Informatique – D.S. 1 *(2 heures)*

Le sujet comporte deux problèmes indépendants, pouvant être traités dans un ordre quelconque.
Les algorithmes doivent être écrits de la manière la plus courte possible, parfaitement lisibles, avec une indentation convenable, sans aucune rature et en respectant scrupuleusement les notations introduites. Ils doivent être documentés par des explications concises et précises sur les points qui le nécessitent.

Les fonctions demandées seront écrites en langage Python. Les tableaux utilisés sont supposés fournis sous forme de liste Python (donc indexés à partir de 0).

Problème A – *Parties d'un ensemble fini*

Le but du problème est d'engendrer l'ensemble $\mathcal{P}(E)$ des parties d'un ensemble fini donné.

Les éléments d'un ensemble E de cardinal n seront stockés dans une liste Python `E` de longueur n, donc indexée de 0 à $n-1$.

Pour caractériser un sous-ensemble A d'un tel ensemble E, on convient d'utiliser un *tableau de présence* `p`, également liste Python de longueur n, dont les éléments valent 0 ou 1 et sont définis de la façon suivante : pour tout k entre 0 et $n-1$, `E[k]` est élément de A si et seulement si `p[k]=1`.

Par exemple, si $n=4$ et si `E=[3,5,7,9]`, alors $E=\{3,5,7,9\}$ et :

- `p=[0,1,0,0]` correspond à la partie $\{5\}$
- `p=[1,0,0,1]` correspond à la partie $\{3,9\}$, etc.

Pour les fonctions demandées, le candidat pourra choisir entre version itérative ou récursive lorsque l'énoncé ne précise rien.

Si une version récursive est choisie, la fonction écrite pourra être elle-même récursive, ou bien faire appel à une fonction auxiliaire récursive.

Il n'est pas demandé de preuve formelle des programmes, mais ceux-ci devront être concis et clairs.

1) *Partie caractérisée par un tableau de présence* : écrire une fonction d'en-tête `Partie(E,p)` renvoyant sous forme de liste la partie de l'ensemble E, stocké dans la liste `E`, définie par le tableau de présence `p`.

2) *Génération de $\mathcal{P}(E)$ – version itérative* : pour cette première méthode, on utilise la propriété suivante, **que l'on ne demande pas de démontrer**. Lorsqu'un entier j varie de 0 à 2^n-1, son écriture en base 2 fournit tous les tableaux de présence correspondant aux 2^n parties d'un ensemble à n éléments (sous réserve bien sûr de considérer les écritures comportant exactement n chiffres, avec éventuellement des 0 pour les chiffres de poids le plus élevé). Par exemple, pour $n=3$, les écritures en base 2 des entiers de 0 à 7 sont : 000, 001, 010, 011, 100, 101, 110, 111.

a) On convient ici que les valeurs (0 ou 1), stockées dans une liste `p`, correspondent aux chiffres de l'écriture en base 2 d'un entier j, **rangés par ordre de poids croissant avec l'indice**.

Par exemple, `p=[1,1,1,0,0,1,0]` correspond à $j=39=2^0+2^1+2^2+2^5$ et `p=[0,0,0,1,0,1,0]` correspond à $j=40=2^3+2^5$.

Écrire une fonction d'en-tête `Ajoute1(p)` recevant une liste `p` contenant — selon le principe ci-dessus — les chiffres de l'écriture en base 2 d'un entier j et renvoyant une liste représentant de même l'entier $j+1$ (en supposant que $j<2^\ell-1$, où ℓ est la longueur de `p`).

On justifiera brièvement la méthode utilisée pour effectuer l'addition.

b) Déduire des questions précédentes une fonction itérative d'en-tête `EnsPartiesI(E)`qui renvoie la liste de toutes les parties de l'ensemble E (stocké dans la liste `E`).

3) *Génération de $\mathcal{P}(E)$ – version récursive* : on peut engendrer récursivement tous les tableaux de présence correspondant aux différentes parties de l'ensemble donné E (et mémoriser la partie associée au tableau `p` dès qu'il est rempli !). Écrire selon ce principe une fonction récursive d'en-tête `EnsPartiesR(E)` qui renvoie la liste de toutes les parties de l'ensemble E (stocké dans la liste `E`), cela sans utiliser la fonction `Ajoute1` ci-dessus.

Problème B – *Transformée de Fourier rapide*

Le but de ce problème est l'étude de la *transformée de Fourier rapide*, qui permet — à l'aide de la stratégie "diviser pour régner" — de calculer (de façon approchée) le produit de deux polynômes de degré n au prix d'un nombre de multiplications en $O(n \log_2 n)$. Le fait d'obtenir des valeurs approchées des coefficients du produit est un inconvénient, mais sans conséquence dans le cas — fréquent — où l'on manipule des polynômes à coefficients entiers. Cela dit, les polynômes considérés par la suite sont des polynômes à coefficients complexes et l'on se contente de la précision fournie par Python.

On rappelle qu'en Python, si `x` et `y` sont deux flottants, l'évaluation de l'expression `complex(x,y)` renvoie le complexe de partie réelle `x` et de partie imaginaire `y`.

Inversement, si `z` est un complexe Python, `z.real` renvoie sa partie réelle et `z.imag` sa partie imaginaire.

Python sait effectuer les opérations algébriques sur les complexes : addition, multiplication et puissances entières, avec la même syntaxe que pour les flottants.

Pour tout n dans $\mathbb{N}^*$ et tout k dans $\mathbb{Z}$, on note $u_{k,n} = e^{2ik\pi/n}$ et l'on considère le vecteur de $\mathbb{C}^n$

$$U = (u_{0,n}, u_{1,n}, \ldots, u_{n-1,n}).$$

À tout polynôme P, de degré au plus $n-1$, défini par

$$P(z) = a_0 + a_1 \cdot z + \cdots + a_{n-1} \cdot z^{n-1} \quad \text{où} \quad (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{C}^n,$$

on associe le vecteur de $\mathbb{C}^n$

$$P\langle U \rangle = \big(P(u_{0,n}), P(u_{1,n}), \ldots, P(u_{n-1,n})\big).$$

Réciproquement, on peut démontrer (et l'on *admettra*) que, pour tout vecteur $Y = (y_0, y_1, \ldots, y_{n-1})$ de $\mathbb{C}^n$, il existe un unique polynôme P, à coefficients complexes, de degré au plus $n-1$, tel que $P\langle U \rangle = Y$. Ce polynôme P est appelé *le polynôme d'interpolation associé à* Y.

On cherche à calculer efficacement, d'une part le vecteur $P\langle U \rangle$ à partir des coefficients de P, d'autre part les coefficients du polynôme P tel que $P\langle U \rangle = Y$ à partir des composantes du vecteur Y.

Les polynômes et les vecteurs sont représentés par des listes Python.
Le polynôme P, défini par $P(z) = a_0 + a_1 \cdot z + \cdots + a_{n-1} \cdot z^{n-1}$, est représenté par la liste $[a_0, a_1, \ldots, a_{n-1}]$.
Le vecteur $Y = (y_0, y_1, \ldots, y_{n-1})$ de $\mathbb{C}^n$ est représenté par la liste $[y_0, y_1, \ldots, y_{n-1}]$.

1) *Quelques utilitaires*

a) Comme il s'agira souvent d'interpréter une liste contenant des nombres complexes proches des coefficients d'un polynôme à coefficients entiers, écrire une fonction d'en-tête `arrondi(P)`, recevant en entrée une liste P de complexes et renvoyant la liste des arrondis "à l'entier le plus proche" des parties réelles des éléments de P. On éliminera en outre de cette liste tous les zéros suivant la dernière valeur non nulle (le polynôme associé à P étant supposé non nul !).

On rappelle que `round(x)` *renvoie l'arrondi à l'entier le plus proche du flottant* `x`.

b) En supposant `pi`, `cos` et `sin` chargés par

```
from math import pi,cos,sin
```

écrire une fonction d'en-tête `u(k,n)`, recevant en entrée deux entiers k et n (avec $n \in \mathbb{N}^*$) et renvoyant le complexe $u_{k,n} = e^{2ik\pi/n}$.

c) Écrire une fonction d'en-tête `separe(L)` recevant en entrée une liste $L = [v_0, v_1, \ldots, v_{n-1}]$ de longueur paire n et renvoyant le couple $(L0, L1)$ où $L0 = [v_0, v_2, v_4, \ldots, v_{n-2}]$ et $L1 = [v_1, v_3, v_5, \ldots, v_{n-1}]$.

d) Écrire une fonction d'en-tête `z_wz(P,w)` recevant en entrée une liste $P = [a_0, a_1, \ldots, a_{n-1}]$ et un complexe w, et renvoyant la liste des coefficients du polynôme associé à la fonction

$$z \mapsto P(wz) = a_0 + a_1 \cdot (w \cdot z) + \cdots + a_{n-1} \cdot (w \cdot z)^{n-1},$$

cela au prix d'un nombre de multiplications de complexes qui doit être un $O(n)$.

Dans toute la suite du problème, n désigne une puissance de 2 : $n = 2^p$, $p \in \mathbb{N}$.

On rappelle que, si pour tout n puissance de 2, $C(n) = 2C(n/2) + O(n)$, alors $C(n) = O(n \log_2 n)$.

2) *Calcul de $Y = P\langle U\rangle$, P étant donné*

a) Pour $n = 2^p$, $p \in \mathbb{N}^*$, on note $m = n/2$ et l'on considère un polynôme P de degré au plus $n-1$, défini par $P(z) = a_0 + a_1 \cdot z + \cdots + a_{n-1} \cdot z^{n-1}$. Montrer que l'on peut écrire $P(z) = P_0\left(z^2\right) + z \cdot P_1\left(z^2\right)$ où P_0, P_1 sont deux polynômes de degré au plus $m-1$ que l'on précisera.

Montrer que les $P(u_{k,n})$, $0 \leq k \leq n-1$, peuvent se calculer à partir des $P_0(u_{j,m})$, $P_1(u_{j,m})$, $0 \leq j \leq m-1$. *On établira :*

$$\forall j \in [\![0, m-1]\!] \quad \left\{ \begin{array}{c} P(u_{j,n}) = P_0(u_{j,m}) + u_{j,n} \cdot P_1(u_{j,m}) \\ P(u_{m+j,n}) = P_0(u_{j,m}) - u_{j,n} \cdot P_1(u_{j,m}) \end{array} \right. .$$

b) En déduire une fonction récursive, d'en-tête `PtoY(P)`, recevant en entrée une liste `P` de longueur $n = 2^p$, contenant les coefficients d'un polynôme P de degré au plus $n-1$, et renvoyant une liste de longueur n représentant le vecteur $P\langle U\rangle$.

Le nombre de multiplications de complexes effectuées lors de l'appel `PtoY(P)` pour `P` de longueur n doit être un $O(n \log_2 n)$, ce que l'on justifiera.

3) *Calcul de P, $Y = P\langle U\rangle$ étant donné*

a) Pour $n = 2^p$, $p \in \mathbb{N}^*$, on note $m = n/2$ et l'on considère un vecteur $Y = (y_0, y_1, \ldots, y_{n-1})$ de $\mathbb{C}^n$. On lui associe les vecteurs $Y_0 = (y_0, y_2, \ldots, y_{2m-2})$ et $Y_1 = (y_1, y_3, \ldots, y_{2m-1})$ de $\mathbb{C}^m$.

U désigne toujours le vecteur $(u_{0,n}, u_{1,n}, \ldots, u_{n-1,n})$ de $\mathbb{C}^n$ et l'on note

$$T = (u_{0,m}, u_{1,m}, \ldots, u_{m-1,m}) \in \mathbb{C}^m.$$

Soient P l'unique polynôme de degré au plus $n-1$, tel que $P\langle U\rangle = Y$, et P_0, P_1 les polynômes de degré au plus $m-1$ tels que, respectivement, $P_0\langle T\rangle = Y_0$ et $P_1\langle T\rangle = Y_1$.

Montrer que :

$$\forall z \in \mathbb{C} \quad P(z) = \frac{1+z^m}{2} \cdot P_0(z) + \frac{1-z^m}{2} \cdot P_1\left(e^{-2i\pi/n} \cdot z\right)$$

En déduire les coefficients de P en fonction de ceux de P_0 et P_1.

b) Déduire du résultat précédent une fonction récursive d'en-tête `YtoP(Y)` recevant en entrée une liste `Y` de longueur $n = 2^p$, contenant les composantes d'un vecteur Y de $\mathbb{C}^n$, et renvoyant une liste de longueur n contenant les coefficients du polynôme P de degré au plus égal à $n-1$ tel que $P\langle U\rangle = Y$.

Le nombre de multiplications de complexes effectuées lors de l'appel `YtoP(Y)` pour `Y` de longueur n doit être un $O(n \log_2 n)$, ce que l'on justifiera.

4) *Produit de deux polynômes*

Déduire des questions précédentes une fonction d'en-tête `mult(A,B)`, recevant en entrée deux listes `A` (de longueur d_A) et `B` (de longueur d_B), contenant les coefficients d'un polynôme A (*resp.* B) de degré d_A (*resp.* d_B), et renvoyant une liste contenant les coefficients du polynôme produit $C = A \times B$.

On prendra garde que d_A et d_B ne sont pas nécessairement des puissances de 2.

Si l'on note $d_C = d_A + d_B$, le nombre de multiplications de complexes effectuées lors de l'appel `mult(A,B)` doit être un $O(d_C \log_2 d_C)$, ce que l'on justifiera.

N.B. Dans le cas du produit de deux polynômes à coefficients entiers, un appel à la fonction `arrondi` du **1)** terminera opportunément le calcul... Ce principe est utilisé pour la multiplication de "grands entiers", écrits sous la forme $P(b)$ où les coefficients de P sont les chiffres de l'écriture du nombre en base b.

Principe de Hofstadter

Il faut toujours plus de temps que prévu, même en tenant compte du principe de Hofstadter.