POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

LOG2810
STRUCTURES DISCRÈTES

TD 6 : ALGORITHMES

Directives pour la remise:

- La remise est individuelle et se fait à la fin de la séance de TD.
- Répondez directement sur ce document Word (docx).
- Lorsque vous avez terminé, générez un PDF avec le nom sous le format: ***SectionDeTD-Matricule.pdf*** (exemple: *04-1234567.pdf*).
- Téléchargez votre fichier PDF dans la boîte de remise située dans la Zone TDs de la page Moodle du cours.
- Choisissez la boîte de remise qui correspond à votre section de TD.
- Aucun retard et aucune remise par courriel ne seront accepté.
- Dans l'intérêt de l'équité pour tous les étudiants, vous devez modifier le fichier Word (docx) fourni. Modifier le fichier EXCLUT le fait d'intégrer des scans de rédaction manuscrite ou d'y écrire avec un stylet.
- Le non-respect des consignes entraînera automatiquement la note 0 pour ce TD.

Objectifs du TD6

Exercice 1: La complexité d'exécution d'un algorithme peut être exprimée à l'aide d'une fonction qui prend en paramètre la taille d'un problème spécifique à résoudre. L'objectif de l'exercice est d'avoir concrètement un ordre de grandeur en tête pour différentes fonctions de complexité algorithmique que l'on peut rencontrer dans la pratique.

Exercice 2: Être en mesure d'ordonner des fonctions de complexité algorithmique selon leur croissance asymptotique.

Exercice 3 et 4: Mettre en pratique les techniques de démonstrations pour la notation grand-Θ et la croissance asymptotique.

Exercice 5: Mettre en pratique les notions de complexité à un algorithme de multiplication matricielle et réaliser qu'il existe des alternatives, qui sont moins efficaces pour les plus petites matrices mais éventuellement plus efficaces lorsqu'il est nécessaire de multiplier de très grandes matrices.

Exercice 6: Appliquer les concepts de problème traitable, intraitable ou indécidable. Réaliser que les problèmes les plus difficiles en logiques promotionnelles sont décidables et le problèmes les plus difficiles en logiques des prédicats (premier ordre) sont indécidables.

Exercice 7: Établir un lien entre un problème démontré indécidable et le problème de l'arrêt.

La remise est individuelle mais le travail en équipe est encouragé. Veuillez inscrire votre nom, prénom et matricule ainsi que les noms des collègues avec lesquels vous avez collaboré pour le TD.

Nom:

Prénom:

Matricule:

Collègues:

Rappels

Croissance asymptotique des fonctions

Différentes fonctions peuvent avoir différents taux de croissance asymptotique. Nous formalisons cela en disant que

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

Cette relation est transitive. Si $f(n) \prec g(n)$ et $g(n) \prec h(n)$ alors $f(n) \prec h(n)$.

Par exemple, $n \prec n^2$; de manière informelle, nous disons que n croît plus lentement que n^2.

Nous pouvons ainsi obtenir une hiérarchie tel que :

$$1 \prec log(log(n)) \prec log(n) \prec n^{\epsilon} \prec n^{c} \prec n^{logn} \prec c^{n} \prec n! \prec n^{n}$$

où $0 < \epsilon < 1 < c$.

Notation grand-O

Démontrer qu'une fonction $f(n)$ est $O(g(n))$ revient à démontrer qu'il existe des constantes Cet k pour lesquelles: $|f(n)| \leq C|g(n)|$ pour $n > k$.

Notation grand-Θ

Démontrer qu'une fonction $f(n)$ est $\Theta(g(n))$ revient à démontrer qu'il existe des constantes C_1, C_2 et k pour lesquelles:
$C_1|g(n)| \leq |f(n)| \leq C_2|g(n)|$ pour $n > k$.

Exercice 1

Pour tous les fonctions $f(n)$ suivantes, déterminez la taille maximale n d'un problème qui peut être résolu en une seconde en assumant que l'algorithme prend $f(n)$ microsecondes (10^{-6} seconde) pour résoudre le problème.

a) $log_2(n)$

Réponse:
$n = 2^{1000000} \approx 10^{301030}$

b) $\sqrt{n}$

Réponse:
$n = 10^{12}$

c) n

Réponse:
$n = 10^{6}$

d) n^2

Réponse:
$n = 10^{3}$

e) n^3

Réponse:
$n = 10^{2}$

f) 2^n

Réponse:
$n = 19$

g) $n!$

Réponse:
$n = 9$

Exercice 2.

Ordonnez les fonctions 1.5^n, n^{100}, $(logn)^3$, $\sqrt{n}logn$, 10^n, $(n!)^2$ et $n^{99} + n^{98}$ de sorte que chaque fonction soit un grand-O de la suivante.
(Exemple: $n, n^3, 2^n$)

Réponse:

L'ordre est simple lorsque nous nous souvenons que les fonctions factorielles croissent plus vite que les fonctions exponentielles, que les fonctions exponentielles croissent plus vite que les fonctions polynomiales, et que les fonctions logarithmiques croissent très lentement. L'ordre est :

$(logn)^3$, $\sqrt{n}logn$, $n^{99} + n^{98}$, n^{100}, 1.5^n, 10^n, $(n!)^2$

Exercice 3

Démontrez que $n^2 + n^3$ est $\Theta(n^3)$

Réponse:
Il faut trouver C_1, C_2 et k pour que : $C_1 n^3 \leq n^2 + n^3 \leq C_2 n^3$ pour $n > k$.
Si nous divisons touts les termes par x^3 nous obtenons: $C_1 \leq \frac{1}{n} + 1 \leq C_2$.
Si nous choisissons $k = 1$, nous avons $1 \leq \frac{1}{n} + 1 \leq 2$. Nous pouvons donc alors choisir par exemple $C_1 = 0.5$ et que $C_2 = 2.5$.

Exercice 4

Démontrez que $log_2(n^2 + 1)$ est $\Theta(log_2(n))$

Réponse:
Il faut trouver C_1, C_2 et k pour que : $C_1 log_2(n) \leq log_2(n^2 + 1) \leq C_2 log_2(n)$ pour $n > k$.

Nous pouvons choisir $C_1 = 1$ et conclure que $log_2(n) \leq log_2(n^2 + 1)$ sachant que la fonction $\log_2$ est strictement monotone croissante et que $n \leq n^2 + 1$ pour tout $n > 0$.

Ensuite, $\log_2(n^2+1) \leq \log_2(2n^2) = 1 + 2\log_2(n) \leq 3\log_2(n)$ lorsque $n > 2$.

Nous pouvons donc choisir par exemple $C_1 = 1$, $C_2 = 3$ et $k = 2$.

Exercice 5

Voici un algorithme pour la multiplication de deux matrices. Pour des matrices $A \in \mathbb{R}^{m\times r}$, $B \in \mathbb{R}^{r\times n}$ et $C \in \mathbb{R}^{m\times n}$ données, l'algorithme suivant remplace C par $C + AB$.

Algorithme 1.
for $i = 1 : m$
    for $j = 1 : n$
        for $k = 1 : r$
            $C(i,j) = C(i,j) + A(i,k) \cdot B(k,j)$
        end
    end
end

Le nombre d'additions et le nombre de multiplications sont égales à $m \times r \times n$.

a) Si toutes les dimensions sont doublées, alors le temps d'exécution augmente par quel facteur?

Réponse:
Le travail augmente par un facteur 8.

b) Lorsque les matrices A et B sont des matrices carrées de dimension $n \times n$, le temps d'exécution de l'algorithme 1 est d'ordre $\Theta(n^a)$. Donnez la valeur de a.

Réponse:
$a = 3.$

c) Faites une recherche en utilisant votre ami Google pour trouver la valeur de b pour le temps d'exécution $\Theta(n^b)$ de l'algorithme de Strassen pour multiplier deux matrices carrées de dimension $n \times n$.

Réponse:
$b = log_2 7 \approx 2.807.$

d) Soit les trois matrices M_1, M_2 et M_3 respectivement de dimensions 10×100, 100×5 et 5×50. Sachant que la multiplication des matrices est associative:

$$(M_1 \cdot M_2) \cdot M_3 = M_1 \cdot (M_2 \cdot M_3)$$

identifiez parmi ces deux possibilités, l'alternative la plus rapide.

Réponse:
La première version demande $10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$
La deuxième version demande $100 \times 5 \times 50 + 10 \times 100 \times 50 = 75000$
La première version est plus rapide par un facteur de 10.

Exercice 6

Dites si les problèmes suivants sont **traitable**, **intraitable** ou **indécidable**.

Définition: Un problème est dit **traitable** si l'on connaît un algorithme pour le résoudre avec une complexité d'exécution $O(n^c)$ où c est une constante.

Définition: Un problème est dit **intraitable** (dans le sens de difficile) si les meilleurs algorithmes que l'on connaît pour le résoudre ont une complexité d'exécution $O(2^n)$ ou une croissance asymptotique encore plus grande.

Définition: Un problème est dit **indécidable** s'il est possible de démontrer qu'aucun algorithme ne peut exister pour le résoudre de manière générale.

a) Un problème général en logique propositionnelle peut être résolu au mieux en générant la table de vérité. Par exemple, le problème de décision SAT consiste à déterminer si une formule est dite satisfaisable, c'est-à-dire s'il existe une assignation des variables qui rend la formule logiquement vraie.

Réponse:
L'algorithme a une complexité d'exécution $O(2^n)$. Donc le problème est intraitable.

b) Une restriction du problème général de satisfiabilité est le problème 2-SAT que l'on peut formuler en utilisant une série de propositions liées par une implication. Alors le problème général peut s'écrire:
$(p \rightarrow q) \wedge (r \rightarrow s) \wedge (t \rightarrow u) \wedge \cdots$

Il existe alors un algorithme qui permet de résoudre le problème avec une complexité d'exécution $O(n)$.

Réponse:
L'algorithme a une complexité d'exécution $O(n)$. Donc le problème est traitable.

c) Le problème de l'arrêt en logique du premier ordre.

Réponse:
Indécidable.

Exercice 7

Certains problèmes en théorie des nombres sont indécidables. Voici un exemple de problème en théorie des nombres tiré de http://www-math.mit.edu/~poonen/papers/h10_notices.pdf.

Quels sont les valeurs de c pour lesquels il n'existe pas de valeurs entières à l'équation $x^3 + y^3 + z^3 = c$

- Nous savons que $x^3 + y^3 + z^3 = 29$ possède une solution avec des nombres entiers soit (1, 1, 3) par exemple.

- Nous savons que $x^3 + y^3 + z^3 = 30$ possède une solution soit (-2218888517, -283059965, 2220422932).

- Maintenant est-ce que $x^3 + y^3 + z^3 = 33$ possède une solution entière? Personne ne sait!

D. Hilbert, dans la liste de 23 problèmes qu'il publia après une célèbre conférence en 1900, demanda à son auditoire de trouver une méthode qui répondrait à toutes ces questions. Plus précisément, le dixième problème de Hilbert demande un algorithme qui prend en entrée un polynôme à plusieurs variables $f(x_1, \ldots, x_n)$ à coefficients entiers et renvoie OUI ou NON selon qu'il existe des entiers $a_1, a_2, \ldots, a_n$ tel que $f(a_1,\ldots,a_n) = 0$. En 1970, Yu. Matiyasevich, s'appuyant sur les travaux antérieurs de M. Davis, H. Putnam et J. Robinson, a démontré qu'un tel algorithme n'existe pas.

Décrivez un algorithme pour déterminer si $x^3 + y^3 + z^3 = 33$ possède une solution avec des nombres entiers, dans la mesure où vous auriez à votre disposition un algorithme pouvant résoudre le problème de l'arrêt.

Réponse:

Sans perte de généralités nous pouvons supposer $x \leq y \leq z$. Ensuite, il suffirait alors de préparer un algorithme pour vérifier toutes les combinaisons possibles des nombres entiers pour les variables x, y et z et de passer cet algorithme comme argument à un autre algorithme qui résout le problème de l'arrêt.