Les calculatrices sont interdites. Numérotez vos feuilles et faites apparaître les questions dans l'ordre du sujet. Si vous repérez une erreur d'énoncé, signalez-la sur votre copie et poursuivez votre composition.

1 Introduction

Dans tout le sujet, sauf mention contraire, on s'intéresse à un graphe non orienté $G = (S, A)$ représenté en Python par listes d'adjacence. Par exemple, les graphes :

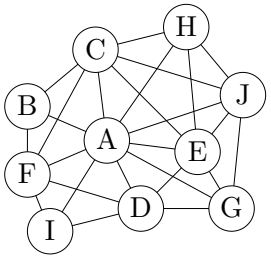

FIGURE 1

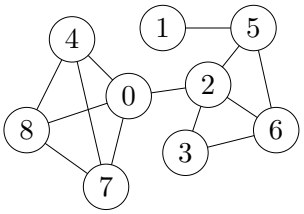

FIGURE 2

sont représentés par :

```
G_fig1 = { "A": ["B", "C", "D", "E", "F", "G", "H", "I", "J"],
           "B": ["A", "C", "F"],
           "C": ["A", "B", "E", "F", "H", "J"],
           "D": ["A", "E", "F", "G", "I"],
           "E": ["A", "C", "D", "G", "H", "J"],
           "F": ["A", "B", "C", "D", "I"],
           "G": ["A", "D", "E", "J"],
           "H": ["A", "C", "E", "J"],
           "I": ["A", "D", "F"],
           "J": ["A", "C", "E", "G", "H"] }
```

```
G_fig2 = {
    0: [2, 4, 7, 8],
    1: [5],
    2: [0, 3, 5, 6],
    3: [2, 6],
    4: [0, 7, 8],
    5: [1, 2, 6],
    6: [2, 3, 5],
    7: [0, 4, 8],
    8: [0, 4, 7] }
```

Dans la suite, on notera « `sommet` » le type Python utilisé pour représenter les sommets de G. Par exemple, pour le graphe de la figure 1, on prendra `sommet = str` et pour le graphe de la figure 2 on prendra `sommet = int`. De plus, le type Python « `graphe` » sera une abréviation pour « `dict[sommet: list[sommet]]` ». En particulier, `G_fig1` et `G_fig2` sont deux variables de type `graphe`.

1. (a) Dans un graphe non orienté, donner la définition d'une "boucle".
   (b) Écrire une fonction `voisins(G: graphe, u: sommet, v: sommet) -> bool` qui indique si `u` et `v` sont voisins dans `G`. On pourra supposer sans le vérifier que `u` et `v` sont des clés du dictionnaire `G`.
   (c) Écrire une fonction `boucles(G: graphe) -> int` qui renvoie le nombre de boucles dans `G`.

Jusqu'à la fin du sujet, on s'intéresse uniquement à des graphes sans boucle. Ainsi, si une variable `G` est de type `graphe`, alors vous pouvez supposer sans le vérifier que `boucles(G) = 0`.

Soit $G = (S, A)$ un graphe non orienté et $C \subset S$. On dit que C est une ***clique*** si pour tous sommets s_1 et s_2, si $s_1 \neq s_2$ alors s_1 et s_2 sont adjacents dans G. En d'autres termes :

$$C \text{ est une clique} \Leftrightarrow \Big[\forall s_1 \in C, \forall s_2 \in C : \Big(s_1 \neq s_2 \Rightarrow \{s_1, s_2\} \in A\Big)\Big]$$

Par exemple, dans le graphe de la figure 1, les ensembles suivants sont des cliques :

$$\{A, B, C, F\} \qquad \{A, C, E, H\} \qquad \{A, C, E, H, J\} \qquad \{E, D, G\} \qquad \{A, I\}$$

2. Soit $G = (S, A)$ le graphe de la figure 2.

(a) Donner sans justification une clique de cardinal 1, une clique de cardinal 2, une clique de cardinal 3 et une clique de cardinal 4 dans G.

(b) Donner sans justification le nombre de cliques de cardinal 3 dans G.

3. Soit $G = (S, A)$ un graphe non orienté quelconque.

(a) Déterminer en le justifiant le nombre de cliques de cardinal 1 dans G.

(b) Déterminer en le justifiant le nombre de cliques de cardinal 2 dans G.

En Python, un ensemble de sommets est représenté par une liste sans doublon « `C: list[sommet]` ». Par exemple, l'ensemble $\{A, B, C, F\}$ est représenté par la liste `["A", "B", "C", "F"]` (l'ordre des éléments n'a pas d'importance).

4. Écrire une fonction `clique(G: graphe, C: list[sommet]) -> bool` qui indique si `C` est une clique de G. On pourra supposer sans le vérifier que les éléments de `C` sont distincts deux à deux et que ce sont tous des sommets de `G`.

2 Nombre de cliques

5. (a) Écrire une fonction `sommets(G: graphe) -> list[sommet]` qui renvoie une liste `S` contenant une et une seule fois chaque sommet de `G`. Par exemple, `sommets(G_fig1)` doit renvoyer (l'ordre des éléments de la liste n'a pas d'importance) :

```
["A", "B", "C", "D", "E", "F", "G", "H", "I", "J"]
```

(b) Quelle est la complexité de votre fonction ? Justifier.

Soit $G = (S, A)$ un graphe non orienté et $n = |S|$ le nombre de sommets dans G. Dans cette partie, on souhaite compter le nombre de sous-ensembles $C \subset S$ tels que C est une clique. Pour cela, on va écrire des entiers en base 2 pour générer tous les sous-ensembles de S. Plus précisément, nous allons construire une bijection φ entre les ensembles :

$$E_1 = \Big\{0, 1, 2, \dots, 2^n - 1\Big\} \qquad \text{et} \qquad E_2 = \Big\{C : C \subset S\Big\}.$$

Soit $i \in E_1$. On note $\alpha(i, n)$ la chaîne de caractères contenant l'écriture en base 2 sur n bits de i. Par exemple $\alpha(372, 10) =$ `"0101110100"`.

6. (a) En donnant le détail des calculs, déterminer $\alpha(678, 12)$.

(b) Écrire une fonction `bin(i: int, n: int) -> str` qui renvoie $\alpha($`i`$,$`n`$)$. Si `i` $\notin [\![$ `0 ; 2**n - 1` $]\!]$, votre fonction déclenchera une erreur.

Décrivons maintenant la bijection $\varphi : E_1 \to E_2$. Soit $i \in E_1$, soit `S` la liste renvoyée par la fonction de la question 5a et `t` la chaîne de caractères $\alpha(i, n)$. On pose :

$$\varphi(i) = \Big\{\texttt{S[k] : k} \in [\![\, \texttt{0 ; n - 1} \,]\!] \texttt{ et t[k] = "1"}\Big\}$$

Pour le graphe de la figure 1 où `S = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J"]`. On obtient $\varphi(372) = \Big\{$`"B", "D", "E", "F", "H"`$\Big\}$.

7. Soit $G = (S, A)$ un graphe non orienté quelconque et $n = |S|$.

(a) Déterminer en le justifiant $\varphi(0)$.

(b) Déterminer en le justifiant $\varphi(2^n - 1)$.

8. (a) Soit `G` un graphe, soit `S` le résultat de l'appel à `sommets(G)`, soit `n` le nombre de sommets dans `G` et soit `i` $\in [\![$ `0 ; 2**n - 1` $]\!]$. Écrire une fonction `phi(S: list[sommet], i: int) -> list[sommet]` qui renvoie $\varphi($`i`$)$.

(b) Écrire une fonction `cpt(G: graphe) -> int` qui renvoie le nombre de cliques dans `G`. Par exemple `cpt(G_fig1) = 70` et `cpt(G_fig2) = 30`.

9. Soit `G` un graphe, soit `S` le résultat de l'appel à `sommets(G)`, soit `n` le nombre de sommets dans `G`, soit `i` $\in [\![0 ; 2^{n} - 1]\!]$ et soit `C` le résultat de l'appel à `phi(S, i)`. Écrire une fonction

```
phiInv(S: list[sommet], C: list[sommet]) -> int
```

qui renvoie l'entier `i`.

10. Écrire une fonction `Gmax(n: int) -> graphe` qui renvoie un exemple de graphe `G` avec `n` sommets tel que `cpt(G) = 2**n`.

3 Matrices d'adjacence et cliques de cardinal 3

Soit $G = (S, A)$ un graphe non orienté et n le nombre de sommets dans G. Dans cette partie, on suppose que $S = [\![0 ; n-1]\!]$. Contrairement aux autres parties, on s'intéresse ici à la matrice d'adjacence de G notée $M \in \mathcal{M}_n(\{0,1\})$. Par exemple, pour le graphe de la figure 2 on obtient `M_fig2` :

```
M_fig2 = [ [0, 0, 1, 0, 1, 0, 0, 1, 1],
           [0, 0, 0, 0, 0, 1, 0, 0, 0],
           [1, 0, 0, 1, 0, 1, 1, 0, 0],
           [0, 0, 1, 0, 0, 0, 1, 0, 0],
           [1, 0, 0, 0, 0, 0, 0, 1, 1],
           [0, 1, 1, 0, 0, 0, 1, 0, 0],
           [0, 0, 1, 1, 0, 1, 0, 0, 0],
           [1, 0, 0, 0, 1, 0, 0, 0, 1],
           [1, 0, 0, 0, 1, 0, 0, 1, 0] ]
```

```
N_fig2 = [ [4, 0, 0, 1, 2, 1, 1, 2, 2],
           [0, 1, 1, 0, 0, 0, 1, 0, 0],
           [0, 1, 4, 1, 1, 1, 2, 1, 1],
           [1, 0, 1, 2, 0, 2, 1, 0, 0],
           [2, 0, 1, 0, 3, 0, 0, 2, 2],
           [1, 0, 1, 2, 0, 3, 1, 0, 0],
           [1, 1, 2, 1, 0, 1, 3, 0, 0],
           [2, 0, 1, 0, 2, 0, 0, 3, 2],
           [2, 0, 1, 0, 2, 0, 0, 2, 3] ]
```

11. (a) Écrire une fonction `mat(G: graphe) -> list[list[int]]` qui prend en entrée le dictionnaire contenant les listes d'adjacence d'un graphe et renvoie sa matrice d'adjacence. Par exemple, `mat(G_fig2)` doit renvoyer `M_fig2`. On pourra supposer sans le vérifier que l'ensemble des sommets est de la forme $[\![0 ; n-1]\!]$.

    (b) Écrire une fonction `adj(M: list[list[int]]) -> graphe` qui prend en entrée la matrice d'adjacence d'un graphe et renvoie le dictionnaire contenant ses listes d'adjacence. Par exemple, `adj(M_fig2)` doit renvoyer `G_fig2`.

Soit $M = \mathcal{M}_n(\{0,1\})$ la matrice d'adjacence de G et $N = M^2$ (où M^2 désigne le carré de M au sens du cours de mathématiques). On admet que G contient une clique de cardinal 3 si et seulement si :

$$\text{Il existe } (i,j) \in [\![0 ; n-1]\!]^2 \text{ tels que } M_{i,j} \neq 0 \text{ et } N_{i,j} \neq 0$$

12. (a) Écrire une fonction `carre(M: list[list[int]]) -> list[list[int]]` qui renvoie `M**2`. Par exemple, `carre(M_fig2) = N_fig2`.

    (b) Soit `M` la matrice d'adjacence du graphe G. Écrire une fonction

```
clique3(M: list[list[int]]) -> bool
```

qui indique si G contient une clique de cardinal 3. Vous devez utiliser l'équivalence évoquée ci-dessus.

4 Clique maximale

Soit $G = (S, A)$ un graphe non orienté et $C_0 \subset S$ une clique. L'objectif de cette partie est de trouver une clique maximale contenant C_0. En d'autres termes, on souhaite construire un ensemble $C \subset S$ vérifiant trois propriétés :

$$C_0 \subset C \qquad C \text{ est une clique} \qquad \forall s \in S \setminus C, \text{l'ensemble } C \cup \{s\} \text{ n'est pas une clique.}$$

13. Dans le graphe de la figure 1, donner sans justification deux cliques maximales contenant $\{E, J\}$.

14. Écrire une fonction `cliqueMaxi(G: graphe, C0: list[sommet]) -> list[sommet]` qui renvoie une clique maximale contenant `C0`. On expliquera succinctement la procédure utilisée et on essaiera d'être le plus efficace possible.

5 Graphe de permutation

Soit $n \in \mathbb{N}^*$. Pour toute fonction bijective $\sigma : [\![1\,;n]\!] \to [\![1\,;n]\!]$, on note $G_\sigma = (S, A)$ le graphe non orienté défini par :

$$S = [\![1\,;n]\!] \qquad \text{et} \qquad A = \Big\{\{i,j\} \mid (i,j) \in [\![1\,;n]\!]^2, i < j, \sigma(i) > \sigma(j)\Big\}$$

Par exemple, avec $n = 6$ et $\sigma_0 : [\![1\,;6]\!] \to [\![1\,;6]\!]$ la fonction telle que :

$$\sigma_0(1) = 2 \qquad \sigma_0(2) = 5 \qquad \sigma_0(3) = 1 \qquad \sigma_0(4) = 4 \qquad \sigma_0(5) = 3 \qquad \sigma_0(6) = 6$$

on obtient le graphe de la figure 3.

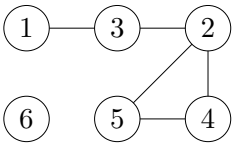

FIGURE 3

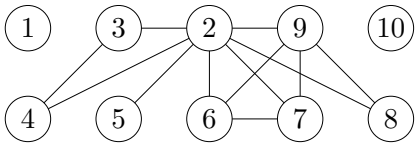

FIGURE 4

15. Donner sans justification une fonction σ telle que G_σ soit le graphe de la figure 4.

En Python, une fonction bijective $\sigma : [\![1\,;n]\!] \to [\![1\,;n]\!]$ sera représentée par la liste `P`$_\sigma$:

`P`$_\sigma$ `= [`σ`(1), `σ`(2), `σ`(3), ..., `σ`(n)]`.

Par exemple, avec la fonction σ_0 définie ci-dessus : `P`$_{\sigma_0}$ `= [2, 5, 1, 4, 3, 6]`.

16. Soit $n \in \mathbb{N}$ et $\sigma : [\![1\,;n]\!] \to [\![1\,;n]\!]$ une fonction bijective. Écrire une fonction `GPerm` qui prend en entrée `P`$_\sigma$ et renvoie G_σ.

Soit `P = [`$\mathtt{x}_1$`, `$\mathtt{x}_2$`, ..., `$\mathtt{x}_n$`]` une liste. On appelle ***sous-liste*** de `P` une liste de la forme `[`$\mathtt{x}_{i_1}$`, `$\mathtt{x}_{i_2}$`, ..., `$\mathtt{x}_{i_k}$`]` avec $1 \leqslant i_1 < i_2 < \ldots < i_k \leqslant n$. On admet la propriété suivante :

> Soit $\sigma : [\![1\,;n]\!] \to [\![1\,;n]\!]$ une fonction bijective et C un ensemble tel que :
>
> $$C = \{i_1, i_2, \ldots, i_k\} \qquad \text{où} \qquad 1 \leqslant i_1 < i_2 < \ldots < i_k \leqslant n \text{ sont des entiers.}$$
>
> Alors, C est une clique de G_σ si et seulement si $\sigma(i_1) > \sigma(i_2) > \ldots > \sigma(i_k)$.

Ainsi, trouver une clique dans G_σ revient à trouver une sous-liste strictement décroissante de `P`$_\sigma$. Par exemple, avec la fonction σ_0 définie ci-dessus, la liste `P`$_{\sigma_0}$ `= [2, 5, 1, 4, 3, 6]` admet pour sous-liste strictement décroissante :

`[5, 4, 3] = [`σ`(2), `σ`(4), `σ`(5)]`

et l'ensemble $\{2, 4, 5\}$ est bien une clique de G_σ.

Soit `P` une liste d'entiers distincts deux à deux. Notre objectif est de trouver une sous-liste strictement décroissante de `P` de longueur maximale. Pour cela, on va construire une liste `S` telle que `S[k]` est la sous-liste de `P` de longueur `k` dont le dernier élément est maximal (parmi toutes les sous-listes de `P` de longueur `k`). Par exemple pour la fonction σ_0 définie ci-dessus :

```
S = [[], [6], [5, 4], [5, 4, 3]]
```

17. (a) Écrire une fonction `makeS(P: list[int]) -> list[list[int]]` qui renvoie la liste `S` décrite ci-dessus. Votre procédure devra s'exécuter en temps $\mathcal{O}(n^2)$ avec n la longueur de `P`
    (b) Soit $n \in \mathbb{N}$ et $\sigma : [\![1\,;n]\!] \to [\![1\,;n]\!]$ une fonction bijective. Écrire une fonction `cliquePerm` qui prend en entrée la liste `P`$_\sigma$ et renvoie une clique de cardinal maximal de G_σ.