Programación 3 -Clases de complejidad

Instituto de Computación

Modelo para problemas de decisión

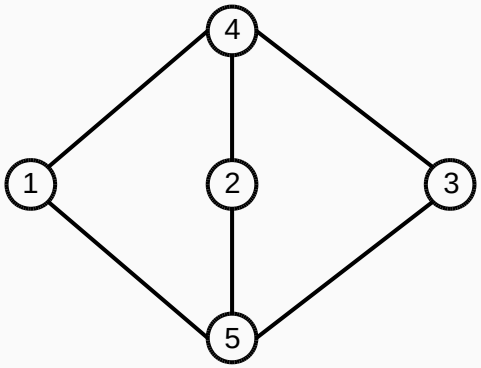
- Un problema de decisión, X, consiste en responder una pregunta binaria (con respuesta sí / no) a partir de ciertas entradas.
- Definiendo un formato de codificación de estas entradas mediante tiras de bits, un algoritmo que resuelve X es simplemente una implementación de una función binaria $f : \{0,1\}^\star \to \{0,1\}$.
- Por ejemplo, un algoritmo para *Independent Set* toma como entrada una tira binaria que representa a un grafo y a un natural k, y devuelve un valor en $\{0,1\}$ (no/sí).
- Las *instancias SÍ* del problema X son todas aquellas para las cuales la respuesta es sí. Podemos pensar X como un subconjunto de $\{0,1\}^\star$, correspondiente a las instancias SÍ.

Modelo para problemas de decisión

- Dado un formato para la codificación binaria de las entradas, un problema queda definido por el conjunto de tiras binarias que representan instancias SÍ.
- Un *algoritmo eficiente* para X cumple que:
  - Toma como entrada una cadena binaria s y devuelve un valor en $\{0, 1\}$.
  - Devuelve 1 si y solo si s representa una instancia SÍ de X ($s \in X$).
  - Computa la respuesta en no más de $p(|s|)$ pasos (es decir en tiempo $O(p(|s|))$), donde $|s|$ denota el largo de s y p es un polinomio.

Ejemplo: representación de instancias de Independent Set

Para el grafo de la figura y $k = 3$ tenemos
$s = 000110100011000110001111100111000 11$



Codificación unaria de $\ell = 3$	0001
(cant. de bits para representar n, k)	
Codificación binaria de $n = 5$ con ℓ bits	101
Fila 1 de matriz de adyacencia	00011
Fila 2 de matriz de adyacencia	00011
Fila 3 de matriz de adyacencia	00011
Fila 4 de matriz de adyacencia	11100
Fila 5 de matriz de adyacencia	11100
Codificación binaria de $k = 3$ con ℓ bits	011

Certificador eficiente

Un algoritmo B es un *certificador eficiente* para un problema de decisión X si

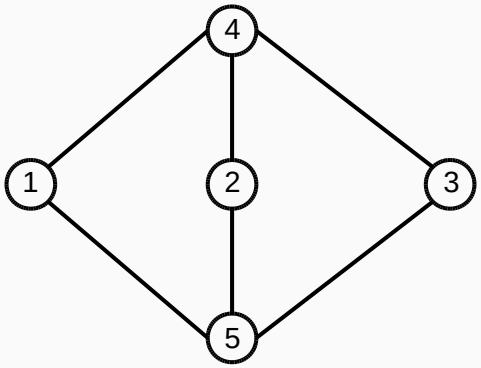
- Toma como entrada dos cadenas binarias, s, t.
- Devuelve una respuesta binaria, sí / no (valor en $\{0,1\}$).
- Termina en tiempo polinomial en $|s| + |t|$.
- Existe un polinomio p tal que para toda cadena $s \in \{0,1\}^\star$ se cumple que

$$s \in X \text{ sii } \exists t \in \{0,1\}^\star, |t| \leq p(|s|), \text{ tal que } B(s,t) = 1\,.$$

En esta definición, s representa una instancia de X y t un *certificado* que, cuando $s \in X$, "comprueba" que la respuesta al problema es afirmativa.

Ejemplo: representación de instancias de Independent Set

Para el grafo de la figura y $k = 3$ tenemos $s = 000110100011000110001111110011100011$, que es de largo 35.



Si B es un certificador eficiente para *Independent Set* basado en esta representación, entonces debe existir un polinomio p tal que

- Existe una cadena t, con $|t| \leq p(35)$, tal que $B(s, t) = 1$.
- Para $s' = 000110100011000110001111110011100100$, que corresponde a una instancia con el mismo grafo pero con $k = 4$, se tiene que cumplir que

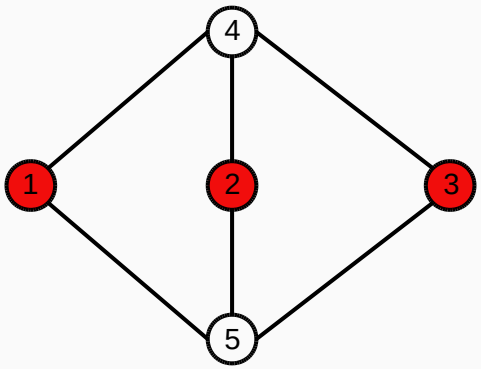
$$B(s', t') = 0, \text{ para toda cadena binaria t', } |t'| \leq p(35)\,.$$

Ejemplo: un certificado para Independent Set

Un certificado podría ser una lista L vértices que representa un conjunto $S \subseteq V$.

Para el grafo de la figura, con $V = \{1, 2, 3, 4, 5\}$, identificamos cada vértice $v \in V$ con la representación binaria del número $v - 1$:

$$
\begin{aligned}
1 &\rightarrowtail 000 \\
2 &\rightarrowtail 001 \\
&\vdots \\
5 &\rightarrowtail 100
\end{aligned}
$$



Para $S = \{1, 2, 3\}$ usaríamos $t = 000\,001\,010$

Codificación binaria del identificador de 1	000
Codificación binaria del identificador de 2	001
Codificación binaria del identificador de 3	010

Ejemplo: un certificador eficiente para Independent Set

- Entradas:
  - una instancia de *Independent Set*, dada por un grafo G y un natural k.
  - un certificado, dado por una lista L de vértices.
- Algoritmo:
  1. Si L tiene menos de k vértices distintos, devolver 0
  2. Hacer *respuesta* $= 1$
  3. Para cada arista (u, v) de G

     3.1 si $u \in L$ y $v \in L$, hacer *respuesta* $= 0$
  4. Devolver *respuesta*

Un certificador eficiente para Independent Set

Corrección (parte 1):

- El tiempo de ejecución es polinomial en $|V|$ y $|L|$ y, por lo tanto, en la suma de las representaciones de las entradas.
- Si existe un conjunto independiente S de tamaño al menos k, entonces:
  - Existe una lista L que representa a S sin elementos repetidos.
  - L tiene tamaño polinomial en $|V|$ (y por lo tanto también polinomial en $|s|$).
  - Con entradas (G, k) y L, el algoritmo devuelve 1 porque:
    - Como $|S| \geq k$, la condición del paso 1 es falsa.
    - Por definición de IS, nunca se satisface la condición de la sentencia *if* del paso 3.1.
- En conclusión, si (G, k) es una instancia SÍ, entonces existe L (el certificado) de largo polinomial en $|s|$, que hace que el algoritmo responda 1 para las entradas $(G, k), L$.

Un certificador eficiente para Independent Set

Corrección (parte 2):

- Si no existe un conjunto independiente de tamaño al menos k, entonces para toda L:
  - O bien L tiene menos de k vértices distintos de G, en cuyo caso se devuelve 0 en el paso 1,
  - o bien existe alguna arista incidente a dos vértices de L, porque de lo contrario el conjunto de vértices de L sería un conjunto independiente y tendría al menos tamaño k. Para tal arista la condición del paso 3.1 se satisface y el algoritmo devuelve 0.
- En conclusión, si (G, k) es una instancia NO, entonces no existe L que haga que el algoritmo responda 1 para las entradas $(G, k), L$.

Clases de problemas $\mathcal{P}$ y $\mathcal{NP}$

- $\mathcal{P}$ es el conjunto de problemas de decisión para los cuales existe un algoritmo eficiente.
- $\mathcal{NP}$ es el conjunto de problemas de decisión para los cuales existe un certificador eficiente.
- $\mathcal{NP}$ es un acrónimo para "nondeterministic polynomial"; NO SIGNIFICA "no polinomial".
- Claramente tenemos $\mathcal{P} \subseteq \mathcal{NP}$: un certificador simplemente puede emitir su respuesta solucionando el problema en tiempo polinomial, ignorando el certificado.
- ¿ $\mathcal{P} = \mathcal{NP}$?

Clase de problemas $\mathcal{NP}-Completos$

- Un problema de decisión X es $\mathcal{NP}-Completo$ sii
  1. $X \in \mathcal{NP}$,
  2. $Y \leq_P X$ para todo $Y \in \mathcal{NP}$.
- En palabras, la clase de problemas $\mathcal{NP}-Completos$ está formada por los problemas más difíciles de $\mathcal{NP}$.
- Sea X un problema $\mathcal{NP}-Completo$ arbitrario. Existe un algoritmo eficiente para X sii $\mathcal{P} = \mathcal{NP}$.
- Consecuencias
  - Si se encuentra un algoritmo eficiente para **algún** problema $\mathcal{NP}-Completo$, entonces podemos resolver **todos** los problemas de $\mathcal{NP}$ eficientemente.
  - Si se muestra que **algún** problema de $\mathcal{NP}$ no se puede resolver eficientemente, entonces tampoco se puede para **ninguno** de los problemas $\mathcal{NP}-Completos$.
- Si $X \in \mathcal{NP}$ y se cumple $Y \leq_P X$, donde Y es $\mathcal{NP}-Completo$, entonces X es $\mathcal{NP}-Completo$.

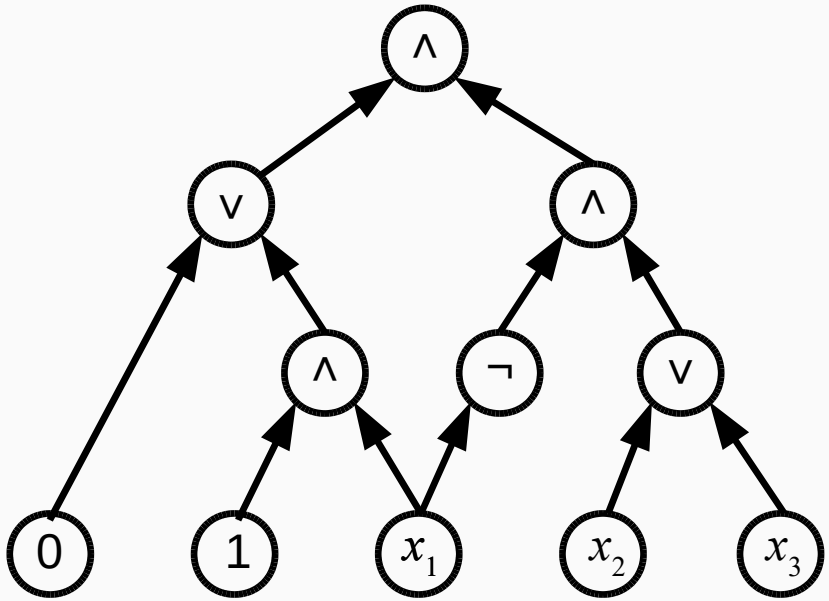
Satisfacción de circuitos (Circuit Satisfiability)

Definimos un *circuito* mediante un DAG con las siguientes características

- Los vértices sin aristas entrantes se denominan *fuentes*. Cada fuente está etiquetada con
  - el nombre de una variable de $\{x_1, x_2, \ldots, x_n\}$ (*fuente variable*),
  - o con una constante en $\{0, 1\}$ (*fuente constante*).
- El resto de los vértices puede ser de tres tipos: $\vee$, $\wedge$, o $\neg$, correspondientes a los operadores lógicos OR, AND, NOT.
  - Los vértices de tipo $\vee$, $\wedge$ tienen dos aristas entrantes.
  - Los vértices de tipo $\neg$ tienen una única arista entrante.
- Todos los vértices tiene al menos una arista saliente, salvo un único vértice, distinguido como *salida* del circuito, que no tiene ninguna.

Satisfacción de circuitos (Circuit Satisfiability)

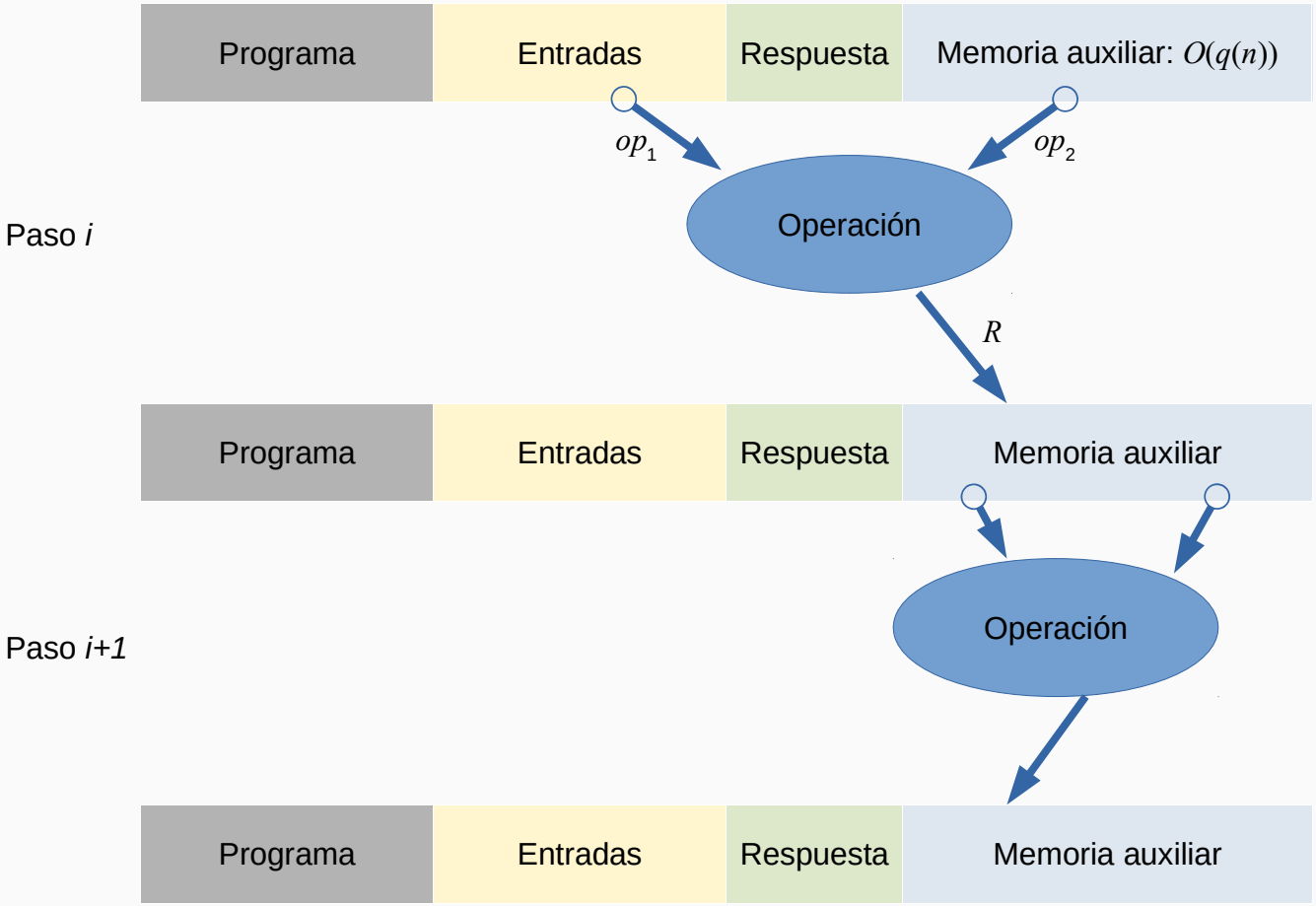
Problema de decisión: Dado un DAG que representa un circuito sobre un conjunto de variables $\{x_1, x_2, \ldots, x_n\}$, ¿existe alguna asignación de valores binarios a las variables tal que la salida del circuito vale 1?



Circuit Satisfiability es $\mathcal{NP}-Completo$

1. *Circuit Satisfiability* es $\mathcal{NP}$. Idea:
   - Usamos como certificado una cadena binaria de largo n con los valores de $x_1, x_2, \ldots, x_n$ que satisfacen el circuito.
2. Para $Y \in \mathcal{NP}$ arbitrario, $Y \leq_P$ *Circuit Satisfiability*. Idea:
   - Consideremos instancias de Y de un cierto tamaño fijo, n, y un algoritmo de certificación eficiente para Y, B.
   - Para este valor de n, la cantidad de pasos que realiza B para una instancia de tamaño n y un certificado de tamaño acotado por $p(n)$ no es mayor que $q(n)$, para cierto polinomio q.
   - Podemos computar la respuesta de B a través de un circuito que simula el sistema de cómputo subyacente.
   - Usamos *Circuit Satisfiability* para verificar si existe un certificado que hace que tal circuito valga 1.

Implementación de algoritmo eficiente con circuito



Implementación de algoritmo eficiente con circuito

- Cada paso de cómputo se puede implementar con un circuito con una cantidad polinomial de vértices.
- A su vez, la cantidad de pasos está acotada por $q(n)$.
- Por lo tanto, podemos obtener la **respuesta final** para cualquier entrada de tamaño n **con un circuito de tamaño polinomial** en n, que se obtiene encadenando $q(n)$ copias del mismo circuito.
- A partir de un programa que implementa B, el circuito para entradas de tamaño n se puede construir en tiempo polinomial en n.

Circuit Satisfiability es $\mathcal{NP}-Completo$

Vemos que $X \leq_P$ *Circuit Satisfiability* para todo $X \in \mathcal{NP}$.

- Sea $X \in \mathcal{NP}$ arbitrario y sea B un algoritmo de certificación eficiente para X.
- Dada una instancia para X representada por una cadena s, construimos un circuito que implementa B, para entradas de tamaño $|s| + p(|s|)$.
- En ese circuito, definimos como fuentes constantes las entradas del circuito correspondientes a s y dejamos como fuentes variables las correspondientes al certificado t.
- Emitimos como respuesta para X la respuesta al problema *Circuit Satisfiability* para la instancia que acabamos de construir.

SAT es $\mathcal{NP}-Completo$

- $SAT \in \mathcal{NP}$: una cadena de n bits define una asignación de valores de verdad para las variables $\{x_1, x_2, \ldots, , x_n\}$ (un certificado), y podemos verificar que cada una de las cláusulas se satisface, utilizando, para cada una, tiempo lineal en el largo de la cláusula.
- Además podemos hacer una reducción de tiempo polinomial de *Circuit Satisfiability* a *SAT*.
- Como *Circuit Satisfiability* es $\mathcal{NP}-Completo$, los dos puntos anteriores implican que *SAT* es $\mathcal{NP}-Completo$

Reducción de tiempo polinomial de Circuit Satisfiability a SAT

- Sea $\{v_1, v_2, \ldots, , v_n\}$ el conjunto de vértices de un circuito en una instancia de *Circuit Satisfiability*
- Construimos una instancia de *SAT* con variables $\{z_1, z_2, \ldots, , z_n\}$
- El conjunto de cláusulas lo construimos partiendo de un conjunto vacío y agregando, para cada vértice v_i que no es fuente variable, cláusulas que dependen del tipo de vértice.
- Sean $\{v_{i_1}, v_{i_2}, \ldots , v_{i_m}\}$ los vértices de tipo fuente variable asociados a las variables $\{x_1, x_2, \ldots , x_m\}$ del circuito.
- Las cláusulas que construimos están diseñadas para que si asignamos a $\{x_1, x_2, \ldots , x_m\}$ los mismos valores que a $\{z_{i_1}, z_{i_2}, \ldots , z_{i_m}\}$, la satisfacción de esas cláusulas implica que a cada variable z_i se asigna el valor que calcula el vértice v_i en el circuito.

Reducción de tiempo polinomial de Circuit Satisfiability a SAT

- Si v_i es una fuente constante igual a 0, agregamos una cláusula de un solo término,

$$\bar{z}_i$$

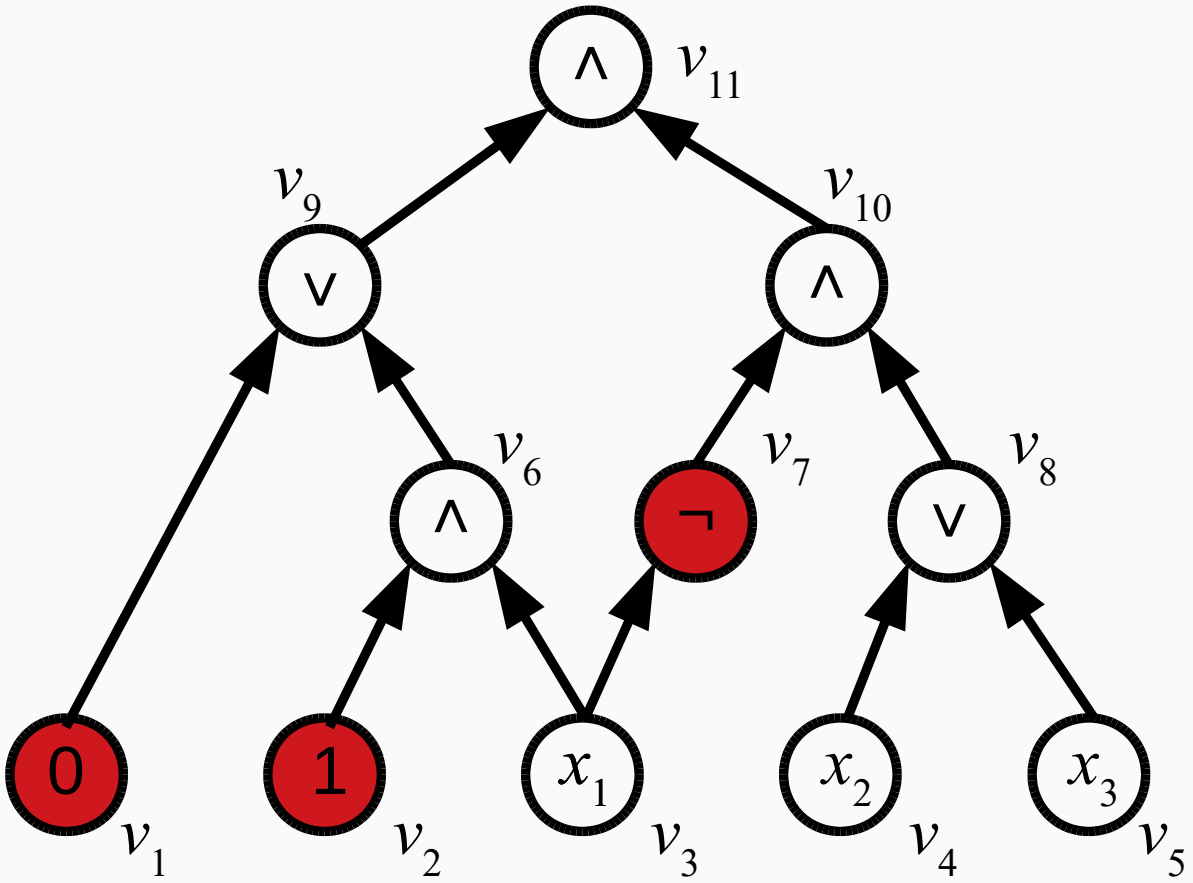
- Si v_i es una fuente constante igual a 1, agregamos una cláusula de un solo término,

$$z_i$$

- Si v_i es de tipo $\neg$, sea v_j el (único) vértice de donde proviene su entrada. Agregamos las cláusulas

$$z_i \vee z_j$$
$$\bar{z}_i \vee \bar{z}_j$$

Reducción de tiempo polinomial de Circuit Satisfiability a SAT



Cláusulas:

$\bar{z}_1$

z_2

$z_7 \vee z_3$

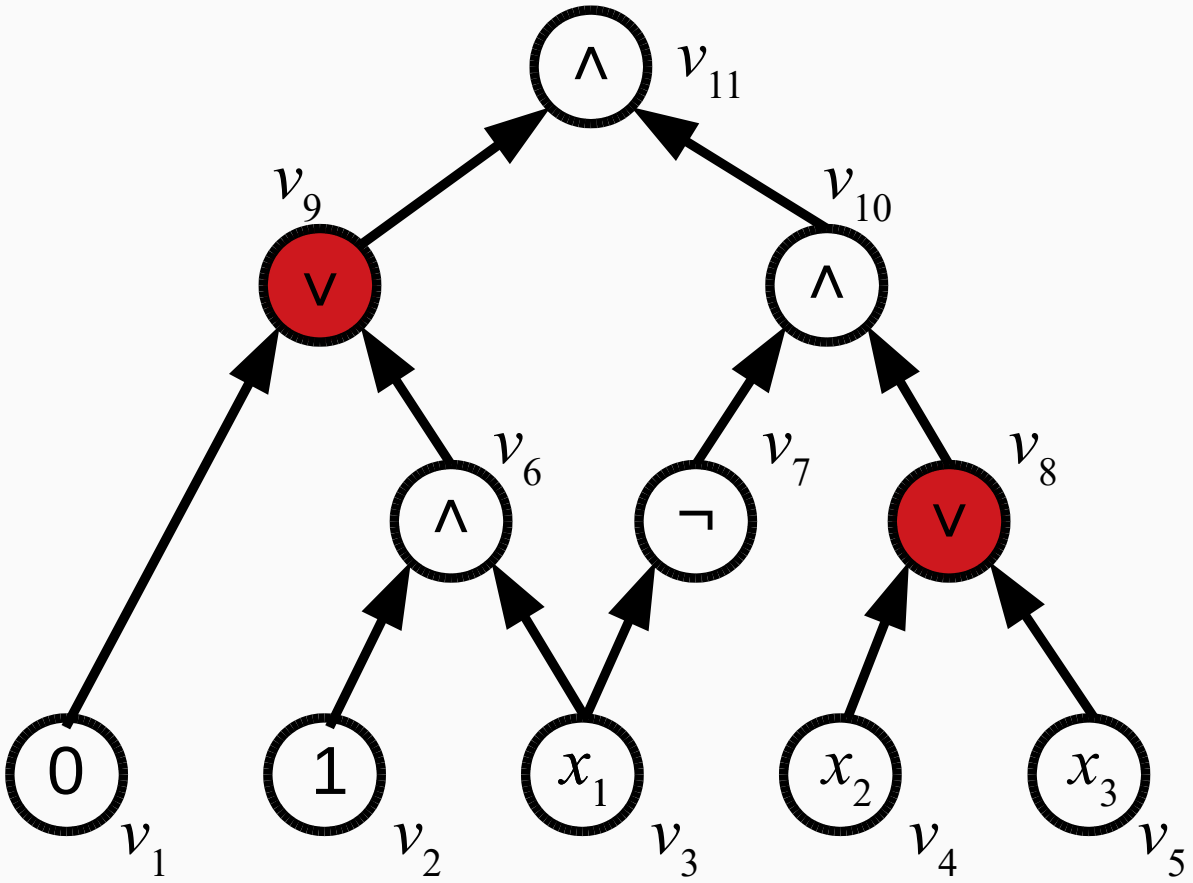
$\bar{z}_7 \vee \bar{z}_3$

Reducción de tiempo polinomial de Circuit Satisfiability a SAT

- Si v_i es de tipo $\vee$, sean v_j y v_k los dos vértice de donde provienen sus entrada. Agregamos las cláusulas

$$z_j \vee z_k \vee \bar{z}_i$$
$$\bar{z}_j \vee z_i$$
$$\bar{z}_k \vee z_i$$

Reducción de tiempo polinomial de Circuit Satisfiability a SAT



Cláusulas:

$z_1 \vee z_6 \vee \bar{z}_9$

$\bar{z}_1 \vee z_9$

$\bar{z}_6 \vee z_9$

$z_4 \vee z_5 \vee \bar{z}_8$

$\bar{z}_4 \vee z_8$

$\bar{z}_5 \vee z_8$

Reducción de tiempo polinomial de Circuit Satisfiability a SAT

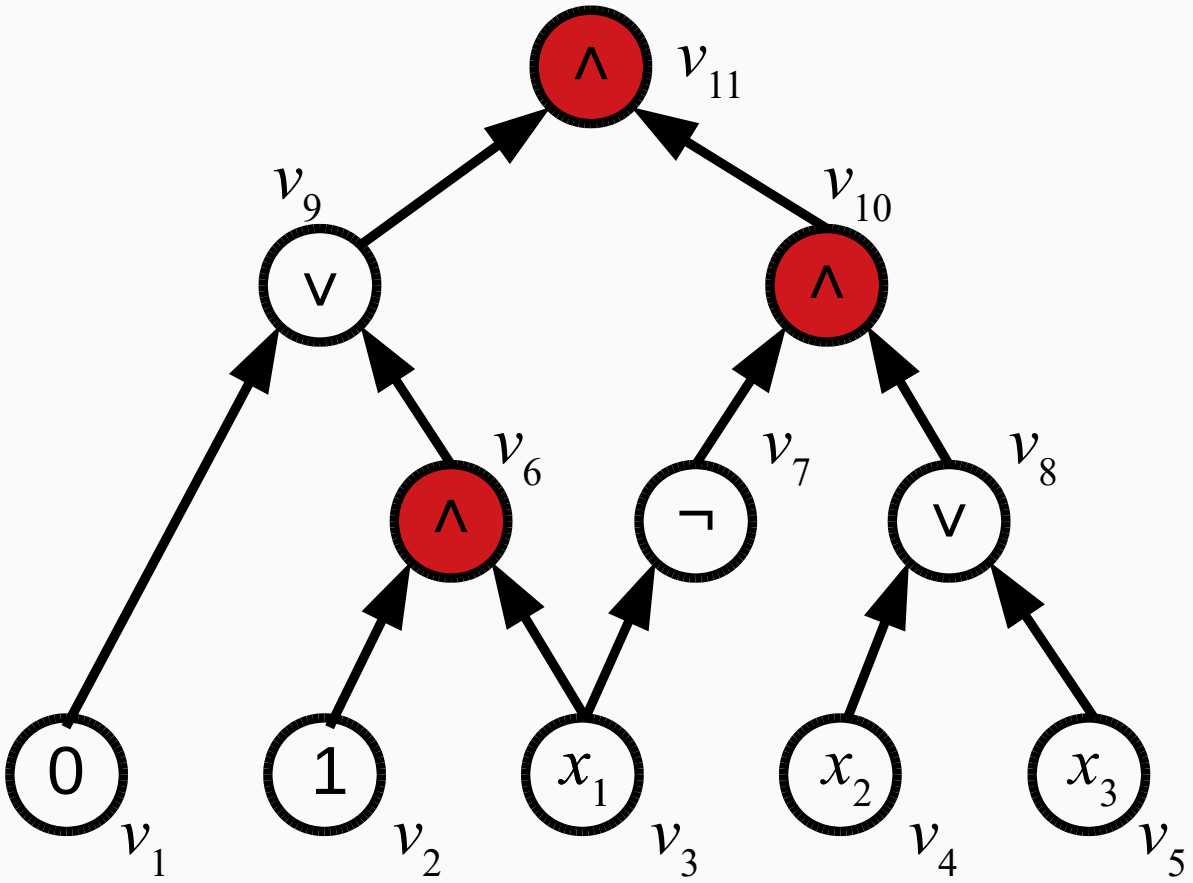
- Si v_i es de tipo $\wedge$, sean v_j y v_k los dos vértice de donde provienen sus entrada. Agregamos las cláusulas

$$\bar{z}_j \vee \bar{z}_k \vee z_i$$
$$z_j \vee \bar{z}_i$$
$$z_k \vee \bar{z}_i$$

- Adicionalmente, si v_i es el vértice final (del tipo que sea), agregamos la cláusula

$$z_i$$

Reducción de tiempo polinomial de Circuit Satisfiability a SAT



Cláusulas:

$\bar{z_2} \vee \bar{z_3} \vee z_6$

$z_2 \vee \bar{z_6}$

$z_3 \vee \bar{z_6}$

$\bar{z_7} \vee \bar{z_8} \vee z_{10}$

$z_7 \vee \bar{z_{10}}$

$z_8 \vee \bar{z_{10}}$

$\bar{z_9} \vee \bar{z_{10}} \vee z_{11}$

$z_9 \vee \bar{z_{11}}$

$z_{10} \vee \bar{z_{11}}$

z_{11}

Corrección de la reducción

- La instancia de *SAT* se construye en tiempo polinomial en el tamaño del circuito.
- Sean $\{v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$ los vértices de tipo fuente variable asociados a $\{x_1, x_2, \ldots, x_m\}$ en una instancia de *Circuit Satisfiability*.
- Supongamos que existe una asignación de verdad que satisface las cláusulas de la instancia *SAT*. Entonces, asignando a $\{x_1, x_2, \ldots, x_m\}$ los mismos valores que a $\{z_{i_1}, z_{i_2}, \ldots, z_{i_m}\}$, el circuito evalúa al mismo valor que la variable z_f, donde v_f es el vértice final. Como existe una cláusula que solo contiene el término v_f, este valor es necesariamente 1.
- Si existe una asignación de verdad que satisface el circuito, entonces, asignando a cada variable z_i el valor que calcula el vértice v_i se satisfacen todas las cláusulas de la instancia *SAT*.

Consecuencias de que SAT es $\mathcal{NP}-Completo$

- SAT es $\mathcal{NP}-Completo \implies 3SAT$ es $\mathcal{NP}-Completo$
- $3SAT$ es $\mathcal{NP}-Completo \implies$ *Independent Set* es $\mathcal{NP}-Completo$
- *Independent Set* es $\mathcal{NP}-Completo \implies$ *Vertex Cover* es $\mathcal{NP}-Completo$
- *Vertex Cover* es $\mathcal{NP}-Completo \implies$ *Set Cover* es $\mathcal{NP}-Completo$
- *Independent Set* es $\mathcal{NP}-Completo \implies$ *Set Packing* es $\mathcal{NP}-Completo$

Resumen

Resumen

- Problemas de decisión: un conjunto de cadenas binarias que representan las instancias SÍ.

Resumen

- Problemas de decisión: un conjunto de cadenas binarias que representan las instancias SÍ.
- Certificado: cadena binaria, de largo polinomial, que "muestra" que una instancia es SÍ.

Resumen

- Problemas de decisión: un conjunto de cadenas binarias que representan las instancias SÍ.
- Certificado: cadena binaria, de largo polinomial, que "muestra" que una instancia es SÍ.
- Certificador eficiente: un algoritmo para "validación de certificados" en tiempo polinomial.

Resumen

- Problemas de decisión: un conjunto de cadenas binarias que representan las instancias SÍ.
- Certificado: cadena binaria, de largo polinomial, que "muestra" que una instancia es SÍ.
- Certificador eficiente: un algoritmo para "validación de certificados" en tiempo polinomial.
- Problemas de la clase $\mathcal{NP}$: se puede verificar un instancia SÍ en tiempo polinomial si disponemos de un certificado para dicha instancia.

Resumen

- Problemas de decisión: un conjunto de cadenas binarias que representan las instancias SÍ.
- Certificado: cadena binaria, de largo polinomial, que "muestra" que una instancia es SÍ.
- Certificador eficiente: un algoritmo para "validación de certificados" en tiempo polinomial.
- Problemas de la clase $\mathcal{NP}$: se puede verificar un instancia SÍ en tiempo polinomial si disponemos de un certificado para dicha instancia.
- Problemas de la clase $\mathcal{NP}-Completo$: los más "difíciles" de $\mathcal{NP}$.

Resumen

- Problemas de decisión: un conjunto de cadenas binarias que representan las instancias SÍ.
- Certificado: cadena binaria, de largo polinomial, que "muestra" que una instancia es SÍ.
- Certificador eficiente: un algoritmo para "validación de certificados" en tiempo polinomial.
- Problemas de la clase $\mathcal{NP}$: se puede verificar un instancia SÍ en tiempo polinomial si disponemos de un certificado para dicha instancia.
- Problemas de la clase $\mathcal{NP}-Completo$: los más "difíciles" de $\mathcal{NP}$.
- Los problemas que se pueden **verificar** en tiempo polinomial, ¿también se pueden **resolver** en tiempo polinomial?

Resumen

- Problemas de decisión: un conjunto de cadenas binarias que representan las instancias SÍ.
- Certificado: cadena binaria, de largo polinomial, que "muestra" que una instancia es SÍ.
- Certificador eficiente: un algoritmo para "validación de certificados" en tiempo polinomial.
- Problemas de la clase $\mathcal{NP}$: se puede verificar un instancia SÍ en tiempo polinomial si disponemos de un certificado para dicha instancia.
- Problemas de la clase $\mathcal{NP}-Completo$: los más "difíciles" de $\mathcal{NP}$.
- Los problemas que se pueden **verificar** en tiempo polinomial, ¿también se pueden **resolver** en tiempo polinomial?
- Buena pregunta.... no lo sabemos :(