Complexité & graphes : TD1

Université de Tours

Département informatique de Blois

Complexité et machines de Turing

*
* *

Problème 1

1. Soient $f : \mathbb{R}^+ \to \mathbb{R}^+$ et $g : \mathbb{R}^+ \to \mathbb{R}^+$, deux fonctions positives. On note $f \in o(g)$ si est seulement si :

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

   On dit que f est *négligeable* devant g.

   Démontrer que $f \in o(g) \Rightarrow f \in O(g)$.

2. Donner l'ordre de complexité des fonctions suivantes[1] :

   - $f_1(n) = 3n^2 + 3n + 1$
   - $f_2(n) = 2^{n+100}$
   - $f_3(n) = \log(\sqrt{n} + n + n2^n)$
   - $f_4(n) = 2^{\log(2n)}$
   - $f_5(n) = n + (\log n)^2$
   - $f_6(n) = \log_n 2^n$

3. Hiérarchie de classes de complexité.

   (a) Montrer que $2^{\sqrt{\log(n)}} \in O(n)$.

   (b) Montrer que $\forall k \geq 1, \log(n)^k \in O\left(2^{\sqrt{\log(n)}}\right)$. On considèrera que $\forall a > 1, \forall k \geq 1 \lim\limits_{X\to+\infty} \frac{X^k}{a^X} = 0$

   (c) Hiérarchiser les trois classes de complexité : $O\left(2^{\sqrt{\log(n)}}\right)$, $O\left(\log(n)^k\right)$ et $O\left(n\right)$ selon la relation d'inclusion $\subseteq$.

Problème 2 (partiel 2019)

Étant donné un réel $\gamma > 0$, le but de cet exercice est de prouver que $e^{\gamma n} \in O(n!)$. Pour cela, on pose les deux suites $(u_n)_{n\geq 1}$ et $(v_n)_{n\geq 1}$ définies telles que :

$$\begin{cases} u_n = e^{\gamma n} \\ v_n = n! \end{cases}$$

1. Montrer que : $\exists n_0 \geq 1, \forall n \geq n_0, \frac{u_{n+1}}{u_n} \leq \frac{1}{2}\frac{v_{n+1}}{v_n}$.
2. En déduire que : $\exists C \in \mathbb{R}, \forall n \geq n_0, u_{n+1} \leq C\left(\frac{1}{2}\right)^{n-n_0} v_{n+1}$. On pourra démontrer à l'aide d'un télescopage.
3. En conclure que $e^{\gamma n} \in O(n!)$.

[1] Sauf précision, on considérera que log désigne $\log_2$, le logarithme de base 2.

Problème 3

1. Donner la complexité des algorithmes suivants :

Algorithme 1 : $A(n,m)$

```
Data : n ≥ 0, m ≥ 0
begin
    i ← 1
    j ← 1
    while j ≤ n do
        if i ≤ m then
            i ← i + 1
        else
            j ← j + 1
```

Algorithme 3 : $C(n,m)$

```
Data : n ≥ 0, m ≥ 0
begin
    i ← 1
    j ← 1
    while j ≤ n do
        if i ≤ m then
            i ← i + 1
        else
            j ← j + 1
            i ← 1
```

Algorithme 2 : $B(n)$

```
Data : n ≥ 0
begin
    i ← 1 while i ≤ n do
        j ← 1 while j ≤ i do
            j ← j + 1
        i ← i × 2
```

Algorithme 4 : $D(n)$

```
Data : n ≥ 0
begin
    if n < 2 then
        return 1
    else
        return D(n − 2) + D(n − 1)
```

2. Calcul de la puissance d'un nombre.
   (a) Soit $a > 0$, écrire un algorithme naïf permettant de calculer a^n pour $n \geq 1$.
   (b) Déterminer un algorithme de complexité $O(\log n)$ et démontrer sa complexité.

Problème 4

1. Avec un ordinateur actuel, vous êtes capable de traiter en 1 heure un problème en $O(n)$ d'une taille maximale de N. Avec la même procédure de calcul, quelle est la taille maximale du problème que vous pourriez traiter en 1 heure avec une machine 100 fois plus rapide ou 1 million de fois plus rapide.
2. Répondez à la même question pour des problèmes en $O(n^2)$, $O(n^5)$, $O(2^n)$ et $O(3^n)$.

Problème 5

Les trois énoncés sont indépendants et portent sur des machines de Turing différentes.

1. Soit la machine de Turing $M = (Q, \Gamma, \Sigma, \delta, q_0, \#, \emptyset)$ où :
   - $Q = \{q_0, q_1, q_2, q_3\}$
   - $\Gamma = \{a, b, A, A', B, B', \#\}$
   - $\Sigma = \{a, b\}$
   - $\delta : Q \times \Sigma \to Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}$

$\delta(q_0, a) = (q_1, A', \mathtt{R})$
$\delta(q_0, b) = (q_3, B', \mathtt{R})$
$\delta(q_1, a) = (q_1, a, \mathtt{R})$
$\delta(q_1, b) = (q_1, b, \mathtt{R})$
$\delta(q_1, A) = (q_1, A, \mathtt{R})$
$\delta(q_1, B) = (q_1, B, \mathtt{R})$
$\delta(q_1, \#) = (q_2, A, \mathtt{L})$
$\delta(q_2, a) = (q_2, a, \mathtt{L})$
$\delta(q_2, b) = (q_2, b, \mathtt{L})$
$\delta(q_2, A) = (q_2, A, \mathtt{L})$
$\delta(q_2, B) = (q_2, B, \mathtt{L})$
$\delta(q_2, A') = (q_0, A, \mathtt{R})$
$\delta(q_2, B') = (q_0, B, \mathtt{R})$
$\delta(q_3, \#) = (q_2, B, \mathtt{L})$
$\delta(q_3, a) = (q_1, a, \mathtt{R})$
$\delta(q_3, b) = (q_1, b, \mathtt{R})$
$\delta(q_3, A) = (q_1, A, \mathtt{R})$
$\delta(q_3, B) = (q_1, B, \mathtt{R})$
$\delta(q_3, \#) = (q_2, B, \mathtt{L})$

(a) Représenter cette machine sous forme de diagramme d'états.

(b) Quel est le contenu du ruban après l'exécution de M sur le mot d'entrée $abab$?

(c) Quel est le comportement général de cette machine pour un mot d'entrée de la forme $(a|b)^*$?

2. Division par 4.

   (a) Comment peut-on reconnaître simplement qu'un nombre écrit en binaire est divisible par 4 ?

   (b) Proposer une machine de Turing permettant de décider si oui ou non un nombre binaire est divisible par 4 ?

   (c) Vérifier que votre machine reconnaît que 20 est divisible par 4.

3. Proposer une machine de Turing permettant de tester si un mot d'entrée en binaire comporte le même nombre de 0 que de 1.