de clase: el impreso de los archivos de su implementación, éstos deben estar debidamente documentados. **Sólo imprima los archivos nuevos.**