Complexité 3

D.E ZEGOUR
École Supérieure d'Informatique
ESI

Complexité

Sommaire

Réduction polynomiale

Np-Complétude

Complexité / Réduction polynomiale

Notion de réduction

Réduction permet de comparer les problèmes entres eux

[Problème A se réduit à un problème B] veut dire si on a une solution pour B, on a aussi une solution pour A

Le problème A n'est alors pas plus difficile que le problème B.

A : multiplication de deux nombres
B : Elever au carré, additionner et diviser par 2.

Remarquons que

$$a \times b = ((a+b)^2 - a^2 - b^2)/2.$$

Ainsi, pour calculer la multiplication de a par b,il suffit d'effectuer 3 additions, 3 élévations au carré, et une division par 2.

Complexité / Réduction polynomiale

Réduction polynomiale

Une réduction permet de ramener la décidabilité d'un problème à celle d'un autre problème.

Une réduction polynomiale fait correspondre à une instance du problème A une instance du problème B
- avec même réponse et
-calculable en temps polynomial.

Complexité / Réduction polynomiale

Réduction polynomiale

Supposons

Problème A codé par le langage L_A sur l'alphabet Σ_A

Problème B codé par le langage L_B sur l'alphabets Σ_B

Une *réduction polynomiale* de A à B est une fonction f de Σ_A^* dans Σ_B^* calculable en temps polynomial (par une machine de Turing) telle que

$\mathbf{w \in L_A \Leftrightarrow f(w) \in L_B}$ Notation : $A \leq_P B$

Exemples

SAT $\leq_P$ 3-SAT

3-SAT $\leq_P$ CLIQUE

Complexité / Réduction polynomiale

Réduction polynomiale

Théorème: $\leq_P$ *est un préordre:*

1. $L \leq_P L$
2. $L1 \leq_P L2, L2 \leq_P L3 \Rightarrow L1 \leq_P L3.$

$L1 \leq_P L2, L2 \in P \Rightarrow L1 \in P$

$L1 \leq_P L2, L1 \notin P \Rightarrow L2 \notin P$

Equivalence de deux problèmes L1 et L2:

$L1 \equiv_P L2 \leftrightarrow L1 \leq_P L2$ *et* $L2 \leq_P L1.$

Remarque:
Les résultats restent vrais si on remplace P par NP

Complexité / Réduction polynomiale

Equivalence polynomiale $SAT \equiv_P 3\text{-}SAT$

Description de SAT

Description de 3-SAT

$SAT \leq_P 3\text{-}SAT$

$3\text{-}SAT \leq_P SAT$ (Triviale)

Complexité / Réduction polynomiale

Rappel (*SAT)*

Etant donné n variables Booléennes x_1, x_2, ..., x_N et une formule de la logique (CNF : forme normale conjonctive)

Existe-il une affectation de valeurs Vrai et Faux aux variables qui rend la formule satisfiable, c.-à-d., Vraie?

Par exemple, supposer la formule (' désigne la négation)

$$(x_1' + x_2 + x_3)\ (x_1 + x_2' + x_3)\ (x_2 + x_3)\ (x_1' + x_2' + x_3')$$

Est-elle s*atisfiable* ?

Complexité / Réduction polynomiale

Rappel (*3-SAT)*

Etant donné n variables Booléennes x_1, x_2, ..., x_N et une formule de la logique en forme normale conjonctive (produit de sommes) avec exactement 3 littéraux différents par clause,

Existe-il une affectation de valeurs Vrai et Faux aux variables qui rend la formule satisfiable, c.-à-d., Vraie?

Exemple $(x_1' + x_2 + x_3)\ (x_1 + x_2' + x_3)\ (x_1' + x_2' + x_3')$

Complexité / Réduction polynomiale

$(x+y+c)(x+y+c')$
$= xx + xy + xc' +$
$yx + yy + yc' +$
$cx + cy + cc'$
$= x + xy + x +$
$yx + y + y$
$= x + xy + y + yx$
$= x+y + yx$
$= x + y(1+x)$
$= x + y$

Réduction polynomiale SAT $\leq_P$ 3-SAT

Soit f une formule booléenne sous forme normale conjonctive.

F = C_1 C_2 C_3 … C_m où chaque C_i est une disjonction de littéraux.

$C_i = x_1 \vee x_2 \vee \ldots \vee x_k$

Remplacer chaque C_i par une conjonction de clauses à au plus 3 littéraux

Cas k =1 : Ajouter à chaque C_i avec une variable (Ci= x) deux variables a_i et b_i
Remplacer x
Par $(x \vee a_i \vee b_i)\ (x \vee a_i \vee b_i')\ (x \vee a_i' \vee b_i)\ (x \vee a_i' \vee b_i')$

Cas k=2 : Ajouter à chaque C_i à deux variables (Ci= x V y) une variable c_i
Remplacer x V y
Par $(x \vee y \vee c_i)\ (x \vee y \vee c_i')$

Complexité / Réduction polynomiale

Réduction polynomiale SAT $\leq_P$ 3-SAT

Cas K>3

Chaque clause $C = x_1 \vee x_2 \vee \ldots \vee x_k$ devient

$C_0 = (x_1 \vee x_2 \vee y_1)\ (y'_1 \vee x_3 \vee y_2)\ (y'_2 \vee x_4 \vee y_3) \ldots (y'_{k-3} \vee x_{k-1} \vee x_k)$

C satisfiable $\Leftrightarrow$ C_0 satisfiable

C_0 satisfiable $\equiv$ il existe une affectation aux variables $y_1, y_2, .., y_{k-3}$ telle que toutes les clauses soient satisfaites(Vraies)

Montrer les deux implications

Complexité / Réduction polynomiale

Réduction polynomiale SAT $\leq_P$ 3-SAT

Cas K>3

Chaque clause $C = x_1 \vee x_2 \vee \ldots \vee x_k$ devient

$C_0 = (x_1 \vee x_2 \vee y_1)\ (y'_1 \vee x_3 \vee y_2)\ (y'_2 \vee x_4 \vee y_3) \ldots (y'_{k-3} \vee x_{k-1} \vee x_k)$

C_0 satisfiable $\Rightarrow$ C satisfiable

Supposons que les clauses de C_0 sont toute satisfaites.
Donc, au moins un des littéraux $x_1, x_2, \ldots, x_k$ doit être vrai.

Autrement y_1 est vrai, ce qui force y_2 à vrai et ainsi de suite. La dernière clause serait fausse.

L'implication est donc montrée.

Complexité / Réduction polynomiale

Réduction polynomiale SAT $\leq_P$ 3-SAT

Cas K>3

Chaque clause $C = x_1 \vee x_2 \vee \ldots \vee x_k$ devient

$C_0 = (x_1 \vee x_2 \vee y_1)\ (y'_1 \vee x_3 \vee y_2)\ (y'_2 \vee x_4 \vee y_3) \ldots (y'_{k-3} \vee x_{k-1} \vee x_k)$

C satisfiable $\Rightarrow$ C_0 satisfiable

$C = x_1 \vee x_2 \vee \ldots x_k$ satisfiable : il existe au moins un i tel que x_i = vrai
on pose $y_1, y_2, \ldots, y_{i-2}$ tous à Vrai et le reste à Faux.

Ceci garantit que les clauses de C_0 sont vraies.

Complexité / Réduction polynomiale

Equivalence polynomiale *3-SAT* $\equiv_P$ *Clique*

Description de SAT

Description de Clique

3-SAT $\leq_P$ Clique

Clique $\leq_P$ 3-SAT

Complexité / Réduction polynomiale

Rappel (*3-SAT)*

Etant donné n variables booléennes x_1, x_2, ..., x_N et une formule de la logique en forme normale conjonctive (produit de sommes) avec exactement 3 littéraux différent par clause,

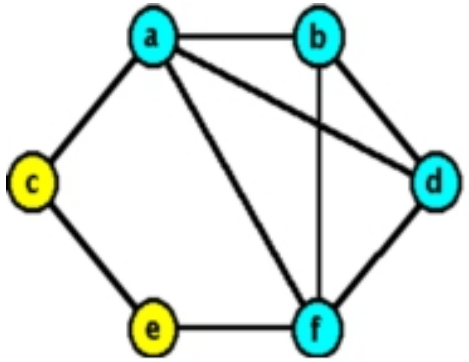
Existe-il une affectation de valeurs Vrai et Faux aux variables qui rend la formule satisfiable, cad., Vraie?

Complexité / Réduction polynomiale

Rappel (*Clique dans un graphe)*

Instance: soit G = (V, E) un graphe non orienté (V: ensemble des nœuds; E:ensemble des arcs) et soit k un entier k <=n

*Une **clique de taille k** est un sous-ensemble W de cardinalité k de V tel que deux sommets quelconques de W sont joints par une arête*



Clique de taille 4 (en bleu)

Complexité / Réduction polynomiale

Réduction polynomiale 3-SAT $\leq_P$ Clique

Soit ϕ une instance de 3-SAT, c'est-à-dire une formule en forme conjonctive telle que chaque clause de ϕ contienne trois littéraux.

$\phi = (l_1 \vee l_2 \vee l_3) \wedge (l_4 \vee l_5 \vee l_6) \wedge \ldots \wedge (l_{3k-2} \vee l_{3k-1} \vee l_{3k})$. Il y a k clauses

On introduit alors le graphe non orienté G dont l'ensemble des sommets est l'ensemble V = { l_1, …, l_{3k} } de tous les littéraux de ϕ.

Deux sommets de G sont reliés par une arête s'ils n'appartiennent pas à la même clause et s'ils ne sont pas contradictoires (l'un n'est pas égal à la négation de l'autre)

L'ensemble E des arêtes est donc défini de la manière suivante.

$E = \{ (l_i, l_j) \mid ENT[(i-1)/3] \neq ENT[(j-1)/3] \text{ et } l_i \neq \neg l_j \}$

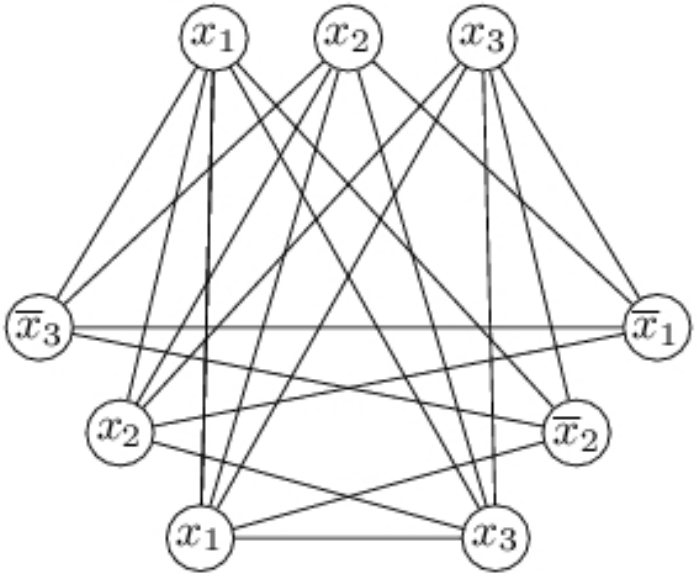
En effet, le numéro de la clause d'un littéral l_i est égal à ENT[(i-1)/3] si les clauses sont numérotées à partir de 0. (0, 1, 2, K-1)

Complexité / Réduction polynomiale

Réduction polynomiale 3-SAT $\leq_P$ Clique

Exemple :

$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$



Φ satisfiable $\Rightarrow$ G contient une clique de taille k

Remarque: deux littéraux d'une même clause ne sont jamais reliés par une arête.

Une clique peut donc contenir au plus un littéral par clause et elle est de taille au plus k.

Complexité / Réduction polynomiale

Réduction polynomiale 3-SAT $\leq_P$ Clique

Montrer : Φ satisfiable $\Rightarrow$ G contient une clique de taille k

Il existe donc une affectation des variables telle que ϕ vaille 1.

Ceci signifie qu'au moins un littéral par clause vaut la valeur 1.

Choisissons un tel littéral dans chacune des clauses pour former un ensemble de k littéraux.

Comme tous ces littéraux valent 1, deux d'entre eux ne peuvent pas être contradictoires et ils sont donc reliés par des arêtes.

C'est donc une clique de taille k dans G.

Complexité / Réduction polynomiale

Réduction polynomiale Clique $\leq_P$ 3-SAT

Montrer : G contient une clique de taille k $\Rightarrow$ Φ satisfiable

Comme les littéraux d'une même clause ne sont pas reliés, cette clique contient un littéral exactement dans chaque clause.

$\phi = (\_ \vee l_1 \vee \_) \wedge (l_2 \vee \_ \vee \_) \wedge \ldots \wedge (\_ \vee l_k \vee \_)$

Montrons alors qu'il existe une affectation qui rend tous ces littéraux l_i (i=1, k) égaux à 1.

Chaque littéral de cette clique est égal à x_i ou à $\neg x_i$.

Pour que ce littéral vaille 1, on impose la valeur 1 ou 0 à la variable correspondante x_i.

Complexité / Réduction polynomiale

Réduction polynomiale Clique $\leq_P$ 3-SAT

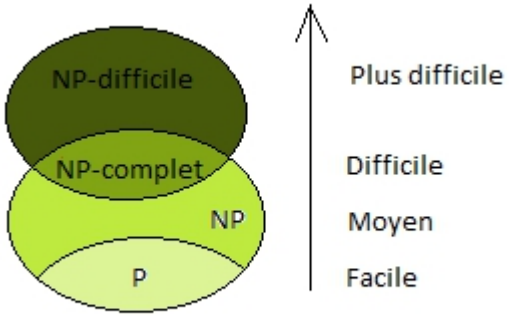
Comme tous les littéraux de la clique sont reliés par une arête, ils ne sont pas contradictoires deux à deux.

Ceci signifie que deux littéraux quelconques de la clique
- concernent deux variables distinctes x_i et x_j avec $i \neq j$ ou alors
- ils concernent la même variable x_i mais ils imposent la même valeur à la variable x_i.

On obtient alors une affectation cohérente des variables apparaissant dans la clique.

En affectant n'importe quelle valeur à chacune des autres variables, on obtient une affectation qui donne la valeur 1 à la formule ϕ.

Complexité / NP-Complétude



Définition

Un problème NP-complet = un problème parmi les plus difficiles de la classe NP.

Plus formellement un problème A est dit *NP-complet*
- si $A \in NP$,
- $\forall B \in NP : B \leq_P A$.

Si seule la seconde condition est vérifiée, le problème A est dit *NP-difficile*.

Résultat remarquable de Stephen Cook (1971) : montre le premier problème NP-complet (SAT)
(Tous les problèmes de la classe NP se réduisent à SAT)
Démo basée sur la machine de Turing

Complexité / NP-Complétude

Théorème

Tous les problèmes NP-complets sont équivalents.

21 problèmes NP-complets de Karp (1972)ont marqué une étape importante de l'histoire de la théorie de la complexité des algorithmes.

Ce sont 21 problèmes réputés difficiles de combinatoire et de théorie des graphes qui sont réductibles entre eux.

SAT / 3-SAT / KNAPSACK / CLIQUE / VERTEX COVER / …

Actuellement, il existe des milliers de problèmes NP-complets (plus de 3000)

Complexité / NP-Complétude

Rappel :
A est NP-difficile : pour tout X ∈ NP : $X \leq_P A$.

Proposition

Si B est NP-complet et si $B \leq_P A$, alors A est NP-difficile.

Démo :
Puisque B est NP-complet, on a
$C \leq_P B$ pour tout problème C de NP.

Si on a aussi $B \leq_P A$,

on obtient $C \leq_P A$
(transitivité).

A est *NP-complet si*
- A ∈ NP,
- ∃ B ∈ NP-complet : $B \leq_P A$

Conséquence:
Montrer qu'un problème A est NP-complet consiste à montrer:
- qu'il est dans la classe NP
- il existe un problème B dans NP-complet qui se réduit à A.

Complexité / NP-Complétude

Preuve de la NP complétude

Prouver la Np complétude d'un langage(problème) A consiste à dérouler les étapes suivantes:

1. Montrer que le langage A est dans NP (deviner une réponse et la vérifier en temps polynomial)
2. Choisir un langage NP-complet B qui peut être réduit à A ($B \leq_p A$).
3. Décrire la fonction de réduction f
4. Montrer que si $x \in B$ alors $f(x) \in A$.
5. Montrer que si $f(x) \in A$ alors $x \in B$.
6. Montrer que f est calculée en temps polynomial

Complexité / NP-Complétude

Question P = NP ?

Stephen Cook à facilité le problème!

Au lieu de montrer que tout problème de NP est dans P (peut etre résolu en temps polynomial),

il suffit de montrer tout simplement

qu'un seul problème NP-complet est dans P.

Complexité / NP-Complétude

Exemple de problèmes NP-complets

Knapsack *(Sac au dos)*

On a un ensemble de n éléments. L'élément i à un poids w[i] et une importance b[i].

Sélectionner un sous ensemble d'éléments de telle sorte que le poids total est inférieur ou égal à W et le total d'importance est supérieur ou égal à B?

Sac à dos : W= 20 Kg // B= 60%
Que faut-il choisir parmi un ensemble d'outils ayant des poids et utilités donnés?

Nombre de possibilités 2^n

Complexité / NP-Complétude

Exemple de problèmes NP-complets

Subset Sum

Problème :

Etant donne un ensemble de n entiers.

Existe-il un sous ensemble dont la somme de ses éléments est égal à B?

Partition

Problème : *Etant donné un ensemble de n entiers, peut-on le partitionner en deux classes de même somme ?*

Complexité / NP-Complétude

Exemple de problèmes NP-complets

Cycle Hamiltonien

Instance: graphe non oriente G = (V,E).

Question: *Est-ce que G contient un cycle Hamiltonien?*

Un graphe non orienté a un cycle Hamiltonien s'il existe un cycle dans le graphe qui visite chaque nœud du graphe exactement une fois.

Complexité / NP-Complétude

Exemple de problèmes NP-complets

Color

Instance: Un graphe G = (V;E), k un entier

Question: *Existe-t-il une façon de colorier les sommets avec k couleurs de telle sorte qu'aucune arête n'aie les extrémités de la même couleur?*

3 - Color

Instance: Un graphe G = (V;E)

Question: *Existe-t-il une façon de colorier les sommets avec 3 couleurs de telle sorte qu'aucune arête n'aie les extrémités de la même couleur?*

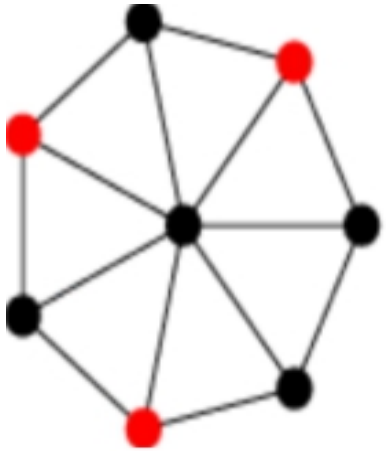
Complexité / NP-Complétude

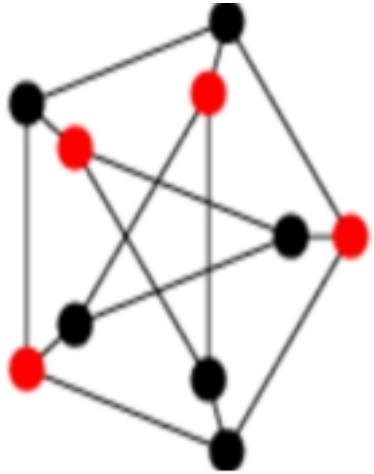
Exemple de problèmes NP-complets

Independent Set

Instance: Etant donné un graphe non orienté G(V,E) avec n nœuds; un entier positive k ≤ n.

Question: Est-ce que G a un ensemble indépendant de taille ≥ k, c'est-à-dire un sous ensemble W de V ayant au moins K nœuds de telle sorte qu'aucune paire de nœuds dans W est reliée par un arc dans E?





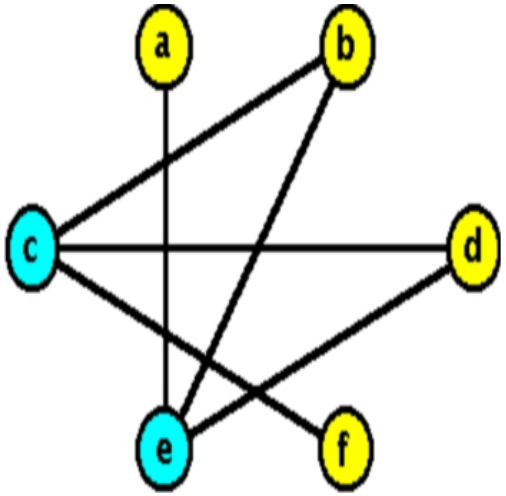
Complexité / NP-Complétude

Exemple de problèmes NP-complet

Vertex Cover (VC)

Instance: Etant donne un graphe non orienté avec n nœuds G(V,E); un entier positif k , k≤n.

Question: *Existe-t-il un sous ensemble W de V ayant une taille k et tel que pour chaque arc (u,v) dans E, au moins une extrémité (u ou v) appartient à W ?*



Complexité / NP-Complétude

Pour montrer la NP-complétude de X (X dans NP), il faut utiliser la réduction Y

X	Y
3-SAT	SAT ≤ 3-SAT
CLIQUE	SAT ≤ CLIQUE
CLIQUE	INDEPENDANCESET ≤ CLIQUE
CLIQUE	3-SAT ≤ CLIQUE
VERTEXCOVER	INDEPENDANCESET ≤ VERTEXCOVER
VERTEXCOVER	CLIQUE ≤ VERTEXCOVER
INDEPENDANTSET	3SAT ≤ INDEPENDANTSET
COLOR	3-SAT ≤ COLOR
3-COLOR	3-SAT ≤ 3-COLOR
COLOR	3-COLOR ≤ COLOR
SUBSET-SUM	3-SAT ≤ SUBSET-SUM
KNAPSACK	PARTITION ≤ KNAPSACK
HAMILTON	3-SAT ≤ HAMILTON

Complexité / NP-Complétude

Rappel (Proposition)

Si B est NP-complet et si B $\leq_p$ A, alors A est NP-difficile

Exemple de problèmes NP-difficiles

TSP (Travelling Salesman Problem)

Instance: graphe non oriente G = (V,E).

On associe à chaque arc un poids(une distance)

Question: *Est-ce que G contient un cycle Hamiltonien de poids minimal?*

$\text{HAMILTON} \leq_P \text{TSP}$

TSP n'est pas dans NP

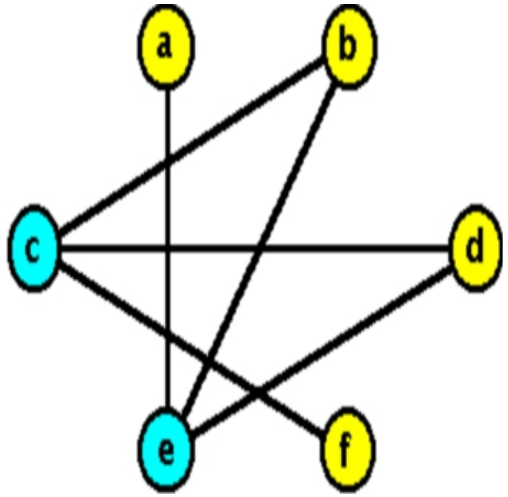
Complexité / NP-Complétude

Exemple de problèmes NP-difficiles

Vertex Cover (VC)

Instance: Etant donne un graphe non orienté avec n nœuds G(V,E).

Question: *Existe-t-il un sous ensemble W de V ayant une taille maximale et tel que pour chaque arc (u,v) dans E, au moins une extrémité (u ou v) appartient à W ?*



CLIQUE $\leq_P$ VERTEXCOVER

VERTEXCOVER n'est pas dans NP

Complexité / NP-Complétude

Exemple de problèmes NP-difficiles

Color

Instance: Un graphe G = (V;E)

Question: *Existe-t-il une façon de colorier les sommets avec un minimum de couleurs de telle sorte qu'aucune arête n'aie les extrémités de la même couleur?*

$3\text{-COLOR} \leq_p \text{COLOR}$

COLOR n'est pas dans NP

Complexité / NP-Complétude

Exemple de problèmes (Non dans NP-complet et non dans NP-difficile)

Factorisation en nombres entiers(ou premier)

Factorisation de grands nombres (centaines de bits)

Utilisée dans les algorithmes cryptage/décryptage (code RSA)

Complexité : exponentielle

Une solution peut être vérfiée en temps polynomial -> dans NP

Pas de preuve qu'il est dans P ou NP-complet

(il n'est donc pas NP-difficile)

Dans la théorie des nombres, l'algorithme le plus efficace est celui de A.K Lenstra et Co (1991) pour factoriser des entiers supérieurs à r^x (r petit et x grand).

Complexité pour factoriser un entier n (composé de $\lfloor \log_2 n \rfloor + 1$ bits)

$$Exp(c\ (\log n)^{1/3}\ (\log \log n)^{2/3}\ ))$$

sachant que $Exp(x) = e^x$
c = 1.93
log : logarithme nipérien (base e)

Complexité / NP-Complétude

Si P = NP ?

Une question qui fascine beaucoup de chercheurs :

Tout problème NP ou NP-complet peut-il être résolu en temps polynomial ?

Entraîne un changement radical dans la science informatique

Exemple : Il faudrait revoir tous les algorithmes liés a cryptographie qui utilise la factorisation (décomposition en nombre premier)

Sinon les algorithms deviennent vulnérables aux attaques.

Complexité / NP-Complétude

Conclusion

Principales classes étudiées :

P, NP, NP-complet, NP-difficile

P : problèmes
- résolus rapidement

NP : problemes
- vérifiés rapidement
- résolus lentement

NP-complet : problemes
- vérifiés rapidement
- résolus lentement
- peuvent être réduits à tout autre problème np-complet

NP-difficile : problemes
- vérifiés lentement,
- résolus lentement
- peuvent être réduit à tout autre problème NP