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. Cette input va sûrement impliquer plus au moins de complexité dans la fonction. On va donc se concentrer sur la pire complexité.
En d'autres mots :
Nombre d'étape de la machine de turing $\Rightarrow$ $f : \mathbb{N} \to \mathbb{N}$: maximum d'étape de la machine de turing sur une entrée de taille n.
Pour cela, on va utiliser le "grand O".

Par exemple : $f(n) = O(g(n))$.
Qui traduit la relation suivante : $\exists n_0, c\, \forall n > n_0 : f(n) \leq c.g(n)$.

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

Dans la même idée, on peut utiliser Ω qui lui utilise donc un $\geq$.

Enfin, on peut définir o tel que :

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

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 tel que $t(n) \geq n$. Chaque $t(n) - time$ multi-tape machine de turing (plusieurs rubans) à 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 ruban 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 à une taille $\leq K \times t(n)$

.

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 : $S : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$
Pour évaluer une machine de turing non déterministe, on peut construire un arbre qui va associé chacun des choix fait. Le **temps d'exécution** = le nombre d'étape maximum pour toutes les entrée de taille n et pour tous les chemins calculable.
Notons que ce n'est possible **que** pour les machines **décidable**.
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 tel que 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'étape $\leq O(t(n))$

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 calculable raisonnable 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 décidable par un $O(t(n)) - time$ d'une machine de turing.
SIPSER : 7.2

P est une collection de langage décidable 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 connecté
$L = \{< G > | G$ est un graphe connecté$\}$
$L \in P$

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

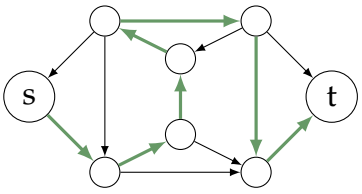
Il reste cependant des problèmes qui sont encore ouvert. Par exemple, lorsque l'on prend un ensemble de nombre et qu'il faut vérifier qu'il existe bel et bien un sous ensemble qui additionné vaut zéro. C'est un problème en temps exponentiel.

3 La class NP

"Non déterministe Polynomial".
Un langage A est un algorithme V tel que : $A = \{w|V \text{ accepte } < w, c > \text{ pour un texte } c\}$
Un temps polynomial vérifie si un vérificateur qui s'exécute en un temps polynomial ($|w|$). A est vérifiable en un temps polynomal $\leftrightarrow \exists$ 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 un langage de class qui a un vérificateur 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 un vérification V en tempos polynomial, construit en suivant en suivant une machine de turing non déterministe. N = "Pour l'entrée w de taille n :

1. Non déterministe prenon c de taille $\leq n^k$
2. Exécuter V ou $< w, c >$
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 $< w, c >$:

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 ...

SUBSET_SUM = $\{< s, t > | S = \{x_1, x_2, ...x_n\}$ et pour certain $\{y_1, y_2, ...y_l\} \subseteq \{x_1, ...x_n\}, \Sigma_i y_i = t\}$
Exemple : $(\{4, 11, 16, 21, 27\}, 25)|$ SUBSET_SUM $\in NP$
Oui, il est bien dans NP.

3.5 P = NP

Actuellement, tout le monde est d'accord de dire que $P \subseteq NP$ mais on n'est pas sûr que $P = NP$.