Cours du 21/04/11

# Calculabilité et complexité

TD3

Exercice 1 :

1. Complexité du traitement A : $\mathcal{O}(1)$
   Complexité du traitement B : $\mathcal{O}(1)$

   Montrer que le traitement B est exécuté $p_k - 2$ fois :
   Pour $n = 12$
   A est effectué 2 fois + 1 fois :
   12 mod 2 = 0, 6mod2 = 3, 3mod3 = 0
   B est effectué 1 fois
   Lorsque n = 3, p = p++
   $n = p_1^{e_1}, p_2^{e_2}, \dots, p_k^{e_k}$ La décomposition en facteurs premiers de n
   ici $n = 2^2\ 3^1$
   $p_k$ est le dernier nombre premier de $n$, ici 3
   $On\ a$ 3-2 = 1 dc B est effectué $p_k - 2$ fois pr 12
   B est effectué a chaque fois que l'on découvre un nouveau p
   $p = 2 \to p_1 \quad \to \quad p_{1-2}\ fois$
   $p_1 + 1 \to p_2 \quad \to \quad p_2 - p_1\ fois$
   $\vdots$
   $p_k + 1 \to p_k \quad \to \quad p_k - p_{k-1}\ fois$
   ———————————————————
   $p_1 - 2 + p_2 - p_1 + \cdots + p_k - p_{k-1} \to \quad p_k - 2$

2. A est exécuté $\sum_{i=1}^{k} e_i$ fois, c'est-à-dire la somme des exposants de la decomposition de n
3. Montrer que $\sum_{i=1}^{k} e_i < n$
   Supposons, $\forall\ i, P_i \geq 2$
   $n \geq \underbrace{2^{e_1}.2^{e_2}\dots\dots 2^{e_k}}_{2^{\sum_{i=1}^{k} e_i}}$
   On suppose donc que $\sum_{i=1}^{k} e_i > n$
   Alors $2^{\sum_{i=1}^{k} e_i} > 2^n$ et donc $n \geq 2^n$ où il y a contradiction !!
   Donc grâce à une démonstration par l'absurde, $\sum_{i=1}^{k} e_i < n$
4. B est exécuté $p_k - 2$ fois et A moins de $n$ fois
   Donc on prend le maximum des 2, et comme $p_k$ est forcement inferieur à $n$, et $p_{k-2} < n$, alors $\max(A, B) < n$
   On peut en conclure donc que $Algo \in \mathcal{O}(n)$
5. La complexité de l'algo en fonction de la taille de n est $2^{|n|}$, donc absolument pas efficace, car c'est exponentiel

Exercice 2 :

A. Formule d'Euler pour les graphes planaires connexes

1. Sommet + face – arrête = 2

Ici on a : 7 sommet + 8 face (7 + celle qui englobe) – 13 arrêtes = 2

2. $3f \leq 2\ a$

Un triangle a 2 faces (la face + l'extérieur) et 3 arrête : $3 * 2 \leq 2 * 3$

$$\underbrace{r_1}_{chaque\ arrete \in au\ plus\ a\ 2f} + r_2 + \ \ldots r_f \leq 2a$$

$r_i$ = nombre de bord de la face i

$3f \leq r_1 + r_2 + \cdots + r_f \leq 2a$

Car $r_1 = 3, r_2 = 3$ et $r_1 + \ r_2 = 6$ et non pas 5 en comptant les arretes commune en double

$a \leq 3s - 6$

D'après la formule d'Euler $(s + f - a = 2)$

$a = s + f - 2$

$3a = 3s + 3f - 6$

$3a \leq 3s + 2a - 6$ (Comme prouvé ci-dessus)

$a \leq 3s - 6$

3. $a \leq 3s - 6\ \in \mathcal{O}(n)$ car s est une constante

Il en va de même pour $3f \leq 2a$ si $c \leq 3s - 6\ \in \mathcal{O}(s)$ alors $2a \in \mathcal{O}(s)$ donc $3f \in \mathcal{O}(s)$

B. Algorithme pour calculer l'enveloppe convexe a partir d'une triangulation

1. CH : 0457203

PCH :$M_4, M_7, M_3, M_4, M_5$ + liste de toutes les arrêtes ordonnées de 1 à 7

$\rightarrow\ \mathcal{O}(s)$ : bornée par le nombre de sommets

2.

a. boucle de $r \rightarrow$ nombre de triangle

Or $r <\ \mathcal{O}(s) < O(n)$

Affectation en $\mathcal{O}(1)$ donc $\mathcal{O}(n)$

b. si $f \in\ \mathcal{O}(n)$ et $g \in \mathcal{O}(f(n) \log f(n))$ alors $f\ \leftarrow\ \mathcal{O}(n \log n)$

$f \in \mathcal{O}(n) \rightarrow\ \exists\ c > 0, \exists n_0\ tel\ que\ \forall n > n_0, f(n) <\ cn$

$g \in\ \mathcal{O}(f(n) \log f(n)) \rightarrow \exists b > 0, \exists n_1\ tel\ que\ \forall n > n_1, g(n) < bf(n) \log f(n)$

Donc $g(n) <\ bc\ n \log f(n) < b\ c\ n \log cn = bcn(\log c \log n)$

Pour $n$ grand on a $g(n) < 2bcn(\log n) < c' \log(n)$

On retrouve donc la définition au dessus, alors $g(n)$ est en $\mathcal{O}(n)$

c. Les meilleurs tris sont en $n \log n$ avec $n$ le nombre d'éléments à trier

On a $\mathcal{O}(n)$ éléments ici, or si on trie on obtiendra $\mathcal{O}(t \log t), t(n) \in \mathcal{O}(n) \rightarrow\ \mathcal{O}(n \log n)$

d. partie 3 :$a \in \mathcal{O}(n)$, executé $n$ fois car $a \leq 3s - 6 \in \mathcal{O}(s) \in \mathcal{O}(n)$
partie 4 : tableau de k en $\mathcal{O}(n)$, et deuxieme boucle en $\mathcal{O}(n)$ car executé $n$ fois

e. Chaque partie est en $\mathcal{O}(n \log n)$, donc si on prend le maximum de chaque partie, on trouvera toujours du $\mathcal{O}(n \log n)$

C. $EC = V = T(r)$
$EnvConv(M, V)$
$EC(n) = T(n) + EnvConv(n)$
$T(n) = EC(n) - EnvConv(n)$
$EnvConv(n) < cn \log n$
$EC(n) > bn \log n$
$T(n) \geq bn \log n - cn \log n$
Mais on a une contradiction car $T(n) < \mathcal{O}(n) \log n$, et $bn \log n - cn \log n \geq \mathcal{O}(n) \log n$

TD4

Exercice 1

1. Calcul des équations : $\frac{y-y_1}{y_2-y_1} = \frac{x-x_1}{x_2-x_1}$ → obtenir les operations coute en $\mathcal{O}(1)$
Calcul d'intersection de 2 droites : résoudre un système : coute en $\mathcal{O}(1)$ car il n'y a que 2 droites
Vérifier les intersections : faire des « if », donc $\mathcal{O}(1)$
La complexité asymptotique de ce problème est proche de $\mathcal{O}(1)$
2. Le boucle coute $\mathcal{O}(1)$ et est executée somme des $n - 1$ premiers entiers fois, donc en $\frac{n(n-1)}{2} = \mathcal{O}(n^2)$
3. Si $n$ petit, algorithme Bentley et Ottmann meilleur, si n grand il est pire
Pour $k \in \mathcal{O}(n)$
$A \in \mathcal{O}((n + k) \log n)$
$k < cn$
$A(n) < b(n + k) \log n < b(n + cn) \log n < b(1 + k)n \log n = a'n \log n$ donc $\mathcal{O}(n)$
$si\ k \in \mathcal{O}(n^2)$
$k < cn^2$
$A(n) < b(n + cn^2) \log n < b(1 + cn)n \log n$ avec $1 + cn < 2c'n$ donc $< 2c'n^2 \log n$
$\rightarrow \mathcal{O}(n^2 \log n)$