Algorithmique

TD N°1 : Généralités

Van du tran

vandu@lix.polytechnique.fr

F(n) = O(g(n)) ⇔ la complexité de f(n) est inférieure à celle de g. quand n tend vers l'infini, (f(n) à une borne supérieure g : elle restera au mieu autour, sinon dessous. f est bornée « par-dessus)
f/g <K

F(n) = o(g(n)) ⇔ g croit beaucoup plus vite que f
f/g → 0

F(n) = Ω(g(n)) ⇔f est bornée par-dessous. : pour tout n, f restera au mieu autour de g, sinon au dessus.
f/g >K

F(n)= ω(g(n)) ⇔ f croit beaucoup plus vite que g.
f/g → ∞

F(n) =Θ(g(n)) ⇔ bornée dessus et dessous. Il existe deux réels a et b tels que pout tout n > à n0 tel que a.gn≤f(n)≤b.g(n)
a<f/g<b

F(n) = ~(g(n)) ⇔ F et g ont une complexité équivalente.

Equivalents

- Comparez :
  - $N^2 = o(n^3) \rightarrow$ tend vers 0
  - $n^3 = \omega(n^2) \rightarrow$ tend vers $+\infty$
  - $Ln(n^2) = \Theta(\ln(n^3))$ (la fraction = 2/3)
  - $Ln(n^3) = \Theta(\ln(n^2))$ (la fraction = 3/2)
  - $N^2 =\sim(n^2+n(-1)^n)$
  - $n^2+n(-1)^n =\sim(n^2)$
  - e^{n^2} ? $e^{(n^2+n*(-1)^n)}$
  - $N^2= o(((2+(-1)^n)n^2))$
  - $((2+(-1)^n)n^2) = \Theta(N^2)$
- Donnez la complexité du bout de code suivant :
  If b
  then print(bonjour)
  else faire 20 fois print (bonsoir)

Complexité :

F=1 ➔ Θ(1)

Ou

F=2O ➔ Θ(1)

- Combien vaut $\sum_{i=1}^{n} 1^{i}$? en donner un équivalent.
  - $\sum_{i=1}^{n} 1^{i} = \frac{n(n+1)}{n} \sim s\,\frac{n^2}{2}$

Donner une équivalent de :

  - $\sum_{i=1}^{n} i^2 \sim \frac{n^3}{3},$
  - $\sum_{i=1}^{n} i^k \sim \frac{n^{k+1}}{k+1}$
  - $\sum_{-1=1}^{n} i^{-1} \sim \frac{n^{-1+1}}{-1+1}$ ➔ division par zéro !!! k=-1 interdit.
  - $\sum_{23=1}^{n} 23 \sim \frac{n^{23}}{24}$
  - $\sum_{\frac{1}{2}=1}^{n} \frac{1}{2} \sim (\frac{2}{3})/\sqrt{4^3}\ )$
  - $\sum_{i=1}^{n} \ln(i) \sim \int \ln(x)\, dx\ n \sim [x\ln(x) - x] \sim n\ln n - n - (\ln(1) - 1)$

Faut il trier ?

Vous pouvez utiliser les fonctions suivantes : (l désigne la longueur de la liste L) :

- estDans(x,L) qui teste si 'lélément x apparait dans la liste L. Cette fonction est de complexité Θ(l).
- estTriee(x,l) fait de meme, mais en supposant que la liste est triée. Cette fonction est de complexité Θ(ln(l))
- elementCommunDansTriees(L1,L2) suppose que L1 et L2 sont triées et regarde si elles ont un élément commun. Cette fonction est de complexité O(L1+L2).
- Tri(L) qui tire la liste L cette fonction est de complexité Θ (lnl).
  On a deux listes non tirées de longueur m et n. on supposera que m ≥ n. on cherche à savoir si'il y a un élément commun aux deux listes. Donner 4 stratégies et comparer els ordres de grandeurs de leurs complexités.

trouveOn a deux listes non tirées de longueur m et n. on supposera que m≥n. on cherche a savoir s'il y a un élément commun aux deux listes !!!!!!; donner 4 stratégies et comparer les ordres de grandeurs de leurs complexités.

1. $\forall x \in Lm, si\ estDans(x, \mathrm{Ln})\ alors\ on\ retourne\ vrai\ sinon\ faux.$
   Θ(m*n)

2. Trier(Ln)
   $\forall x \in Lm,$
   $si\ estDans\ (x, Ln)$ alors
   Retourne vrai
   Sinon retourne faux
   Θ(nlnn) + Θ(mlnn) = Θ(mlnn)

3. Trier(LM)
   Pour i de 0 à n

   si estDansTriee(x, Ln)

   Retourne vrai

   Sinon

   Retourne faux

   finSin

   finPour

   $\Theta(m\ln m)$.

4. Tri(Lm)
   Tri(Ln)
   rendre elementCommunDansTriées(Lm,Ln)

   $\Theta(m\ln m) + \Theta(n\ln n) + O(m+n) = \Theta(m\ln m)$

   C'est la Deuxième solution la plus performante.

Complexités de l'arithmétique

1. Quelle est la complexité de l'algo d'addition d'entiers vus en primaire ?

   a on suppose a>b

   \+ b

   ____

   a+b

   $\Theta(n)$ avec n = nombre de chiffres de a.

   $n = \log_{10} Max(a,b)$.
2. Quelle est la complexité des algos utilisant les batons ?

   2 + 5

   II + IIIII =III+IIII=IIII+II=…=IIIIII

   $\Theta(b)$

   5 + 2

   IIIII + II = =…=IIIIII

   $\Theta(\min(a,b))$
3. Justifiez formellement que le premier est meilleur

   Log est bien meilleur.
4. Qu'en est-il pour la multiplication ?

   $\Theta(nm)$ avec n=nb chiffre de a, et m nb chiffre de b

Taille logarithme d'un nombre n

A

N= nombre de chiffre de a en base 2.
n=O($\log_2 a$)

Rapport : n_2 = log(adiv2) +2

$\text{Log}_2 n$ opérations pour le programme : il s'agit du nombre de chiffre de n en binaire.

Multiplication Russe

1. 69*135

Opération	Quotient	Reste	Multiplicande	Résultat
69/2	34	1	135	135
34/2	17	0	270	135
17/2	8	1	540	675
8/2	4	0	1080	675
4/2	2	0	2160	675
2/2	1	0	4320	675
1/2	0	1	8649	9315

2.
```
function Russe (     in: a, b
                     out : r
              ){
        r=0;
        tant que a>0 faire
              si a est impair
                     r <- r+b
              fsi
              a <- a div 2
              b <- b * 2
        ftq
}
fonction recRusse(IN : a,b
                           inout :r){
si a != 0
        si a est impair
              alors recRusse((a/2à,(b+2), (r+b))
        sinon
              recRusse((a/2), (b*2), 2
        fsi
fsi
}
```

Durant toute la durée de l'exécution de l'algorithme, a*b + r reste égal.

Les pieces

=Θ(n*(ndiv2)*(ndiv5)*(ndiv10)*(ndiv20)=Θ(n^5/2000) = O(n^5)

Petits programmes

```
double power(int x, int n){
	double res = 1;
	for (int i=0;i<n;i++)
	{
		res=res*x;
	}
	return res;
}
```

```
double recPower(int x, int n){
	r<-1
	if n mod 2 = 0
		r <- x * x
		for i = 0 to (n-1)/2
			do
				r2 <- r*r
			done
	return (r.r2)
}
```

➔x^n = x* xn-1 →Θ (n)

➔x^(2p) = (x^p)^2 ➔ Θ ($\log_2 n$)

```
Fonction base (entier :k, entier : n){
	S←[]
	Compteur ←1
	Tant que n>0 faire
		S[compteur] ← (n modulo k)*compteur + s
		N ← division_entiere k
		Compteur ← compteur +1
	Ftq


	While compteur > 0
		afficher s[compteur]
		compteur ← compteur -1
	done
}
```

Ou

```
Function base(int k, int n){
	Res = reclase(k,n);
	Court res
}
Function recbase(int k,int n){
	String res='';
	If(n diventiere k ≤ 0){
		Return '';
	}
	res = n%k
	Return recbase(n, n div_entiere k)  + res
}
```

Le nombre d'itération est [$\log_k n$]+1, la complexité $\Theta(\log_k n)$

<u>Hanoi</u>

H(n,p,q)

- H(1,p,q) vrai pour tout $p \neq q$
- H(2,p,q)
  - H(1,p,6-p-q) vrai
  - P→ q
  - H(1,6-p-q,q) Vrai
  - ⇨ H(2,p,q) vrai $\forall$ $p \neq q$
- H(n-1,p,q) vrai $\forall$ p,q
  - H(n-1,p,6-p-q) vrai
  - P→q
  - H(n-1,6-p-q,q) vrai
  - ⇨ H(n,p,q) vrai $\forall$ $p \neq q$

```
Procédure H(n,p,q)
      Si n>0 alors
            H(n-1,p,6-p-q)
            Affiche (p→q)
            H(n-1,6-p-q,q)
      Fsi
Fin
```