Leçon 31 : Classes P et NP. Problèmes NP-complets. Exemples.

Prérequis :

On présente ici les notions informellement.

I - Introduction

I.1 - Problèmes de décision

Def 1 : Un problème de décision Π est la donnée d'un ensemble E d'instances et d'un ensemble $E^+ \subset E$ d'instances positives.

Exemple 2 : 1) $E = \mathbb{N}$; $E^+ = \{ n \in \mathbb{N} \mid n \text{ premier} \}$

2) E = { formules du 1er ordre } ; E^+ = { formules universellement vraies }

Notations 3 : Pour un pb de décision $\Pi = (E, E^+)$, on identifie parfois Π et E^+, en considérant E implicite.

Exemple 4 : SAT est le problème de savoir si une formule propositionnelle est satisfiable ou pas. On note parfois $x \wedge y \in SAT$, $x \wedge \neg x \notin SAT$.

Exemple 5 : Problème App

Entrée : T un tableau d'entiers, $x \in \mathbb{N}$

Sortie : $x \in T$

exercice : le réécrire sous forme E, E^+

Remarque 6 : Quand on considère E en informatique, on ne considère pas les objets mathématiques mais un codage de ces objets. On peut ajouter ce codage à la définition d'un problème, sinon on considère un codage "raisonnable".

Def 7 : La taille d'une instance d'un problème est la taille du codage de l'instance,

Exemple 8 : Des tailles pour des structures de données classiques :

* $O(\log_2(n))$ pour $n \in \mathbb{N}$ (codage binaire des entiers)
* $O(|T| \times k)$ pour un tableau T contenant des éléments de taille k
* $O(|A| \times \log_2(|S|))$ pour un graphe $G = (S, A)$ (liste d'arêtes)

I.2 - Algorithmes

On appelle ici algorithme un programme C avec mémoire infinie.

Def 9 : On dit qu'un algorithme A résout un problème Π si pour toute entrée e de Π, $A(e)$ termine et renvoie la sortie correspondante.

Exemple 10 : L'algorithme qui calcule le pgcd par soustractions successives puis compare le résultat à 1 résout le problème Π "premier entre eux" : $E : \mathbb{N} \times \mathbb{N}$, $E^+ = \{ a, b \in \mathbb{N}^2 \mid a \wedge b = 1 \}$.

Def 11 : On dit qu'un algorithme a une complexité en $f : E \times S \to \mathbb{N}$ si pour toute entrée e, et toute sortie s, l'algorithme termine en moins de $f(e, s)$ opérations élémentaires.

Remarque 12 : Ici, "opérations élémentaires" est défini informellement comme une opération raisonnable (affectation, opération arithmétique...). Le formalisme vient avec les machines de Turing qui sont hors programme.

Exemple 13 : dans l'exemple 10, l'algorithme a une complexité en
$$f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, \quad a, b \mapsto a + b$$

Remarque 14 : Si $g > f$ et A a une complexité en f, elle l'a aussi en g.

→ Avec notre définition, un algorithme linéaire est quadratique.

Ne rien écrire hors du cadre.

II - La classe P et NP

II.1) La classe P

Def 15 : On dit qu'un algorithme A est polynomial s'il existe un polynome T et une fonction f tq A est de complexité f et $\forall (e,s)$, $f(e,s) \leq T(|e|)$ où $|e|$ est la taille de l'instance e.

Remarque 16 : L'algorithme qui teste si i divise n pour tout $i \leq n$ a une complexité $O(n) = O(2^{|n|})$ donc n'est pas polynomial.

Def 17 : La classe P contient tous les problèmes de décision Π pour lesquels il existe un algorithme polynomial qui résout Π.

Exemple 18 : App $\in$ P. L'algorithme polynomial associé parcourt les éléments de T jusqu'à trouver x (ou la fin de T), en $O(|T|)$.

Remarque 19 : P contient des problèmes de décision, pas des algorithmes.

Exemple 20 : Pour le problème App sur l'instance (T,x), si je fais une boucle de taille $2^{|T|}$ avant la recherche, l'algorithme est exponentiel et résout un problème de P.

Def 21 : Soit Π_1, Π_2 deux problèmes de décision. On dit que Π_1 se réduit polynomialement vers Π_2, s'il existe $f : E_1 \to E_2$ tq :

- $\forall e_1 \in E_1$, $\Pi_1(e_1) = \Pi_2(f(e_1))$ (réponse préservée)
- $\exists$ un algorithme polynomial qui calcule f (en toute rigueur, on pourrait dire résout)

On note $\Pi_1 \leq \Pi_2$.

Remarque 22 : Informellement, Π_1 est plus simple que Π_2, à un polynome près.

Exemple 23 : Max_T qui pour une instance (T,x) dit si x est le max de T, se réduit polynomialement en Min_T en remplaçant tous les éléments de T par leur opposé, et x par $-x$.

Propriété 24 : La relation être une réduction polynomial de est transitive.

Propriété 25 : Si $\Pi_1 \leq \Pi_2$ et $\Pi_2 \in P$, alors $\Pi_1 \in P$.

Développement 1 : 2-SAT $\in$ P, par réduction vers composante connexe.

II.2 - La classe NP

Def 26 : Un vérificateur v associé à un pb de décision $\Pi(E, E^+)$ est un algorithme qui prend en entrée un couple $(e,c) \in E \times C$ où C est un ensemble d'éléments appelés certificats, et qui renvoie un booléen tq : $e \in E^+$ ssi $\exists c \in C$ tq $v(e,c) = 1$.

Def 27 : La classe NP est la classe des problèmes de décision qui admettent un vérificateur polynomial.

Propriété 28 : SAT $\in$ NP. Le vérificateur associé prend comme certificat les valeurs de vérité des variables et en déduit la valeur de la formule.

Exemple 29 : Voyageur de Commerce

$E = G = (V,A)$ graphe complet non orienté, $\omega : A \to \mathbb{N}$, $k \in \mathbb{N}$

Question : Existe-t-il un cycle passant par tous les sommets une unique fois, et de poids total $\leq k$?

Voyageur de Commerce est dans NP.

Exercice 30 : Donnez le vérificateur associé.

Remarque 31 : $\Pi \in NP \Rightarrow$ toute instance positive de Π admet un certificat polynomial. Le rôle entre E^+ (instances positives) et $E \setminus E^+$ (instances négatives) n'est pas symétrique. En les échangeant dans la définition, on obtient la classe co-NP.

Nom : Sujet : 31 Jury : Page : 3/3

Propriété 32 : $\Pi_1 = (E, E^+) \in NP \Leftrightarrow \Pi_2 = (E, E \setminus E^+) \in co\text{-}NP$.

Exemple 33 : Le problème de savoir si le nombre de facteurs premiers de n est $\leq k$ est dans NP et dans co-NP.

Propriété 33 : $P \subset NP$ (et $P \subset co\text{-}NP$)

dem : Soit $\Pi = (E, E^+) \in P$. Alors il existe A polynomial qui résout Π. Le vérificateur $v : E \times \emptyset \to \{0,1\}$ qui renvoie $A(e)$ est polynomial.

Remarque 34 : Savoir si l'inclusion est stricte ou si $P = NP$ est un grand problème informatique ouvert.

III - NP-complétude

III.1 - Définition

Def 35 : Un pb de décision Π est NP-difficile si tout pb de NP se réduit polynomialement vers lui.

Intuitivement, Π est plus dur que tt pb de NP.

Def 36 : $\Pi \in$ NP-complet si Π est NP-difficile <u>et</u> $\Pi \in NP$.

Théorème 37 : (théorème de Cook) SAT est NP-complet.

Developpement 1 : Illustration du théorème de Cook.

III.2 - Quelques pb NP-complets

Propriété 38 : Π_1 est NP-complet ssi $\Pi_1 \in NP$ et $\exists\, \Pi_2$ NP-complet tq $\Pi_2 \leq \Pi_1$

Méthode 39 : (Pour montrer Π_1 NP-complet)

1) Prouver $\Pi_1 \in NP$
2) Choisir Π_2 NP-complet
3) Construire une transformation polynomiale de Π_2 vers Π_1
4) Montrer que la réponse est préservée

Developpement 2 : 3-SAT est NP-complet (réduction depuis SAT)

Exemple 40 : Problème : Chemin hamiltonien

Entrée : $G = (V, A)$ non orienté,

Question : Existe-t-il un cycle de G qui prend une unique fois par chaque sommet ?

Propriété 41 : Chemin hamiltonien est NP-complet. (réduction depuis 3-SAT)

Propriété 42 : Voyageur de Commerce est NP-complet (réduction depuis chemin hamiltonien).

IV - Classes P et NP en informatique

IV.1 - Pour les problèmes qui ne sont pas de décision ?

Remarque 43 : La classe P s'étend bien aux problèmes qui ne sont pas des problèmes de décision, mais qu'en est-il pour la classe NP ?

Méthode 44 : Pour les pb d'optimisation (quelle est la valeur min/max de ...) on se ramène à un pb de décision associé en ajoutant un seuil. (existe-il une solution de valeur $\leq k$?). En faisant une dichotomie sur le seuil, on fait le chemin inverse.

Exemple 45 : Pour voyageur de Commerce, si on sait résoudre le pb de décision, on se ramène au pb de minimisation en faisant une dichotomie sur k (compris entre 0 et $\sum \omega_{ij}$), ce qui est polynomial car on lance l'algorithme $\log(\sum_{(i,j) \in A} \omega_{ij})$ fois.

Exercice 46 : Comment faire si k a priori non borné ?

IV.2 - P ou NP-complet ?

Remarque 47 : Il est souvent facile de montrer $\Pi \in NP$. On se demande alors plutôt P ou NP-complet ?

P	NP-complet

IV.3 - Plus que NP

Proposition : Il existe des pb $\Pi \notin NP$. Par exemple le pb de l'arrêt.

Ne rien écrire hors du cadre.