# Diviser pour régner

Les trois phases du paradigme "diviser pour régner" :
- Diviser : division du problème en un certain de nombres de sous-problèmes (semblables au problème initial mais de taille moindre, par exemple de taille moitié).
- Conquérir et Régner : résolution des sous-problèmes
  - par des appels récursifs,
  - directement lorsque le sous-problème est de taille "petite".
- Combiner : combinaison des solutions des sous-problèmes pour constituer la solution globale.

## 1 Puissances

On propose ci-dessous deux algorithmes pour calculer la puissance $n$ d'un réel $x$ (rédigé dans le langage Xcas) :

**Xcas**
```
puiss(x,n):={
si n==0 alors return 1;
sinon return x*puiss(x,n-1);
fsi;
}:;
```

**Xcas**
```
puiss2(x,n):={
local y;
si n==0 alors return 1;fsi;
si n==1 alors return x;fsi;
y:=puiss2(x,iquo(n,2));
si irem(n,2)==0
alors return y*y;
sinon return x*y*y;
fsi;
}:;
```

1. Quel est le nombre de multiplications effectuées pour calculer $x^{100}$ avec chacun des deux algorithmes ?
2. $n$ étant de la forme $2^p$, quel est le nombre de multiplications pour le calcul de $x^n$ pour chacun des deux algorithmes ?

## 2 Recherche dans une liste

On se propose de comparer les deux algorithmes suivants (langage Xcas) qui ont pour objectif de chercher le rang d'un élément dans une liste dont les éléments sont supposés triés dans l'ordre strictement croissant.
Remarque 1 : les listes sont numérotées en xcas de 0 à dim(liste)-1.
Remarque 2 : pour alléger le code, on suppose que la valeur cherchée est effectivement dans la liste.

Xcas
```
sequentiel(liste,valeur):={
local rang;
rang:=0;
tantque liste[rang]<>valeur faire
rang:=rang+1;
ftantque;
retourne rang;
};;
```

Xcas
```
dichotomique(liste,valeur):={
local debut,fin,k,trouve;
trouve:=false;
debut:=0;fin:=dim(liste)-1;
repeter
k:=iquo(debut+fin,2);
si liste[k]==valeur alors
trouve:=true;
sinon
si liste[k]<valeur alors debut:=k+1
    sinon fin:=k-1;fsi;
fsi;
jusqu_a trouve;
retourne k;
};;
```

1. Combien, au plus, d'itérations de la boucle auront lieu lors d'une recherche pour chacun des deux programmes ? Pour simplifier on se placera dans le cas où le nombre d'éléments dans la liste est de la forme $n = 2^p$.
2. On suppose que $n = 2^{100}$. Admettons que la machine fasse un million de boucles en 1 seconde, quel temps (dans le pire des cas) sera utilisé pour chacun des deux programmes lors d'une recherche ?
3. Écrire (xcas) une version récursive de chacun des deux programmes (avec les mêmes hypothèses simplificatrices faites plus haut).

# 3 La suite de Fibonacci

On rappelle la définition de la suite de Fibonacci :

$$F_0 = F_1 = 1 \quad \text{et} \quad \forall n \geqslant 1, F_{n+1} = F_n + F_{n-1}$$

1. Établir :
$$\forall n \geqslant 1, \forall p \geqslant 1, F_{n+p} = F_n F_p + F_{n-1} F_{p-1}$$
2. On pose $T_n := (F_{n-1}, F_n)$ pour $n \geqslant 1$. Exprimer $T_{2n}$ et $T_{2n+1}$ en fonction de $F_n$ et $F_{n-1}$.
3. En déduire un programme fibo de calcul de $F_n$ du type "diviser pour régner". Donner une estimation du nombre d'appels à fibo pour calculer $F_n$.

# 4 Quick sort

## 4.1 Partition

On considère l'algorithme suivant (langage xcas) dans lequel T est une liste :

**Xcas**

```
partition(T,deb,fin,clef):={
local d,f,tmp;
d:=deb;f:=fin;
repeter
        tantque d<=fin faire
        si T[d]<=clef alors d:=d+1 sinon break;fsi;
        ftantque;
        tantque f>=deb faire
        si T[f]>clef alors f:=f-1 sinon break;fsi;
        ftantque;
        si d<f alors
        tmp=<T[d];T[d]=<T[f];T[f]=<tmp;   // on échange T[d] et T[f]
        d:=d+1;f:=f-1;
        fsi;
jusqu_a d>f;
retourne f;
};;
```

1. Que renvoie partition ([5,10,12,3,7,4,2,6,9,8,1,13],0,11,7) ?
2. Justifier que l'algorithme se termine.
3. Justifier qu'après application de l'algorithme, tous les éléments de $T$[deb] à $T[f]$ sont inférieurs ou égaux à clef et tous les éléments de $T[f+1]$ à $T$[fin] sont supérieurs strictement à clef. Pour cela, démontrer par récurrence l' "invariant" : « A la fin de l'itération $k$ de la boucle répéter, tous les éléments $T$[deb] à $T[d-1]$ sont inférieurs ou égaux à clef et tous les éléments de $T[f+1]$ à $T$[fin] sont supérieurs strictement à clef. »
4. Quel est le nombre de comparaisons avec la clef dans le pire des cas (dans les deux boucles tant que) ? le nombre d'échanges dans le pire des cas (dans le « si $d < f$ ») ?

## 4.2 Quick Sort

On considère l'algorithme suivant (écrit en langage xcas) :

**Xcas**

```
tri_rap(T,deb,fin):={
local d,f,clef;
si deb<fin alors
        d:=deb;f:=fin;clef=<T[deb];
        repeter
                tantque d<=fin faire
                si T[d]<=clef alors d:=d+1 sinon break;fsi;
                ftantque;
                tantque f>=deb faire
                si T[f]>clef alors f:=f-1 sinon break;fsi;
                ftantque;
                si d<f alors
                tmp=<T[d];T[d]=<T[f];T[f]=<tmp;
                d:=d+1;f:=f-1;
                fsi;
        jusqu_a d>f;
T[deb]=<T[f];T[f]=<clef; //échange de T[deb] et T[f]
T=<tri_rap(T,deb,f-1);T=<tri_rap(T,f+1,fin);
fsi;
return T;
};
```

1. Quel est l'effet de tri_rap sur $T := [2,8,6,12,5]$ par exemple ?
2. Démontrer, par récurrence sur la taille $n = \text{fin} - \text{deb} + 1$ du tableau, l'effet de tri_rap sur une liste $T$.
3. Lorsque le tableau de taille $n$ donné en entrée est déjà trié, estimer la complexité.

# 5 Karatsuba

1. On considère l'algorithme suivant (donné en langage xcas) :

**Xcas**

```
gaga(P,Q):={
local dp,dq,R,x,k,j;
dp:=dim(P)-1;dq:=dim(Q)-1;
R=<makelist(x->0,0,dp+dq);
pour k de 0 jusque dp faire
pour j de 0 jusque dq faire
R[k+j]=<R[k+j]+P[k]*Q[j];
fpour;fpour;
retourne R;
};
```

   (a) Que renvoie l'appel gaga ([3,4,5],[4,7,9,10]) ?

(b) Combien de multiplications et additions lors de l'exécution de ce programme ?

2. Montrer comment multiplier les deux polynômes $P(X) = a_1X + a_0$ et $Q(X) = b_1X + b_0$ en utilisant seulement trois multiplications « élémentaires » où l'une des multiplications est $(a_0+a_1)\times(b_0+b_1)$.
3. En déduire un algorithme du type "diviser pour régner" permettant de multiplier deux polynômes de degré au plus $n-1$. Soit $P$ le polynôme $P(X) = \sum_{j=0}^{n-1} a_j X^j$, on peut diviser ainsi :

$$P(X) = \left(a_0 + a_1X + \cdots + a_{k-1}X^{k-1}\right) + X^k\left(a_k + a_{k+1}X + \cdots + a_{n-1}X^{\ell-1}\right)$$

où l'on a posé $k = \lfloor \frac{n}{2} \rfloor$ et $\ell = n-k = \lceil \frac{n}{2} \rceil$.
4. Écrire le détail pour le produit de $P(x) = a_0 + a_1x + a_2x^2$ et $Q(x) = b_0 + b_1x + b_2x^2$.
5. Évaluer le nombre de multiplications et le nombre d'additions (en se limitant au cas où $n$ est une puissance de 2 pour simplifier).