Université Savoie Mont Blanc, UFR SceM
INFO602, L3 INFO

5 mai 2021
Durée : 1h

# Examen, INFO602, Session 1

**Documents autorisés :** tous documents du cours/td/tp, notes manuscrites (nb : pas de livres)

*Les exercices sont indépendants. Le barème est indicatif. Il dépasse volontairement 20.*

**Exercice 1. Notations $O$, $\Omega$, $\Theta$ (/6)**

Complétez le tableau ci-dessous en indiquant si $f$ est un grand $O$, un grand $\Omega$ ou un grand $\Theta$ de la fonction $g$ (pour $n$ tendant vers l'infini). Attention, Juste $= +0,4$, Faux $= -0,4$, Rien $= +0$, ceci pour éviter que vous ne répondiez au hasard.

| | | $g$ | | |
|---|---|---|---|---|
| | | $n^2 + 3n \log n$ | $2^{n-1}$ | $3n + \frac{n}{2} \log n$ |
| $f$ | $n\sqrt{n} + 10$ | | | |
| | $n^2(n+1)$ | | | |
| | $n \log n$ | | | |
| | $2^{n+1} + 2n^2$ | | | |
| | $n + 3 \log n$ | | | |

On remplace les fonctions $f$ et $g$ par des fonctions $f'$ et $g'$ telles que $f' = \Theta(f)$ et $g' = \Theta(g)$ pour simplifier les comparaisons.

| | | $g$ | | |
|---|---|---|---|---|
| | | $n^2$ | $2^n$ | $n \log n$ |
| $f$ | $n^{\frac{3}{2}}$ | $O$ | $O$ | $\Omega$ |
| | $n^3$ | $\Omega$ | $O$ | $\Omega$ |
| | $n \log n$ | $O$ | $O$ | $\Theta$ |
| | $2^n$ | $\Omega$ | $\Theta$ | $\Omega$ |
| | $n$ | $O$ | $O$ | $O$ |

**Exercice 2. Fonction mystère (/4)**

Que calcule $F$ ? Quelle est sa complexité en pire cas ? Plus généralement, si on remplace 10 par un entier $a \geq 2$ quelconque, quelle fonction mathématique classique approche $F$ ?

```
// n est un nombre entier positif ou nul.
Fonction F( E n : Entier ) : entier ;
début
    if n == 0 then Retourner 0;
    else Retourner 1+F(n/10);
fin
```

$F$ calcule le nombre de chiffres décimaux nécessaires pour représenter le nombre $n$ (en considérant qu'il faut 0 chiffre pour représenter le nombre 0).
Sa complexité suit la loi $T(0) = O(1)$, $T(n) = T(n/10) + O(1)$. En utilisant le "Master Theorem" pour les fonctions de la forme $T(n) = aT(n/b) + f(n)$, on a

— $a = 1$, $b = 10$, $f(n) = O(1)$
— donc $e = \log_b a = \log_{10} 1 = 0$, et $f(n)$ se compare à $n^e = n^0 = 1$
— on est dans le cas 2 du théorème, donc $T(n) = \Theta(n^e \log n) = \Theta(\log n)$.

Une autre façon est de remarquer que $n/10$ enlève toujours un chiffre décimal à $n$ et que le programme s'arrête lorsque $n$ n'a plus de chiffres. Donc le nombre d'appel récursif à $n$ est proportionnel au nombre de chiffres décimaux de $n$, soit environ $\log_{10} n$. La complexité est alors aussi en $\Theta(\log_{10} n) = \Theta(\log n)$.
Plus généralement, si on remplace 10 par un entier $a \geq 2$, cette fonction calcule le nombre de chiffres en base $a$ nécessaire et suffisant pour représenter le nombre $n$. $F$ est donc une fonction qui approche le logarithme en base $a$ de $n$, soit la fonction $x \mapsto \log_a x$, ou encore $x \mapsto \frac{\ln x}{\ln a}$.

**Exercice 3. Complexité d'une fonction récursive (/4)**

On se donne un algorithme dont le temps d'exécution est de la forme : $T(1) = O(1)$, et $T(n) = 3T(n/2) + 6n$.

Quelle est la complexité en pire cas de cet algorithme en fonction de $n$ ? Justifiez votre résultat.

NB : il s'agit de la complexité de l'algorithme de multiplication de grands entiers sur $n$ bits proposé par Karatsuba en 1960.

Sa complexité suit la loi $T(0) = O(1)$, $T(n) = 3T(n/2) + 6n$. En utilisant le "Master Theorem" pour les fonctions de la forme $T(n) = aT(n/b) + f(n)$, on a

— $a = 3$, $b = 2$, $f(n) = \Theta(n)$
— donc $e = \log_b a = \log_2 3 \approx 1.585$, et $f(n) = \Theta(n)$ se compare à $n^{\log_2 3} \approx n^{1.585}$
— on est dans le cas 1 du théorème, donc $T(n) = \Theta(n^e) = \Theta(n^{\log_2 3}) \approx \Theta(n^{1.585})$.

L'algorithme de Karatsuba est donc nettement plus rapide que le produit de deux nombres sur $n$ bits calculé naïvement, dont la complexité est $\Theta(n^2)$.

**Exercice 4. Polygones simples, convexité et points intérieurs (/7,5)**

Si $Q = (q_i)_{i=0..n-1}$ est un polygone simple, il existe une façon relativement simple de savoir si un point $p$ est à l'intérieur de $Q$. On trace un rayon $[p, r)$ à partir de $p$, où $r$ est un point différent de $p$, et on compte le nombre de fois où $[p, r)$ intersecte le bord de $Q$. Si ce nombre est impair alors le point est à l'intérieur, sinon le point est à l'extérieur (voir illustration page suivante).

*Nb : dans la suite, pour simplifier, on supposera toujours que le rayon $[p, r)$ ne traverse pas un sommet de $Q$, ni se s'aligne avec un côté de $Q$. C'est une hypothèse réaliste si $r$ est bien choisi ou si $r$ est tiré au hasard.*

*Nb : on vous redonne dans la page suivante quelques fonctions de base de géométrie algorithmique, que vous pouvez utiliser dans vos algorithmes.*

1. (/2(+1 si explication)) Adaptez la fonction Intersection-Segments du cours (rappelée plus loin) pour en faire une fonction Intersection-Rayon$(p_1, p_2, p_3, p_4)$ qui retourne vrai si et seulement le rayon $[p_1, p_2)$ intersecte le segment fermé $[p_3, p_4]$.

Il faut remarquer que ORIENTATION$(p, q, r)$ calcule l'aire algébrique du triangle $[pqr]$ (positif si ccw et négatif si cw). Si $d_1$ est positif, il faut que soit $d_2$ soit négatif (et $p_2$ de l'autre côté), soit que $p_2$ appartienne au triangle $[p_3, p_4, p_1]$, et donc $d_2$ plus petit que $d_1$. On déduit la condition $d_1 > 0$ et $d_2 < d_1$. Symétriquement, si $d_1 < 0$ alors il faut aussi $d_2 > d_1$.

`// Détermine si le rayon` $[p_1, p_2)$ `et le segment` $[p_3, p_4]$ `s'intersectent.`

**Fonction** INTERSECTION-RAYON-SEGMENT*(* **E** $p_1, p_2, p_3, p_4$ *: Point ) : booléen ;*

**début**

$d_1 \leftarrow$ ORIENTATION$(p_3, p_4, p_1)$ ;

$d_2 \leftarrow$ ORIENTATION$(p_3, p_4, p_2)$ ;

$d_3 \leftarrow$ ORIENTATION$(p_1, p_2, p_3)$ ;

$d_4 \leftarrow$ ORIENTATION$(p_1, p_2, p_4)$ ;

**si** $((d_1 < 0$ *et* $d_1 < d_2)$ *ou* $(d_1 > 0$ *et* $d_2 < d_1))$ *et* $((d_3 < 0$ *et* $d_4 > 0)$ *ou* $(d_3 > 0$ *et* $d_4 < 0))$ **alors** **Retourner** *Vrai* ;

**sinon si** $d_1 = 0$ *et* SUR-SEGMENT$(p_3, p_4, p_1)$ **alors** **Retourner** *Vrai* ;

**sinon si** $d_2 = 0$ *et* SUR-SEGMENT$(p_3, p_4, p_2)$ **alors** **Retourner** *Vrai* ;

**sinon si** $d_3 = 0$ *et* SUR-SEGMENT$(p_1, p_2, p_3)$ **alors** **Retourner** *Vrai* ;

**sinon si** $d_4 = 0$ *et* SUR-SEGMENT$(p_1, p_2, p_4)$ **alors** **Retourner** *Vrai* ;

**sinon** **Retourner** *Faux* ;

**fin**

2. (/2,5) Ecrire maintenant l'algorithme qui teste si un point $p$ est à l'intérieur de $Q$. On attend ici un algorithme simple de complexité linéaire en le nombre de sommets de $Q$. Son prototype sera **Fonction** ESTINTÉRIEUR ?*(* **E** *p : Point,* **E** *Q : Polygone ) : booléen*

   NB : On écrira $Q.n$ pour avoir le nombre de sommets de $Q$, et $Q[i]$ pour accéder au $i$-ème sommet, avec l'indice $i$ pris modulo $Q.n$. N'oubliez pas de choisir un $r$.

**Fonction** ESTINTÉRIEUR ?*(* **E** *p : Point,* **E** *Q : Polygone ) : booléen ;*

**Var :** $i, n$ : entier ;

**Var :** $q$ : Point ;

**début**

$n \leftarrow 0$ ;

$q \leftarrow (Q[0] + Q[1] + Q[2])/2$ ;

**pour** *i de 0 à* $Q.n - 1$ **faire**

**si** INTERSECTION-RAYON-SEGMENT$(p, q, Q[i], Q[i+1])$ **alors**

$n \leftarrow n + 1$ ;

**fin**

**fin**

**Retourner** $n\%2 == 1$

**fin**

3. (/1,5) Si maintenant $Q$ est un polygone convexe, combien de fois le rayon $[p, r)$ peut-il intersecter le bord de $Q$ ?

Si $Q$ est convexe, alors si $p$ est intérieur le rayon touchera exactement 1 fois le bord de $Q$. Si $p$ est extérieur il touchera soit 0 fois soit 2 fois le bord du polygone.

4. (/1,5) Prenons maintenant $Q$ un polygone non simple comme le pentacle. Dessinez-le et remplissez en gris les zones où cet algorithme retournera intérieur et laissez en blanc les zones où cet algorithme retournera extérieur. Y trouvez-vous une logique ?

La règle est appelée "even-odd" rule. En gros chaque fois qu'on traverse une frontière on passe de extérieur à intérieur ou de intérieur à extérieur.