Chapitre 7 : Codage des entrées

ENSIIE - Théorie de la complexité

Dimitri Watel (dimitri.watel@ensiie.fr)

2022

Première remarque

Compresser une instance

Si on peut compresser une instance, on augmente artificiellement la complexité du problème, qui dépend de la taille de cette instance. On peut voir cette augmentation comme le temps qu'il faut pour décompresser l'instance. Inversement, on peut tenter décompresser l'instance pour la rendre plus simple. Certains problèmes compressés résistent à la décompression et restent malgré tout difficiles : ces problèmes sont appelés *forts*. Les autres sont appelés *faibles*.

Taille de l'entrée usuelle

Définition

La taille de l'entrée est définie comme le nombre de bits nécessaires pour l'encoder.

Exemples

- pour un graphe : nombre de nœuds, nombre d'arêtes
- pour une liste : nombre d'éléments
- pour un entier k : $\log_2(k)$
- pour une liste L d'entiers : $\sum_{k \in L} \log_2(k)$
- ...

On note généralement $|x|$ ou n cette taille.

Codage binaire VS Codage unaire

Codage binaire

Coder un entier k en *binaire* sur une machine de Turing consiste à l'écrire en base 2 avec $\log_2(k)$ bits entourés par deux symboles blancs. La place mémoire de k est $log_2(k) + 2$.

Codage unaire

Coder un entier k en *unaire* sur une machine de Turing consiste à l'écrire avec k cases remplies de 1 entourées par deux cases blanches. La place mémoire de k est $k + 2$.

Complexité et encodage

Constatation

Soit un problème de décision Π, une instance x de Π contenant un entier W et un algorithme $\mathcal{A}$ de complexité $O(W)$ résolvant Π.

- Si W est codé en unaire, $\mathcal{A}$ est polynomial.
- Si W est codé en binaire, $\mathcal{A}$ est exponentiel.

NP-Complétude forte

Définition

Soit un problème de décision Π, on dit que Π est *fortement NP-Complet* (ou *NP-Complet au sens fort*) s'il est NP-Complet quand on encode les entiers contenus dans les entrées en unaire.

NP-Complétude faible

Définition

Soit un problème de décision Π, on dit que Π est *faiblement NP-Complet* (ou *NP-Complet au sens faible*) s'il est NP-Complet mais qu'il existe un algorithme polynomial pour le résoudre si on encode les entiers contenus dans les entrées en unaire.

Taille de l'entrée usuelle : 2^{e} essai

Définition

Soit une entrée x contenant un ensemble x_0 d'objets non numériques et un ensemble de k entiers $(x_1, x_2, \ldots, x_k)$, on définit la taille de x de plusieurs manières :

- sa taille totale $|x|$ usuelle : le nombre de bits pour tout encoder ;
- la taille $l(x)$ indépendante de la valeur des entiers : $|x_0| + k$;
- la taille $\max(x)$ relative aux entiers : $\max_{[\![1;k]\!]} x_i$.

NP-Complétude forte : définition alternative

Définition

Soit un problème de décision Π, on dit que Π est *fortement NP-Complet* (ou *NP-Complet au sens fort*) s'il est NP-Complet quand on se restreint aux instances où $\max(x)$ est polynomialement borné par rapport à $I(x)$.

Théorème

Les deux définitions de NP-Complétude forte sont équivalentes.

NP-Complétude faible

Définition : complexité pseudo-polynomiale

Soit un problème de décision Π, un algorithme résolvant Π est *pseudo-polynomial* si sa complexité en temps est polynomiale en $I(x)$ et $\max(x)$.

Définition

Soit un problème de décision Π, on dit que Π est *faiblement NP-Complet* (ou *NP-Complet au sens faible*) s'il est NP-Complet mais qu'il existe un algorithme pseudo-polynomial pour le résoudre.

Théorème

Les deux définitions de NP-Complétude faible sont équivalentes.

Fort $\neq$ Faible

Théorème

Si P $\neq$ NP alors les problèmes NP-Complets au sens fort ne sont pas NP-Complets au sens faible et inversement.

Montrer qu'un problème est fortement NP-Complet.

Théorème

Soit Π un problème NP-Complet ne contenant pas d'entier, alors Π est NP-Complet au sens fort.

Théorème

Soit Π_1 un problème fortement NP-Complet et Π_2 un problème de décision. S'il existe une réduction polynomiale de Karp telle que, pour toute instance x de Π_1 transformée en instance y de Π_2 , $\max(y)$ est borné polynomialement par rapport à $I(x)$ et/ou $\max(x)$ alors Π_2 est NP-Complet au sens fort.

Dernière remarque

Compresser une instance

On a vu la compression au travers de l'exemple le plus courrant : le codage des entiers. Il est possible de reproduire ce schémas avec d'autres objets comme les graphes succints ou les formules booléennes succintes.

Machine de Turing arithmétique

Définition

Une machine de Turing arithmétique est une machine de Turing capable de manipuler les entiers en temps constant, quelle que soit leur taille, par exemple avec les opérations suivantes :

- déplacement sur la bande d'un entier à l'autre
- copie d'un entier sur la bande
- somme
- produit
- division
- comparaison
- ...

Problème fortement polynomial

Définition

Un problème de décision Π est *fortement polynomial* s'il existe une machine de Turing arithmétique qui résoud toute instance x de Π avec une complexité en temps polynomiale en $I(x)$ et une complexité en espace polynomiale en $|x|$.

Théorème

Tout problème de décision Π fortement polynomial est dans P.

Problème faiblement polynomial

Définition

Un problème de décision Π est *faiblement polynomial* s'il est dans P et s'il n'est pas fortement polynomial.

La forte/faible polynomialité n'a pas de lien avec la compression ou la décompression des entiers : on cherche ici à trouver des problèmes dont la complexité s'affranchi le plus possible de la taille des entiers.