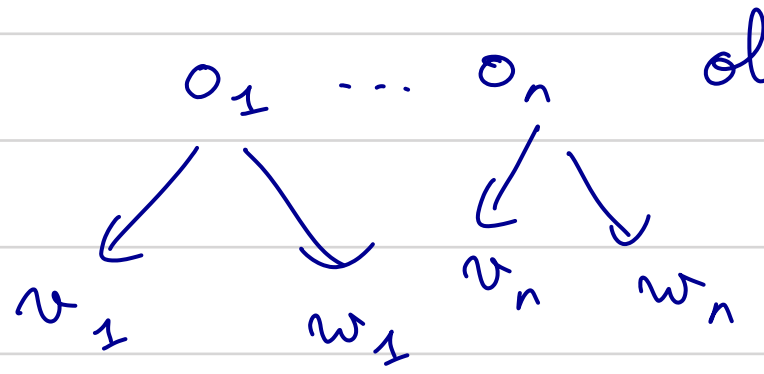
Problèmes NP-complets

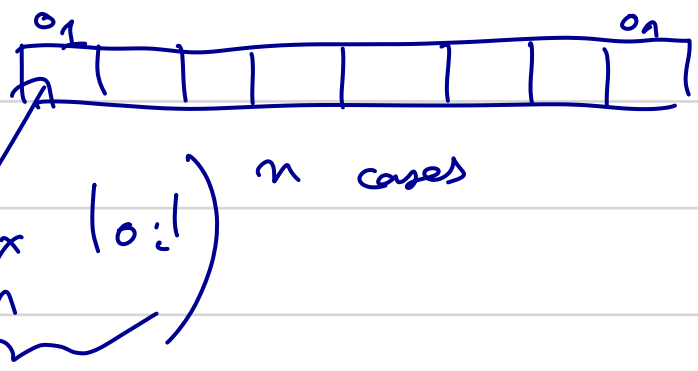
Tri : $O(n \log n)$ par tri fusion
Multiplication de matrice : $O(n^{\log_2 7})$ par Strassen
Sac à dos 0-1 : $O(W \times n)$ par prog. dynamique



On cherche $S \subseteq \{1, \dots, n\}$ t.q.
$\sum_{i \in S} w_i \leq W$ et $\sum_{i \in S} v_i$ maximale

On dit qu'un problème est résolvable en <u>temps polynomial</u> s'il existe un algorithme pour résoudre ce problème dont la complexité asymptotique en temps est en $O(P(|I|))$ où P est un polynôme et $|I|$ est la taille de l'entrée du problème (le nombre de bits nécessaire pour écrire l'entrée).

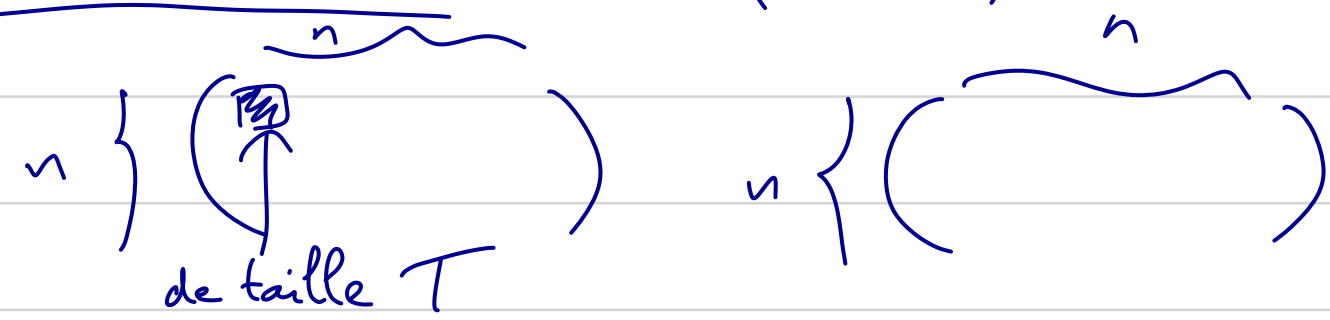
<u>Tri</u> : $O(n \log n)$



$|I| = O(n \times \max_{1 \leq i \leq n} |o_i|)$ (sous-terme : T)

Problème en temps polynomial.

<u>Multiplication de matrices</u> $O(T_n^{2 \log_2 7})$



$|I| = O(n^2 \times T)$

Problème en temps polynomial

Sac à dos $O(n \times W)$ $v_i \leq T$ $w_i \leq T$ $W \leq T$
$n \times T$

$$|I| = O(\log T + n \log T)$$
$$= O(n \log T)$$

La prog. dynamique ne donne pas une résolution en temps polynomial du pb du sac à dos. On dit parfois que l'algorithme est pseudo-polynomial (polynomial en la valeur de l'entrée)

P et NP

Problèmes de décision : problème est soit Vrai soit Faux

Variantes de décision des pb plus haut :

- Est-ce que le tableau en entrée est trié ?
  $O(n \times T)$
- Vérifier si une matrice C est le produit de deux matrices A et B $O(n^{\log_2 7} \times T)$
- Existe-t-il $S \subseteq \{1, \dots, n\}$ t.q. $\sum_{i \in S} w_i \leq W$ et $\sum_{i \in S} v_i \geq V$ pour un V donné en entrée.

Toujours le pb de décision a une complexité asymptotique au moins aussi faible que le problème de calcul.

La classe de complexité P (ou PTIME) est la classe de l'ensemble des problèmes de décision résolubles en temps polynomial par un algorithme classique.

On introduit un nouvel opérateur dans les algorithmes Deviner qui devine une structure de taille polynomiale en (la taille de) l'entrée.

Sac à dos Non Déterministe

Entrée : $(v_i)_{1\le i\le n}, (w_i), V, W$

Sortie : Vrai s'il existe $S \subseteq \{1 \dots n\}$ t.q.

$\sum_{i\in S} w_i \le W$ et $\sum_{i\in S} v_i \ge V$

Deviner un sous-ensemble $S \subseteq \{1, \dots, n\}$

Calculer $\sum_{i\in S} w_i$ et $\sum_{i\in S} v_i$

Si $\sum_{i\in S} w_i \le W$ et $\sum_{i\in S} v_i \ge V$

Renvoyer Vrai

Sinon

Renvoyer Faux

On dit qu'un algorithme utilisant l'opérateur Deviner (on parle d'algorithme non-déterministe) résout un problème si :

- quand il existe un guess qui conduit l'algorithme à renvoyer Vrai, alors le problème a pour solution Vrai
- sinon, le problème a pour solution Faux

La classe de complexité NP (nondeterministic polynomial time) est la classe de l'ensemble des problèmes de décision résolubles en temps polynomial par un algorithme non-déterministe.

$\boxed{P \subseteq NP}$ Sac À Dos $\in NP$

$NP \not\subseteq$ non polynomial

$\boxed{NP \subseteq EXPTIME}$: tous les problèmes dans NP sont résolubles en temps exponentiel par un algorithme classique

Digression sur la satisfiabilité

On fixe un ensemble fini $X = \{x_1 \dots x_n\}$ de variables pouvant prendre les valeurs Vrai ou Faux (variables booléennes, proposition).

Une formule logique sur X (on parle aussi de formule du calcul des propositions, de la logique propositionnelle) est une formule construite avec les opérateurs suivants:

* si $x \in X$, alors x est une formule
* si f est une formule, alors $(\neg f)$ est une formule (la négation de f)
* si $f_1 \dots f_m$ sont des formules, alors $(f_1 \vee \dots \vee f_m)$ et $(f_1 \wedge \dots \wedge f_m)$ sont des formules (la disjonction / conjonction de $f_1 \dots f_m$)

(Quand il n'y a pas d'ambiguité, on enlève les parenthèses)

Exemple : $X = \{x_1, x_2, x_3, x_4\}$

$$f_0 = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1 \vee x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$$

Soit $v : X \to \{\text{Vrai}, \text{Faux}\}$ (on appelle v une valuation). On définit pour une formule f l'évaluation de f par v, $v(f) \in \{\text{Vrai}, \text{Faux}\}$ par:

* si $f = x \in X$, $v(f) = v(x)$
* si $f = (\neg f')$, $v(f) = \neg v(f')$
* si $f = (f_1 \vee \dots \vee f_m)$, $v(f) = v(f_1) \vee \dots \vee v(f_m)$
* si $f = (f_1 \wedge \dots \wedge f_m)$, $v(f) = v(f_1) \wedge \dots \wedge v(f_m)$

Ex. $v = \{x_1 \mapsto \text{Vrai}, x_2 \mapsto \text{Faux}, x_3 \mapsto \text{Vrai}, x_4 \mapsto \text{Faux}\}$

$v(f_0) = \text{Vrai} \wedge \text{Vrai} \wedge \text{Vrai} = \text{Vrai}$

Déf. Une formule est en DNF (disjunctive) CNF (conjunctive normal form) si elle s'écrit

$$(c_1 \wedge \dots \wedge c_m)$$ ← clause

où chaque c_i s'écrit

$$(l_{i1} \vee \dots \vee l_{ik_i})$$ ← littéral

où chaque l_{ij} est de la forme $x \in X$ ou $(\neg x)$ pour $x \in X$.

f_0 est une CNF.

Rem. : L'évaluation d'une formule par valuation se calcule en temps linéaire en la taille de la formule.

Déf. : Une formule f est satisfiable s'il existe une valuation v de ses variables t.q. $v(f) = \text{Vrai}$.

f_0 est satisfiable.

Déf. Le problème SAT est, étant donné une formule logique, déterminer si f est satisfiable.

Proposition $\quad SAT \in NP$

(deviner une valuation v, vérifier si $v(f) = \text{Vrai}$)

NP-complétude

Déf. Une réduction (many-to-one de Karp, en temps polynomial) entre un problème de décision P et un problème de décision P' est un algorithme classique en temps polynomial qui prend en entrée une entrée du problème P et qui produit en sortie une entrée du problème P' t.q. $P'(\mathcal{A}(I)) = P(I)$ pour toute entrée I du problème P. On écrit $P \leq P'$.

Déf. Un problème P est dit **NP-difficile** si pour tout problème $P' \in NP$, alors $P' \leq P$.
Un problème P est dit **NP-complet** si P est NP-difficile et $P \in NP$.

Théorème (admis) SAT est NP-complet. C'est également vrai si on restreint SAT aux formules CNF avec 3 littéraux par clause (SAT-3CNF).
(fo était un exemple de 3CNF).

Application: Un SAT-solver permet de résoudre tous les problèmes qui sont dans NP (après traduction en SAT).

P vs NP

$P \stackrel{?}{=} NP \longrightarrow$ 7 pb du millénaire
1 M$.

Théorème : SacÀDos est NP-complet.

Mq $SAT \leq SSE \leq$ SacÀDos
(somme des sous-ensembles)

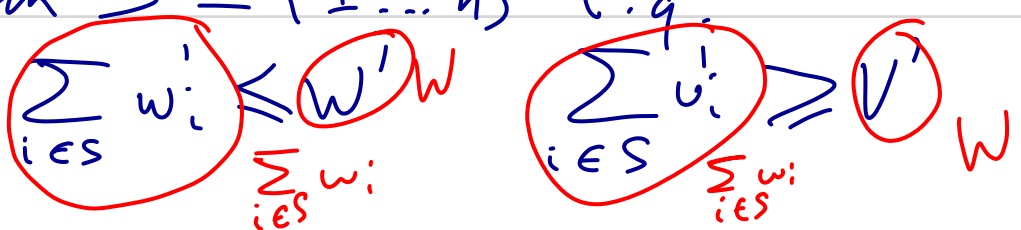
$w_1 \dots w_n$ n entiers ≥ 0
$W \geq 0$
Existe-t-il $S \subseteq \{1, \dots, n\}$ t.q. $\sum_{i \in S} w_i = W$?

Mq. SSE $\leq$ SacÀDos

On prend une entrée de SSE $(w_1 \dots w_n, W)$
On construit une entrée du pb sac à dos :

$$\begin{cases} w'_i := w_i \\ v'_i := w_i \\ W' := W \\ V' := W \end{cases}$$

Une solution du pb de sac à dos est un $S \subseteq \{1 \dots n\}$ t.q.



$$W \leq \sum_{i \in S} w_i \leq W$$

$3CNFSAT \leq SSE$

Ex. $f_0 = \underbrace{(x_1 \vee x_2 \vee \neg x_3)}_{c_1} \wedge \underbrace{(x_3 \vee \neg x_1 \vee \neg x_2)}_{c_2} \wedge \underbrace{(x_2 \vee \neg x_3 \vee \neg x_4)}_{c_3}$

$(n=4,\ m=3)$

On introduit $2n+2m$ entiers $w_1 .. w_n\ w'_1 .. w'_n\ z_1 .. z_m$

	$\overbrace{\quad}^{m}$	$\overbrace{\quad\quad}^{n}$		$\overbrace{\quad}^{m}$	$\overbrace{\quad\quad}^{n\ z'_1 \ldots z'_m}$
$w_1 =$	001	0001	$w'_1 =$	010	0001
$w_2 =$	111	0010	$w'_2 =$	000	0010
$w_3 =$	010	0100	$w'_3 =$	101	0100
$w_4 =$	000	1000	$w'_4 =$	100	1000

$z_1 = z'_1 = 0010000$

$z_2 = z'_2 = 0100000$

$z_3 = z'_3 = 1000000$

$W = \overbrace{333}^{m}\ \overbrace{1111}^{n}$

001 0001
000 0010
010 0100
100 1000

Pour $v = \{x_1 \mapsto \text{Vrai}, x_2 \mapsto \text{Faux}, x_3 \mapsto \text{Vrai}, x_4 \mapsto \text{Faux}\}$

$v(f_0) = \text{Vrai}$

$S = \{w_1, w'_2, w_3, w'_4, z_1, z'_1, z_2, z'_2, z_3, z'_3\}$

$$\sum_{v \in S} v = w_1 + w'_2 + w_3 + w'_4 + 222\,0000$$
$$= 111\,1111$$
$$= 333\,1111$$

---

f une formule 3CNF avec n variables $x_1 .. x_n$ et m clauses $c_1 .. c_m$.

On introduit $2m + 2n$ entiers naturels $z_1, z'_1 \ldots z_m z'_m$ $w_1, w'_1 \ldots w_m w'_m$

Pour $1 \leq j \leq m$, $z_j = z'_j = 10^{n+j-1}$

Pour $1 \leq i \leq m$, $w_i = 10^{i-1} + \sum_{j=1}^{m} 10^{n+j-1}$

$1 \leq i \leq m$, $w'_i = 10^{i-1} + \sum_{j=1,\ x_i \in c_j}^{m} 10^{n+j-1} + \sum_{j=1,\ \neg x_i \in c_j}^{m} 10^{n+j-1}$

$$W = \sum_{i=1}^{n} 10^{i-1} + \sum_{j=1}^{m} 3 \times 10^{n+j-1} = \underbrace{3 \dots 3}_{m} \; \underbrace{1 \dots 1}_{n}$$

On va montrer f satisfiable $\iff \exists S \subseteq \{w_1 .. w_n, w'_1 ... w'_n, z_1 .. z_m, z'_1 ... z'_m\}$

t.q. $\sum_{v \in S} v = W$

$\Rightarrow$ Soit ν une valuation t.q. $\nu(f) = \text{Vrai}$.

On choisit $S' = \{ w_i \mid \nu(x_i) = \text{Vrai} \} \cup \{ w'_i \mid \nu(x_i) = \text{Faux} \}$

$$\sum_{v \in S'} v = \underbrace{k_1 \dots k_m}_{m} \; \underbrace{1 \dots 1}_{n}$$

avec $\forall 1 \leq j \leq m \quad 1 \leq k_j \leq 3$

On complète S avec des z_j et z'_j t.q.

$$\sum_{v \in S} v = W$$

$\Leftarrow$ Supposons $S \subseteq \underbrace{\{ w_i \dots w'_i \dots z_j \dots z'_j \}}_{Z}$

t.q. $\sum_{v \in S} v = W$.

⚠ Il n'y a jamais de retenue quand on fait la somme car dans Z, il y a au plus 5 variables dont le $k^{ème}$ chiffre est 1.

Clairement, pour tout i, soit $w_i \in S$ et $w'_i \notin S$
soit $w'_i \in S$ et $w_i \notin S$

Considérons $\nu : x_i \mapsto$ Vrai si $w_i \in S$
$\mapsto$ Faux si $w'_i \in S$

Pour tout j, on a au moins un littéral de c_j qui est mis à Vrai. En effet, pour tout j, il existe nécessairement soit un $w_i \in S$ avec $x_i \in c_j$
un $w'_i \in S$ avec $\neg x_i \in c_j$. □