Algorithmique
TD n° 3 : Implementation et comparaison

Exercice 1 : Algorithme de Karatsuba pour les polynomes

L'algorithme de Karatsuba (vu en cours) peut être utilisé pour la multiplication de polynômes.

Petit rappel. On note a_0, a_1, $\dots a_{n-1}$ les coefficients du polynôme de degré $n-1$ P_1 de telle sorte que : $P_1(x) = \sum_{i=0}^{n-1} a_i \cdot x^i$. On a de même pour P_2 avec $P_2(x) = \sum_{i=0}^{n-1} b_i \cdot x^i$.

On souhaite donc calculer le produit $P_1(x) \cdot P_2(x) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_i \cdot b_j \cdot x^{i+j}$. Si on procède de manière naïve, on obtient donc un algorithme en $\Theta(n^2)$.

On peut faire mieux avec l'algorithme de Karatsuba qui est en $\Theta(n^{\log_2 3})$.

Prenons $\begin{cases} P_1(x) = \alpha_1(x) \cdot x^m + \alpha_0(x) \\ P_2(x) = \beta_1(x) \cdot x^m + \beta_0(x) \end{cases}$ avec $m = \frac{n}{2}$ et $\alpha_0, \beta_0, \alpha_1, \beta_1$ des polynômes de degré $\leqslant m$.

On a : $P_1(x) \cdot P_2(x) = \underbrace{\alpha_1(x) \cdot \beta_1(x)}_{K_2(x)} x^{2m} + \underbrace{(\alpha_0(x) \cdot \beta_1(x) + \alpha_1(x) \cdot \beta_0(x))}_{K_1(x)} x^m + \underbrace{\alpha_0(x) \cdot \beta_0(x)}_{K_0(x)}$

Et $K_1(x)$ peut donc s'obtenir avec un unique produit (de polynômes de degré $\leqslant \frac{n}{2}$) avec :

$$K_1(x) = K_2(x) + K_0(x) - (\alpha_1(x) - \alpha_0(x)) \cdot (\beta_1(x) - \beta_0(x))$$

Algorithme. On va représenter un polynôme P de degré n par un tableau T de taille $n+1$ où $T[0]$ est le coefficient de degré 0,... et $T[n]$ celui de degré n. On va programmer l'algorithme décrit ci-dessous. Attention : lorsqu'on écrit A+B ou A-B, il s'agit d'une opération sur les tableaux A et B. De plus, on écrit `"([0]*m)»K1"` pour indiquer que l'on ajoute `m` chiffres `0` au tableau `K1` (*ie* on multiplie le polynôme correspondant par x^m).

```
def karatsuba(A,B) :
    si |A| != |B|, on complète avec des 0 pour obtenir |A| == |B|
    si |A|==1 :
        res = A.B  // res est un tableau de taille 1
    sinon :
        m = (|A|+1)/2
        A0 = A[0...m-1],    A1 = A[m...|A|-1]
        B0 = B[0...m-1],    B1 = B[m...|A|-1]

        K2 = karatsuba(A1,B1)
        K0 = karatsuba(A0,B0)
        aux = karatsuba(A1-A0,B1-B0)
        K1 = (K0+K2)-aux
        K1=([0]*m)>>K1
        K2=([0]*(2*m))>>K2
        res = K2+K1+K0
    retourner res
```

Dans les fonctions ci-dessous, `P`, `P1` et `P2` sont des des tableaux d'entiers et désignent des polynômes. `n`, `k` et `c` sont des entiers.

1. Programmer la fonction `polyalea(n)` qui renvoie un polynôme (donc un tableau...) de degré n dont les coefficients sont des entiers entiers choisis de manière aléatoire dans l'ensemble $\{-100, \dots, 100\}$.
2. Programmer les fonctions suivantes :
    1. `somme(P1,P2)` qui renvoie un polynôme correspondant à la somme $P_1(x) + P_2(x)$;

2. `difference(P1,P2)` qui renvoie un polynôme correspondant à la différence $P_1(x) - P_2(x)$;
3. `decalage(P1,k)` qui renvoie un polynôme correspondant à $P_1(x) \cdot x^k$;
4. `prodfac(P1,c,k)` qui renvoie un polynôme correspondant à $P_1(x) \cdot (c \cdot x^k)$;

3. Programmer la fonction `produit(P1,P2)` qui renvoie le produit $P_1(x) \cdot P_2(x)$ en le calculant selon la méthode naïve.

4. Programmer la fonction `karatsuba(P1,P2)` qui renvoie le produit $P_1(x) \cdot P_2(x)$ en le calculant selon l'algorithme de Karatsuba.

5. **Comparer les deux calculs du produits selon les degrés des polynômes.** Pour cela, construire une fonction test qui, étant donné un entier `nbt` et un degré `n`, répète `nbt` fois : choisir deux polynômes aléatoires de degré `n` et mesurer le temps d'exécution des deux algorithmes sur ces deux polynômes. Puis calculer la moyenne des temps d'exécution des deux algorithmes.

Programmation. On vous suggère d'utiliser Python pour ces exercices mais le choix du langage est libre.
Si vous utilisez Python, nous vous rappelons que le choix d'entiers aléatoires se fait avec la fonction `random.randint(bi,bs)` où `bi` et `bs` désignent les bornes inférieures et supérieures (comprises). Pour mesurer le temps d'exécution d'une fonction, on utilisera

```
start = time.process_time()
ma_fct(A,B)
end = time.process_time()
print('temps d'exec. :'+str(end-start))
```