INFO-F408: Computability & complexity

Rémy Detobel

23 Octobre, 2017

1 Time Complexity

Cela n'a pas de sens de regarder le temps réel d'exécution car cela va dépendre de la machine qui exécute le code. On va donc plutôt se concentrer sur les étapes exécutées.
On va donc réduire un programme à une fonction prenant en paramètre la taille de l'input, la complexité du paramètre utilisé pour exécuter cette fonction. La complexité du comportement de la fonction va *a priori* varier selon cette entrée. On va donc se concentrer sur la complexité dans le pire des cas.
En d'autres mots :
Nombre d'étape de la machine de Turing $\Rightarrow f : \mathbb{N} \to \mathbb{N}$: nombre maximum d'étapes de la machine de Turing sur une entrée de taille n.
Pour cela, on va utiliser la notation "grand O".

Par exemple : $f(n) = O(g(n))$.
Qui traduit la relation suivante : $\exists n_0 \in \mathbb{N}, c \in \mathbb{R} | \forall n > n_0 : f(n) \leqslant c \cdot g(n)$.

$$f(n) = n$$
$$f(n) = O(n \log n)$$

Dans la même idée, on peut utiliser Ω qui lui utilise donc un $\geqslant$ ($f = O(g) \iff g = \Omega(f)$, et $f = \Theta(g) \iff f = O(g)$ et $f = \Omega(g)$).

Enfin, on peut définir o tel que :

$$f(n) = o(g(n))$$
$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

On peut donc écrire :

$$\sqrt{n} = o(n)$$
$$n = o(n \log \log n)$$
$$n^2 = o(n^3)$$

Théorème 7.8 : posons $t(n)$ une fonction telle que $\forall n \in \mathbb{N} : t(n) \geqslant n$. Chaque $t(n)$-time multitape machine de Turing (plusieurs rubans) a un équivalent en temps $\mathbf{O(t^2(n))}$ – **times** avec un seul ruban (*single-tape*) sur la machine de Turing.
Souvenons nous du théorème 3.13 : si l'on simule une machine de Turing à plusieurs rubans sur une machine de Turing à un ruban. $\delta : Q \times \Gamma^K \to Q \times \Gamma^K \times \{L, R\}^K$,
on peut observer plusieurs choses :

1. Pour chaque étape de la machine de Turing à plusieurs ruban, O(1) scan sur le contenu du ruban.
2. le contenu total a une taille $\leqslant K \times t(n)$

Souvenons nous du théorème 3.16 : chaque machine de Turing non déterministe a une équivalence en une machine de Turing déterministe.

Rappel d'une machine de Turing non déterministe : $S : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$
Pour évaluer une machine de Turing non déterministe, on peut construire un arbre qui va associer chacun des choix fait. Le **temps d'exécution** = le nombre d'étapes maximum pour toutes les entrées de taille n et pour tous les chemins calculables.
Notons que ce n'est possible **que** pour les machines **décidables**.
Théorème 7.11 : Chaque $t(n)$-time machine de Turing non déterministe a un équivalent en temps $2^{O(t(n))}$-time machine de Turing déterministe.
On définit b, le nombre maximum de choix possible pour chaque élément dans l'arbre.
On va donc pouvoir dire qu'il y aura :

1. au maximum $b^{t(n}$ nœuds
2. pour chaque nœud, le nombre d'étapes $\leqslant O(t(n))$

Au total : $O(t(n)b^{t(n)}) = 2^{O(t(n))}$ sur une machine de Turing à 3 rubans.
Notons que $(2^{O(t(n))})^2 = 2^{O(t(n))}$.

2 La class P

"Tous les modèles calculables raisonnables sont équivalent en temps polynomial.".
Ici, polynomial est $O(n^K)$ pour une certaine constante K.
$t : \mathbb{N} \to \mathbb{R}^+$ classe de complexité du temps $TIME(t(n))$ est une collection de langage décidables par une TM $O(t(n))$-time.
SIPSER : 7.2

P est une collection de langages décidables dans un temps polynomial sur une machine de Turing déterministe. On note également $P = \bigcup_K TIME(n^k)$

2.1 Exemple

Entrée : le graphe G
Question : est-ce que G est connexe ?
$L = \{\langle G \rangle | G \text{ est un graphe connexe}\}$
$L \in P$

RELPRIME (premier entre eux) = $\{\langle x, y \rangle | x \text{ et } y \text{ sont des nombres premier entre eux}\}$
$(gcd(x, y) = 1)$
Dans l'exécution de l'algorithme d'Euclide : $O(\log x + \log y))$
RELPRIME $\in$ P

2.2 Précision

Aucun problème NP ne peut être résolu en temps polynomial. Cependant, plusieurs problèmes ont une solution en temps exponentiel connue, mais pour lesquels il n'a pas été *démontré* qu'aucun algorithme polynomial existe (e.g. SUBSET_SUM).

3 La class NP

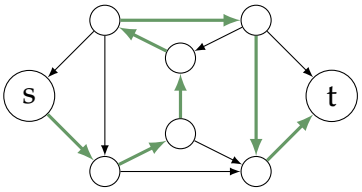
"Non déterministe Polynomial".
Une vérification pour un langage A est un algorithme V tel que :

$$A = \{w | V \text{ accepte } \langle w, c \rangle \text{ pour un texte } c\}.$$

Un temps polynomial vérifie si un vérificateur qui s'exécute en un temps polynomial ($|w|$). A est vérifiable en un temps polynomial $\leftrightarrow \exists$ un vérificateur en un temps polynomial.

3.1 Exemple

Chemin Hamiltonien : pour un graphe dirigé G et 2 sommets/point s, t
Est-ce qu'il existe un chemin depuis s vers t qui passe par tous les sommets une seule fois ?
Exemple :



3.2 Définition

Def 7.19 : NP est la classe de langages qui admettent une vérification en temps polynomial.
Théorème 7.20 Un langage B est dans NP si si et seulement si B est décidable dans une certaine machine de Turing dans un temps polynomial non déterministe.
Preuve
$\Rightarrow$ donner une vérification V en temps polynomial construit en suivant une machine de Turing non déterministe.
N = "Pour l'entrée w de taille n :

1. Prenons c, de manière non déterministe, de taille $\leqslant n^k$
2. Exécuter V sur l'entrée $\langle w, c \rangle$
3. si V est accepté, on accepte, sinon on rejette."

$\Leftarrow$ Supposons que l'on a une machine de Turing N non déterministe qui décide du langage en un temps polynomial. Montrer qu'il existe un vérificateur en un temps polynomial.
V = "Pour l'entrée $\langle w, c \rangle$:

1. Simuler N sur w, et traiter chaque symbole de c comme une description du choix non déterministe pour faire a chaque étape
2. Accepter si la branche est acceptée

"

3.3 Deux définition équivalent pour NP

1. Il existe un décideur non déterministe en un temps polynomial (”poly-time non deterministic decider“)
2. Il existe un vérificateur en un temps polynomial (”polynomial-time verifier“).

3.4 ...

$\text{SUBSET\_SUM} = \{\langle S, t\rangle | S = \{x_1, x_2, ...x_n\}$ et il existe $\{y_1, y_2, ...y_k\} \subseteq \{x_1, ...x_n\}, \Sigma_{i=1}^{k} y_i = t\}$.

Exemple : $\big(\{4, 11, 16, 21, 27\}, 25\big)$ | SUBSET_SUM $\in$ NP ?
Oui, il est bien dans NP.

3.5 $\mathbf{P \stackrel{?}{=} NP}$

Il est prouvé que $P \subseteq NP$, mais on ne sait pas si l'inclusion est stricte ou si $P = NP$ (il est conjecturé que $P \neq NP$).