TD n°3

Rappel de cours

Il arrive fréquemment que pour un même problème, différents algorithmes permettant de le résoudre soient connus. L'étude de certains critères peut conduire à préférer l'un plutôt que l'autre :

La simplicité	La correction
Le coût de la mise en oeuvre	Le temps d'exécution
Le coût de la maintenance	L'espace mémoire occupé
...	

1 Temps d'exécution

Calculer le temps d'exécution d'un algorithme revient à calculer le temps d'exécution de chaque instruction et à le multiplier par le nombre de fois ou elle est exécutée. Le temps d'exécution des actions élémentaires (affectation, comparaison, opération, entrée/sortie) étant variable d'une machine à l'autre, on majore généralement celles-ci par une ou plusieurs constantes. Les coûts de différents algorithmes peuvent ainsi être comparé sans biais.

Si l'on prend l'exemple des algorithmes de tris, on se rend rapidement compte que le temps d'exécution d'un algorithme donné sur un tableau de taille N va dépendre de l'ordre initial de ses éléments. On définit donc trois mesures de complexité :

Soit l'algorithme A travaillant sur une donnée d de taille n (d est par exemple un tableau de taille n).

- La complexité dans le meilleur des cas : $Inf_A(d,n) = inf\{coût_A(d) | \#(d) = n\}$

- La complexité dans le pire des cas : $Sup_A(d,n) = sup\{coût_A(d) | \#(d) = n\}$

- La complexité dans le cas moyen, avec D_n l'ensemble des données de taille n :

$$Moy_A(D_n) = \frac{\sum_{d \in D_n} coût_A(d)}{\#(D_n)}$$

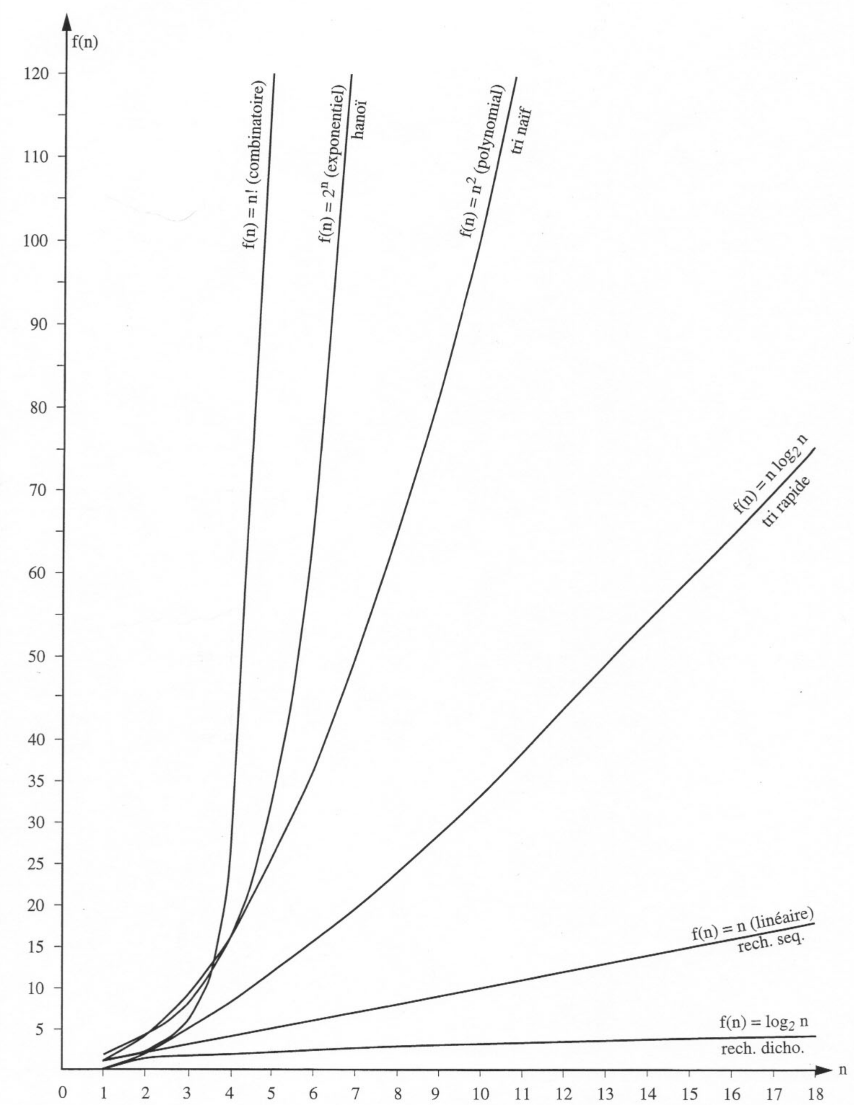
2 Exemple : Produit de N facteurs

```
public int produit(int [] t){
 int i=0; int produit=1;
 while(produit!=0 && i<t.length){
    produit=produit*t[i]; i=i+1;
 }
 return produit;
```

}

1. Exhibez un tableau pour lequel le temps d'exécution est minimal. Calculez-le.
   $\Longrightarrow$ un tableau où t[0]=0.
   $Cout(produit) = 2t_{aff} + (2t_{comp} + 1t_{op} + 1t_{aff} + 1t_{op} + 1t_{aff}) + 1t_{comp} + 1t_{aff}$
   Cout(produit)= 10t
   $\Longrightarrow$ **complexité en O(1)**

2. Exhibez le pire cas.
   $\Longrightarrow$ tableau de taille n sans élément égal à 0.
   Cout(produit) = 4t + 6t*(n)
   $\Longrightarrow$ **complexité en O(n)**



Principales fonctions de complexité

$\log_2 n < n < n\log_2 n < n^2 < 2^n < n!$