

Programación 3 -Clases de complejidad

Instituto de Computación

Modelo para problemas de decisión

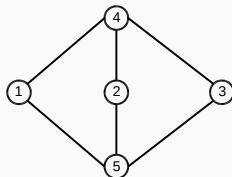
- Un problema de decisión, X , consiste en responder una pregunta binaria (con respuesta sí / no) a partir de ciertas entradas.
- Definiendo un formato de codificación de estas entradas mediante tiras de bits, un algoritmo que resuelve X es simplemente una implementación de una función binaria $f : \{0, 1\}^* \rightarrow \{0, 1\}$.
- 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 $\{0, 1\}^*$, 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
 $s = 00011010001100011000111110011100011$



Codificación unaria de $\ell = 3$ (cant. de bits para representar n, k)	0001
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

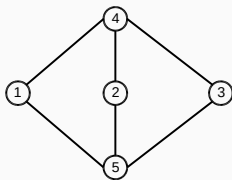
- Toma como entrada dos cadenas binarias, s , t .
- Devuelve una respuesta binaria, sí / no (valor en $\{0, 1\}$).
- Termina en tiempo polinomial en $|s| + |t|$.
- Existe un polinomio p tal que para toda cadena $s \in \{0, 1\}^*$ se cumple que

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

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 $s = 00011010001100011000111110011100011$, 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 $s' = 00011010001100011000111110011100100$, que corresponde a una instancia con el mismo grafo pero con $k = 4$, se tiene que cumplir que

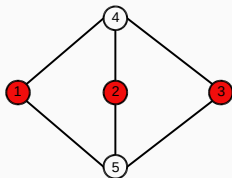
$B(s', t') = 0$, para **toda** cadena binaria t' , $|t'| \leq p(35)$.

Ejemplo: un certificado para Independent Set

Un certificado podría ser una lista L de 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 $v - 1$:

1 \mapsto 000
2 \mapsto 001
⋮
5 \mapsto 100



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.

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 a condición del paso 3.1 se satisface y el algoritmo devuelve 0.

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} -Completo

- Un problema de decisión X es \mathcal{NP} -Completo si
 - 1. $X \in \mathcal{NP}$,
 - 2. $Y \leq_P X$ para todo $Y \in \mathcal{NP}$.
- En palabras, la clase de problemas \mathcal{NP} -Completo está formada por los problemas más difíciles de \mathcal{NP} .
- Sea X un problema \mathcal{NP} -Completo arbitrario. Existe un algoritmo eficiente para X si $\mathcal{P} = \mathcal{NP}$.
- 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} -Completo.
- Si $X \in \mathcal{NP}$ y se cumple $Y \leq_P X$, donde Y es \mathcal{NP} -Completo, entonces X es \mathcal{NP} -Completo.

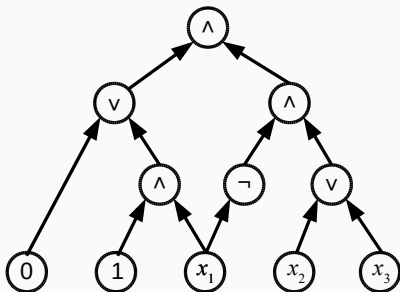
Satisfacción de circuitos (Circuit Satisfiability)

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 $\{x_1, x_2, \dots, x_n\}$ (*fuelle variable*),
 - o con una constante en $\{0, 1\}$ (*fuelle 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 $\{x_1, x_2, \dots, x_n\}$, ¿existe alguna asignación de valores binarios a las variables tal que la salida del circuito vale 1?



Circuit Satisfiability es \mathcal{NP} -Completo

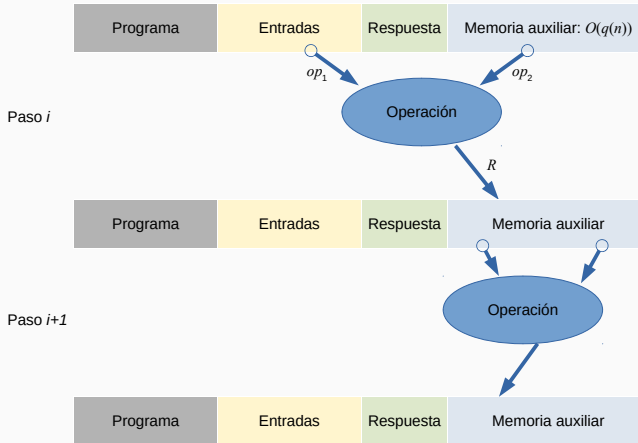
1. *Circuit Satisfiability* es \mathcal{NP} . Idea:

- Usamos como certificado una cadena binaria de largo n con los valores de x_1, x_2, \dots, 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 $\{x_1, x_2, \dots, x_n\}$ (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

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

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

- 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.

- 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 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 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} .

- 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?

- 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 :(