

E.D. Algorithmes et Structures de Données n° 1

Thème : Complexité des Algorithmes

Exercice I.1 De l'intérêt d'améliorer la taille des ordinateurs

Question 1 A titre d'illustration des notions de *taille de problèmes* et de *nombre d'opérations*, on considère un vecteur $x=(x_1, x_2, \dots, x_n)$ de n composantes, où chaque composante x_i peut prendre les valeurs 0 ou 1. Dans tous les exemples qui suivent la *taille du problème* est la taille du vecteur x , c'est à dire n . (Ici, la taille du problème est donc le nombre de composantes du vecteur x , mais pour d'autres problèmes, cela pourrait être un nombre de clients à traiter, un nombre de villes à relier entre elles par des lignes téléphoniques, bref c'est un moyen de dimensionner le problème).

Pour chacun des algorithmes suivants évaluer le nombre d'opérations :

<pre>Algo 1 Pour i allant de 1 à n faire afficher(x_i) fait</pre>	affichage des n composantes du vecteur x
<pre>Algo 2 Pour i allant de 1 à n faire Pour j allant de 1 à n faire afficher(x_i+x_j) fait fait</pre>	
<pre>Algo 3 Pour i allant de 1 à n faire Pour j allant de 1 à n faire Pour k allant de 1 à n faire Pour l allant de 1 à n faire Pour m allant de 1 à n faire afficher(x_i+x_j+x_k+x_l+x_m) fait fait fait fait fait fait</pre>	

algo 4: Affichage de toutes les valeurs possibles du vecteur x.

Exemple pour n=3 :

x = (0, 0, 0) x = (0, 0, 1) x = (0, 1, 0) x = (0, 1, 1)
 x = (1, 0, 0) x = (1, 0, 1) x = (1, 1, 0) x = (1, 1, 1)

```

Algo enumeration (n,i : entier, x : vecteur)
Debut
  si (i>n)
    alors afficher(x)
    sinon   xi := 0;
             enumeration(n,i+1,x);
             xi := 1;
             enumeration(n,i+1,x);
  finsi
fin
  
```

Appel de l'algorithme :
 enumeration(3,1,x);

Question 2 Compléter le tableau ci-dessous

Nombre d'opérations en fonction de la taille n des Données	Avec un ordinateur X		Avec un ordinateur Y 100 fois plus rapide que X	
	Taille maximale (n max) des problèmes traités en 1h	Nombre d'opérations effectuées en 1h	Taille maximale (n max) des problèmes traités en 1h	Nombre d'opérations effectuées en 1h
N	N ₁	N ₁	N' ₁ = 100 N ₁	
n ²	N ₂		N' ₂	
n ⁵	N ₃		N' ₃	
2 ⁿ	N ₄		N' ₄	

Exercice I.2

Soient les 3 algorithmes suivants permettant de calculer la valeur d'un polynôme

$$p(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

en un point x. Comparer leur complexité. (La valeur de x est donnée).

```

algo 1
début
  p := a0 ;
  pour i := 1 à n faire
    calculer pi = xi ;   -- x.x.x...x  i-1 fois
    p := p + ai pi ;
  fait;
fin;
  
```

```

algo 2      -- il utilise le fait que  $x^i = x^{i-1} * x$ 
début
  p := a0;   q := 1;
  pour i := 1 à n faire
    q := q * x ;
    p := p + ai * q ;
  fait;
fin;

```

```

algo 3      --algorithme de Horner; il calcule à reculons  $a_n, a_n x + a_{n-1}$ ,
  puis  $(a_n x + a_{n-1}) x + a_{n-2} \dots$ 
début
  p := an ;
  pour i := n à 1, pas -1, faire
    p := p*x + ai-1 ;
  fait;
fin;

```

Exercice I.3

Rappel de quelques définitions.

Soient 2 fonctions f et g de \mathbf{N} dans \mathbf{N} :

$f = O(g)$ si il existe $n_0 \in \mathbf{N}$ et c constante > 0 , tels que $f(n) \leq c g(n)$ pour tout $n \geq n_0$

Question 1 Soit un algorithme demandant exactement $f(n) = 12n+7$ opérations , montrer que cet algorithme est en $O(n)$.

Question 2 De même, montrer que si $f(n) = a_0 n^p + a_1 n^{p-1} + a_2 n^{p-2} + \dots + a_{p-1} n + a_p$, et $a_0 > 0$ alors $f = O(n^p)$.

Exercice I.4

Soit l'algorithme suivant qui effectue la recherche dichotomique du rang (place) d'un nombre A dans une suite triée (par ordre croissant) de n nombres mis dans un tableau à une dimension (vecteur $L[i]; i=1, \dots, n$). Cet algorithme fonctionne sous l'hypothèse que A est présent dans la liste.

```

Algorithme :
début
  place = 1 ;
  f = n ;
  tant que place < f faire
    milieu = (place + f) / 2 ;
    si L[milieu] < A alors
      place = milieu + 1 ;
    sinon f = milieu ;
    finsi;
  fait;
fin;

```

Question Donner la complexité de cet algorithme.