NOM : BOLLE **Prénom :** Quentin **Jury :**

Algèbre ← Entourez l'épreuve → **Analyse**

Sujet choisi : Classe de Complexité. Exemples. 915

Autre sujet :

Papadimitriou, Computational Complexity

Motivation : ne pas étudier les problèmes algorithmique séparément, mais les réunir pour établir des théorèmes généraux.

I) Outils

1) Modèle de calcul

*) On utilisera une machine de Turing :
- avec un ruban d'entrée en lecture seule
- avec des rubans de travail
- avec un ruban de sortie en écriture seule

*) La machine peut être déterministe ou non-déterministe (sans mention : "machine" → déterministe)

*) On verra que l'on peut choisir des machines de Turing avec des critères spécifiques (exemple : un seul ruban de travail).

2) Coût

*) Si M est une machine travaillant sur l'entrée x, on note $T(M,x)$ le temps de calcul (que l'on supposera fini) et $E(M,x)$ le nombre de cases utilisées par les rubans de travail.

*) On note $T_M(n) = \max\limits_{|x|=n} T(M,x)$

et $E_M(n) = \max\limits_{|x|=n} E(M,x)$

On a ainsi défini la complexité temporelle (resp. spatiale) de la machine.

3) Problème de décision et codage

*) Un problème de décision prend en argument des données (ex : graphe ; nombre) et renvoie "oui" ou "non".

Exemple : CONNEXITE : Argument : un graphe G + deux sommets x et y

Renvoie "oui" s'il existe dans G un chemin de x vers y. Renvoie "non" sinon

*) Pour relier un problème de décision aux machines de Turing, on associe à chaque problème un langage décidable. Cette association se réalise grâce à un codage.

Exemple : pour un graphe → liste des sommets et arcs
→ matrice d'adjacence
- pour un entier → représentation binaire
→ représentation unaire

4) Réduction et complétude

*) Une réduction des langages L_1 vers L_2 est une fonction calculable qui vérifie la propriété suivante : $x \in L_1 \iff f(x) \in L_2$

*) Soit M la machine de Turing associée à la réduction f.

→ Si T_M est bornée par une fonction polynômiale on parle de réduction (en temps) polynômiale

→ Si E_M est bornée par une fonction $n \mapsto c \log n$, on parle de réduction (en espace) logarithmique

*) On note $L_1 <_p L_2$ (resp. $L_1 <_{log} L_2$) s'il existe une réduction polynômiale (resp. logarithmique) de L_1 vers L_2.

*) On pose $< \in \{<_p, <_{log}\}$ et L un langage

→ Si, pour tout langage L' d'une classe C, on a $L' < L$, alors on dit que L est <u>C-difficile</u>

→ Si on a en plus $L \in C$, on dit que L est <u>C-complet</u>

<u>Remarque</u> : la complétude est un outil fondamental en théorie de la complexité. Un problème C-complet est un problème qui renferme en lui l'essence de la complexité de C : c'est un lien fort entre algorithmique et complexité.

II) <u>Comment classer ?</u>

1) <u>Inutilité des constantes multiplicatives</u>

*) Si $T_M(n) = \Theta(T_{M'}(n))$ ou $E_M(n) = \Theta(E_{M'}(n))$, les deux théorèmes suivants montrent que M et M' ont la même "puissance".

*) <u>Théorème d'accélération linéaire</u> (version temporelle) :
Soit L un langage décidé par une machine M et soit $\varepsilon > 0$. Alors L est décidé par une machine M' où
$T_{M'}(n) = \varepsilon\, T_M(n) + n + 2$.

*) <u>Théorème d'accélération linéaire</u> (version spatiale) :
Soit L un langage décidé par une machine M et soit $\varepsilon > 0$. Alors L est décidé par une machine M' où
$E_{M'}(n) = \varepsilon\, E_M(n) + 2$

*) Bilan : on note $TEMPS(f(n))$ (resp. $ESPACE(f(n))$) l'ensemble des langages décidés par une machine M où $T_M(n) = O(f(n))$ (resp. où $E_M(n) = O(f(n))$).

2) <u>Codage fidèle</u>

*) Différents codages existent pour un même problème. Cela peut modifier sensiblement la complexité.

*) Deux codages sont dits <u>fidèles</u> si l'on peut passer de l'un à l'autre par une fonction "polynômiale"

<u>Exemples</u> : • Pour les graphes, le codage par liste est fidèle au codage par matrice
• pour les entiers, la représentation unaire n'est pas fidèle à la représentation binaire

*) Les classes de complexité doivent en tenir compte !
Si $TEMPS(f(n)) \in C$, on veut que $TEMPS(f(n^k)) \in C$, $k \in \mathbb{N}^*$
Pareil pour la complexité spatiale.

3) <u>Bilan</u>

*) On peut désormais définir des classes usuelles :

→ Temps : $P = \bigcup_{k\in\mathbb{N}} TEMPS(n^k)$, $EXP = \bigcup_{k\in\mathbb{N}, c>1} TEMPS(c^{n^k})$

→ Espace : $L = ESPACE(\log n)$, $PSPACE = \bigcup_{k\in\mathbb{N}} ESPACE(n^k)$

En non déterministe :

→ Temps : $NP = \bigcup_{k\in\mathbb{N}} NTEMPS(n^k)$, $NEXP = \bigcup_{k\in\mathbb{N}, c>1} NTEMPS(c^{n^k})$

→ Espace : $NL = NESPACE(\log n)$, $NPSPACE = \bigcup_{k\in\mathbb{N}} NESPACE(n^k)$

*) On note $co\text{-}C = \{\overline{L} \mid L \in C\}$ le complément de C.

On a $co\text{-}P = P$, $co\text{-}EXP = EXP$, $coL = L$, ...

Mais on a $co\text{-}NP \overset{?}{=} NP$.

III) Résultats principaux de la théorie

1) Relations générales

* On a
  - TEMPS $(f(n)) \subset$ NTEMPS $(f(n))$
  - NTEMPS $(f(n)) \subset$ ESPACE $(f(n))$
  - ESPACE $(f(n)) \subset$ NESPACE $(f(n))$
  - NESPACE $(f(n)) \subset$ TEMPS $(c^{f(n)})$ $(c > 1)$

* Bilan : on a $L \subset NL \subset P \subset NP \subset PSPACE \subset EXP$

2) Hierarchie

* Une fonction f est constructible en temps (resp. en espace) s'il existe une machine M construisant tout élément de taille n après $f(n)$ transitions (resp. en $f(n)$ espace).

* Théorème de hierarchie temporelle :
Si $f_2(n) \geq n$ est constructible en temps et si $f_1(n) \log f_1(n) = o(f_2(n))$ alors on a TEMPS $(f_1(n)) \subsetneq$ TEMPS $(f_2(n))$.

Théorème de hierarchie spatiale :
Si $f_2(n) \geq \log n$ est constructible en espace et si $f_1(n) = o(f_2(n))$, alors on a ESPACE $(f_1(n)) \subsetneq$ ESPACE $(f_2(n))$.

* Bilan : $P \subsetneq EXP$ et $L \subsetneq PSPACE$

3) Deux théorèmes majeures sur le non-déterminisme spatiale

* On suppose $f(n) \geq \log n$ constructible en espace.

DEV 1 ← * Théorème de Savitch : NESPACE $(f(n)) \subseteq$ ESPACE $(f(n)^2)$
Corollaire : PSPACE = NSPACE

* Théorème d'Immerman-Szelepscényi : NESPACE $(f(n))$ = co-NESPACE $(f(n))$
Corollaire : NL = co NL

3) Sur la classe NP

* Caracterisation (certificat)
Un langage L est dans NP ssi il existe un polynôme P et une machine M tel que :
$x \in L \Leftrightarrow \exists\, u \in \underbrace{\Sigma}_{\text{alphabet}}^{P(|x|)}$, M accepte $\langle x, u \rangle$

* NP-complétude :
On s'intéresse à la réduction polynômiale.
(on montre que $L_1 <_{log} L_2 \Rightarrow L_1 <_p L_2$)
Soit les problèmes suivants :
→ CIRCUIT-SAT : peut-on satisfaire un circuit booléen ?
→ SAT : peut-on satisfaire une formule logique sous forme normale conjonctive ?

Théorème de Cook : SAT est NP-complet (par réduction logarithmique) → DE
Demonstration : par l'intermédiaire de CIRCUIT-SAT

Problèmes NP-Complet : 3SAT, CHEMIN-HAMILTONIEN, COLORIAGE, SAC-A-DOS

Conjecture : on imagine que $P \neq NP$, ce qui est équivalent à affirmer que l'on ne peut résoudre aucun problème NP-Complet en temps polynômial.