Problemes tractables i intractables

Es tracta d'estudiar les característiques d'allò que **és** computable.

No tots els problemes costen igual de resoldre.

Es pot fer una classificació dels problemes resolubles en funció de la seua **complexitat**?

Problema: com mesurem els costs? amb MTs? de quin tipus?

F. J. Ferri, **Dept. d'Informàtica** *Universitat de València*
NP-COMPLETITUD

Problemes tractables i intractables

Per tal de fer l'estudi independent del model de computació farem només la distinció entre problemes **tractables** (cost polinòmic) i **intractables** (cost exponencial).
Tècnicament parlarem de costs polinòmics enfront de costs polinòmics indeterministes (que en la pràctica es corresponen amb els exponencials).
Es pot distinguir entre tres tipus de problemes:

- problemes per als quals s'ha trobat solució polinòmica.
- problemes per als que s'ha demostrat que no hi ha solució polinòmica.
- problemes per als quals no **s'ha trobat** solució polinòmica.

Costs associats a MT

Siga $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \triangle, F \rangle$ i $w \in \Sigma^\star$ amb $|w| = n$.

complexitat o cost temporal de M ($T_M(n)$): nombre màxim de moviments duts a terme per M amb qualsevol w de talla n com a entrada fins a arribar a una configuració de parada.

complexitat o cost espacial de M ($T_M(n)$): nombre màxim de caselles de la cinta de M utilitzades (escrites o llegides) durant l'acceptació o no de qualsevol cadena w de talla n.

Costs associats a MT

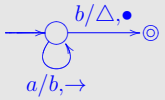
Si les màquines són indeterministes, poden haver diferents comportaments per a una mateixa cadena.

$T_M(n)$ està indefinida per a n si hi ha alguna cadena de longitud n per a la qual M pot no parar.

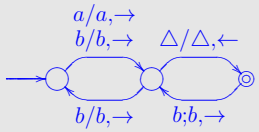
Si $T_{M,x}(n)$ és el **mínim** nombre de moviments que necessita M per acceptar o rebujar la cadena x de talla n, $T_M(n)$ és el **màxim** dels $T_{M,x}(n)$ per a totes les cadenes de talla n.

Costs associats a MT

M_1



M_2



$$T_{M_1}(n) = n$$

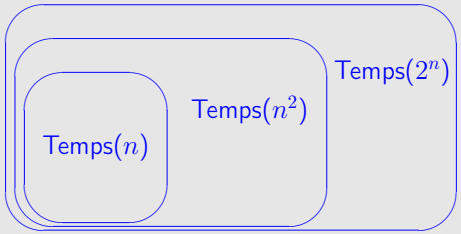
$$T_{M_2}(n) = \begin{cases} n & \text{si } n \text{ parell} \\ \text{indefinit} & \text{si } n \text{ senar} \end{cases}$$

Complexitats dels llenguatges

Es diu que un llenguatge, L, té una complexitat temporal $f(n)$ si és acceptat per una MT determinista la complexitat temporal de la qual és menor o igual que $c \cdot f(n)$ per a alguna constant c i per a tot n llevat d'un nombre finit de valors. Es diu que L pertany a la classe Temps$(f(n))$.

De la mateixa manera es pot definir la complexitat temporal no determinista així com les corresponents complexitats espacials. Les classes corresponents reben els noms de NTemps$(f(n))$, Espai$(f(n))$ i NEspai$(f(n))$.

Complexitats dels llenguatges



Per a que aquestes definicions tinguen sentit caldria especificar exactament el tipus de MT. Es consideraran MT multicinta en tots els casos.

Exemple

Siga L_c el llenguatge format per cadenes capicues de $(a, b)^\star$.

Siga M_1 la MT de una cinta que accepta (decideix) L_c marcant símbols iguals al principi i final de la cadena.

$$T_{M_1}(n) \in O(n^2)$$

Siga M_2 la MT amb dos capçals que decideix L_c comprovant símbols al principi i al final amb els dos capçals.

$$T_{M_2}(n) \in O(n)$$

Exemple

Siga $L_p = \{1^m | m$ és primer$\}$.
Siga M_1 la MT que decideix L_p fent successives divisions per enters inferiors fins que trobe un divisor (la divisió en unari és lineal).

$$T_{M_1}(n) \in O(n^2)$$

Siga M_2 una MT indeterminista que genera un enter qualsevol, calcula el residu de la divisió i si és zero accepta. Aleshores $L_p = L(M_2)$.

$$T_{M_2}(n) \in O(n)$$

Relació entre classes de complexitats

Per ser les MT deterministes un cas particular de les indeterministes:

$$\text{Temps}(f(n)) \subseteq \text{NTemps}(f(n))$$
$$\text{Espai}(f(n)) \subseteq \text{NEspai}(f(n))$$

Com que no es pot accedir a més caselles que moviments s'efectuen:

$$\text{Temps}(f(n)) \subseteq \text{Espai}(f(n))$$
$$\text{NTemps}(f(n)) \subseteq \text{NEspai}(f(n))$$

Relació entre complexitat determinista i indeterminista

Resulta convenient restringir les complexitats definides a les funcions anomenades **construibles** o de **conteig de passos** (step-counting).

> Una funció f s'anomena construible si existeix una MT que arriba a una configuració de parada exactament després de $f(n)$ moviments per a tota cadena de longitud n.

Les funcions "normals" són construibles: n, n^k, 2^n, etc.

Relació entre complexitat determinista i indeterminista

$$\text{NTemps}(f(n)) \subseteq \text{Espai}(f(n)), \text{ si } f \text{ és construible}$$

Si recordem la forma en què se simula una MT indeterminista, l'espai utilitzat en les diferents cintes no pot superar el nombre de moviments $f(n)$ (llevat d'una constant).

Cal també utilitzar una MT que compute f per tal d'avortar branques de computació que no accepten o rebujen la cadena en els primers $f(n)$ moviments.

Relació entre complexitat determinista i indeterminista

Si $L \in \text{NEspai}(f(n))$ i f és construible, aleshores

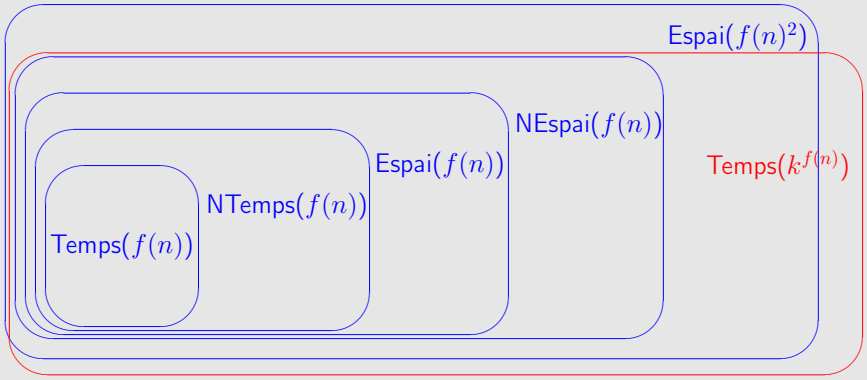
$$\exists k : \begin{cases} L \in \text{Temps}(k^{f(n)}) \\ L \in \text{Espai}(kf(n)^2) \end{cases}$$

Si una MT indeterminista accepta una cadena fent servir $f(n)$ caselles, aleshores el nombre màxim de possibles configuracions que la simulació determinista hauria de provar estarà fitat per una quantitat que depén en particular de $|\Gamma|^{f(n)}$.

La fita espacial quadràtica (teorema de Savitch) s'obté en implementar un procediment recursiu que comprova **totes** les possibles seqüencies de configuracions basat en divideix i venceràs. Aquest procediment implica un arbre de recursió de profunditat $f(n)$ (no es poden repetir configuracions) on en cada nivell s'emmagatzemen configuracions.

Relació entre complexitat determinista i indeterminista

Tot plegat:



Les classes P i NP

$$\mathcal{P} = \bigcup_{k\geq 1} \mathsf{Temps}(n^k)$$
$$\mathcal{NP} = \bigcup_{k\geq 1} \mathsf{NTemps}(n^k)$$

De la mateixa manera es defineixen les classes $\mathcal{P}$espai i $\mathcal{NP}$espai.
Es compleix que:

$$\mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{P}\text{espai} = \mathcal{NP}\text{espai}$$

Però,

$$\mathcal{P} = \mathcal{NP}?$$

Les classes P i NP

- Els problemes en $\mathcal{P}$ són resolubles en temps polinòmics (tractables).
- Hi ha molts problemes que no s'han pogut resoldre en temps determinista polinòmic però en canvi tenen una solució indeterminista polinòmica (estan en $\mathcal{NP}$).
- Intuitivament són problemes on una MT (o un algorisme) pot "endevinar" la solució de manera indeterminista per a després comprovarla (de manera determinista) en temps polinòmic.
- Exemples: Problema del viatjant, de les n-reines, de la motxilla.
- Molts d'aquests problemes es poden formular com a problema de decisió o de minimització (en funció de si es demana o no la **solució** que cal trobar en qualsevol cas).

Reducció polinòmica entre problemes

Diem que un problema P es redueix polinòmicament al problema Q (i s'escriu $P \preceq_p Q$) si la solució en temps polinòmic de Q implica la solució en temps polinòmic de P.

Diem que el llenguatge $L_1 \in \Sigma_1$ es redueix polinòmicament al llenguatge $L_2 \in \Sigma_2$ (i s'escriu $L_1 \preceq_p L_2$) si existeix una funció polinòmicament computable $f : \Sigma_1 \longrightarrow \Sigma_2$ de manera que es complesca que

$$x \in L_1 \Leftrightarrow f(x) \in L_2$$

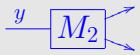
Reducció polinòmica entre problemes

Si $L_1 \preceq_p L_2$, aleshores

a) $L_2 \in \mathcal{P} \Rightarrow L_1 \in \mathcal{P}$

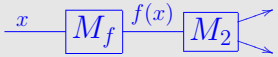
b) $L_2 \in \mathcal{NP} \Rightarrow L_1 \in \mathcal{NP}$

a) Si $L_1 \preceq_p L_2$ i $L_2 \in \mathcal{P}$,



Aleshores $L_1 \in \mathcal{P}$ perque

b) és igual



Problemes NP-complets

Siga $\mathcal{C}$ una determinada classe de llenguatges.
Un llenguatge L es diu $\mathcal{C}$-difícil (o difícil per a la classe $\mathcal{C}$) si tot llenguatge de $\mathcal{C}$ es redueix polinòmicament a L. És a dir,

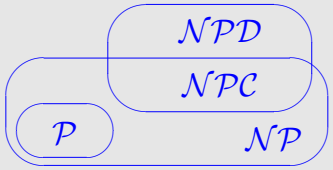
$$L' \preceq_p L \quad \forall L' \in \mathcal{C}$$

Si a més a més, el llenguatge L pertany a $\mathcal{C}$, es diu que L és $\mathcal{C}$-complet.

Que un llenguatge síga $\mathcal{NP}$-difícil, vol dir que no poden haver en $\mathcal{NP}$ llenguatges (polinòmicament) més difícils que ell.

Es defineix co-$\mathcal{C} = \{L \mid \overline{L} \in \mathcal{C}\}$. Es compleix que co-$\mathcal{P}$=$\mathcal{P}$, però no se sap si co-$\mathcal{NP}$ és o no igual a $\mathcal{NP}$.

Problemes NP-complets



Si $\exists L \in \mathcal{P}$ i $L \in \mathcal{NPC}$, aleshores $\mathcal{P} = \mathcal{NP}$

Problemes NP-complets

Si $L_1 \preceq_p L_2$ i $L_2 \preceq_p L_3$ aleshores $L_1 \preceq_p L_3$.

Només cal composar les dues funcions que transformen cadenes d'un llenguatge a un altre.

Si $L_1 \in \mathcal{NPC}$ i $L_1 \preceq_p L_2$, aleshores $L_2 \in \mathcal{NPD}$

Si L_1 és $\mathcal{NPC}$ serà també $\mathcal{NPD}$ i, per definició, es complirà que $L \preceq_p L_1$ per a tot llenguatge L de $\mathcal{NP}$.
Com que $L_1 \preceq_p L_2$, aleshores

$$L \preceq_p L_2 \quad \forall L \in \mathcal{NP}$$

Caracterització de problemes $\mathcal{NP}$-complets

És L $\mathcal{NP}$-complet?

- disposar de L_1, $\mathcal{NP}$-complet.
- demostrar que $L_1 \preceq_p L$.
- demostrar que $L \in \mathcal{NP}$.

El problema per aplicar-ho és que cal almenys un primer problema $\mathcal{NP}$-complet.

El problema de la satisfactibilitat

Siga un conjunt de variables booleanes. Una clàusula és una disjunció de literals (variables o negacions de variables).
Una assignació de valors de veritat és associar el valor cert o fals a cada variable booleana.

> Donat un conjunt de m clàusules amb n variables booleanes, existeix una assignació de valors de veritat que faça certes les m clàusules?

Si la resposta és si, es diu que el conjunt de clàusules se satisfà.

El problema de la satisfactibilitat

$$x_1 \vee \overline{x_2}$$
$$\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}$$
$$\overline{x_1} \vee x_2 \vee x_3$$

se satisfà per als valors de veritat $\{1, 0, 1\}$. En canvi, el conjunt

$$\overline{x_1}$$
$$x_1 \vee \overline{x_2}$$
$$x_1 \vee x_2$$

no se satisfà.

El problema de la satisfactibilitat

El problema de la Satisfactibilitat (P_{SAT})

Dades: Un conjunt C de m clàusules amb n variables.

Enunciat: Se satisfà C?

- P_{SAT} és resoluble: es poden tantejar les 2^n possibles assignacions de veritat.
- P_{SAT} està en $\mathcal{NP}$. Es pot generar una assignació de valors de veritat de manera indeterminista i comprovar (en temps $n \cdot m$) si se satisfà o no.
- no se sap si P_{SAT} està en $\mathcal{P}$.

El teorema de Cook

P_{SAT} és $\mathcal{NP}$-difícil (Qualsevol problema en $\mathcal{NP}$ es pot reduir a P_{SAT}?)

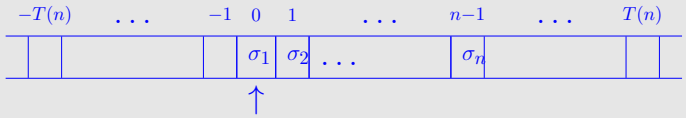
Siga $L \in \mathcal{NP}$, i L_{SAT} el llenguatge associat a P_{SAT}.
$L = L(M_1)$ i $T_{M_1}(n) = T(n) \in \mathcal{O}(n^k)$ per a alguna MT, M_1.

Suposarem sense pèrdua de generalitat M_1 té una única cinta i que els seus estats estan numerats de 0 a p de forma que q_0 és l'estat inicial i q_p el (únic) final. Per conveniència, numerarem també l'alfabet de cinta de 0 a m i assignarem al símbol blanc en número 0. En resum

$$Q = \{q_0, \ldots, q_p\} \qquad \Gamma = \{\tau_0, \ldots, \tau_m\}$$

El teorema de Cook

Donada una configuració d'entrada qualsevol $q_0\sigma_1\cdots\sigma_n$, la MT arribarà a una configuració de parada en menys de $T(n)$ moviments.
L'espai utilitzat estarà també limitat per $T(n)$:



Definirem variables booleanes que descriuran el lloc on es troba el capçal en cada moment (H_{ij}), l'estat actual ($Q_{i\ell}$) i el símbol actual (S_{ijk}).

El teorema de Cook

- Per a $i = 0, 1, \ldots, T(n)$ i per a $j = -T(n), \ldots, T(n)$

$$H_{ij} = \begin{cases} 1 & \texttt{si} \\ 0 & \texttt{si no} \end{cases}$$

en el i-èssim moviment el capçal es troba a la j-èssima posició.

- Per a $i = 0, 1, \ldots, T(n)$ i per a $\ell = 0, \ldots, p$

$$Q_{i\ell} = \begin{cases} 1 & \texttt{si} \\ 0 & \texttt{si no} \end{cases}$$

en el i-èssim moviment el nou estat és q_ℓ.

- Per a $i = 0, 1, \ldots, T(n)$, per a $j = -T(n), \ldots, T(n)$ i per a $k = 0, \ldots, m$

$$S_{ijk} = \begin{cases} 1 & \texttt{si} \\ 0 & \texttt{si no} \end{cases}$$

en el i-èssim moviment la j-èssima casella conté el k-èssim símbol, τ_k.

El teorema de Cook

Una computació de M amb $\sigma_1 \ldots \sigma_n$ implica una assignació de valors de veritat per a les variables definides però no al revés.

Quines condicions hauràn de complir les assignacions de valors de veritat per tal que es corresponguen amb una computació?

- el capçal no pot estar en dues posicions simultàniament.
- només pot haver-hi un únic estat actual.
- el contingut de cada posició conté un únic símbol.

El teorema de Cook

$$\begin{cases} \vee H_{ij} & \forall i \\ \overline{H_{ij}} \vee \overline{H_{ij'}} & \forall i, j, j' : j < j' \end{cases}$$

$$\begin{cases} \vee Q_{i\ell} & \forall i \\ \overline{Q_{i\ell}} \vee \overline{Q_{i\ell'}} & \forall i, \ell, \ell' : \ell < \ell' \end{cases}$$

$$\begin{cases} \vee S_{ijk} & \forall i \\ \overline{S_{ijk}} \vee \overline{S_{ijk'}} & \forall i, j, k, k' : k < k' \end{cases}$$

El teorema de Cook

A més a més,

- Inicialment el capçal es troba en la primera posició: H_{00}
- L'estat inicial és el 0: Q_{00}
- Els continguts de les n primeres caselles són els símbols de w. En la resta de caselles hi ha blancs.

$$\begin{cases} S_{0tk_t} & t = 0, \ldots, n-1 \ \ \underline{\texttt{si}} \ \ w = \tau_{k_1} \cdots \tau_{k_n} \\ S_{0t0} & \forall t : t < 0, t \geq n \end{cases}$$

- Després de l'últim moviment s'arriba a l'estat de parada, p.

$$Q_{T(n)p}$$

- p és un estat de parada. Aleshores $Q_{ip} \implies Q_{i+1,p}$.

$$\overline{Q_{ip}} \vee Q_{i+1,p}$$

El teorema de Cook

Per últim, cal considerar la relació entre les variables associades a un instant i i a l'instant anterior.
Per cada transició $(q_{s'}, \tau_{t'}, D) \in \delta(q_s, \tau_t)$:

$$H_{ij} \wedge Q_{is} \wedge S_{ijt} \implies H_{i+1,j+d} \wedge Q_{i+1,s'} \wedge S_{i+1,jt'}$$

$$\text{on } d = \begin{cases} +1 & \text{si } D = \rightarrow \\ \\ -1 & \text{si } D = \leftarrow \end{cases}$$

A més a més, $\overline{H_{ij}} \wedge S_{ijt} \implies S_{i+1,jt}$

El teorema de Cook

Equivalentment (per als valors adequats dels índexs),

1. $\overline{H_{ij}} \vee \overline{Q_{is}} \vee \overline{S_{ijt}} \vee H_{i+1,j+d}$
2. $\overline{H_{ij}} \vee \overline{Q_{is}} \vee \overline{S_{ijt}} \vee Q_{i+1,s'}$
3. $\overline{H_{ij}} \vee \overline{Q_{is}} \vee \overline{S_{ijt}} \vee S_{i+1,jt'}$
4. $H_{ij} \vee \overline{S_{ijt}} \vee S_{i+1,jt}$

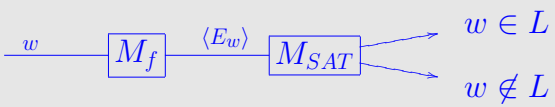
El teorema de Cook

Definides d'aquesta manera, es compleix:

Si les clàusules se satisfan
$\Downarrow$
Existeix una assignació per a les variables
$\Downarrow$
Existeix una computació d'acceptació per a M amb w
$\Downarrow$
La cadena w és acceptada per M.

El teorema de Cook

El conjunt de definit, E_w té $\mathcal{O}(T(n)^2)$ clàusules sobre $\mathcal{O}(T(n)^2)$ variables i es té que:



la qual cosa implica segons la definició que

$$L \preceq_p L_{SAT}$$

i, per tant, L_{SAT} és $\mathcal{NP}$-difícil.

El problema del recobriment exacte

Problema del recobriment exacte (P_{RE})

Dades: Un conjunt $U = \{u_1, \ldots, u_n\}$
Un conjunt de m subconjunts de U, $\mathcal{S} = \{S_1, \ldots, S_m\}$.

Enunciat: Existeix un subconjunt $\mathcal{C}$ d'elements de $\mathcal{S}$ format per conjunts disjunts la unió dels quals siga U?

El problema del recobriment exacte

Podem representar una instància de P_{RE}, $(U, \mathcal{S})$, gràficament si pensem en una màtriu les files de la qual siguen els vectors característics dels conjunts de $\mathcal{S}$. El problema consisteix doncs a seleccionar un conjunt de files que, per columnes, sumen el vector $(1, \ldots, 1)$.

El problema del recobriment exacte

$$\boxed{P_{RE} \text{ és } \mathcal{NP}\text{-difícil } (P_{SAT} \preceq_p P_{RE})}$$

Siga F una instància qualsevol per al P_{SAT}:

$$\begin{array}{c} c_1 : \lambda_{11} \vee \cdots \vee \lambda_{1m_1} \\ \vdots \\ c_i : \lambda_{i1} \vee \cdots \vee \lambda_{im_i} \\ \vdots \\ c_\ell : \lambda_{\ell 1} \vee \cdots \vee \lambda_{\ell m_\ell} \end{array}$$

cada λ_{jk} serà x_i o $\overline{x_i}$, $i = 1..n$.

El problema del recobriment exacte

construim la instància $(U_F, \mathcal{S}_F)$ per al P_{RE}:

$$U_F = \{\underbrace{x_1, \ldots, x_n}_{\text{variables}}, \underbrace{c_1, \ldots, c_\ell}_{\text{clàusules}}, \underbrace{p_{11}, \ldots, p_{1m_1}, p_{21}, \ldots, p_{\ell m_\ell}}_{\text{literals}}\}$$

$$\mathcal{S}_F = \{V_1, \ldots, V_n, F_1, \ldots, F_n, C_{11}, \ldots, C_{\ell m_\ell}, P_{11}, \ldots, P_{\ell m_\ell}\}$$

$$V_i = \{x_i\} \cup \{p_{jk} : \lambda_{jk} = \overline{x_i}\} \qquad C_{jk} = \{c_j, p_{jk}\}$$
$$F_i = \{x_i\} \cup \{p_{jk} : \lambda_{jk} = x_i\} \qquad P_{jk} = \{p_{jk}\}$$

$\mathcal{S}_F$ conté un recobriment exacte de $U_F \iff F$ és satisfactible

El problema del recobriment exacte

$\mathcal{C}$ recobreix $U_F \implies F$ se satisfà

Considerem cada clàusula c_j, $j = 1..\ell$

-contindrà un únic dels $C_{jk} = \{c_j, p_{jk}\}$.
-p_{jk} es refereix a un literal que haurà d'aparéixer en les zones V o (exclusiu) F.

- si apareix en F (x_i, no negat), c_j se satisfarà si x_i=cert.
  ($\mathcal{C}$ no contindra F_i però si (conseqüentment) V_i)
- si apareix en V ($\overline{x_i}$, negat), c_j se satisfarà si x_i=fals.
  ($\mathcal{C}$ no contindra V_i però si (conseqüentment) F_i)

L'assignació $x_i = \begin{cases} Cert & \text{si } V_i \in \mathcal{C} \\ & \\ Fals & \text{si } F_i \in \mathcal{C} \end{cases}$ satisfarà tot F.

El problema del recobriment exacte

$\boxed{F \text{ se satisfà}} \implies \mathcal{C}$ recobreix U_F

Donada l'assignació que satisfà F es construeix un recobriment començant per les zones V i F:

$$\begin{cases} V_i \in \mathcal{C} \texttt{si } x_i = \text{CERT} \\ F_i \in \mathcal{C} \texttt{si } x_i = \text{FALS} \end{cases}$$

Això inclou en $\mathcal{C}$ alguns p_{jk} que es refereixen precisament a literals negats quan la variable és certa i no negats quan es falsa.
Per tant, han d'haver necessàriament altres literals per cada clàusula que siga cert si F és satisfactible. Seleccionem per cada una de les ℓ columnes (zona C) un d'aquests literals $\{c_j, p_{jk}\}$ per cada j.
Una vegada cobertes les $n+\ell$ primeres columnes, es cobreixen les que (encara) no ho estiguen seleccionant els literals de la zona P que calguen.

El problema de la motxilla

Problema de la motxilla (P_{MO})

Dades: Un conjunt d'enters no negatius, $S = \{a_1, \ldots, a_n\}$
Un enter K.

Enunciat: Existeix un subconjunt P de S tal que $\sum_{a_i \in P} a_i = K$?

$\boxed{P_{MO} \text{ és } \mathcal{NP}\text{-difícil}}$

Es pot fer una reducció des de P_{RE} ...

Exemples. El problema k-SAT

El problema k-SAT és una versió particular del problema SAT en què el nombre de literals en cada clàusula és, com a molt, k.

3-SAT $\preceq_p$SAT Trivial!

SAT $\preceq_p$3-SAT Comentari: caldria convertir (en temps polinòmic) qualsevol instància, w, per al SAT en una altra instància, w_3, (per al 3-SAT) de manera que:

$$w_3 \text{ se satisfà} \Leftrightarrow w \text{ se satisfà}$$

2-SAT $\in \mathcal{P}$ Comentari: només (?) cal trobar un algorisme que resolga el problema (conteste SI o NO) en temps polinòmic. Sempre.