DM

Tous les algorithmes sont à écrire en pseudo-code selon les modèles vus en cours. Pour les algorithmes les plus complexes (pseudocode long, utilisation de variables internes non évidentes, utilisation de propriétés particulières, ...), il est fortement recommandé de donner une explication concise de leur principe de fonctionnement (si je ne parviens pas à comprendre l'idée derrière un pseudo-code incorrect, je ne peux pas mettre de points). Le devoir est à rendre au format numérique à l'adresse `fpirot@lisn.fr`, au plus tard le 02/04/2023 à 23h59.

Exercice 1 (Arithmétique de Polynômes)**.**
Le but de cet exercice est d'implanter des opérations arithmétiques de base sur les polynômes.

On représente un polynôme $P = a_d X^d + a_{d-1} X^{d-1} + \cdots + a_0$ de degré d sous la forme du tableau `t` de ses coefficients, de taille $d+1$, dont les éléments sont des flottants. On a donc `t[i]` $= a_i$ pour tout $0 \leq i \leq d$. De plus, on impose que `t[d]` $\neq 0$, de sorte que la taille de `t` soit précisément $\deg(P)+1$ (on représentera le polynôme $P = 0$ avec un tableau vide, en utilisant la convention inhabituelle que $\deg(P) = -1$ dans ce cas).

1. Implanter une fonction Derivee(P) qui renvoie la dérivée du polynôme P. 1 pt
2. Implanter une fonction Primitive(P) qui renvoie la primitive du polynôme P. 1 pt
3. On cherche à implémenter la somme de polynômes. Cela nécessite de faire attention aux monômes de plus gros degré qui pourraient s'annuler, ce qui peut faire que $\deg(P+Q) < \max\{\deg(P), \deg(Q)\}$.
   (a) Implanter une fonction DegreeSomme(P,Q) qui renvoie le degré de la somme des polynômes P et Q (si $P+Q=0$, on renverra -1). 1 pt
   (b) Implanter une fonction Somme(P,Q) qui renvoie la somme des polynômes P et Q. 1 pt
4. Implanter une fonction itérative Produit(P,Q) (utilisant uniquement des boucles Pour) qui renvoie le produit des polynômes P et Q. Quelle est sa complexité ? 2 pts
5. On cherche maintenant à utiliser le paradigme *Diviser pour Régner* afin d'obtenir un algorithme récursif de meilleure complexité pour le produit de polynômes. Pour tout entier $r \geq 1$, il est possible de décomposer un polynôme P sous la forme $P_0 X^r + P_1$, avec $\deg(P_0) \leq \deg(P) - r$ et $\deg(P_1) \leq r-1$. Le produit de P avec un polynôme $Q = Q_0 X^r + Q_1$ peut alors s'obtenir avec la formule

$$PQ = P_0 Q_0 \, X^{2r} + ((P_0 + P_1)(Q_0 + Q_1) - P_0 Q_0 - P_1 Q_1) \, X^r + P_1 Q_1. \qquad (\star)$$

   (a) En utilisant $(\star)$, implanter une fonction récursive ProduitRec(P,Q) qui renvoie le produit de deux polynômes P et Q de degré au plus d via 3 appels récursifs sur des polynômes de degré au plus $\lfloor d/2 \rfloor$. 2 pts
   (b) Exprimer la complexité $C(d)$ de votre fonction sur des polynômes de degré au plus d avec une formule de récurrence. 1 pt
   (c) En utilisant le Master Theorem, donner la valeur de $C(d)$. 1 pt

Exercice 2 (Structure pour la recherche et l'insertion)**.**
Nous avons vu en TD que la recherche d'un élément au sein d'un tableau trié de n éléments peut se faire en temps $O(\log n)$. En revanche, l'insertion d'un élément x dans un tableau trié (partiellement rempli) de taille n se fait en temps $\Theta(n)$ dans le pire cas, même s'il n'y a pas besoin de réallouer de mémoire, puisque cela demande de décaller d'une case vers la droite toutes les valeurs supérieures à x dans le tableau. Le but de cet exercice est de mettre au point une structure de données S qui propose un meilleur compromis entre la complexité de la recherche et de l'insertion d'un élément en son sein.

Lorsque S contient $n \geq 1$ nombres, on peut la décrire comme suit. S contient une liste $\mathsf{S.bits} = [b_0, \ldots, b_m]$ contenant les bits de la représentation binaire de n (b_0 est le bit de poids faible et b_m le bit de poids fort, et on a $m = \lfloor \log_2 n \rfloor$). Les n nombres contenus dans S sont rangés dans des tableaux $t_0, t_1, \ldots, t_m$, où t_i est de taille 2^i, et est entièrement rempli si $b_i = 1$ ou vide (toutes ses cases valent `None`) si $b_i = 0$. De plus, chaque tableau t_i est trié, mais on n'a aucune hypothèse sur l'ordre relatif de deux nombres contenus dans des tableaux différents. On accède à ces tableaux via une liste $\mathsf{S.content} = [t_0, \ldots, t_m]$.

Pour simplifier le pseudocode, on pourra supposer que les listes sont implémentées comme en Python (ce sont donc en fait des tableaux dynamiques, qui permettent d'accéder directement à n'importe quelle position i).

1. Implanter une fonction `isInside(x,S)` qui teste la présence d'un nombre x au sein de la structure S. Quelle est sa complexité (une complexité sous-linéaire est attendue) ? — 2 pts
2. On s'intéresse maintenant à l'insertion d'un nouvel élément dans S. Cette opération repose sur le même principe que l'opération `incr` sur le compteur dans le TD2. Si $b_0 = 0$, le tableau `t0` est vide, et on peut donc insérer x directement dans `t0`. Si $b_0 = 1$ et $b_1 = 0$, on insère x et l'élement de t_0 directement dans t_1 (et on vide t_0). Si $b_0 = 1$, $b_1 = 1$, et $b_2 = 0$, on insère x et les éléments de t_0 et t_1 dans t_2 (et on vide t_0 et t_1). Et ainsi de suite, quitte à devoir créer un nouveau tableau t_{m+1}.
   (a) Implanter une fonction `Fusion`(t_0,t_1) qui prend en entrée deux tableaux triés t_0 et t_1 de taille respective n_0 et n_1, et renvoie un tableau trié t contenant l'union des éléments de t_0 et t_1. Sa complexité devra être $O(n_0 + n_1)$. — 2 pts
   (b) Implanter une fonction `Insert(x,S)` qui insère un nombre x dans la structure S (on pourra faire plusieurs appels à `Fusion`, et utiliser une fonction `dump(t)` de complexité linéaire qui vide le tableau `t` passé en argument). — 3 pts
   (c) Exprimer la complexité de `Insert(x,S)` en fonction du nombre de 1 consécutifs à la fin de la représentation binaire de n (le nombre d'éléments contenus dans S avant l'insertion). — 1 pt
   (d) Quelle est la complexité de répéter N fois l'opération `Insert`, en partant d'une structure S vide (une complexité sous-quadratique est attendue) ? — 2 pts

Bonus Décrire la procédure à suivre pour supprimer un nombre (dont on nous donne la position via un pointeur) de S, et évaluer sa complexité en fonction de la représentation binaire de n. Quelle est la complexité dans le pire cas de N insertions et/ou suppressions successives de nombres dans S ? — 3 pts