

Tema 7. Complexitat

Estructures de Dades i Algorismes

FIB

Transparències d'Antoni Lozano
(amb edicions menors de Gabriel Valiente)

Q1 2017–2018

1 Classes

- Problemes decisionals
- Temps polinòmic i exponencial
- Indeterminisme

2 Reduccions

- Concepte de reducció
- Exemple de reducció
- Propietats

3 NP-completesa

- Teoria de la NP-completesa
- Problemes NP-complets

1 Classes

- Problemes decisionals
- Temps polinòmic i exponencial
- Indeterminisme

2 Reduccions

- Concepte de reducció
- Exemple de reducció
- Propietats

3 NP-completesa

- Teoria de la NP-completesa
- Problemes NP-complets

- 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 les eines més bàsiques per **classificar** els problemes segons la seva complexitat.

Per classificar els problemes, considerarem les seves versions decisionals.

Definició

Un **problema decisional** és un problema en el qual la sortida és **sí** o **no**

Equivalentment, un problema és decisional quan s'ha de determinar si l'**entrada** (també anomenada **instància**) satisfà o no una certa propietat.

Molts problemes vistos fins ara són decisionals:

- **connectivitat**: donat un graf, decidir si és connex.
- **3-colorabilitat**: donat un graf, decidir si és 3-colorable.
- **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 .

o es poden transformar en decisionals:

- **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 longitud màxima k .

Certes versions decisionals d'alguns problemes no tenen gaire sentit.

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 n'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 FALS

si no

retornar CERT

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, \dots, r_k , amb $k \leq n$, decidir si es poden col·locar n reines en un tauler $n \times n$ sense que cap n'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

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

execucions de l'algorisme que el resol.

Altres problemes decisionals:

- ➊ **primalitat**: donat un natural, decidir si és primer.
- ➋ **viatjant de comerç**: 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 les ciutats i torni a l'origen.
- ➌ **aturada fitada**: donats un natural k , un programa P i una entrada per al programa e , determinar si quan executem P amb entrada e , el programa s'atura en com a molt k passos.

Problemes decisionals

Un problema decisional es representa formalment mitjançant un conjunt: el conjunt de les entrades amb resposta **sí**.

Per exemple, el conjunt que representa el problema de connectivitat és el conjunt dels grafs connexos.

I el conjunt que representa el problema de primalitat és el conjunt dels nombres primers.

En general, si T és una propietat que els elements d'un conjunt d'entrades E poden tenir o no, i ens plantegem el següent problema decisional:

Problema A

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

aleshores podem descriure formalment A com el conjunt:

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

Problemes decisionals

Les entrades dels problemes pertanyeran a certs **dominis de dades** (és a dir, conjunts que podem representar en un ordinador).

Per exemple:

- els nombres naturals
- les tuples de naturals
- els grafs
- els dags amb pesos en els arcs
- les 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 de x** , representada amb $|x|$, és el nombre de símbols necessaris per a codificar 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 així:

Primalitat

Donat un natural x , determinar si x és primer.

O bé formalment com el conjunt de les entrades positives:

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

Un exemple de funció de mida per als naturals és la que compta el nombre de dígitos de la representació binària:

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

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 així:

Primalitat

Donat un natural x , determinar si x és primer.

O bé formalment com el conjunt de les entrades positives:

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

Un exemple de funció de mida per als naturals és la que compta el nombre de dígit de la representació binària:

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

Problemes decisionals

Ara que ja podem descriure els problemes com a objectes matemàtics, els podem agrupar en classes en funció de la seva complexitat.

- Considerarem classes de problemes segons els recursos necessaris per resoldre'ls.
- Una classe agrupa problemes, de la mateixa manera que un problema agrupa entrades.
- Cal distingir entre tres nivells d'abstracció:
 - Les **entrades**
Per exemple, les seqüències d'enters
 - Els **problemes**: conjunts d'entrades
Per exemple, les seqüències d'enters ordenades
 - Les **classes**: conjunts de problemes
Per exemple, els que podem resoldre en temps lineal

Temps polinòmic i exponencial

Suposem que $t : \mathbb{N} \rightarrow \mathbb{N}$ 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:

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

Diem que un problema decisonal A és decidable en temps t si existeix un algorisme de cost t que el decideix (el resol); és a dir, si existeix $\mathcal{A} : E \rightarrow \{0, 1\}$ de cost t tal que, per a tot $x \in E$:

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

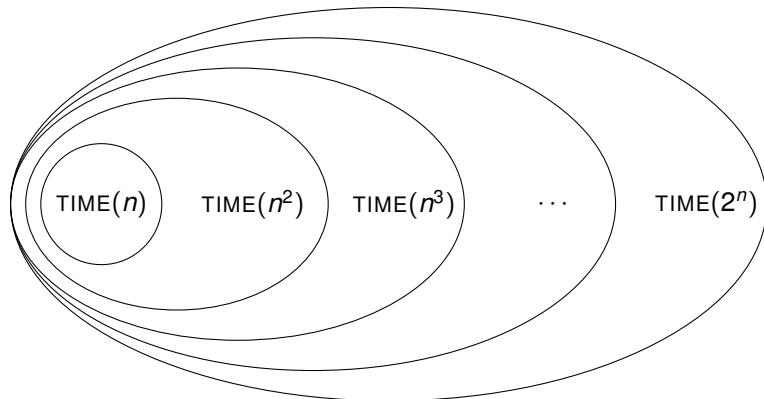
$$x \notin A \Rightarrow \mathcal{A}(x) = 0$$

Temps polinòmic i exponencial

Classe $\text{TIME}(t)$

Donada una funció $t : \mathbb{N} \rightarrow \mathbb{N}$, agrupem els problemes decidibles en temps t :

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



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

En el tema 1 havíem vist les dues taules següents que assenyalen diferències quantitatives 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

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	$N_5 + 4.19$	$N_5 + 6.29$

Classe P

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

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

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

P són els problemes que podem resoldre amb un algorisme polinòmic.

Classe EXP

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

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

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

EXP són els problemes que podem resoldre amb un algorisme exponencial.

Temps polinòmic i exponencial

Es considera que els problemes de la classe P són **tractables**, mentre que els de la classe EXP són **intractables**.

Exemples

- CONNECTIVITAT $\in P$
- ACCESSIBILITAT $\in P$
- PRIMALITAT $\in P$
- CAMÍ CURT $\in P$
- 2-COLORABILITAT $\in P$
- 3-COLORABILITAT $\in EXP$ (no se sap si és a P)
- VIATJANT $\in EXP$ (no se sap si és a P)
- ATURADA FITADA $\in EXP$ (i se sap que no és a P)

Teorema

$P \subsetneq EXP$.

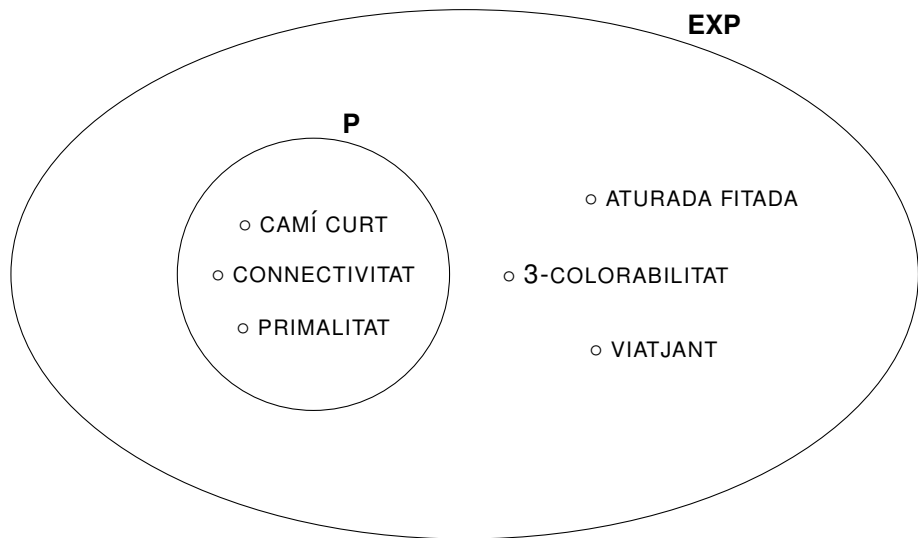
La inclusió estricta del teorema es pot dividir en dues parts:

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

$$P = \bigcup_{k \geq 0} \text{TIME}(n^k) \subseteq \bigcup_{k \geq 0} \text{TIME}(2^{n^k}) = EXP$$

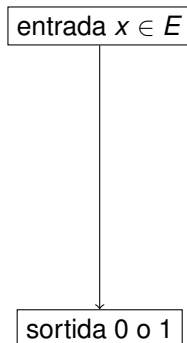
② $P \neq EXP$. La demostració està fora de l'abast de l'assignatura.

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 $\mathcal{A} : E \rightarrow \{0, 1\}$ per a un conjunt de dades E es pot veure com un camí:



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

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

TRIAR(x)

que retorna un nombre y entre 0 i x .

Aleshores:

- \mathcal{A} comença el càlcul de manera determinista fins la primera instrucció TRIAR.
- Per a cada valor retornat per TRIAR, el càlcul es divideix en diferents branques amb el valor corresponent.
- Diem que \mathcal{A} retorna 1 si ho fa en **alguna** de les branques de l'arbre de càlcul.

Exemple: compostos

El problema

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

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

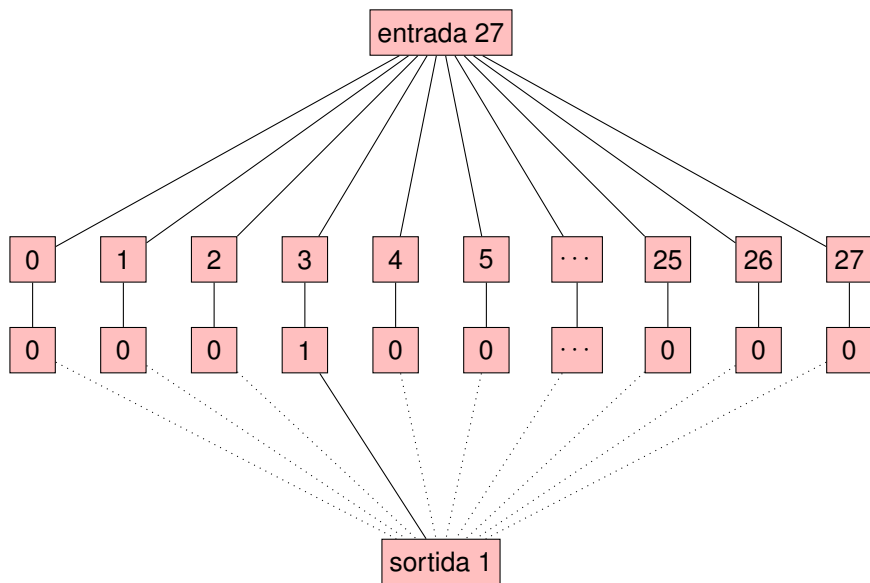
entrada x

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

si $1 < y < x$ **i** y divideix x **llavors**

retornar 1

retornar 0



- 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.
La mida dels testimonis no és més gran que la de l'entrada: $|y| \leq |x|$
 - Un algorisme polinòmic que, donat un y , **verifica** si y divideix x .
- Hi ha molts problemes per als quals hi ha testimonis curts, que es poden verificar en temps polinòmic.

Exemple: 3-colorabilitat

El problema

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

també té un algorisme exhaustiu de temps exponencial

entrada $G = (V, E)$

$n \leftarrow |V|$

per a cada tupla (c_1, \dots, c_n) on $\forall i \leq n \ c_i \in \{0, 1, 2\}$

si (c_1, \dots, 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

$c_i \leftarrow \text{TRIAR}(2)$

si (c_1, \dots, c_n) és una 3-coloració de G **llavors**

retornar 1

si no

retornar 0

La **definició formal** dels algorismes polinòmics indeterministes separa:

- el càlcul del testimoni i
- el càlcul determinista.

Decidibilitat en temps polinòmic indeterminista

Un problema decisional A definit sobre un conjunt d'entrades E es diu que és **decidable en temps polinòmic indeterminista** si existeix

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

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

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

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

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

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

- 1 que les entrades positives de A tenen testimonis de mida polinòmica i que les entrades negatives de A no tenen testimonis de mida polinòmica
(cal indicar quins són els testimonis)
- 2 que els testimonis es poden verificar en temps polinòmic
(cal trobar un verificador)

Compostos

Considerem el problema

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

- 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

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

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 $C = (c_1, c_2, \dots, c_n)$, on $n = |V|$ i $c_i \in \{0, 1, 2\}$ 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

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

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

Classe NP

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

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

Com es relaciona NP amb P i EXP?

Diferència fonamental entre P i NP:

- els testimonis dels problemes de P es poden **trobar** en temps polinòmic
- els testimonis dels problemes de NP es poden **verificar** en temps polinòmic

Quadrats perfectes i compostos

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

2 i 3-colorabilitat

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

Teorema

$P \subseteq NP$.

Demostració

Tot algorisme determinista també és indeterminista (però no fa ús de la instrucció TRIAR).

Vist d'una altra manera, per a tot $A \in P$, podem crear verificadors \mathcal{V} tals que

$$\mathcal{V}(x, y) = 1 \Leftrightarrow x \in A$$

independentment de y . Per tant, $A \in NP$.

Diferència entre NP i EXP:

- els problemes de NP tenen testimonis verificables en temps polinòmic
- els problemes d'EXP poden tenir testimonis exponencialment llargs

Per resoldre els problemes de NP hi ha un algorisme estàndard exponencial que cerca un testimoni i el verifica

Teorema

$NP \subseteq EXP$.

Demostració

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

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

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

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

entrada x

per a tot y tal que $|y| \leq p(|x|)$

si $\mathcal{V}(x, y) = 1$ **llavors**

retornar 1

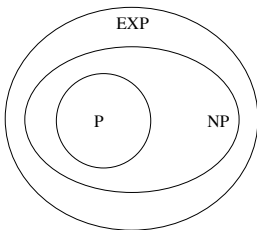
retornar 0

És fàcil veure que l'algorisme anterior és exponencial i decideix A .
Per tant, $A \in EXP$.

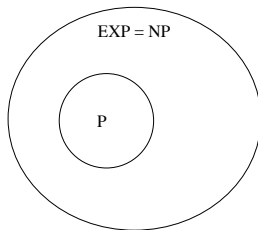
Indeterminisme

- No se sap si $P = NP$.
- Podem assegurar que $P \neq NP$ o $NP \neq EXP$ (perquè se sap que $P \neq EXP$).
- Per tant, hi ha tres possibilitats:

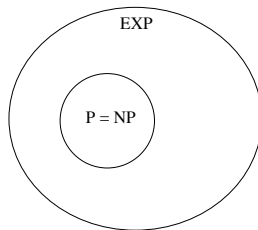
(a)



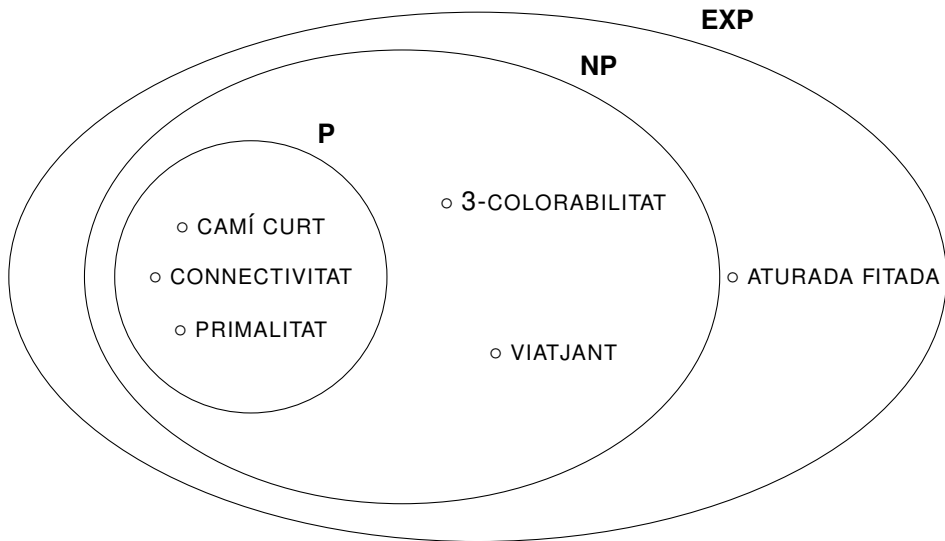
(b)



(c)



Prendrem (a) com a hipòtesi de treball.



1 Classes

- Problemes decisionals
- Temps polinòmic i exponencial
- Indeterminisme

2 Reduccions

- Concepte de reducció
- Exemple de reducció
- Propietats

3 NP-completesa

- Teoria de la NP-completesa
- Problemes NP-complets

Concepte de reducció

Si hem de resoldre un problema A i disposem d'un algorisme per resoldre B , podem usar-lo per resoldre A ?

Reduccions

Siguin A i B dos problemes decisionals amb conjunts d'entrades E i E' , resp.

Diem que *A es redueix a B en temps polinòmic* si existeix un algorisme polinòmic $\mathcal{F} : E \rightarrow E'$ 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}).
Diem que \mathcal{F} és una reducció polinòmica de A a B .

La idea és que si A es redueix a B , podem usar B per resoldre A :
composem la reducció de A a B amb l'algorisme per a B .

Si l'algorisme per a B és polinòmic, aquest algorisme per a A també ho és!

Exemple de reducció

Problema de la MOTXILLA (o KNAPSACK)

Donada una motxilla amb capacitat C , un valor V i n objectes amb

- pesos p_1, p_2, \dots, p_n
- i valors v_1, v_2, \dots, v_n

trobar una selecció $S \subseteq \{1, \dots, n\}$ dels objectes

- que no superi la capacitat de la motxilla: $\sum_{i \in S} p_i \leq C$
- amb valor almenys V : $\sum_{i \in S} v_i \geq V$

Problema d'INEQUACIONS LINEALS 0-1

Donada una matriu A de $m \times n$ enters i donat un vector b de m enters, determinar si existeix un vector $x \in \{0, 1\}^n$ tal que $Ax \geq b$.

Exemple de reducció

Suposem que volem resoldre MOTXILLA,
i tenim un programa per a INEQUACIONS LINEALS 0-1 (p. ex. Maple).

Donada una entrada de MOTXILLA:

- capacitat C
- valor mínim V
- nombre d'objectes n
- pesos p_1, p_2, \dots, p_n
- valors v_1, v_2, \dots, v_n ,

considerem variables 0-1 x_i que signifiquen “agafo l' i -èsim objecte”.

Lavors hi ha un subconjunt d'objectes per a l'entrada de MOTXILLA si i
només si el següent sistema d'inequacions lineals té solució:

$$\sum_{i=1}^n p_i x_i \leq C \quad \sum_{i=1}^n v_i x_i \geq V$$

Exemple de reducció

Suposem que volem resoldre MOTXILLA, i tenim un programa per a INEQUACIONS LINEALS 0-1 (p. ex. Maple).

Donada una entrada de MOTXILLA:

- capacitat C
- valor mínim V
- nombre d'objectes n
- pesos p_1, p_2, \dots, p_n
- valors v_1, v_2, \dots, v_n ,

considerem variables 0-1 x_i que signifiquen “agafo l' i -èsim objecte”.

Lavors hi ha un subconjunt d'objectes per a l'entrada de MOTXILLA si i només si el següent sistema d'inequacions lineals té solució:

$$\sum_{i=1}^n (-p_i)x_i \geq -C \qquad \sum_{i=1}^n v_i x_i \geq V$$

Exemple de reducció

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int C, V, n;
    cin >> C >> V >> n;
    vector<int> p(n), v(n);
    for (int i = 0; i < n; ++i) cin >> p[i];
    for (int i = 0; i < n; ++i) cin >> v[i];

    // Generem el sistema (entrada per a INEQUACIONS LINEALS 0-1)
    cout << "evalb(nops({isolve}){" << endl;
    for (int i = 0; i < n; ++i)
        cout << "0 <= x" << i << ", x" << i << " <= 1, " << endl;

    for (int i = 0; i < n; ++i) cout << " + (" << -p[i] << ") * x" << i;
    cout << " >= " << -C << ", " << endl;

    for (int i = 0; i < n; ++i) cout << " + (" << v[i] << ") * x" << i;
    cout << " >= " << V << endl;

    cout << "})) > 0);" << endl;
}
```


Exemple de reducció

Exemple d'entrada per a MOTXILLA

```
3
6
3
1 2 3
2 4 5
```

Entrada generada per a INEQUACIONS LINEALS 0-1

```
evalb(nops({isolve}({
0 <= x0, x0 <= 1,
0 <= x1, x1 <= 1,
0 <= x2, x2 <= 1,
+ (-1) * x0 + (-2) * x1 + (-3) * x2 >= -3,
+ (2) * x0 + (4) * x1 + (5) * x2 >= 6
}))) > 0);
```

Podem resoldre MOTXILLA així: donada una entrada e de MOTXILLA, generem el sistema d'inequacions, i el passem al programa que el resol. Llavors:

- Si el sistema té solució, $e \in \text{MOTXILLA}$
- Si el sistema no té solució, $e \notin \text{MOTXILLA}$

Com que el programa per generar el sistema d'inequacions funciona en temps polinòmic, tenim que $\text{MOTXILLA} \leq^p \text{INEQUACIONS LINEALS}$

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

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

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 $\mathcal{G} \circ \mathcal{F}$ demostra que $A \leq^p C$.

Corol·lari

Les reduccions formen, doncs, un preordre.

Tancament de P per reduccions

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

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 $\mathcal{F} \circ \mathcal{B}$ és un algorisme polinòmic per a A .

1 Classes

- Problemes decisionals
- Temps polinòmic i exponencial
- Indeterminisme

2 Reduccions

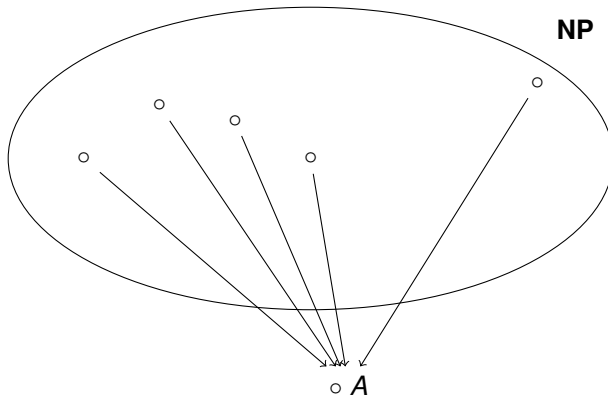
- Concepte de reducció
- Exemple de reducció
- Propietats

3 NP-completesa

- Teoria de la NP-completesa
- Problemes NP-complets

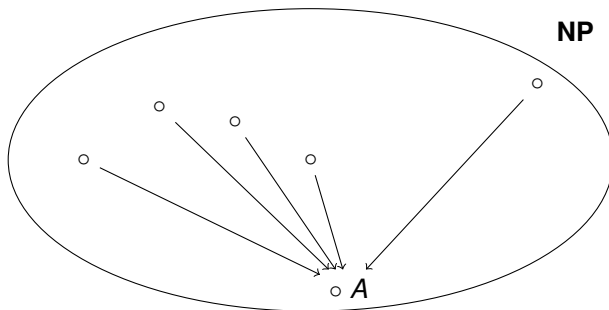
Definició

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



Definició

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



Proposició

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

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

\Leftarrow Sigui $A \in P$.

Com que A és NP-complet, llavors per a tot $B \in NP$, $B \leq^p A$.

Si $A \in P$, pel tancament de P per reduccions, llavors $B \in P$.

Per tant $NP \subseteq P$. Com que ja sabem que $P \subseteq NP$, finalment $P = NP$.

Però... es coneix cap problema NP-complet?

Fórmules booleanes

- Una **fórmula booleana** és una expressió formada amb connectives \vee (disjunció), \wedge (conjunció) i \neg (negació) i variables booleanes

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.

Per exemple,

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

Satisfactibilitat

Una fórmula booleana és **satisfactible** si hi ha una assignació de valors cert/fals a les variables que fa la fórmula certa.

Per exemple,

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

és satisfactible perquè $x = 1, y = 0, z = 0$ la satisfà: $F(1, 0, 0) = 1$.

Definim

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

$$\text{CNF-SAT} = \{ F \mid F \text{ és una fórmula booleana en CNF satisfactible} \}.$$

Teorema de Cook-Levin (1971)

SAT i CNF-SAT són NP-complets.

Satisfactibilitat

Una fórmula booleana és **satisfactible** si hi ha una assignació de valors cert/fals a les variables que fa la fórmula certa.

Per exemple,

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

és satisfactible perquè $x = 1, y = 0, z = 0$ la satisfà: $F(1, 0, 0) = 1$.

Definim

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

$$\text{CNF-SAT} = \{ F \mid F \text{ és una fórmula booleana en CNF satisfactible} \}.$$

Teorema de Cook-Levin (1971)

SAT i CNF-SAT són NP-complets.

Demostrarem que CNF-SAT és NP-complet.

Cal veure:

- 1 CNF-SAT \in NP
- 2 CNF-SAT és NP-difícil

(1) CNF-SAT \in NP

- Els **testimonis** són assignacions booleanes a variables que satisfan F
- En tota codificació raonable d'una fórmula F en CNF 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:

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

Teoria de la NP-completesa

La idea principal de la demostració que CNF-SAT és NP-difícil és que qualsevol algorisme es pot implementar eficientment amb un circuit amb portes AND, OR i NOT (= fórmula en CNF) si l'entrada es codifica en binari.

Lema

Donat un algorisme $\mathcal{A} : E \rightarrow \{0, 1\}$ amb cost en temps polinòmic, podem trobar en temps polinòmic una fórmula booleana en CNF $F_{\mathcal{A}}$ tal que:

$$F_{\mathcal{A}}(x) = 1 \Leftrightarrow \mathcal{A}(x) = 1 \quad \text{per a tot } x \in E$$

(2) CNF-SAT és NP-difícil.

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

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

Sigui $\mathcal{V}_x(y)$ un algorisme que comprova $|y| \leq q(|x|)$ i $\mathcal{V}(x, y) = 1$. Llavors,

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

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

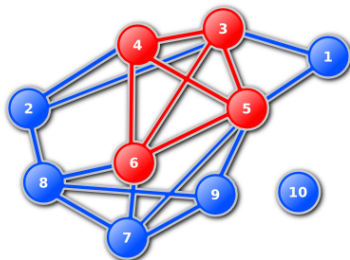
Problemes NP-complets

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

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

(un graf és **complet** si conté totes les arestes entre els seus vèrtexs).

Donat el graf G



podem observar que

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

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 $|C| \leq |(G, k)| = p(|(G, k)|)$.
- **Verifiquem** en temps polinòmic si un conjunt C de vèrtexs és testimoni: tot parell de vèrtexs de C ha de formar una aresta en G ($\leq n^2$ comprovacions).

CLICA és NP-difícil

Demostrarem que $\text{CNF-SAT} \leq^p \text{CLICA}$.

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

Podem expressar aquesta propietat en general.

Proposició

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

CLICA és NP-difícil

Demostrarem que $\text{CNF-SAT} \leq^p \text{CLICA}$.

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

Podem expressar aquesta propietat en general.

Proposició

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

CNF-SAT \leq^p CLICA

Sigui F una fórmula booleana en CNF amb:

- clàusules C_1, \dots, C_m
- literals l_1, \dots, l_r

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

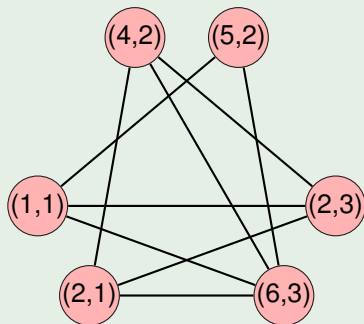
- $V = \{(i, j) \mid l_i \text{ apareix a } C_j\}$
(Els vèrtexs representen ocurrències de literals en clàusules.)
- $E = \{ \{(i, j), (k, l)\} \mid j \neq l \wedge \neg l_i \neq l_k \}$
(Les arestes representen parells de literals en clàusules que poden ser certs alhora.)

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



En general, tenim que

$$F \in \text{CNF-SAT} \Leftrightarrow (G, m) \in \text{CLICA}.$$

- ⇒ Sigui α una assignació que satisfà F . Llavors, hi ha m literals que α fa certs alhora i, per tant, formen un subgraf complet en G .
- ⇐ Si G té un subgraf complet de m vèrtexs, cada vèrtex ha de correspondre a una clàusula diferent. Per tant, fent cert un literal de cada clàusula alhora satisfem F . O sigui que F és satisfactible.

Definicions

- H és **subconjunt independent** si els vèrtexs no són adjacents dos a dos.
- H és **recobriment de vèrtexs** si té un extrem de tota aresta del graf.

Exercici

Donats els problemes següents:

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

demostru

- 1 $CLICA \leq^p SI$
- 2 $SI \leq^p RV$

Molts problemes 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 per als 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 decidable en temps polinòmic amb l'algorisme següent:

entrada F

si F conté dos literals contradictoris llavors

retornar FALS

si no

retornar CERT

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 **decidable en temps polinòmic** amb l'algorisme següent:

entrada F

si F conté dos literals contradictoris **llavors**

retornar FALS

si no

retornar CERT

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 decidable 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 **decidable 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

$$\begin{aligned} F(x, y, z) = & (\neg x \Rightarrow y) \wedge (z \Rightarrow x) \wedge (x \Rightarrow y) \wedge (y \Rightarrow \neg z) \\ & \wedge (\neg y \Rightarrow x) \wedge (\neg x \Rightarrow \neg z) \wedge (\neg y \Rightarrow \neg x) \wedge (z \Rightarrow \neg y) \end{aligned}$$

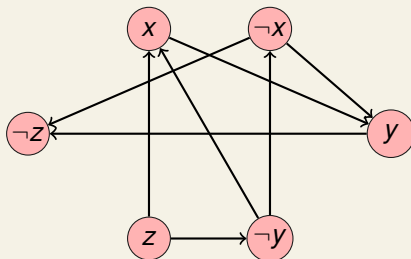
que es basen en les equivalències

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

La fórmula booleana amb implicacions

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

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



Lema

F és insatisfactible si i només si $\exists x$ tal que G té camins de x a $\neg x$ i de $\neg x$ a x .

Satisfactibilitat 3-fitada (3-SAT)

Donada un fórmula booleana en CNF F de n variables amb ≤ 3 literals per clàusula, determinar si és satisfactible.

3-SAT és NP-complet.

Per demostrar-ho, cal provar:

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

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

CNF-SAT \leq^p 3-SAT

El mètode següent transforma una fórmula booleana en CNF en una altra de satisfactible en 3-CNF.

Donada una fórmula booleana F en CNF,

- 1 Sigui F' la fórmula *cert*
- 2 Per a cada clàusula $C = (a_1 \vee \dots \vee a_k)$ de F :
 - si $k \leq 3$, afegir C a F'
 - si $k > 3$, afegir a F' la clàusula

$$(a_1 \vee a_2 \vee z_2) \wedge (\neg z_2 \vee a_3 \vee z_3) \wedge (\neg z_3 \vee a_4 \vee z_4) \dots (\neg z_{k-2} \vee a_{k-1} \vee a_k)$$

on z_2, \dots, z_{k-2} 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

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

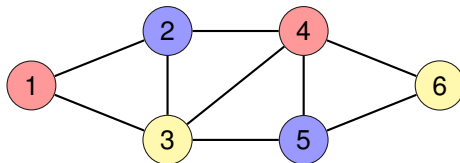
- És evident que si C és certa amb una assignació α , C' es pot satisfer amb α i valors adequats de z_1 i z_2 .
- Si C' és certa amb una assignació β , algun a_i serà cert i C serà certa amb β .

Definició

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

$$\chi : V \rightarrow \{1, \dots, k\}$$

tal que $\chi(u) \neq \chi(v)$ per a $\{u, v\} \in E$. La funció χ és una **k-coloració**.



3-coloració

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:

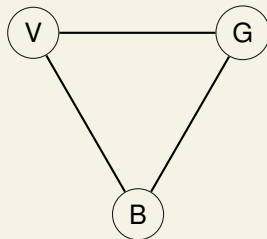
- 1-COLOR
- 2-COLOR

CNF-SAT \leq^p 3-COLOR

Sigui F una fórmula booleana en CNF.

Construïrem un graf G que serà 3-colorable si i només si F és satisfactible.

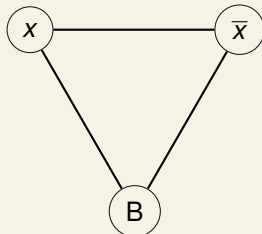
- Hi haurà 3 vèrtexs especials anomenats V, G, B.



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.



Problemes NP-complets

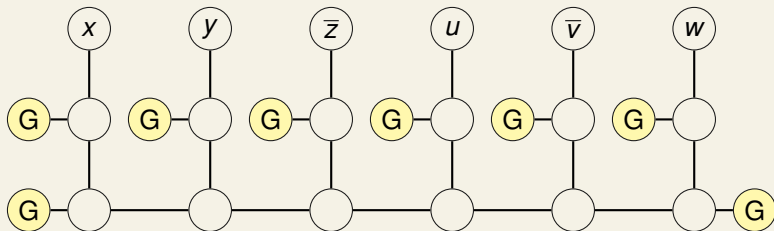
Per cada clàusula, afegim un subgraf.

Suposem que el nombre de literals de la clàusula és parell.

Lavors per exemple per a la clàusula

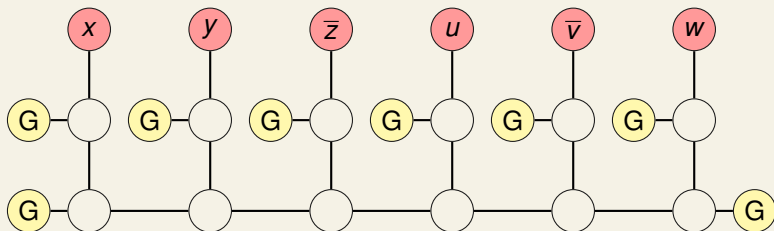
$$(x \vee y \vee \bar{z} \vee u \vee \bar{v} \vee w).$$

afegim



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.

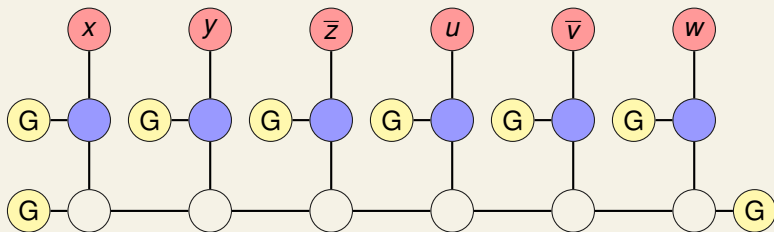
Si tots els de dalt tenen color vermell...



...no podem completar la 3-coloració.

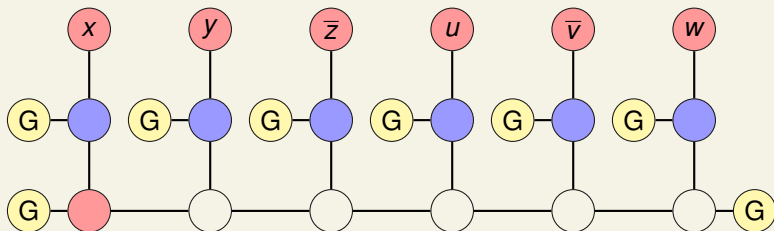
Problemes NP-complets

Si tots els de dalt tenen color vermell...



...no podem completar la 3-coloració.

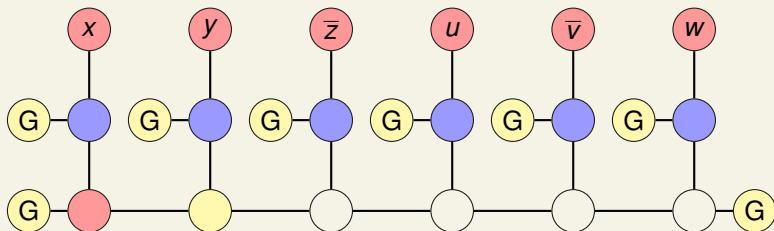
Si tots els de dalt tenen color vermell...



...no podem completar la 3-coloració.

Problemes NP-complets

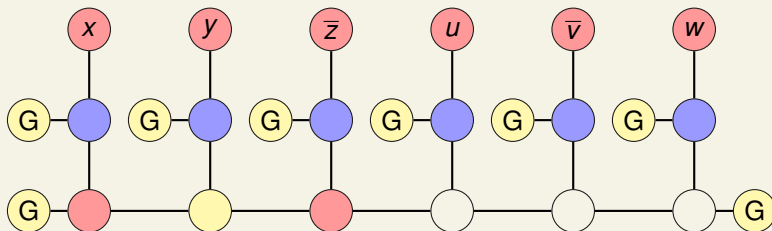
Si tots els de dalt tenen color vermell...



...no podem completar la 3-coloració.

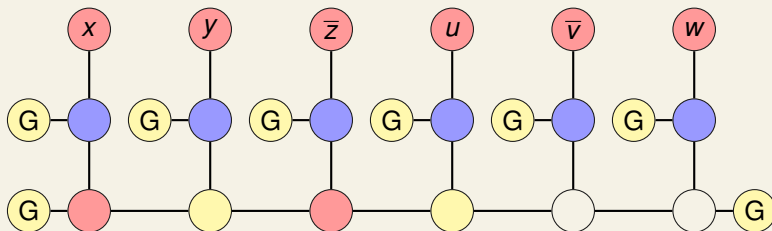
Problemes NP-complets

Si tots els de dalt tenen color vermell...



...no podem completar la 3-coloració.

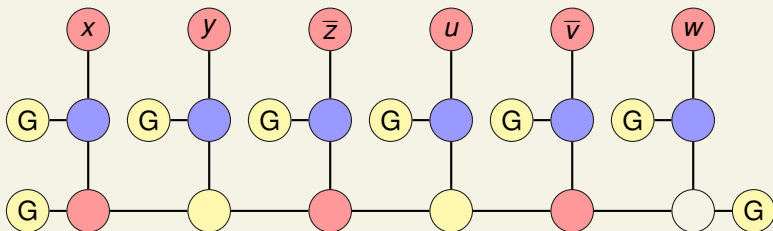
Si tots els de dalt tenen color vermell...



...no podem completar la 3-coloració.

Problemes NP-complets

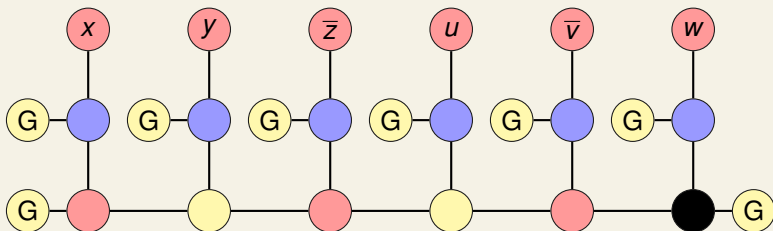
Si tots els de dalt tenen color vermell...



...no podem completar la 3-coloració.

Problemes NP-complets

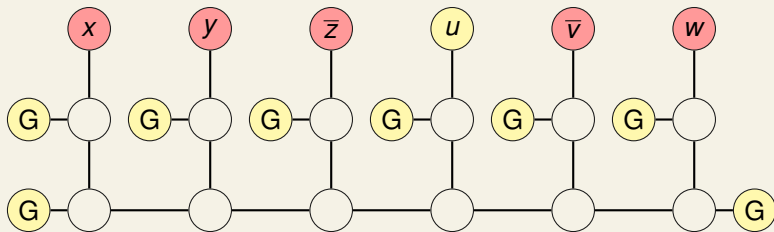
Si tots els de dalt tenen color vermell...



...no podem completar la 3-coloració.

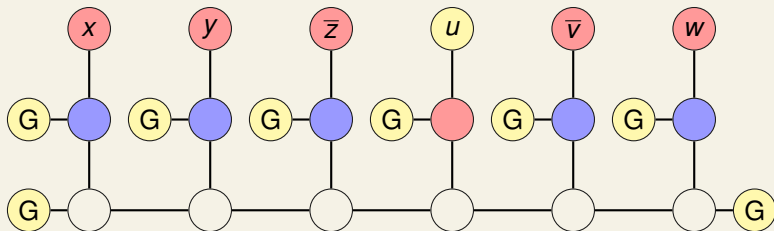
Problemes NP-complets

Si almenys un de dalt té color groc...



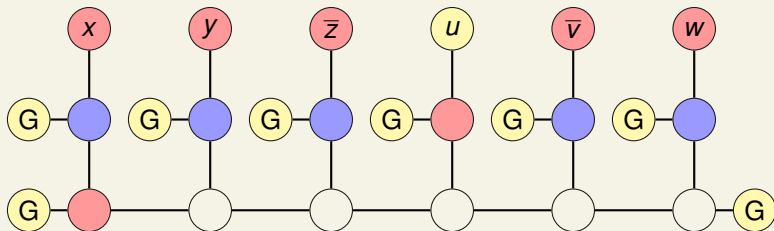
...podem obtenir una 3-coloració global.

Si almenys un de dalt té color groc...



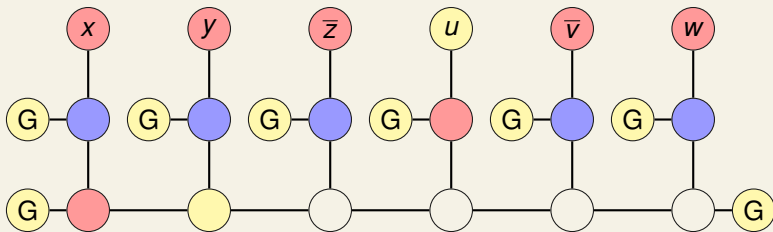
...podem obtenir una 3-coloració global.

Si almenys un de dalt té color groc...



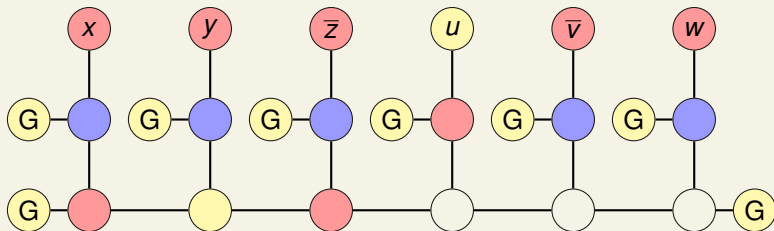
...podem obtenir una 3-coloració global.

Si almenys un de dalt té color groc...



...podem obtenir una 3-coloració global.

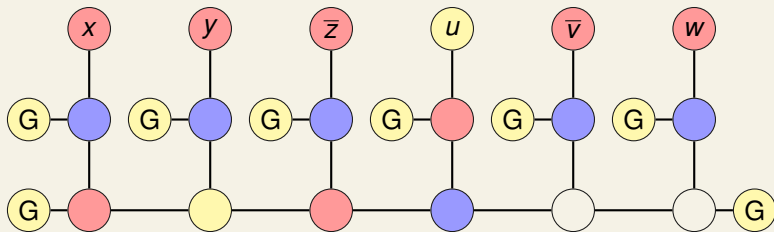
Si almenys un de dalt té color groc...



...podem obtenir una 3-coloració global.

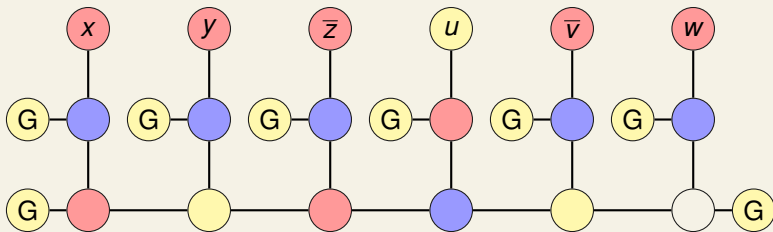
Problèmes NP-complets

Si almenys un de dalt té color groc...



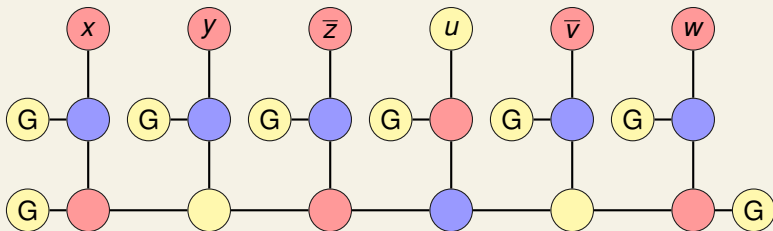
...podem obtenir una 3-coloració global.

Si almenys un de dalt té color groc...



...podem obtenir una 3-coloració global.

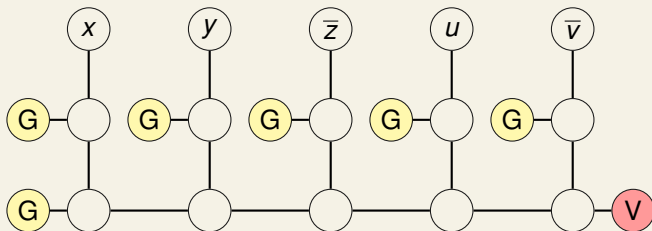
Si almenys un de dalt té color groc...



...podem obtenir una 3-coloració global.

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

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



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

F és satisfactible $\Leftrightarrow G$ és 3-colorable.

Si una assignació booleana satisfà F , pintem de color groc els literals que són certs, i de color vermell els que no ho són.

Recíprocament, si G és 3-colorable, llavors cada clàusula ha de tenir almenys un literal de color groc. Aquests són els que fem certs.

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

Proposició

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

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

Corol·lari

Per a tot $k > 3$, $k\text{-COLOR}$ és NP-complet.

Per tant, tenim:

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

Problemes NP-complets

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

Proposició

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

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

Corol·lari

Per a tot $k > 3$, $k\text{-COLOR}$ és NP-complet.

Per tant, tenim:

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

Fins ara hem vist l'arbre de reduccions següent.

