TD 6 - complexité en temps introduction

Encodage de l'entrée

Encodage. Supposons que chaque instance puisse être représentée par une chaîne finie de symboles. Nous appellons *fonction d'encodage* (ou *schéma d'encodage*) la fonction qui à une instance associe son encodage sous la forme d'une chaîne de symboles.

Un *schéma d'encodage* est une façon d'encoder et de décrire les instances d'un problème. Le schéma d'encodage utilisé diffère au plus polynomialement d'un autre schéma d'encodage, à condition que ceux-ci soient raisonnables. La notion de *raisonnable* est difficile à définir, mais les deux conditions suivantes en donnent une idée :

- l'encodage d'une instance doit être concis et non édulcoré avec des informations ou symboles inutiles;
- les nombres apparaissant dans l'instance doivent être représentés en binaire (ou décimal, ou octal, ou dans n'importe quelle base hormis en unaire).

Finalement, un problème est caractérisé par le langage (c'est-à-dire l'ensemble des mots) des encodages de ses instances positives. Résoudre un problème revient à reconnaître les instances positives de ses encodages.

Exercice 1 *Binaire vs unaire*

Soient deux nombres entiers x et y.

- ▲ Notons $|x|$ (respectivement $|y|$) le nombre de bits nécessaires à la représentation (c'est-à-dire à l'encodage) de x (respectivement de y). Si ces nombres sont encodés en binaire, combien de bits sont nécessaires à leurs représentations.
- ◆ Écrire un algorithme qui effectue l'addition de x et y représentés en binaire.
- ⬟ Notons $k = \max(|x|, |y|)$. On observe que $|x| + |y| \leqslant 2 \cdot k$. Quelle est la complexité de votre algorithme en fonction de k ?
- ⬢ Supposons maintenant que x et y soient représentés en unaire. Donnez un algorithme qui effectue l'addition de ces deux nombres. Quelle est la complexité de votre algorithme ?

Exercice 2 *De l'importance d'un encodage* raisonnable *de l'entrée*

Considérons le problème suivant et l'algorithme SolveKS pour le résoudre :

Problème **Sac à dos**

entrée :
- un ensemble de n entiers positifs $p_1, \dots, p_n \in \mathbb{N}$, les poids des objets ;
- un ensemble de n entiers positifs $v_1, \dots, v_n \in \mathbb{N}$, les valeurs des objets ;
- un entier positif $W \in \mathbb{N}$, la capacité du sac ;

sortie : un sous-ensemble $S \subseteq \{1, \dots, n\}$ des objets, tel que $\sum_{i \in S} p_i \leqslant W$ et qui maximise la valeur $opt = \sum_{i \in S} v_i$.

Algorithm 1: SolveKS$(p_1, \dots, p_n, v_1, \dots, v_n, W)$

```
for j = 0 to W do
    T[0, j] ← 0
for i = 1 to n do
    for j = 0 to W do
        if p_i > j then T[i, j] ← T[i − 1, j] ;
        else T[i, j] ← max{T[i − 1, j] , T[i − 1, j − p_i] + v_i} ;
return opt = T[n, W]
```

- ▲ Établir la complexité de l'algorithme SolveKS.
- ◆ Cette complexité est-elle polynomiale en la taille de l'entrée ?

Problèmes de décision

Problème de décision. Un problème de décision est une question (sur l'instance du problème) qui admet pour réponse « oui » ou pour réponse « non ». On peut ainsi distinguer les instances positives des instances négatives au problème.

Exercice 3 *c'est oui ou c'est non ?*
Formulez chacun des problèmes suivants sous la forme d'un problème de décision.

- ▲ Problème **Sac à dos** ;
- ◆ Problème **Coloration**
  entrée : un graphe $G = (V, E)$.
  sortie : une coloration propre de G utilisant le moins de couleurs possibles.
- ⬟ Problème **CNF-SAT**
  entrée : – un ensemble $X = \{x_1, \dots, x_n\}$ de variables booléennes ;
  — une formule propositionnelle $F = c_1 \wedge c_2 \wedge \dots \wedge c_m$ où chaque clause c_i est de la forme $l_1 \vee l_2 \vee \dots \vee l_{k_i}$; chaque littéral est soit une variable booléenne x_i, soit sa négation $\overline{x_i}$.
  sortie : une affectation de valeurs aux variables booléennes de X, tel que la formule F soit satisfaite.
- ⬢ Problème **Multiplication de matrices**
  entrée : deux matrices A et B.
  sortie : la matrice C résultant du produit $A \cdot B$.

Modèle de calcul : la machine à k rubans

Machine à k rubans. Une machine de Turing $\mathcal{M}$ à k rubans possède k rubans munis d'une tête de lecture-écriture. Quand $\mathcal{M}$ est dans l'état q et qu'elle lit un symbole sur chacun des k rubans, $\mathcal{M}$ remplace chacun de ces k symboles, se déplace à droite ou à gauche (indépendamment) sur chacun des rubans, et passe dans un nouvel état.

La fonction de transition est définie par $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$.

Exercice 4 *efficacité d'une machine à 1 ruban et d'une machine à 2 rubans*
Regardons le langage $A = \{0^k 1^k : k \geqslant 0\}$.

- ▲ Proposez une machine de Turing à un seul ruban pour décider le langage A. Quel est son temps d'exécution ?
- ◆ Décrivez le fonctionnement d'une machine de Turing (toujours à un seul ruban) pour décider le langage A qui s'exécute en temps $O(n \cdot log(n))$.
- ⬟ Supposons maintenant que nous disposions d'une machine à deux rubans. Décrire le fonctionnement de cette machine pour décider A en temps $O(n)$.

Exercice 5 *slow down à un ruban*
Notons $\mathcal{M}$ une machine de Turing déterministe à k rubans dont le temps d'exécution est borné par une fonction $t(n)$, telle que $t(n) \geqslant n$.

Montrer que $\mathcal{M}$ peut être simulée par une machine de Turing $\mathcal{M}'$ à un seul ruban, en temps $O(t^2(n))$.