Tema 7. Complexitat

Estructures de Dades i Algorismes

FIB

Antoni Lozano

Q2 2018–2019

Versió de 6 de maig de 2019

Classes

L'anàlisi d'algorismes estudia la quantitat de recursos que necessita un algorisme per resoldre un problema.

La teoria de la complexitat considera els algorismes possibles que resolen un mateix problema.

- Mentre l'anàlisi d'algorismes se centra en els algorismes, la teoria de la complexitat s'interessa pels problemes
- Veurem els conceptes bàsics per classificar els problemes segons la seva complexitat

Classes

L'anàlisi d'algorismes estudia la quantitat de recursos que necessita un algorisme per resoldre un problema.

La teoria de la complexitat considera els algorismes possibles que resolen un mateix problema.

- Mentre l'anàlisi d'algorismes se centra en els algorismes, la teoria de la complexitat s'interessa pels problemes
- Veurem els conceptes bàsics per classificar els problemes segons la seva complexitat

Classes

L'anàlisi d'algorismes estudia la quantitat de recursos que necessita un algorisme per resoldre un problema.

La teoria de la complexitat considera els algorismes possibles que resolen un mateix problema.

- Mentre l'anàlisi d'algorismes se centra en els algorismes, la teoria de la complexitat s'interessa pels problemes
- Veurem els conceptes bàsics per classificar els problemes segons la seva complexitat

Problemes decisionals

Per poder classificar millor els problemes, considerarem les seves versions decisionals.

Definició

Un problema decisional és un problema en el qual s'ha de determinar si l'entrada satisfà una certa propietat.

Problemes decisionals

Molts problemes vistos fins ara són decisionals o es poden transformar en decisionals.

Alguns problemes decisionals sobre grafs:

- **connectivitat**: donat un graf, decidir si és connex
- **accessibilitat**: donat un graf $G = (V, E)$ i dos vèrtexs $i, j \in V$, decidir si hi ha un camí a G entre i i j
- **camí curt**: donat un graf $G = (V, E)$, dos vèrtexs $i, j \in V$ i un natural k, decidir si hi ha un camí a G entre i i j de llargària màxima k
- **camí llarg**: donat un graf $G = (V, E)$, dos vèrtexs $i, j \in V$ i un natural k, decidir si hi ha un camí a G entre i i j de llargària mínima k
- **3-colorabilitat**: donat un graf, decidir si és 3-colorable

Problemes decisionals

Certs problemes no tenen gaire sentit en versió decisional.

Problema de les n-reines decisional (1a versió)

Donat un natural n, decidir si es poden col·locar n reines en un tauler $n \times n$ sense que cap amenaci cap altra.

Se sap que hi ha solucions per a tot $n \neq 2, 3$. Per tant, l'algorisme següent decideix el problema en temps $\Theta(1)$.

```
REINES(n)
    si n = 2 o n = 3 llavors
        retornar 0
    si no
        retornar 1
```

Problemes decisionals

El que ens interessa és trobar una solució, no saber si existeix.

Problema de les n-reines decisional (2a versió)

Donat un natural n i k valors $r_1, \ldots r_k$, amb $k \leq n$, decidir si es poden col·locar n reines en un tauler $n \times n$ sense que cap amenaci cap altra i de manera que per a tot i tal que $1 \leq i \leq k$, la reina de la fila i ocupi la columna r_i.

Aquesta versió, tot i ser decisional, permet trobar una solució amb

$$(n-1) + (n-2) + \cdots + 2 = \sum_{i=2}^{n-1} i = \frac{n(n-1)}{2} - 1 \in \Theta(n^2)$$

execucions de l'algorisme que el resol.

Problemes decisionals

Altres problemes decisionals:

1. **primalitat**: donat un natural, decidir si és primer
2. **viatjant**: donades n ciutats, les distàncies entre elles i un nombre de kilòmetres k, decidir si hi ha un recorregut d'un màxim de k kilòmetres que passi per totes i torni a l'origen

Un problema decisional és una col·lecció d'infinites entrades.

Si un problema contingués un nombre finit d'entrades, es podria resoldre amb un algorisme de cost constant (p. ex., 8-reines).

Problemes decisionals

Un problema decisional es representa formalment mitjançant un conjunt.

Si T és una propietat que es pot comprovar en els elements d'un conjunt d'entrades E, podem plantejar el problema decisional:

Problema A
Donat $x \in E$, determinar si es compleix $T(x)$.

Formalment, podem descriure A com el conjunt:

$$A = \{ \ x \in E \mid T(x) \ \} \ .$$

Problemes decisionals

Les entrades dels problemes pertanyen a certs dominis de dades com ara:

- nombres naturals
- tuples de naturals
- grafs
- dags amb pesos en els arcs
- fórmules booleanes

En cada cas, considerarem una funció de mida.

Funció de mida

Donat un $x \in E$, on E és un domini de dades, la mida (o talla) de x, representada amb $|x|$, és el nombre de símbols d'una codificació estàndard de x.

Problemes decisionals

Donat un problema A definit sobre un conjunt d'entrades E, distingirem entre

- les entrades positives: les que pertanyen a A
- les entrades negatives: les que pertanyen a $E - A$

Primalitat

El problema de la primalitat el podem descriure informalment:

> **Primalitat**
> Donat un natural x, determinar si x és primer.

O bé formalment com el conjunt de les entrades positives:

$$P = \{x \in \mathbb{N} \mid x \text{ és primer } \}.$$

Una funció de mida per als naturals és la que compta el nombre de dígits de la representació binària:

$$|x| = \text{ nombre de dígits de } x \text{ en binari} = \lfloor \log_2 x \rfloor + 1.$$

Problemes decisionals

Donat un problema A definit sobre un conjunt d'entrades E, distingirem entre

- les entrades positives: les que pertanyen a A
- les entrades negatives: les que pertanyen a $E - A$

Primalitat

El problema de la primalitat el podem descriure informalment:

Primalitat
Donat un natural x, determinar si x és primer.

O bé formalment com el conjunt de les entrades positives:

$$P = \{x \in \mathbb{N} \mid x \text{ és primer } \}.$$

Una funció de mida per als naturals és la que compta el nombre de dígits de la representació binària:

$$|x| = \text{ nombre de dígits de } x \text{ en binari} = \lfloor \log_2 x \rfloor + 1.$$

Problemes decisionals

Un cop podem descriure els problemes com a objectes matemàtics (conjunts, si són problemes decisionals), els podem agrupar en classes en funció de la seva complexitat.

- Considerarem classes de problemes que es poden resoldre amb certs recursos
- Una classe agrupa problemes de la mateixa manera que un problema agrupa entrades
- Cal distingir entre tres nivells d'abstracció:
  - Entrades (cadenes de caràcters)
  - Problemes (conjunts d'entrades)
  - Classes (conjunts d'problemes)

Temps polinòmic i exponencial

Suposem que $t : \mathbb{N} \to \mathbb{R}^+$ és una funció.

Algorismes de cost t

Diem que un algorisme $\mathcal{A}$ té cost t si el seu cost en cas pitjor pertany a $O(t)$.

Problemes decidibles en temps t

Si un algorisme $\mathcal{A}$ rep entrades d'un conjunt E i té una sortida binària, escriurem:

$$\mathcal{A} : E \to \{0, 1\}.$$

Diem que un problema decisional A és decidible en temps t si existeix un algorisme $\mathcal{A} : E \to \{0, 1\}$ de cost t tal que, per a tot $x \in E$:

$$x \in A \Rightarrow \mathcal{A}(x) = 1$$

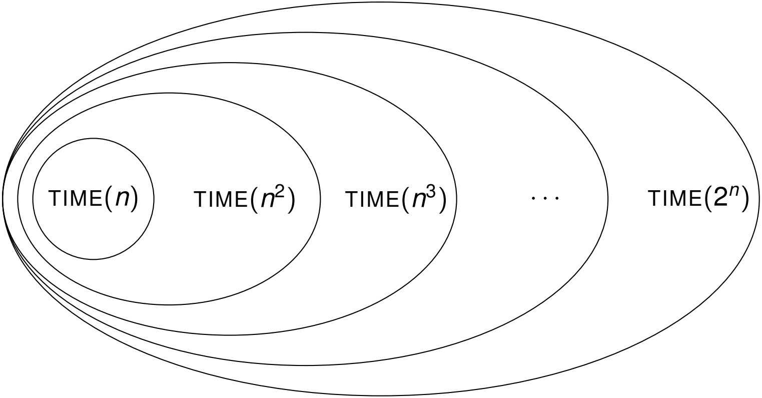
$$x \notin A \Rightarrow \mathcal{A}(x) = 0$$

Temps polinòmic i exponencial

Classe TIME(t)

Per a una funció $t : \mathbb{N} \to \mathbb{R}^+$, agrupem els problemes decidibles en temps t:

$$\mathrm{TIME}(t) = \{A \mid A \text{ és decidible en temps } t\}.$$



Temps polinòmic i exponencial

Recordem que hi ha una gran diferència entre tenir un algorisme polinòmic o un d'exponencial per a un problema.

En el tema 1 hem vist dues taules que assenyalen:

- diferències quantitatives (taula 1)
- diferències qualitatives (taula 2)

entre polinomis i exponencials.

Temps polinòmic i exponencial

Taula 1 (Garey/Johnson, *Computers and Intractability*)

Comparació de funcions polinòmiques i exponencials.

cost	10	20	30	40	50
n	0.00001 s	0.00002 s	0.00003 s	0.00004 s	0.00005 s
n^2	0.0001 s	0.0004 s	0.0009 s	0.0016 s	0.0025 s
n^3	0.001 s	0.008 s	0.027 s	0.064 s	0.125 s
n^5	0.1 s	3.2 s	24.3 s	1.7 min	5.2 min
2^n	0.001 s	1.0 s	17.9 min	12.7 dies	35.7 anys
3^n	0.059 s	58 min	6.5 anys	3855 segles	2×10^8 segles

Temps polinòmic i exponencial

Taula 2 (Garey/Johnson, *Computers and Intractability*)

Efecte de les millores en la tecnologia sobre algorismes polinòmics i exponencials.

cost	tecnologia actual	tecnologia $\times 100$	tecnologia $\times 1000$
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.64N_3$	$10N_3$
n^5	N_4	$2.5N_4$	$3.98N_4$
2^n	N_4	$N_4 + 6.64$	$N_4 + 9.97$
3^n	N_5	$N_5 + 4.19$	$N_5 + 6.29$

Temps polinòmic i exponencial

Classe P

Definim la classe P com la reunió de les classes de temps polinòmiques:

$$\mathrm{P} = \bigcup_{k>0} \mathsf{TIME}(n^k).$$

És a dir, un problema pertany a P si és decidible en temps n^k per a algun k.

Classe EXP

Definim la classe EXP com la reunió de les classes de temps exponencials:

$$\mathrm{EXP} = \bigcup_{k>0} \mathsf{TIME}(2^{n^k}).$$

És a dir, un problema és a EXP si és decidible en temps 2^{n^k} per a algun k.

Temps polinòmic i exponencial

Exemples

Problemes a P:

- **connectivitat**
- **accessibilitat**
- **primalitat**
- **camí curt**
- **2-colorabilitat**

Problemes a EXP (no se sap si a P):

- **camí llarg**
- **3-colorabilitat**
- **viatjant**

Altres problemes a EXP:

- **dames, escacs i go generalitzats**

Temps polinòmic i exponencial

Teorema

$\mathrm{P} \subsetneq \mathrm{EXP}$.

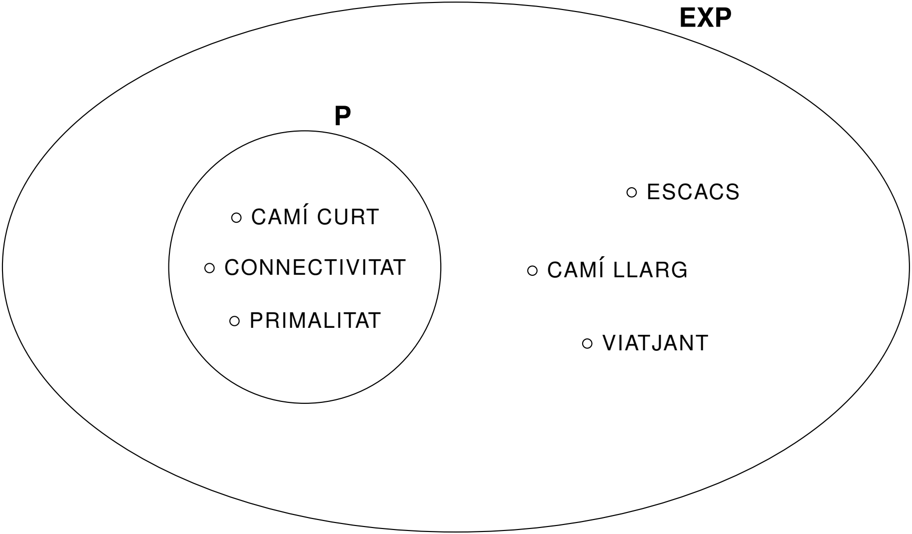
La inclusió estricta del teorema es pot dividir en dues parts:

1. $\mathrm{P} \subseteq \mathrm{EXP}$. Evident a partir de les definicions:

$$\mathrm{P} = \bigcup_{k>0} \mathsf{TIME}(n^k) \subseteq \bigcup_{k>0} \mathsf{TIME}(2^{n^k}) = \mathrm{EXP}$$

2. $\mathrm{P} \neq \mathrm{EXP}$. Es demostra per la tècnica de diagonalització

Temps polinòmic i exponencial



Indeterminisme

- Els algorismes vistos fins ara són deterministes: segueixen un únic camí de càlcul des de l'entrada fins al resultat
- L'execució d'un algorisme $\mathcal{A} : E \to \{0, 1\}$ per a un conjunt de dades E es pot veure com un camí:

$$\begin{array}{c}\boxed{\text{entrada } x \in E}\\ \Big\downarrow \\ \boxed{\text{sortida 0 o 1}}\end{array}$$

Indeterminisme

Un algorisme indeterminista pot arribar a un resultat a través de diferents camins. El seu funcionament s'assembla més a un arbre.

Algorismes indeterministes (idea informal)

Un algorisme $\mathcal{A} : E \rightarrow \{0, 1\}$ és *indeterminista* si pot fer ús d'una nova funció

$$\text{TRIAR}(y)$$

que, per a una entrada x, retorna un nombre entre 1 i y, amb $y \leq x$. El càlcul de $\mathcal{A}$ amb entrada x comença de manera determinista fins la primera instrucció TRIAR i es pot veure com un arbre:

- **arbre de computació**: cada crida a TRIAR(y) genera y branques de càlcul on, en la branca i-èsima, la crida retorna el valor i
- **valor retornat**: es diu que $\mathcal{A}$ retorna 1 si alguna fulla de l'arbre de computació retorna 1; altrament, $\mathcal{A}$ retorna 0
- **cost**: el cost de $\mathcal{A}$ és el de la branca de més cost de l'arbre de computació

Indeterminisme

Un algorisme indeterminista pot arribar a un resultat a través de diferents camins. El seu funcionament s'assembla més a un arbre.

Algorismes indeterministes (idea informal)

Un algorisme $\mathcal{A} : E \to \{0, 1\}$ és *indeterminista* si pot fer ús d'una nova funció

$$\text{TRIAR}(y)$$

que, per a una entrada x, retorna un nombre entre 1 i y, amb $y \leq x$. El càlcul de $\mathcal{A}$ amb entrada x comença de manera determinista fins la primera instrucció TRIAR i es pot veure com un arbre:

- **arbre de computació**: cada crida a $\text{TRIAR}(y)$ genera y branques de càlcul on, en la branca i-èsima, la crida retorna el valor i
- **valor retornat**: es diu que $\mathcal{A}$ retorna 1 si alguna fulla de l'arbre de computació retorna 1; altrament, $\mathcal{A}$ retorna 0
- **cost**: el cost de $\mathcal{A}$ és el de la branca de més cost de l'arbre de computació

Exemple: compostos

El problema

$$\text{COMPOSTOS} = \{x \mid \exists y \quad 1 < y < x \ \text{ i } y \text{ divideix } x\ \}$$

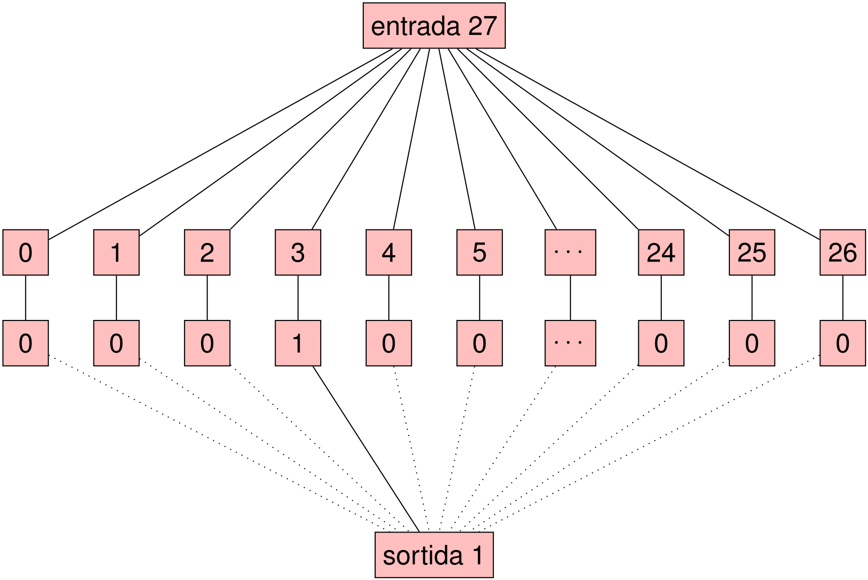
té un algorisme determinista trivial de temps exponencial

entrada x
per a $y = 2$ **fins** $x - 1$
  si y divideix x **llavors**
    retornar 1
retornar 0

i un algorisme indeterminista de temps polinòmic

entrada x
$y \leftarrow \text{TRIAR}(x - 1)$
si $y > 1$ **i** y divideix x **llavors**
  retornar 1
retornar 0

Indeterminisme



Indeterminisme

- En l'exemple anterior, diem que el 3 és un testimoni del fet que el nombre 27 és compost.
- És a dir, en el problema COMPOSTOS existeixen:
  - Possibles testimonis ($y < x$) del fet que un nombre x és compost
  - Un algorisme verificador dels testimonis de temps polinòmic que, donat un y, comprova si y divideix x

Indeterminisme

A diferència de COMPOSTOS, el problema ESCACS GENERALITZAT no té testimonis curts que permetin comprovar que un jugador té una estratègia guanyadora.

Però hi ha molts problemes per als quals és fàcil trobar testimonis curts. Per a tots ells, hi ha algorismes indeterministes de temps polinòmic.

Exemple: 3-colorabilitat

El problema de la 3-colorabilitat, representat pel conjunt

$$3\text{-COLOR} = \{\ G \mid G \text{ és 3-colorable }\}$$

també té un algorisme exhaustiu de temps exponencial

entrada $G = (V, E)$
$n \leftarrow |V|$
per a cada tupla $(c_1, \ldots, c_n)$ on $\forall i \leq n\ \ c_i \in \{1, 2, 3\}$
  si $(c_1, \ldots, c_n)$ és una 3-coloració de G **llavors**
    retornar 1
retornar 0

Exemple: 3-colorabilitat

i un algorisme indeterminista de temps polinòmic

entrada $G = (V, E)$
$n \leftarrow |V|$
per a $i = 1$ fins n
  $c_i \leftarrow \textsc{Triar}(3)$
si $(c_1, \ldots, c_n)$ és una 3-coloració de G **llavors**
  retornar 1
si no
  retornar 0

Indeterminisme

La definició formal dels algorismes polinòmics indeterministes separa:

- el càlcul del testimoni
- el càlcul determinista

Indeterminisme

Decidibilitat en temps polinòmic indeterminista

Sigui Σ un alfabet i A un problema decisional definit sobre un conjunt d'entrades E. Diem que A és decidible en temps polinòmic indeterminista si existeix

- un algorisme polinòmic $\mathcal{V} : E \times \Sigma^* \to \{0, 1\}$ (anomenat verificador) i
- un polinomi $p(n)$

tals que per a tot $x \in E$, tenim

$$x \in A \Rightarrow \mathcal{V}(x, y) = 1 \text{ per a algun } y \in \Sigma^* \text{ tal que } |y| = p(|x|)$$

$$x \notin A \Rightarrow \mathcal{V}(x, y) = 0 \text{ per a tot } y \in \Sigma^* \text{ tal que } |y| = p(|x|)$$

Si $x \in A$, els y tals que $\mathcal{V}(x, y) = 1$ se'n diuen testimonis o certificats.

Indeterminisme

Per veure que un problema A és decidible en temps polinòmic indeterminista caldrà comprovar que:

1. les entrades positives de A tenen testimonis de mida polinòmica
   (cal identificar els testimonis i la seva mida)
2. els testimonis es poden verificar en temps polinòmic
   (cal trobar un verificador)

Indeterminisme

Compostos

Considerem el problema

$$\text{COMPOSTOS} = \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad 1 < y < x \ \text{ i } y \text{ divideix } x\}$$

1. Els testimonis per a x són tots els $y \neq 1, x$ que divideixen x
2. El polinomi és $p(n) = n$
3. El verificador és

$\mathcal{V}(x, y)$
  si $(1 < y < x)$ **i** (y divideix x) **llavors**
    retornar 1
  si no
    retornar 0

COMPOSTOS és decidible en temps polinòmic indeterminista perquè

$$x \in \text{COMPOSTOS} \Leftrightarrow \mathcal{V}(x, y) = 1 \text{ per a algun } y \text{ t.q. } |y| = p(|x|)$$

3-colorabilitat

Considerem el problema

$$3\text{-COLOR} = \{\ G \mid G \text{ és 3-colorable}\ \}$$

1. Els testimonis per a $G = (V, E)$ són totes les 3-coloracions C de G de la forma $C = (c_1, c_2, \ldots, c_n)$, on $n = |V|$ i $c_i \in \{1, 2, 3\}$ per a tot $i \ \leq n$
2. El polinomi (amb representacions raonables de G i C) pot ser $p(n) = n$
3. El verificador és

   $\mathcal{V}(G, C)$
   si C és una 3-coloració de G **llavors**
   retornar 1
   si no
   retornar 0

Indeterminisme

Tots els problemes decidibles en temps polinòmic indeterminista els agrupem en una classe.

Classe NP

Definim la classe NP (de *nondeterministic polynomial time*) com:

$$\mathrm{NP} = \{A \mid A \text{ és decidible en temps polinòmic indeterminista}\}.$$

Com es relaciona NP amb P i EXP?

Tots els problemes decidibles en temps polinòmic indeterminista els agrupem en una classe.

Classe NP

Definim la classe NP (de *nondeterministic polynomial time*) com:

$$\mathrm{NP} = \{A \mid A \text{ és decidible en temps polinòmic indeterminista}\}.$$

Com es relaciona NP amb P i EXP?

Indeterminisme

Diferència fonamental entre P i NP:

1. les solucions dels problemes de P es poden trobar en temps polinòmic
2. les solucions dels problemes de NP es poden verificar en temps polinòmic

Exemple: Quadrats perfectes i compostos

1. QUADRATS $= \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad 1 \leq y < x \quad \text{i} \quad x = y^2\}$
2. COMPOSTOS $= \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad 1 < y < x \quad \text{i} \quad y \text{ divideix } x\}$

Exemple: 2 i 3-colorabilitat

1. 2-COLOR $= \{\, G \mid G \text{ és 2-colorable} \,\}$
2. 3-COLOR $= \{\, G \mid G \text{ és 3-colorable} \,\}$

Indeterminisme

Diferència fonamental entre P i NP:

1. les solucions dels problemes de P es poden trobar en temps polinòmic
2. les solucions dels problemes de NP es poden verificar en temps polinòmic

Exemple: Quadrats perfectes i compostos

1. QUADRATS $= \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad 1 \leq y < x \quad \text{i} \quad x = y^2 \}$
2. COMPOSTOS $= \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad 1 < y < x \quad \text{i} \quad y \text{ divideix } x \}$

Exemple: 2 i 3-colorabilitat

1. 2-COLOR $= \{ G \mid G \text{ és 2-colorable} \}$
2. 3-COLOR $= \{ G \mid G \text{ és 3-colorable} \}$

Indeterminisme

Diferència fonamental entre P i NP:

1. les solucions dels problemes de P es poden trobar en temps polinòmic
2. les solucions dels problemes de NP es poden verificar en temps polinòmic

Exemple: Quadrats perfectes i compostos

1. QUADRATS $= \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad 1 \leq y < x \quad \text{i} \quad x = y^2\}$
2. COMPOSTOS $= \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad 1 < y < x \quad \text{i} \quad y \text{ divideix } x\}$

Exemple: 2 i 3-colorabilitat

1. 2-COLOR $= \{\, G \mid G \text{ és 2-colorable} \,\}$
2. 3-COLOR $= \{\, G \mid G \text{ és 3-colorable} \,\}$

Indeterminisme

Teorema

$\mathrm{P} \subseteq \mathrm{NP}$.

Demostració

Tot algorisme determinista es pot considerar que és indeterminista però no fa ús de TRIAR.

Vist d'una altra manera, si $A \in \mathrm{P}$ és reconegut per un algorisme polinòmic $\mathcal{A}$, podem crear un verificador $\mathcal{V}$ que, per a tot x:

$$x \in A \Rightarrow \mathcal{V}(x, y) = 1 \text{ per a tot } y \in \Sigma^* \text{ tal que } |y| = |x|$$

$$x \notin A \Rightarrow \mathcal{V}(x, y) = 0 \text{ per a tot } y \in \Sigma^* \text{ tal que } |y| = |x|$$

Per calcular $\mathcal{V}(x, y)$, només cal simular $\mathcal{A}(x)$ i retornar el mateix valor 0 o 1 (independentment de y). Per tant, $A \in \mathrm{NP}$.

Indeterminisme

Teorema

$\mathrm{P} \subseteq \mathrm{NP}$.

Demostració

Tot algorisme determinista es pot considerar que és indeterminista però no fa ús de TRIAR.

Vist d'una altra manera, si $A \in \mathrm{P}$ és reconegut per un algorisme polinòmic $\mathcal{A}$, podem crear un verificador $\mathcal{V}$ que, per a tot x:

$$x \in A \Rightarrow \mathcal{V}(x, y) = 1 \text{ per a tot } y \in \Sigma^* \text{ tal que } |y| = |x|$$

$$x \notin A \Rightarrow \mathcal{V}(x, y) = 0 \text{ per a tot } y \in \Sigma^* \text{ tal que } |y| = |x|$$

Per calcular $\mathcal{V}(x, y)$, només cal simular $\mathcal{A}(x)$ i retornar el mateix valor 0 o 1 (independentment de y). Per tant, $A \in \mathrm{NP}$.

Indeterminisme

Diferències entre NP i EXP:

1. els problemes de NP tenen solucions verificables en temps polinòmic
2. els problemes d'EXP poden tenir solucions exponencialment llargues
3. per resoldre els problemes de NP hi ha un algorisme estàndard per trobar testimonis, però en general no per als d'EXP

Indeterminisme

Teorema

$\mathrm{NP} \subseteq \mathrm{EXP}$.

Demostració

Sigui $A \in \mathrm{NP}$. Llavors, existeix un polinomi $p(n)$ i un verificador $\mathcal{V}$ tals que

$$x \in A \Rightarrow \mathcal{V}(x, y) = 1 \text{ per a algun } y \in \Sigma^* \text{ tal que } |y| = p(|x|)$$

$$x \notin A \Rightarrow \mathcal{V}(x, y) = 0 \text{ per a tot } y \in \Sigma^* \text{ tal que } |y| = p(|x|)$$

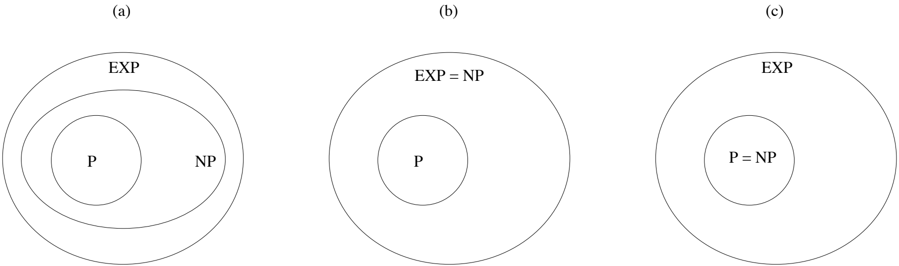
Podem considerar un algorisme exponencial per a A que cerca un testimoni:

entrada x
per a tot y tal que $|y| = p(|x|)$
    si $\mathcal{V}(x, y) = 1$ **llavors**
        retornar 1
retornar 0

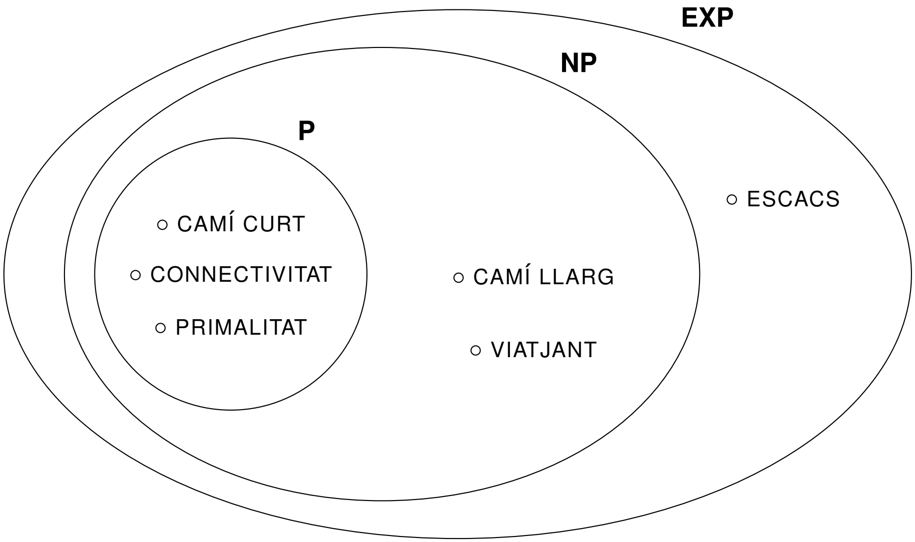
És evident que l'algorisme anterior és exponencial i decideix A. Per tant, $A \in \mathrm{EXP}$.

Indeterminisme

- Sabem que $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXP}$
- Sabem que $\mathrm{P} \neq \mathrm{EXP}$
- Per tant, podem assegurar que $\mathrm{P} \neq \mathrm{NP}$ o $\mathrm{NP} \neq \mathrm{EXP}$ i tenim tres possibilitats:



Prendrem (a) com a hipòtesi de treball.



EXP
NP
P
CAMÍ CURT
CONNECTIVITAT
PRIMALITAT
CAMÍ LLARG
VIATJANT
ESCACS

Concepte de reducció

La història de la tassa de te

Reduccions

Siguin A i B dos problemes decisionals. Diem que *A es redueix a B en temps polinòmic* si existeix un algorisme polinòmic $\mathcal{F}$ tal que

$$x \in A \Rightarrow \mathcal{F}(x) \in B$$

$$x \notin A \Rightarrow \mathcal{F}(x) \notin B$$

En aquest cas, escrivim $A \leq^p B$ (o $A \leq^p B$ *via* $\mathcal{F}$) i diem que $\mathcal{F}$ és una reducció polinòmica de A a B.

Exemples de reduccions

Paritat

Considerem el llenguatge dels nombres parells

$$\textsc{parells} = \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad x = 2y\ \}$$

i el dels senars

$$\textsc{senars} = \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad x = 2y + 1\ \}$$

Veiem que podem reduir PARELLS a SENARS (PARELLS $\leq^p$ SENARS) amb un algorisme $\mathcal{F}$ tal que $\mathcal{F}(x) = x + 1$. És evident que per a tot x:

$$x \in \textsc{parells} \Leftrightarrow \mathcal{F}(x) \in \textsc{senars}.$$

Fixem-nos que podem reduir SENARS a PARELLS amb el mateix algorisme $\mathcal{F}$, és a dir, SENARS $\leq^p$ PARELLS via $\mathcal{F}$. En general, però, la relació $\leq^p$ no és simètrica.

Paritat

Considerem el llenguatge dels nombres parells

$$\textsc{Parells} = \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad x = 2y\}$$

i el dels senars

$$\textsc{Senars} = \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \quad x = 2y + 1\}$$

Veiem que podem reduir PARELLS a SENARS (PARELLS $\leq^p$ SENARS) amb un algorisme $\mathcal{F}$ tal que $\mathcal{F}(x) = x + 1$. És evident que per a tot x:

$$x \in \textsc{Parells} \Leftrightarrow \mathcal{F}(x) \in \textsc{Senars}.$$

Fixem-nos que podem reduir SENARS a PARELLS amb el mateix algorisme $\mathcal{F}$, és a dir, SENARS $\leq^p$ PARELLS via $\mathcal{F}$. En general, però, la relació $\leq^p$ no és simètrica.

Exemples de reduccions

Particions

Considereu els dos problemes següents.

Partició
Donats els naturals $x_1, x_2, \ldots, x_n$, determinar si es poden dividir en dos grups que sumin el mateix.

Motxilla
Donats els naturals $x_1, x_2, \ldots, x_n$ i una capacitat $C \in \mathbb{N}$, determinar si es pot trobar una selecció dels x_i's que sumi exactament C.

Formalment:

$$\text{PARTICIÓ} = \{(x_1, \ldots, x_n) \mid \exists I \subseteq \{1, \ldots, n\} \quad \sum_{i \in I} x_i = \sum_{i \notin I} x_i\}$$

$$\text{MOTXILLA} = \{(x_1, \ldots, x_n, C) \mid \exists I \subseteq \{1, \ldots, n\} \quad \sum_{i \in I} x_i = C\}$$

Particions

Considereu els dos problemes següents.

Partició
Donats els naturals $x_1, x_2, \ldots, x_n$, determinar si es poden dividir en dos grups que sumin el mateix.

Motxilla
Donats els naturals $x_1, x_2, \ldots, x_n$ i una capacitat $C \in \mathbb{N}$, determinar si es pot trobar una selecció dels x_i's que sumi exactament C.

Formalment:

$$\text{PARTICIÓ} = \{(x_1, \ldots, x_n) \mid \exists I \subseteq \{1, \ldots, n\} \quad \sum_{i \in I} x_i = \sum_{i \notin I} x_i\}$$

$$\text{MOTXILLA} = \{(x_1, \ldots, x_n, C) \mid \exists I \subseteq \{1, \ldots, n\} \quad \sum_{i \in I} x_i = C\}$$

Particions

L'algorisme

$\mathcal{F}(x_1, \ldots, x_n)$
$\quad S \leftarrow \sum_{i=1}^{n} x_i$
$\quad$ **si** S és senar **llavors**
$\quad\quad$ **retornar** $(x_1, \ldots, x_n, S+1)$
$\quad$ **si no**
$\quad\quad$ **retornar** $(x_1, \ldots, x_n, S/2)$

és una reducció polinòmica de PARTICIÓ a MOTXILLA:

$$(x_1, \ldots, x_n) \in \text{PARTICIÓ} \Leftrightarrow \mathcal{F}(x_1, \ldots, x_n) \in \text{MOTXILLA}.$$

Exercici

Definim la col·lecció següent de problemes de coloració:

> ***k*-Colorabilitat** (k-COLOR)
> Donat un graf no dirigit G, determinar si els vèrtexs de G es poden acolorir amb un màxim de k colors, de manera que a tot parell de vèrtexs adjacents els corresponguin colors diferents.

Demostreu que, per a tot k, es compleix:

$$k\text{-COLOR} \leq^p (k+1)\text{-COLOR}.$$

Definició

Un camí hamiltonià d'un graf G és un camí que passa per tots els vèrtexs sense repetir-ne cap.

Exercici

Definim el problema del camí hamiltonià (HP) i el del camí hamiltonià entre dos vèrtexs (HP_2):

- $\mathrm{HP} = \{G \mid G$ té un camí hamiltonià$\}$
- $\mathrm{HP}_2 = \{\, (G, u, v) \mid G$ té un camí hamiltonià amb extrems u, $v\}$

Proposeu:

1. una reducció que demostri $\mathrm{HP} \leq^p \mathrm{HP}_2$
2. una reducció que demostri $\mathrm{HP}_2 \leq^p \mathrm{HP}$

Propietats

Propietats: reflexivitat

Per a tot A, $A \leq^p A$.

N'hi ha prou a considerar l'algorisme que calcula la funció identitat:

$\mathcal{F}(x)$
  retornar x

És evident que, per a tot x

$$x \in A \;\Leftrightarrow\; \mathcal{F}(x) = x \in A.$$

Propietats

Propietats: transitivitat

Per a tot A, B, C, si $A \leq^p B$ i $B \leq^p C$, llavors $A \leq^p C$.

Si

- $A \leq^p B$ via $\mathcal{F}$ i
- $B \leq^p C$ via $\mathcal{G}$,

llavors la composició $\mathcal{G} \circ \mathcal{F}$ ($\mathcal{F} | \mathcal{G}$ en notació *pipe* de UNIX) demostra que $A \leq^p C$.

Considerem que $\mathcal{G} \circ \mathcal{F}(x) = \mathcal{G}(\mathcal{F}(x))$.

Exercici

Demostreu

$$3\text{-}\textsc{color} \leq^p k\text{-}\textsc{color}$$

per a tot $k \geq 4$ amb dos mètodes diferents:

1. fent servir la transitivitat de les reduccions
2. donant una reducció explícita

Propietats

Corol·lari

Les reduccions formen un preordre.

Qüestió

Observeu que, si bé les reduccions formen un preordre, en canvi no formen un ordre parcial perquè no compleixen la propietat antisimètrica:

- $\forall A, B \quad A \leq^p B \ \wedge \ B \leq^p A \ \Rightarrow \ A = B$

Propietats

Corol·lari

Les reduccions formen un preordre.

Qüestió

Observeu que, si bé les reduccions formen un preordre, en canvi no formen un ordre parcial perquè no compleixen la propietat antisimètrica:

- $\forall A, B \quad A \leq^p B \wedge B \leq^p A \Rightarrow A = B$

Propietats

Tancament de P per reduccions

Per a tot A, B, si $A \leq^p B$ i $B \in \mathrm{P}$, llavors $A \in \mathrm{P}$.

Si

- $\mathcal{B}$ és un algorisme polinòmic per a B i
- $\mathcal{F}$ és un algorisme polinòmic que demostra $A \leq^p B$,

llavors la composició $\mathcal{B} \circ \mathcal{F}$ és un algorisme polinòmic per a A:

1. $\mathcal{B} \circ \mathcal{F}$ és polinòmic perquè és composició d'algorismes polinòmics
2. $\mathcal{B} \circ \mathcal{F}(x)$ accepta $\Leftrightarrow$ $\mathcal{B}$ accepta $\mathcal{F}(x) \Leftrightarrow \mathcal{F}(x) \in B \Leftrightarrow x \in A$

Propietats

Tancament de P per reduccions

Per a tot A, B, si $A \leq^p B$ i $B \in \mathrm{P}$, llavors $A \in \mathrm{P}$.

Si

- $\mathcal{B}$ és un algorisme polinòmic per a B i
- $\mathcal{F}$ és un algorisme polinòmic que demostra $A \leq^p B$,

llavors la composició $\mathcal{B} \circ \mathcal{F}$ és un algorisme polinòmic per a A:

1. $\mathcal{B} \circ \mathcal{F}$ és polinòmic perquè és composició d'algorismes polinòmics
2. $\mathcal{B} \circ \mathcal{F}(x)$ accepta $\Leftrightarrow$ $\mathcal{B}$ accepta $\mathcal{F}(x)$ $\Leftrightarrow$ $\mathcal{F}(x) \in B \Leftrightarrow x \in A$

Propietats

Tancament de P per reduccions

Per a tot A, B, si $A \leq^p B$ i $B \in \mathrm{P}$, llavors $A \in \mathrm{P}$.

Si

- $\mathcal{B}$ és un algorisme polinòmic per a B i
- $\mathcal{F}$ és un algorisme polinòmic que demostra $A \leq^p B$,

llavors la composició $\mathcal{B} \circ \mathcal{F}$ és un algorisme polinòmic per a A:

1. $\mathcal{B} \circ \mathcal{F}$ és polinòmic perquè és composició d'algorismes polinòmics
2. $\mathcal{B} \circ \mathcal{F}(x)$ accepta $\Leftrightarrow$ $\mathcal{B}$ accepta $\mathcal{F}(x)$ $\Leftrightarrow$ $\mathcal{F}(x) \in B \Leftrightarrow x \in A$

Propietats

Equivalència polinòmica

Donats dos problemes decisionals A, B, escrivim $A \equiv^p B$ si $A \leq^p B$ i $B \leq^p A$.

Problema: Classes d'equivalència de P

1. Demostreu que $\equiv^p$ és una relació d'equivalència (reflexiva, simètrica i transitiva)
2. Demostreu que per a tot A, B, si $A \in \mathrm{P}$ i $B \neq \emptyset, \Sigma^*$, llavors $A \leq^p B$
3. Deduïu quina és la partició de P en classes d'equivalència induïda per la relació $\equiv^p$

Propietats

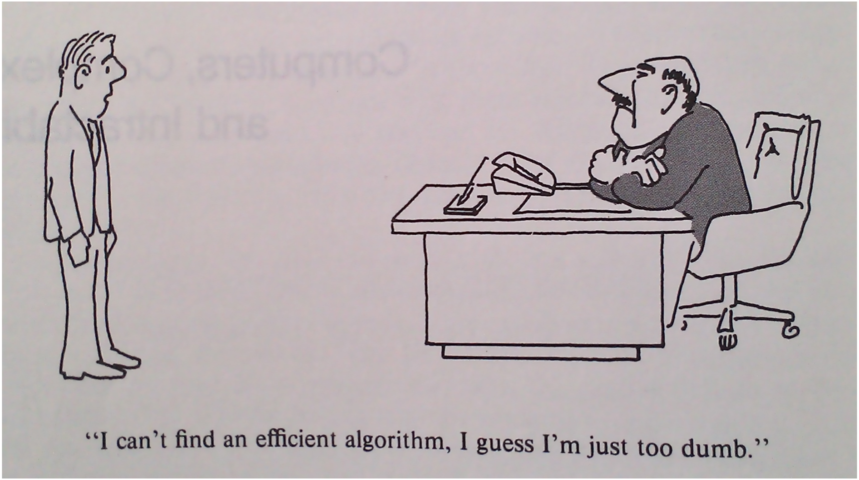
Equivalència polinòmica

Donats dos problemes decisionals A, B, escrivim $A \equiv^p B$ si $A \leq^p B$ i $B \leq^p A$.

Problema: Classes d'equivalència de P

1. Demostreu que $\equiv^p$ és una relació d'equivalència (reflexiva, simètrica i transitiva)
2. Demostreu que per a tot A, B, si $A \in \mathrm{P}$ i $B \neq \emptyset, \Sigma^*$, llavors $A \leq^p B$
3. Deduïu quina és la partició de P en classes d'equivalència induïda per la relació $\equiv^p$

Teoria de la NP-completesa



Garey & Johnson, *Computers and Intractability*

Teoria de la NP-completesa

"I can't find an efficient algorithm, because no such algorithm is possible!"

Garey & Johnson, *Computers and Intractability*

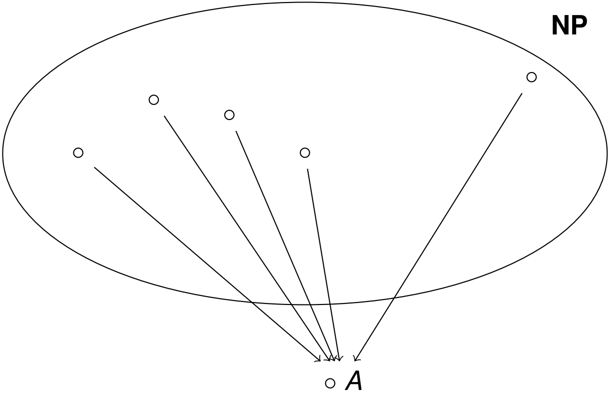
Teoria de la NP-completesa

"I can't find an efficient algorithm, but neither can all these famous people."

Garey & Johnson, *Computers and Intractability*

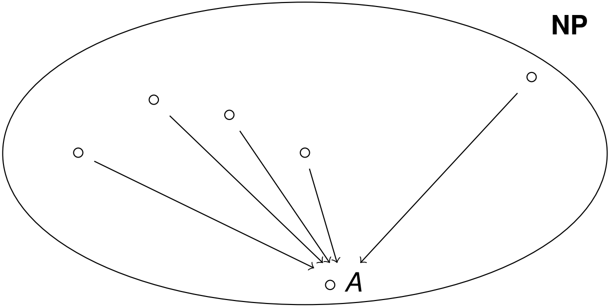
Definició

Un problema A és NP-difícil si per a tot problema $B \in \mathrm{NP}$ tenim que $B \leq^p A$.



Definició

Un problema A és NP-complet si és NP-difícil i $A \in \mathrm{NP}$.



Qualsevol problema NP-complet "representa" tota la classe NP respecte de P.

Més formalment...

Proposició

Sigui A un problema NP-complet. Llavors, $\mathrm{P} = \mathrm{NP}$ si i només si $A \in \mathrm{P}$.

$\Rightarrow$ Com que A és NP-complet, $A \in \mathrm{NP}$ i, per tant, $A \in \mathrm{P}$.

$\Leftarrow$ Sigui $A \in \mathrm{P}$.

1. Com que A és NP-complet, sabem que per a tot $B \in \mathrm{NP}$, $B \leq^p A$
2. Pel tancament de P per reduccions, sabem que si $B \leq^p A$, llavors $B \in \mathrm{P}$

Per 1 i 2, $\mathrm{NP} \subseteq \mathrm{P}$ i, per tant, $\mathrm{P} = \mathrm{NP}$.

Però... existeixen els NP-complets?

Qualsevol problema NP-complet “representa” tota la classe NP respecte de P.

Més formalment...

Proposició

Sigui A un problema NP-complet. Llavors, $\mathrm{P} = \mathrm{NP}$ si i només si $A \in \mathrm{P}$.

$\boxed{\Rightarrow}$ Com que A és NP-complet, $A \in \mathrm{NP}$ i, per tant, $A \in \mathrm{P}$.

$\boxed{\Leftarrow}$ Sigui $A \in \mathrm{P}$.

1. Com que A és NP-complet, sabem que per a tot $B \in \mathrm{NP}$, $B \le^p A$
2. Pel tancament de P per reduccions, sabem que si $B \le^p A$, llavors $B \in \mathrm{P}$

Per 1 i 2, $\mathrm{NP} \subseteq \mathrm{P}$ i, per tant, $\mathrm{P} = \mathrm{NP}$.

Però... existeixen els NP-complets?

Qualsevol problema NP-complet "representa" tota la classe NP respecte de P.

Més formalment...

Proposició

Sigui A un problema NP-complet. Llavors, $\mathrm{P} = \mathrm{NP}$ si i només si $A \in \mathrm{P}$.

$\boxed{\Rightarrow}$ Com que A és NP-complet, $A \in \mathrm{NP}$ i, per tant, $A \in \mathrm{P}$.

$\boxed{\Leftarrow}$ Sigui $A \in \mathrm{P}$.

1. Com que A és NP-complet, sabem que per a tot $B \in \mathrm{NP}$, $B \leq^p A$
2. Pel tancament de P per reduccions, sabem que si $B \leq^p A$, llavors $B \in \mathrm{P}$

Per 1 i 2, $\mathrm{NP} \subseteq \mathrm{P}$ i, per tant, $\mathrm{P} = \mathrm{NP}$.

Però... existeixen els NP-complets?

Qualsevol problema NP-complet “representa” tota la classe NP respecte de P.

Més formalment...

Proposició

Sigui A un problema NP-complet. Llavors, $\mathrm{P} = \mathrm{NP}$ si i només si $A \in \mathrm{P}$.

$\boxed{\Rightarrow}$ Com que A és NP-complet, $A \in \mathrm{NP}$ i, per tant, $A \in \mathrm{P}$.

$\boxed{\Leftarrow}$ Sigui $A \in \mathrm{P}$.

1. Com que A és NP-complet, sabem que per a tot $B \in \mathrm{NP}$, $B \le^p A$
2. Pel tancament de P per reduccions, sabem que si $B \le^p A$, llavors $B \in \mathrm{P}$

Per 1 i 2, $\mathrm{NP} \subseteq \mathrm{P}$ i, per tant, $\mathrm{P} = \mathrm{NP}$.

Però... existeixen els NP-complets?

Fórmules booleanes

- Una fórmula booleana (f.b.) és un predicat sobre variables booleanes sense quantificadors.
- Farem servir les connectives $\vee$ (disjunció), $\wedge$ (conjunció) i $\neg$ (negació).

Per exemple,

$$F(x, y, z) = (x \vee y \vee \neg z) \wedge \neg(x \wedge y \wedge z)$$

és una fórmula booleana.

Forma normal conjuntiva (CNF)

- Un literal és una variable afirmada o negada (x, $\neg x$)
- Una clàusula és una disjunció de literals ($x \vee \neg y \vee z$)
- Una fórmula booleana està en forma normal conjuntiva (CNF) si és una conjunció de clàusules

$$F(x, y, z) = (x \vee \neg y \vee z) \wedge (\neg x \vee \neg z)$$

Fórmules booleanes

- Una fórmula booleana (f.b.) és un predicat sobre variables booleanes sense quantificadors.
- Farem servir les connectives $\vee$ (disjunció), $\wedge$ (conjunció) i $\neg$ (negació).

Per exemple,

$$F(x, y, z) = (x \vee y \vee \neg z) \wedge \neg(x \wedge y \wedge z)$$

és una fórmula booleana.

Forma normal conjuntiva (CNF)

- Un literal és una variable afirmada o negada (x, $\neg x$)
- Una clàusula és una disjunció de literals ($x \vee \neg y \vee z$)
- Una fórmula booleana està en forma normal conjuntiva (CNF) si és una conjunció de clàusules

$$F(x, y, z) = (x \vee \neg y \vee z) \wedge (\neg x \vee \neg z)$$

Satisfactibilitat

Una fórmula booleana és satisfactible si existeix una assignació de valors de veritat a les variables per a la qual la fórmula és certa. Per exemple,

$$F(x, y, z) = (x \vee \neg y \vee z) \wedge (\neg x \vee \neg z)$$

és satisfactible amb $x = 1$, $y = 0$, $z = 0$. Escrivim $F(100) = 1$.

Definim

$$\text{SAT} = \{\, F \mid F \text{ és una fórmula booleana satisfactible} \,\}$$

$$\text{CNF-SAT} = \{\, F \mid F \text{ és una f.b. en CNF satisfactible} \,\}$$

Teorema de Cook-Levin (1971)

SAT i CNF-SAT són NP-complets.

Satisfactibilitat

Una fórmula booleana és satisfactible si existeix una assignació de valors de veritat a les variables per a la qual la fórmula és certa. Per exemple,

$$F(x, y, z) = (x \vee \neg y \vee z) \wedge (\neg x \vee \neg z)$$

és satisfactible amb $x = 1$, $y = 0$, $z = 0$. Escrivim $F(100) = 1$.

Definim

$$\begin{aligned} \text{SAT} &= \{ F \mid F \text{ és una fórmula booleana satisfactible} \} \\ \text{CNF-SAT} &= \{ F \mid F \text{ és una f.b. en CNF satisfactible} \} \end{aligned}$$

Teorema de Cook-Levin (1971)

SAT i CNF-SAT són NP-complets.

Teorema de Cook-Levin (1971)

SAT i CNF-SAT són NP-complets.

Per demostrar el teorema de Cook-Levin, cal veure:

1. CNF-SAT $\in \mathrm{NP}$
2. CNF-SAT és NP-difícil

(1) CNF-SAT $\in \mathrm{NP}$

- Els testimonis són les assignacions de les variables booleanes a $\{0, 1\}$
- En qualsevol codificació raonable d'una fórmula F de n variables, $n \leq |F|$. Com que un testimoni α consta de n bits, $|\alpha| = n \leq |F|$
- Per tant, triant $p(n) = n$, tenim que $|\alpha| \leq p(|F|)$
- Podem verificar si una assignació α satisfà F en temps polinòmic:
  - substituïm les variables pels valors donats per α
  - avaluem les connectives de dins cap a fora

Exemple

En el cas de la fórmula booleana en CNF

$$F(x, y, z) = (x \vee \neg y \vee z) \wedge (x \vee \neg z)$$

i l'assignació $\alpha = 100$ (és a dir, $x = 1$, $y = 0$, $z = 0$), el verificador avaluaria:

- (substituir valors) $F(\alpha) = (1 \vee \neg 0 \vee 0) \wedge (1 \vee \neg 0)$
- (calcular negacions) $F(\alpha) = (1 \vee 1 \vee 0) \wedge (1 \vee 1)$
- (calcular disjuncions) $F(\alpha) = (1) \wedge (1)$
- (calcular conjuncions) $F(\alpha) = 1$

Lemma

Donat un algorisme $\mathcal{A} : E \to \{0,1\}$ amb cost d'espai polinòmic en cas pitjor, podem trobar en temps polinòmic una f.b. en CNF $F_{\mathcal{A}}$ tal que per a tot $y \in E$:

$$F_{\mathcal{A}}(y) = 1 \Leftrightarrow \mathcal{A}(y) = 1.$$

(2) CNF-SAT és NP-difícil.

Sigui $A \in \mathrm{NP}$. Llavors, hi ha un polinomi q i un verificador $\mathcal{V}$ t.q. per a tot x:

$$x \in A \Leftrightarrow \exists y \quad |y| = q(|x|) \wedge \mathcal{V}(x,y) = 1.$$

Sigui $\mathcal{V}_x(y)$ un nou verificador, per a x fixat, tal que

$$\mathcal{V}_x(y) = 1 \Leftrightarrow |y| = q(|x|) \wedge \mathcal{V}(x,y) = 1.$$

Llavors,

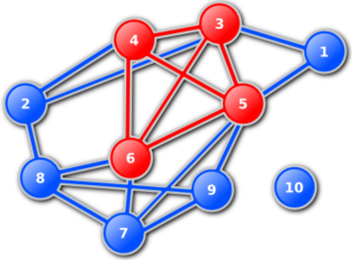
$$x \in A \Leftrightarrow \exists y \;\; F_{\mathcal{V}_x}(y) \;\Leftrightarrow F_{\mathcal{V}_x}(y) \in \text{CNF-SAT}.$$

Per tant, $A \leq^p$ CNF-SAT.

Trobar un primer NP-complet permet trobar-ne més per reduccions. Recordem que

$$\text{CLICA} = \{\ (G, k) \mid G \text{ té un subgraf complet de } k \text{ vèrtexs}\ \}.$$

Donat el graf G



podem observar que

- $(G, 4) \in$ CLICA
- $(G, 5) \notin$ CLICA

Teorema

CLICA és NP-complet

Per demostrar la NP-completesa de CLICA cal veure que:

1. CLICA $\in$ NP
2. CLICA és NP-difícil

(1) CLICA $\in$ NP

Sigui (G, k) una entrada de CLICA.

- Els testimonis són els vèrtexs dels subgrafs complets de G de k vèrtexs (en l'exemple anterior, el conjunt $C = \{3, 4, 5, 6\}$)
- El polinomi $p(n) = n$ és suficient perquè un testimoni C compleix $|C| \ \leq |(G, k)| = p(|(G, k)|)$
- Podem verificar en temps polinòmic si un conjunt C és un testimoni: tot parell de vèrtexs de C ha de formar una aresta en G ($\binom{n}{2} \leq n^2$ comprovacions)

Teorema

CLICA és NP-complet

Per demostrar la NP-completesa de CLICA cal veure que:

1. CLICA $\in$ NP
2. CLICA és NP-difícil

(2) CLICA és NP-difícil

Demostrarem que CNF-SAT $\leq^p$ CLICA. Aleshores,

- Com que CNF-SAT és NP-difícil, tot $S \in$ NP compleix $S \leq^p$ CNF-SAT
- Per transitivitat, tot $S \in$ NP complirà $S \leq^p$ CLICA
- Per tant, CLICA serà NP-difícil

Problemes NP-complets

Podem expressar aquesta propietat en general.

Proposició

Sigui A un problema NP-complet i B un problema tal que $B \in \mathrm{NP}$ i $A \leq^p B$. Llavors, B també és NP-complet.

- Com que A és NP-difícil, qualsevol $S \in \mathrm{NP}$ satisfà $S \leq^p A$
- Per transitivitat, qualsevol $S \in \mathrm{NP}$ satisfà $S \leq^p B$
- Per tant, B és NP-difícil

Problemes NP-complets

CNF-SAT $\leq^p$ CLICA

Sigui F una fórmula booleana en CNF amb:

- clàusules $C_1, \ldots, C_m$
- literals $l_1, \ldots, l_r$

L'algorisme de reducció és $\mathcal{R}(F) = (G, m)$, on $G = (V, E)$ és:

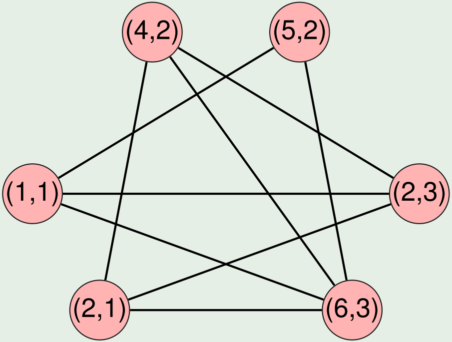
- $V = \{(i, j) \mid l_i \text{ apareix a } C_j\}$
  (Els vèrtexs representen ocurrències de literals en clàusules)
- $E = \{\ \{(i, j), (k, l)\} \mid j \neq l \wedge \neg l_i \neq l_k\ \}$
  (Les arestes representen parells de literals que poden ser certs alhora)

Exemple

$F(x_1, x_2, x_3) = C_1 \wedge C_2 \wedge C_3$, on

- $C_1 = (x_1 \vee x_2)$, $C_2 = (\neg x_1 \vee \neg x_2)$, $C_3 = (x_2 \vee \neg x_3)$
- $l_1 = x_1$, $l_2 = x_2$, $l_3 = x_3$, $l_4 = \neg x_1$, $l_5 = \neg x_2$, $l_6 = \neg x_3$

La reducció és $\mathcal{R}(F) = (G, 3)$, on G és el graf

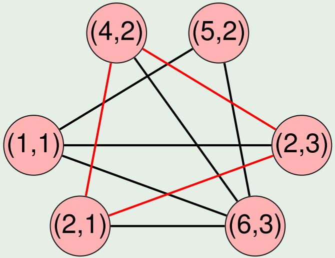


En general, tenim que $F \in$ CNF-SAT $\Leftrightarrow (G, m) \in$ CLICA:

$\boxed{\Rightarrow}$ Sigui α una assignació que satisfà F. Llavors, hi ha m ocurrències de literals que α fa certes alhora i que formen un subgraf complet en G

$\boxed{\Leftarrow}$ Si G té un subgraf complet de m vèrtexs, cada vèrtex ha de correspondre a una clàusula diferent. Per tant, es pot fer cert un literal de cada clàusula alhora i F és satisfactible

Exemple anterior amb $l_2 = 1, l_4 = 1$



Problemes NP-complets

Definicions

- *H* és un subgraf complet de *G* si conté totes les arestes possibles entre els seus vèrtexs.
- *H* és un subconjunt independent de *G* si consisteix en vèrtexs aïllats.
- *H* és un recobriment de vèrtexs de *G* si té un extrem de tota aresta de *G*.

Exercici

Donats els problemes següents:

- CLICA $= \{ (G, k) \mid$ G té un subgraf complet de k vèrtexs $\}$
- SI $= \{ (G, k) \mid$ G té un subconjunt independent de k vèrtexs $\}$
- RV $= \{ (G, k) \mid$ G té un recobriment de k vèrtexs $\}$

demostreu

1. CLICA $\leq^p$ IS
2. IS $\leq^p$ RV
3. RV $\leq^p$ CLICA

Problemes NP-complets

Molts NP-complets tenen "casos particulars" que són a P.

Per exemple, en CNF-SAT podem fixar el nombre de literals per clàusula per obtenir una família infinita de problemes.

Satisfactibilitat *k*-fitada (k-SAT)
Donada un fórmula booleana en CNF F de n variables amb $\leq k$ literals per clàusula, determinar si és satisfactible.

Veurem com classificar k-SAT pels diferents valors de k.

Satisfactibilitat 1-fitada (1-SAT)
Donada un fórmula booleana en CNF F de n variables amb 1 literal per clàusula, determinar si és satisfactible.

Per exemple,

$$F(x, y, z, t) = (x) \wedge (\neg y) \wedge (z) \wedge (\neg t).$$

1-SAT és decidible en temps polinòmic amb l'algorisme següent:

entrada F
si F conté dos literals contradictoris **llavors**
  retornar 0
si no
  retornar 1

Satisfactibilitat 1-fitada (1-SAT)
Donada un fórmula booleana en CNF F de n variables amb 1 literal per clàusula, determinar si és satisfactible.

Per exemple,

$$F(x, y, z, t) = (x) \wedge (\neg y) \wedge (z) \wedge (\neg t).$$

1-SAT és decidible en temps polinòmic amb l'algorisme següent:

entrada F
si F conté dos literals contradictoris **llavors**
  retornar 0
si no
  retornar 1

Satisfactibilitat 2-fitada (2-SAT)
Donada un fórmula booleana en CNF F de n variables amb ≤ 2 literals per clàusula, determinar si és satisfactible.

Per exemple,

$$F(x, y, z) = (x \vee y) \wedge (x \vee \neg z) \wedge (\neg x \vee y) \wedge (\neg y \vee \neg z).$$

2-SAT és decidible en temps polinòmic

- transformant la fórmula en un graf dirigit
- aplicant al graf un algorisme de camins

Satisfactibilitat 2-fitada (2-SAT)
Donada un fórmula booleana en CNF F de n variables amb ≤ 2 literals per clàusula, determinar si és satisfactible.

Per exemple,

$$F(x, y, z) = (x \vee y) \wedge (x \vee \neg z) \wedge (\neg x \vee y) \wedge (\neg y \vee \neg z).$$

2-SAT és decidible en temps polinòmic

- transformant la fórmula en un graf dirigit
- aplicant al graf un algorisme de camins

Esbós de l'algorisme

Donada una fórmula booleana en 2-CNF

$$F(x,y,z) = (x \vee y) \wedge (x \vee \neg z) \wedge (\neg x \vee y) \wedge (\neg y \vee \neg z)$$

es reescriu fent servir implicacions

$$F(x,y,z) = (\neg x \Rightarrow y) \wedge (z \Rightarrow x) \wedge (x \Rightarrow y) \wedge (y \Rightarrow \neg z)$$

que es basen en les equivalències

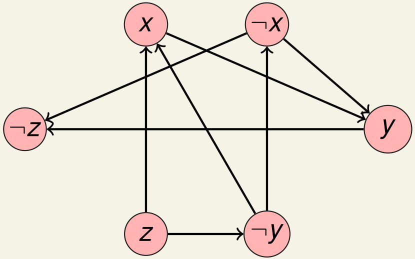
- $(a \vee b) \equiv (\neg a \Rightarrow b) \equiv (\neg b \Rightarrow a)$
- $(a) \equiv (a \vee a) \equiv (\neg a \Rightarrow a) \equiv (a \Rightarrow \neg a)$

Problemes NP-complets

La fórmula booleana amb implicacions

$$F(x, y, z) = (\neg x \Rightarrow y) \wedge (z \Rightarrow x) \wedge (x \Rightarrow y) \wedge (y \Rightarrow \neg z)$$

es transforma en un digraf G i s'aplica el lema.



Lema

F és insatisfactible si i només si conté una variable v per a la qual G té camins de v a $\neg v$ i de $\neg v$ a v.

Satisfactibilitat 3-fitada (3-SAT)
Donada un fórmula booleana en CNF F de n variables amb ≤ 3 literals per clàusula, determinar si és satisfactible.

Teorema

3-SAT és NP-complet.

Per demostrar-ho, cal provar:

1. 3-SAT $\in$ NP (semblant a CNF-SAT)
2. 3-SAT és NP-difícil: reducció CNF-SAT $\leq^p$ 3-SAT

Problemes NP-complets

Satisfactibilitat 3**-fitada** (3-SAT)
Donada un fórmula booleana en CNF F de n variables amb $\leq$ 3 literals per clàusula, determinar si és satisfactible.

Teorema

3-SAT és NP-complet.

Per demostrar-ho, cal provar:

1. 3-SAT $\in \mathrm{NP}$ (semblant a CNF-SAT)
2. 3-SAT és NP-difícil: reducció CNF-SAT $\leq^p$ 3-SAT

CNF-SAT $\leq^p$ 3-SAT

El mètode següent transforma un fórmula booleana en CNF en una altra d'equivalent en 3-CNF.

Donada una f.b. F en CNF,

1. Sigui F' una f.b. buida
2. Per a cada clàusula $C = (a_1 \vee \cdots \vee a_k)$ de F:
   - si $k \leq 3$, afegir C a F'
   - si $k > 3$, afegir a F' la clàusula

     $$(a_1 \vee a_2 \vee z_1) \wedge (\neg z_1 \vee a_3 \vee z_2) \wedge (\neg z_2 \vee a_4 \vee z_3) \ldots (\neg z_{k-3} \vee a_{k-1} \vee a_k)$$

     on $z_1, \ldots, z_{k-3}$ són variables noves.
3. Retornar F'

Exemple

Donada una clàusula de cinc literals $C = (a_1 \vee a_2 \vee a_3 \vee a_4 \vee a_5)$, la reducció retorna

$$C' = (a_1 \vee a_2 \vee z_1) \wedge (\neg z_1 \vee a_3 \vee z_2) \wedge (\neg z_2 \vee a_4 \vee a_5).$$

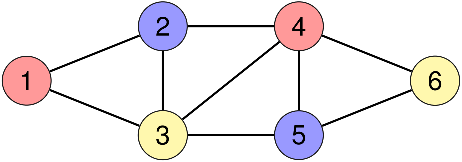
- És evident que si C és certa amb una assignació α, C' es pot satisfer amb α i valors adequats de z_1 i z_2
- Si C' és certa amb una assignació β, algun a_i serà cert i C serà certa amb l'assignació als a_i's de β

Definició

Un graf $G = (V, E)$ de n vèrtexs és k-colorable si existeix una funció total

$$\chi : V \to \{1, \ldots, k\}$$

t.q. $\chi(u) \neq \chi(v)$ per a cada aresta $\{u, v\} \in E$. La funció χ és una k-coloració.



3-coloració

Problemes NP-complets

Amb el nombre de colors k com a paràmetre extern, podem plantejar el problema de la colorabilitat en funció de k.

> k-**Colorabilitat** (k-COLOR)
> Donat un graf G, determinar si és k-colorable.

Per als casos següents se'n coneixen algorismes polinòmics:

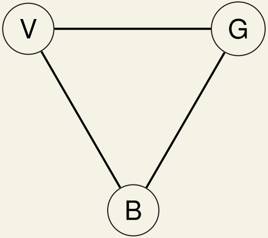
- 1-COLOR
- 2-COLOR

Problemes NP-complets

CNF-SAT $\leq^p$ 3-COLOR

Sigui F una fórmula booleana en CNF. Construirem un graf G que serà 3-colorable si i només si F és satisfactible.

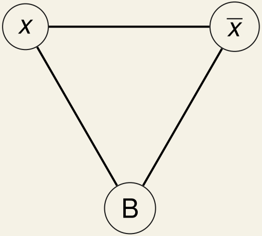
- Hi haurà 3 vèrtexs especials anomenats V, G, B.



Podem suposar que, en qualsevol coloració, tenen els colors:

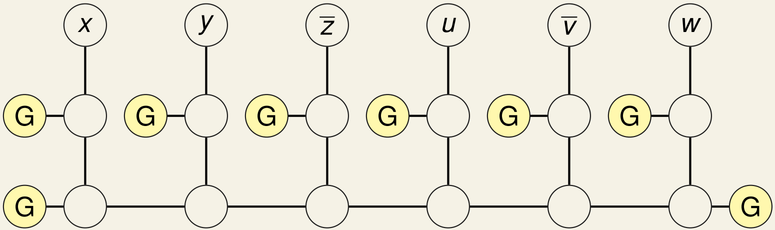
V $\rightarrow$ vermell, G $\rightarrow$ groc, B $\rightarrow$ blau

- Afegim un vèrtex per cada literal i connectem cada literal i el seu complementari al vèrtex B.



- Per cada clàusula, afegim un subgraf com el següent. En aquest cas

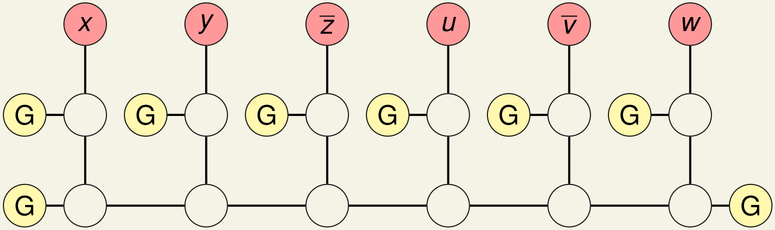
$$(x \vee y \vee \overline{z} \vee u \vee \overline{v} \vee w).$$



Propietat: Una coloració dels vèrtexs superiors amb vermell o groc es pot estendre a una 3-coloració global si i només si almenys un és groc.

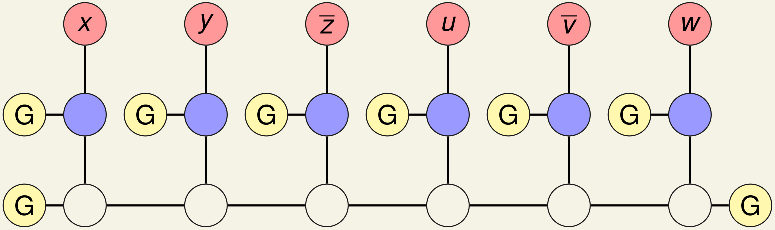
Problemes NP-complets

Si tots els de dalt són vermells...



...no podem completar la 3-coloració.

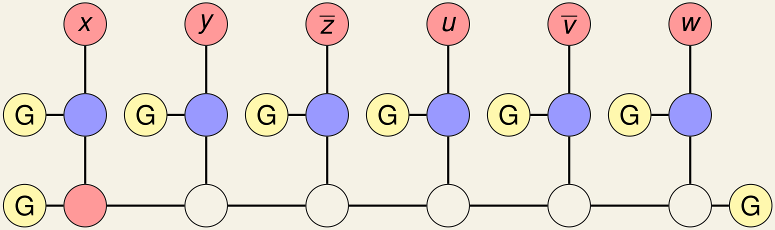
Si tots els de dalt són vermells...



...no podem completar la 3-coloració.

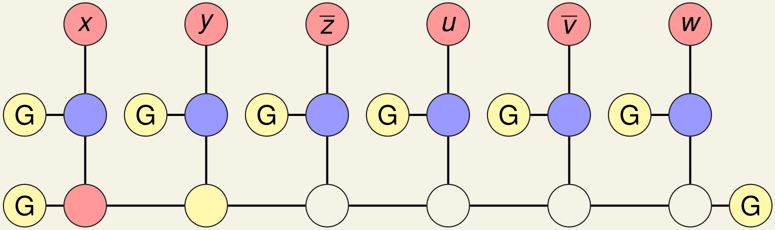
Problemes NP-complets

Si tots els de dalt són vermells...



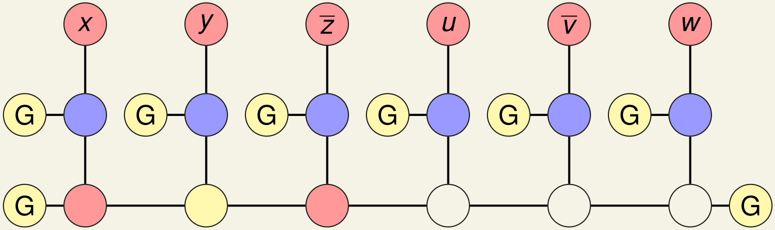
...no podem completar la 3-coloració.

Si tots els de dalt són vermells...



...no podem completar la 3-coloració.

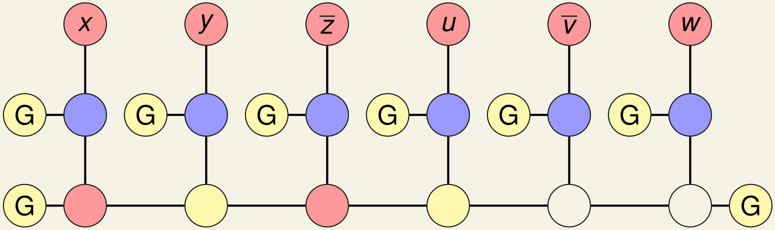
Si tots els de dalt són vermells...



...no podem completar la 3-coloració.

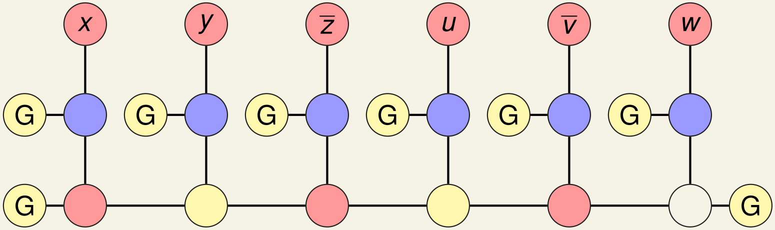
Problemes NP-complets

Si tots els de dalt són vermells...



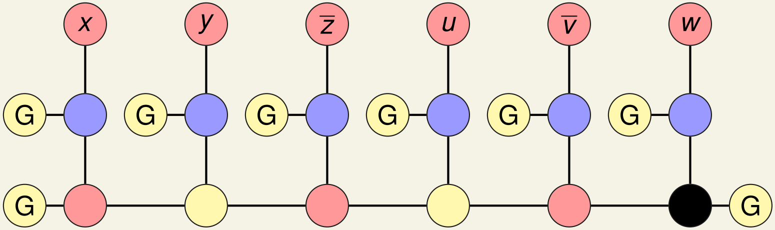
...no podem completar la 3-coloració.

Si tots els de dalt són vermells...



...no podem completar la 3-coloració.

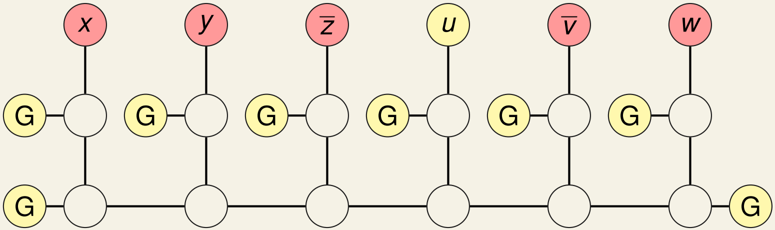
Si tots els de dalt són vermells...



...no podem completar la 3-coloració.

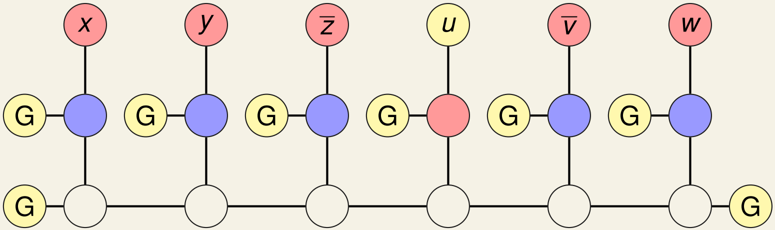
Problemes NP-complets

Si almenys un de dalt és groc...



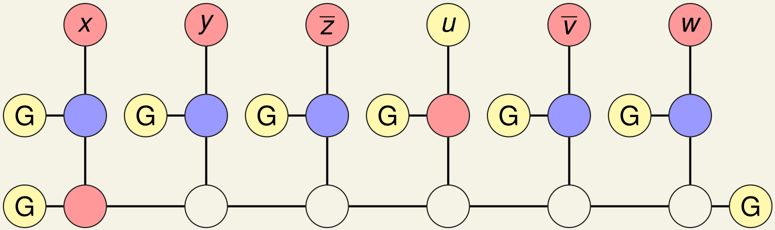
...podem obtenir una 3-coloració global.

Si almenys un de dalt és groc...



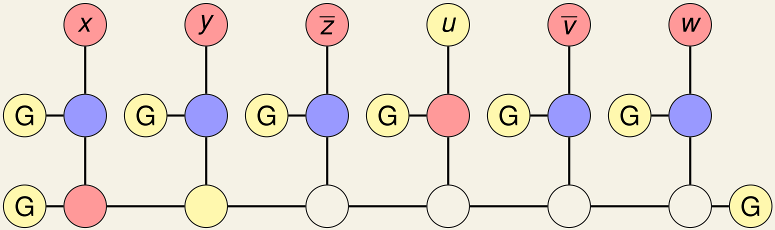
...podem obtenir una 3-coloració global.

Si almenys un de dalt és groc...



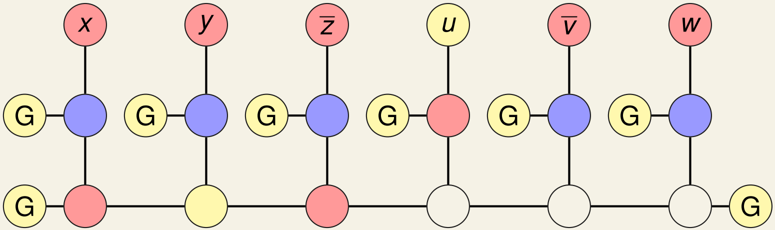
...podem obtenir una 3-coloració global.

Si almenys un de dalt és groc...



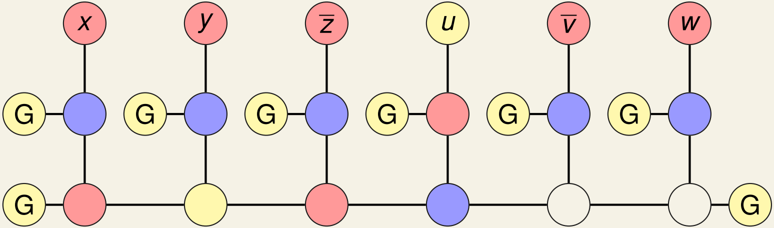
...podem obtenir una 3-coloració global.

Si almenys un de dalt és groc...



...podem obtenir una 3-coloració global.

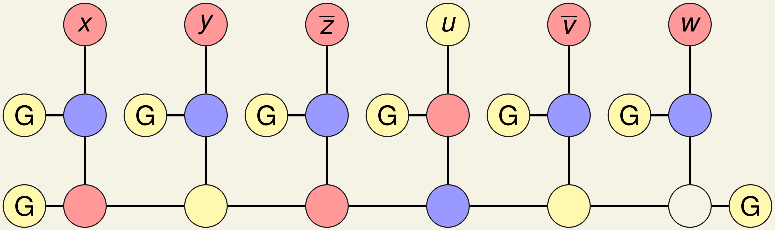
Si almenys un de dalt és groc...



...podem obtenir una 3-coloració global.

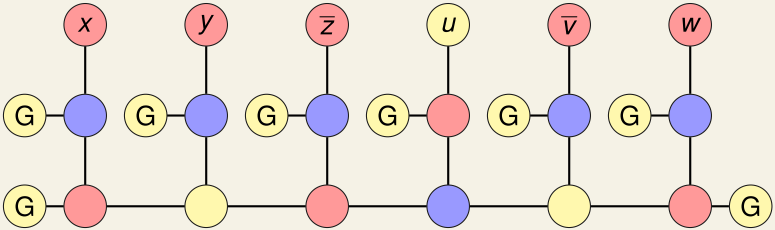
Problemes NP-complets

Si almenys un de dalt és groc...



...podem obtenir una 3-coloració global.

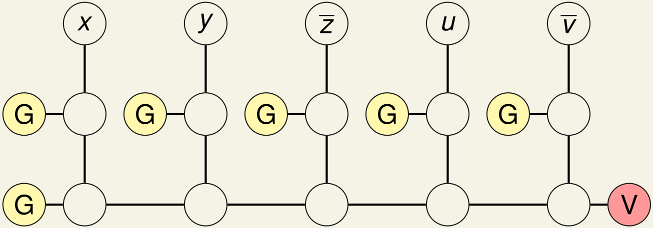
Si almenys un de dalt és groc...



...podem obtenir una 3-coloració global.

En cas que el nombre de literals sigui senar, el vèrtex de la dreta serà V.
Per exemple,

$$(x \vee y \vee \overline{z} \vee u \vee \overline{v})$$



Problemes NP-complets

Si G és el graf amb tots els vèrtexs i arestes definits abans, llavors

$$F \text{ és satisfactible} \Leftrightarrow G \text{ és 3-colorable.}$$

Com que G es pot construir en temps polinòmic, tenim que

$$\text{CNF-SAT} \leq^p \text{3-COLOR}.$$

Teorema

3-COLOR és NP-complet.

Problemes NP-complets

Per la resta de problemes k-COLOR, podem observar el següent.

Proposició

Per a tot $k > 3$, 3-COLOR $\leq^p$ k-COLOR.

La reducció consisteix, donat un graf G, a afegir-li un subgraf complet de $k - 3$ vèrtexs connectats a tots els de G.

Corol·lari

Per a tot $k > 3$, k-COLOR és NP-complet.

Per tant, tenim:

- k-COLOR $\in \mathrm{P}$ per a $k \leq 2$
- k-COLOR és NP-complet per a $k \geq 3$

Problemes NP-complets

Per la resta de problemes k-COLOR, podem observar el següent.

Proposició

Per a tot $k > 3$, 3-COLOR $\leq^p$ k-COLOR.

La reducció consisteix, donat un graf G, a afegir-li un subgraf complet de $k - 3$ vèrtexs connectats a tots els de G.

Corol·lari

Per a tot $k > 3$, k-COLOR és NP-complet.

Per tant, tenim:

- k-COLOR $\in \mathrm{P}$ per a $k \leq 2$
- k-COLOR és NP-complet per a $k \geq 3$

Problemes NP-complets

Què podem dir de la colorabilitat de grafs planars? Considerem la sèrie de problemes següent.

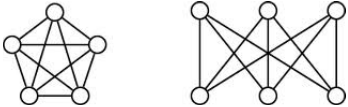
> k-**Colorabilitat planar** (k-COLOR-PL)
> Donat un graf planar G, determinar si és k-colorable.

La planaritat es pot comprovar en temps polinòmic

Definició de planaritat

Un graf és planar si es pot dibuixar en el pla sense creuaments d'arestes.

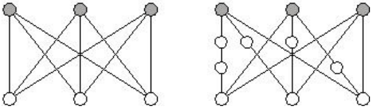
Aplicacions: disseny de circuits, gràfics.



K_5 $K_{3,3}$

Teorema de Kuratowski

Un graf és planar si i només si no conté cap subgraf homeomorf a K_5 o $K_{3,3}$.



$K_{3,3}$ i graf homeomorf

Problemes NP-complets

Teorema de Kuratowski

Un graf és planar si i només si no conté cap subgraf homeomorf a K_5 o $K_{3,3}$.

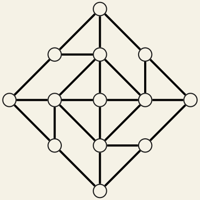
Test de planaritat

- **Força bruta**: $O(n^6)$
  - Contreure arestes de grau 2
  - Comprovar si cada subconjunt de 5 vèrtexs és un K_5
  - Comprovar si cada subconjunt de 6 vèrtexs és un $K_{3,3}$
- **Eficient**: $O(n)$
  - Aplicar DFS

Problemes NP-complets

3-COLOR $\leq^p$ 3-COLOR-PL

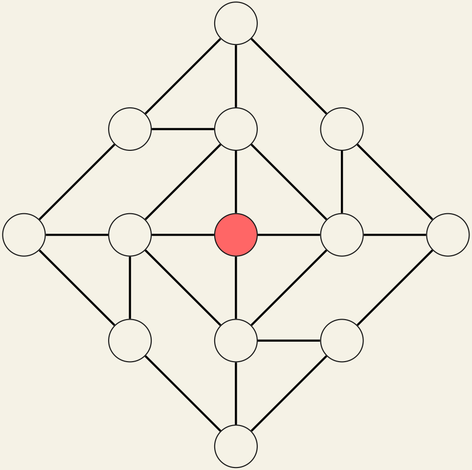
Donat un graf G, considerem un dibuix de G, possiblement amb creuaments d'arestes. Cada creuament el substituïm pel giny W:



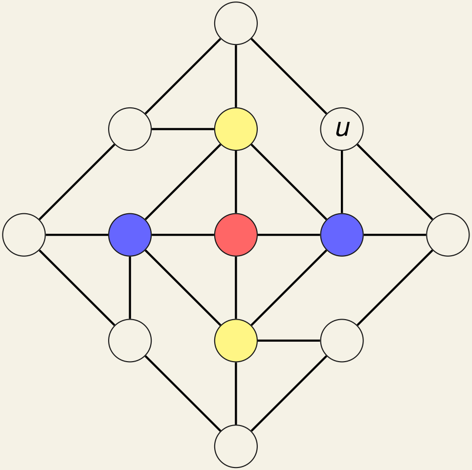
W té propietats interessants:

1. en tota 3-coloració de W, els extrems oposats tenen el mateix color
2. tota coloració dels extrems on els oposats tenen el mateix color es pot estendre a una 3-coloració de W

Problemes NP-complets



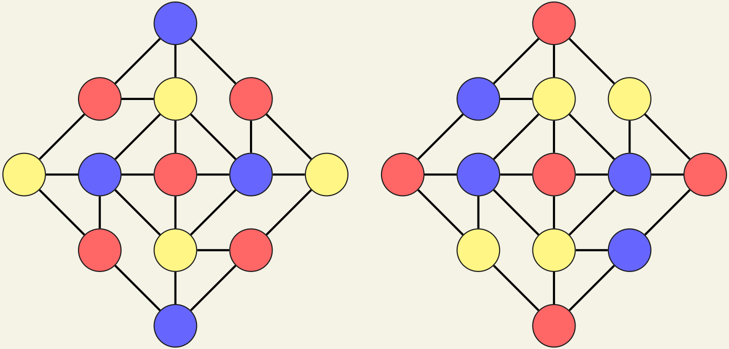
Hi ha dos colors disponibles per al vèrtex u.



Hi ha dos colors disponibles per al vèrtex u.

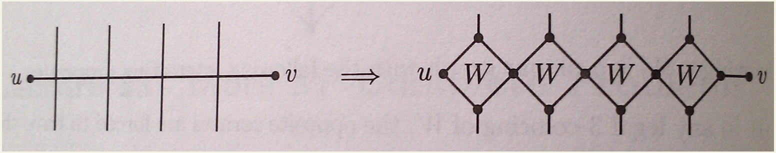
Problemes NP-complets

Això dóna lloc a dues coloracions (fins a isomorfisme):



És fàcil comprovar que es compleixen les propietats (1) i (2).

El graf que resulta fent les substitucions

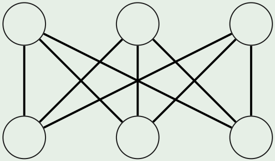


en el dibuix de G

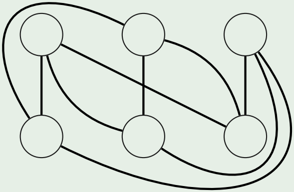
- és planar i
- és 3-colorable si i només si G és 3-colorable

Exemple

Suposem que tenim el graf $K_{3,3}$ com a entrada de 3-COLOR:

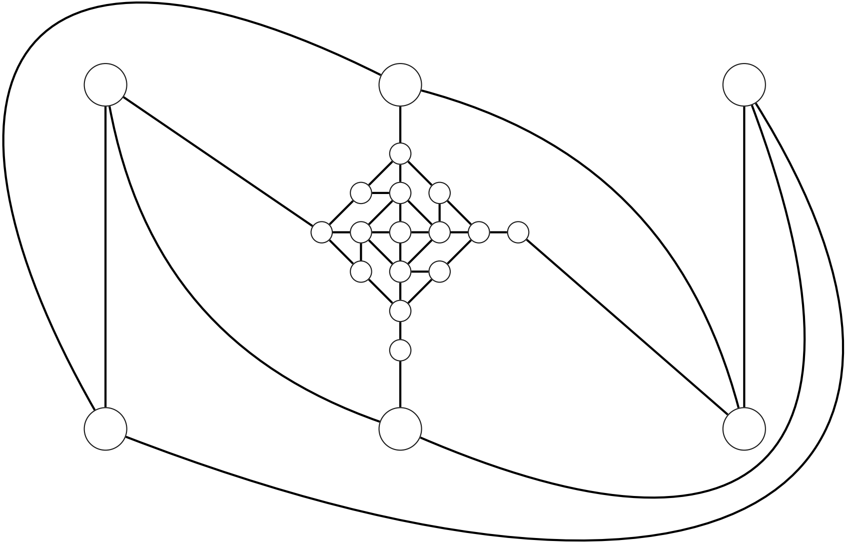


Però considerem el dibuix següent que redueix els creuaments a un:

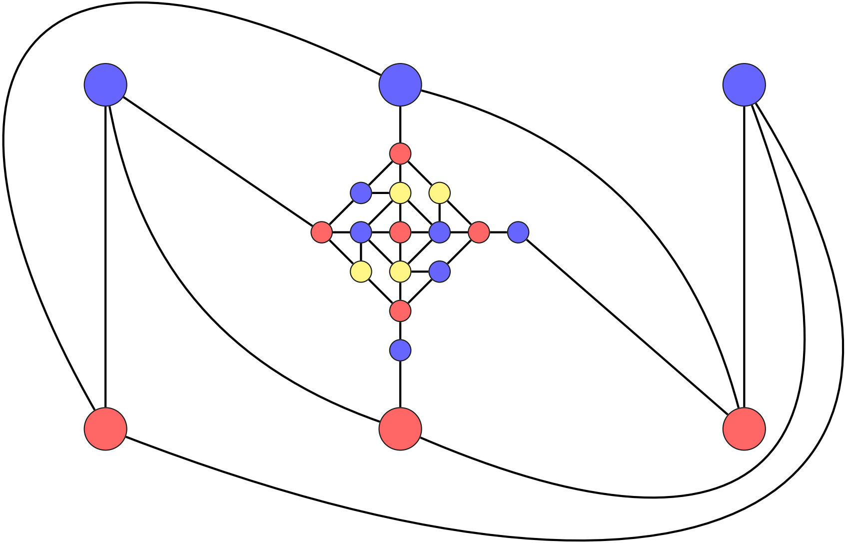


Problemes NP-complets

Una 3-coloració de $K_{3,3}$ indueix una 3-coloració de (i a l'inrevés):



Una 3-coloració de $K_{3,3}$ indueix una 3-coloració de (i a l'inrevés):



Corol·lari

3-COLOR-PL és NP-complet.

Per tant, tenim:

- k-COLOR-PL $\in \mathrm{P}$ per a $k \leq 2$
- 3-COLOR-PL és NP-complet
- k-COLOR-PL $\in \mathrm{P}$ per a $k \geq 4$

Corol·lari

3-COLOR-PL és NP-complet.

Per tant, tenim:

- k-COLOR-PL $\in \mathrm{P}$ per a $k \leq 2$
- 3-COLOR-PL és NP-complet
- k-COLOR-PL $\in \mathrm{P}$ per a $k \geq 4$
  (pel teorema dels 4 colors)

Problemes NP-complets

Fins ara hem vist l'arbre de reduccions següent.