# PROG
# TP#6

2020-10-21

## Multiplications de polynômes avec l'algorithme de Karatsuba

Le but de cet exercice est d'implémenter l'algorithme de Karatsuba pour multiplier deux polynômes *de même taille* à coefficients dans $\mathbb{Z}/2^{32}\mathbb{Z}$ stockés sur 32 bits. On rappelle que celui-ci utilise la décomposition récursive suivante : soit $P_0$, $P_1$, $Q_0$, $Q_1$ quatre polynômes de degré $n-1$, on souhaite calculer le produit $R := (P_1X^n + P_0) \times (Q_1X^n + Q_0)$ ; on pose :

— $A := P_0 \times Q_0$ ;
— $B := (P_0 + P_1) \times (Q_0 + Q_1)$ ;
— $C := P_1 \times Q_1$ ;

ce qui permet de calculer $R = CX^{2n} + (B - (A + C))X^n + A$

**Q.1 :**

1. Définissez et implémentez une structure `struct poly_u` permettant de représenter un polynôme via un tableau de coefficients stockés sur des entiers non signés de 32 bits et son degré.
2. Écrivez des fonctions d'allocation et de désallocation de ces structures, de prototypes `struct poly_u *alloc_poly_u(int deg)` et `void free_poly_u(struct poly_u *p)` respectivement.
3. Écrivez une fonction d'affichage d'un polynôme.

**Q.2 :**

1. Écrivez une fonction `mulpu` qui calcule naïvement le produit de deux polynômes, et testez là sur de petits exemples.
2. Testez les performances et évaluez le comportement asymptotique de `mulpu` sur des polynômes de degrés variables. Testez également l'impact du compilateur et de ses optimisations.

**Q.3 :**

1. Écrivez une fonction `mulpuk1` qui calcule le produit de deux polynômes de même degré en appliquant *une fois* la décomposition ci-dessus (c'est à dire en utilisant `mulpu` pour calculer les termes $A$, $B$ et $C$). Testez votre fonction en comparant son résultat avec `mulpu`.
2. Testez les performances de `mulpuk1` sur des polynômes de degrés variables, et comparez avec `mulpu`.

**Q.4 :**

1. Écrivez une fonction récursive `mulpukr` qui calcule le produit de deux polynômes de même degré en appliquant la décomposition ci-dessus tant que ses arguments ont un degré inférieur à un certain seuil `DEG_THRESHOLD` que vous définirez. Testez votre fonction en comparant son résultat avec `mulpu`.
2. Testez les performances de `mulpukr` sur des polynômes de degrés variables, ainsi qu'avec des valeurs variables pour `DEG_THRESHOLD`.
3. Quelle valeur de seuil vous donne les meilleurs résultats ? Celle-ci dépend-elle du degré initial de vos entrés ?

*N.B.* Vous devriez être capable de multiplier des polynômes de degré supérieur à 1 000 000 en un temps raisonnable.

**Remarques :** Faites particulièrement attention à l'allocation et à la désallocation mémoire. Réfléchissez à vos besoins, et déterminez dans quelles fonctions l'allocation est la plus judicieuse.