AGRÉGATION
TP ALGORITHMIQUE EN SAGE

CHRISTOPHE RITZENTHALER

1. Listes

On propose le programme SAGE suivant de recherche d'un maximal dans une liste d'entiers.

```
def plus_grand_element(liste):
     resultat = liste[0]
     for i in range(1, len(liste)):
          if liste[i] > resultat:
               resultat = liste[i]
     return resultat
```

(1) Implémenter le.
(2) À l'aide de la fonction `ZZ.random_element()`, créer des listes aléatoires de longueur croissante et donner le temps pour le calcul de la fonction ci-dessus avec `cputime()`.
(3) Automatiser la procédure précédente et tracer le graphe du temps dans le pire des cas (en prenant le maximum pour 5 tirages de longueur donnée) pour des longueurs entre 10^4 et 10^6 (avec un pas de 10^4). On pourra avoir besoin de `list_plot()` et de stocker des éléments dans une liste avec la fonction `append`.

2. Entiers, nombres premiers, congruences, corps finis

Réaliser les opérations suivantes.

(1) Calculer $\sum_{k=0}^{n} \frac{1}{16^k}\left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right)$ pour $n = 1, 5, 10$ en utilisant la fonction `sum`. En utilisant `R=RealField(p)` en déduire une valeur approchée que l'on comparera à π (la variable p est le nombre de bits de précision de la sortie et on peut déclarer π par `R(pi)`).
(2) Écrire une fonction qui construit la liste des 100 premiers nombres premiers en utilisant `next_prime()`.
(3) Factoriser 10! (avec la commande `factor()`) et récupérer la liste des exposants des facteurs premiers.
(4) Pour un entier $n > 1$ donné ($n = 10$), écrire une fonction qui renvoie la probabilité d'un évènement $k \in [0, \dots, n^2]$ pour une loi binomiale de probabilité de succès $1/n$. Tracer le graphe pour $k = 0, \dots, 2n$ et le comparer à la loi normale $\mathcal{N} = \frac{1}{\sqrt{2\pi\sigma}} \cdot \exp(-(x-M)^2/(2\sigma))$ où $M = n$ est la moyenne et $\sigma = n - 1$ la variance.
(5) Écrire 105 en base 2 avec la commande `str()`.
(6) Calculer le reste et le quotient de la division de 105 par 57.
(7) Déterminer le plus petit nombre pseudo-premier en base 2 (i.e. un entier n composé tel que $2^{n-1} \equiv 1 \pmod{n}$).
(8) En utilisant la fonction `crt`, calculer une solution de

$$\begin{cases} x & \equiv 3 \pmod{5}, \\ x & \equiv 7 \pmod{11}. \end{cases}$$

On peut définir l'anneau $\mathbb{Z}/n\mathbb{Z}$ par `R=IntegerModRing(n)`. On peut alors réaliser tous les calculs dans cet anneau. Pour relever un résultat $b = R(a)$ à $\mathbb{Z}$, on peut écrire $\mathbb{Z}(b)$.

(9) Déterminer les ordres multiplicatifs des éléments de $(\mathbb{Z}/15\mathbb{Z})^*$.

Les corps finis se définissent par la fonction `FiniteField(q)` ou `GF(q)`. Lorsque q n'est pas premier, il faut définir un générateur de $\mathbb{F}_q$ comme $\mathbb{F}_p$-algèbre. Par exemple pour $\mathbb{F}_{3^5}$, on écrira `F.<a>=GF(q)`. Pour trouver le polynôme minimal utilisé par Sage pour définir l'extension, on demande `F.polynomial()`.

(10) Soit la courbe affine $y^2 = x^3 + 5x + 3$ sur $\mathbb{F}_{5^3}$. Trouver tous les points sur cette courbe.

3. Polynômes à une ou plusieurs indéterminées

Pour définir un anneau de polynômes en les variables $x_1, \ldots, x_n$ sur un anneau A, cela se fait grâce à `R.<x1,...,xn>=PolynomialRing(A)`.

(1) Définir l'anneau $R = \mathbb{F}_5[x]$ puis le polynôme $P = x^3 + 3x + 2$. Est ce que P est irréductible ?
(2) Réduire le polynôme cyclotomique ϕ_{25} dans R. On le note Q. Le factoriser.
(3) Calculer le reste de la division euclidienne de Q par P puis le pgcd étendu de Q et P.
(4) Définir le corps de rupture de P grâce à

```
F53.<a>=GF(5^3, name='a', modulus=P)
```

(5) Implémenter l'interpolation de Lagrange.
(6) Trouver les points d'intersection des ellipses d'équations $P = 0$ et $Q = 0$ avec $P = X^2 - XY + Y^2 - 1$ et $Q = 2X^2 + Y^2 - Y - 2$. Tracer les courbes. La fonction pour le résultant est `P.resultant(Q,X)`.
(7) Fabriquer l'équation implicite de la courbe paramétrée par $x = t^2 + t + 1$, $y = (t^2 - 1)/(t^2 + 1)$ et tracer la courbe des deux manières.
(8) Soit F une famille de courbes à un paramètre réel t. L'enveloppe de F est l'ensemble des points (x, y) qui vérifient les deux équations

$$F(x, y, t) = 0, \quad \frac{\partial F(x, y, t)}{\partial t} = 0$$

pour au moins un t. Calculez l'enveloppe de la famille suivante, en faisant à chaque fois un graphique contenant quelques courbes de la famille et le tracé de l'enveloppe :

$$F(x, y, t) = x^2 + (y - t)^2 - 1/2(t^2 + 1).$$

(9) On se propose de déterminer la formule donnant l'aire d'un triangle en fonction des longueurs de ses trois côtés a, b, c. Grâce à des calculs de résultants, montrer que

$$S^2 = \frac{1}{16}(a + b - c)(b + a + c)(a - b + c)(-c + a - b).$$

4. Matrices

Une matrice M de taille (m, n) sur un anneau A peut se définir avec la commande `matrix(A,m,n,L])` où L est une liste de coefficients dans A. On peut aussi définir l'espace des matrices qu'on considère par `R=MatrixSpace(A,m,n)` puis un éléments par `R(L)`, une matrice aléatoire par `R.random_matrix()`, etc. On peut accéder à un coefficients (i, j) de M par `M[i,j]`. On peut accéder/modifier toute la colonne j par `M[: , j]` ou une partie (à partir de la ligne i) par `M[i : , j]`.

Remark 4.1. Attention ! Comme pour les listes, les matrices sont indicées à partir de 0.

(1) En considérant le pivot de Gauss comme l'action de certaines matrices à gauche, effectuer celui-ci sur la matrice

$$M = \begin{bmatrix} 6 & 2 & 2 \\ 5 & 4 & 4 \\ 6 & 4 & 5 \\ 5 & 1 & 3 \end{bmatrix}$$

définie sur $\mathbb{F}_7$. Pour échanger deux lignes, on pourra utiliser `swap_rows`.

La résolution d'un système linéaire $Av = b$ se fait avec la commande `solve_right`.

(2) résoudre un système linéaire aléatoire sur $\mathbb{Q}$.

(3) Calculer la décomposition de Dunford $M = D + N$ (avec D diagonalisable, N nilpotente et $DN = ND$) de la matrice

$$M = \begin{bmatrix} 2 & 3 & 2 \\ -1 & -2 & -6 \\ 1 & 1 & 5 \end{bmatrix}$$

à l'aide d'une suite d'itérations de Newton

$$X_{n+1} = X_n - \frac{P(X_n)}{P'(X_n)}$$

où P est le radical du polynôme caractéristique de M en partant de M. Quand la suite devient stationnaire, on obtient D.

5. Séries Formelles

On souhaite effectuer la division euclidienne d'un polynôme F de degré m sur $\mathbb{Q}$ par un polynôme G de degré $n < m$ sur $\mathbb{Q}$. Pour procéder de manière quasi-optimale, on utilise l'algorithme suivant

- Calculer $T^m F(1/T)$ et $T^n G(1/T)$;
- Calculer le quotient $(T^m F(1/T))/(T^n G(1/T)) \pmod{T^{m-n+1}}$ par une inversion de série formelle suivie d'un produit. On obtient ainsi

$$T^{m-n} Q(1/T) + T^m R(1/T)/(T^n G(1/T))$$

où Q, R sont le quotient et le reste de la division euclididienne de F par G ;
- en déduire Q en inversant l'ordre des coefficients ;
- en déduire R par la différence.

(1) Implémenter l'algorithme de calcul de la division euclidienne ci-dessus.

On utilisera `PowerSeriesRing(QQ)` pour définir les séries formelles. L'opération de troncature est `truncate()`.

6. Approfondissements

7. Graphes

Sage possède des fonctions pour gérer les graphes, comme suit

- Initialiser un graphe orienté (resp. non orienté) vide avec le constructeur `DiGraph()` (resp. `Graph()`).
- Ajouter les sommets avec la commande `add_vertex` ou `add_vertices`
- Ajouter les arcs (resp. arêtes) avec la commande `add_edge` ou `add_edges`.

Pour connaitre les fonctions implémentés sur les graphes, il suffit d'entrer le nom d'une instance de graphe (par exemple ici on a initialisé `G=Graph()`), suivi du point '.' et faire tab pour que s'affiche la liste des 257 fonctionMéthodes prédéfinies dans la librairie. Par exemple la fonction `G.topological_sort()` retourne une liste topologique si le graphe est sans circuit.
On rappelle que la matrice d'incidence d'un graphe orienté $G = (S, A)$ est la matrice Δ tel que $\delta_{xa} = 1$ si x est l'origine de l'arc a, -1 si x est l'extrémité de l'arc a et 0 sinon.

(1) Construire et afficher le graphe donné par la matrice d'adjacence

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

8. Groupes finis et représentations

Il est possible de travailler avec un grand nombre de types de groupes dans Sage, parmi lesquels le groupe symétrique `SymmetricGroup()`, le groupe dihédral `DihedralGroup()`, les groupes matriciels `MatrixGroup` . . . pour lesquels on peut recueillir beaucoup d'informations (ordre `G.order()`, abélien `G.is_abelian()`, simplicité `G.is_simple()`, liste des éléments `G.list()`). Il est aussi possible de décrire un sous-groupe d'un groupe symétrique en spécifiant des générateurs, par exemple le groupe de symétrie du cube

```
cube=PermutationGroup(["(3,2,6,7)(4,1,5,8)","(1,2,6,5)(4,3,7,8)",
 "(1,2,3,4)(5,6,7,8)"])
```

On peut vérifier qu'il est isomorphe à $\mathcal{S}(4)$ par `cube.is_isomorphic(SymmetricGroup(4))`.

(1) Pour chaque élément g de `cube`, déterminer l'orbite du sommet 1 par l'action de g.
(2) Dessiner le cube avec la fonction `polytopes.n_cube(3)`.
(3) Donner la table des caractères de `cube` avec la fonction `character_table`. On pourra appeler l'aide sur cette fonction par `cube.character_table?`.

8.1. **Calcul exact sur des extensions.** On considère le polynôme $P = x^3 + 3x + 1$.

(1) écrire formellement toutes ces racines sur $\bar{\mathbb{F}}_{17}$ puis dans $\bar{\mathbb{Q}}$.
(2) Écrire de manière exacte les points d'intersection des coniques $x^2 + y^2 - 1 = 0$ et $x^2 + 3y^2 - 2 = 0$ dans $\bar{\mathbb{Q}}$.

8.2. **Énumeration dans les groupes.** On a le résultat suivant : soient S, T deux éléments d'ordre fini d'un groupe G. On construit $E_0 = \{e\}$ puis $E_{k+1} = E_k \cup \{MS, M \in E_k\} \cup \{MT, M \in E_k\}$. S'il existe k tel que $E_{k+1} = E_k$ alors $E_k = \langle S, T \rangle$.

Programmer cette procédure pour le groupe S_5 et les éléments $(1, 2)$ et $(1, 2, 3, 4)$ puis pour le groupe $\mathrm{GL}_3(\mathbb{Z}/6\mathbb{Z})$ et les éléments

$$S = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, T = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

8.3. **Théorie des nombres : test de Fermat.**

- Écrire une procédure `TestFermat` qui prend en entrée deux entiers n et a et qui rend en sortie `true` si n vérifie $a^{n-1} \equiv 1$ (mod n) et `false` sinon. Que dire de la primalité de n si le programme rend `true` ? s'il rend `false` ?
- Écrire une procédure `EstPseudoPremier` qui prend en entrée deux entiers n et a et qui rend en sortie true si n est a-pseudo-premier et `false` sinon.
- Vérifier que 341 est le plus petit nombre entier qui soit 2-pseudo-premier. Est-il 3-pseudo- premier ?
- Un nombre de Carmichael est un entier composé n qui est a-pseudo-premier pour tout entier a de $1, ..., n-1$ premier avec n. Écrire une procédure `EstCarmichael` qui teste si un nombre entier est un nombre de Carmichael.
- Trouver le plus petit nombre de Carmichael N.
- Écrire une procédure `ProportionFermat` qui prend en entrée un entier $n \geq 3$ et rend en sortie le nombre A/B où A est le nombre d'entiers a compris entre 1 et $n-1$ et premiers avec n pour lesquels `TestFermat(n,a) =True` et B est le nombre d'entiers a compris entre 1 et $n-1$ et premiers avec n.
- Vérifier que `ProportionFermat(15)=1/2` et `ProportionFermat(N)=1`.
- Construire la liste des $[n; ProportionFermat(n)]$ pour n entier composé compris entre 3 et 1000 (ne pas afficher la liste) puis représenter les points obtenus sur un graphe. Commenter.

8.4. **Matrices.**

Exercice 1. Écrire un programme qui détermine si deux matrices A et B sont semblables et renvoie la matrice U de passage telle que $A = U^{-1}BU$ (on pourra renvoyer None dans le cas où les matrices ne sont pas semblables).

Exercice 2. (Racine carrée d'une matrice symétrique semi-définie positive). Soit A une matrice symétrique semi-définie positive (c'est-à-dire qui vérifie $x^tAx \geq 0$ pour tout vecteur x). Montrer qu'on peut calculer une matrice X, elle aussi symétrique semi-définie positive, telle que $X^2 = A$.