Nous vous proposons dans ce document plusieurs problèmes. Pour les résoudre il vous faudra mettre en œuvre vos connaissances et compétences en algorithmique. Il ne s'agit pas d'exercices d'application, une bonne dose de réflexion est nécessaire pour pouvoir concevoir une solution algorithmique.

La majorité des problèmes sont posés sur des *graphes*. Vous trouverez les définitions nécessaires dans la section suivante.

Structure de graphe

Un graphe G est une structure mathématique $\langle V, E \rangle$ où :

- ☞ V est un ensemble fini, dit ensemble des sommets du graphe G
- ☞ E est une relation binaire sur V, c'est-à-dire, un sous-ensemble du produit cartésien $V \times V$. Si la relation E est symétrique alors le graphe est dit *non orienté* et les éléments de E sont appelés *arêtes*. Dans le cas contraire le graphe est dit *orienté* et les éléments de E sont appelés *arcs*.

Puisque l'ensemble des sommets V est fini, nous pouvons, sans perte de généralité, le supposer égal à $\{1, \ldots, n\}$ où n est le nombre de sommets du graphe.

Les graphes sont un outil de modélisation assez puissant. En effet, un grand nombre de problèmes de la vie de tous les jours peuvent être modélisés comme des problèmes sur des graphes. En effet, considérons la figure 1 représentant le plan du métro parisien. Un problème que vous résolvez assez souvent est celui de déterminer un chemin pour

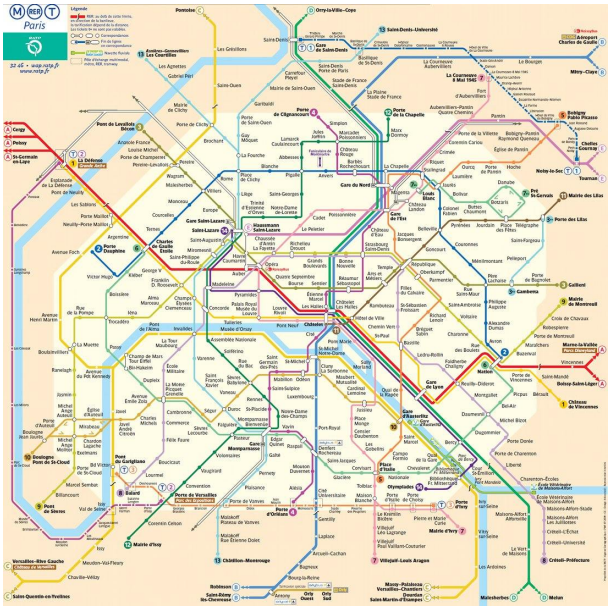

FIGURE 1 – Plan de métro île de France.

aller d'une station donnée vers une autre. Dans ce cas, le problème se modélise comme suit : Considérons le graphe G tel que :

- ☞ l'ensemble de ses sommets correspond aux stations de métro,
- ☞ la relation E est définie comme suit : deux stations u et v sont en relation si et seulement si il existe une ligne un métro allant de u à v en un seul arrêt (d'autres définitions sont possibles). Puisque toutes les lignes de métros sont à double sens alors nous pouvons envisager un graphe non orienté.

Ainsi, déterminer un chemin pour aller d'une station u à une autre v se réduit à déterminer dans le graphe G si les deux sommets u et v sont accessibles ou pas. De même que déterminer un plus court chemin suivant un certain critère (nombre d'arrêts, nombre de correspondances, le temps de parcours, ...) est un problème de graphes.

Un autre exemple de problème [1] dans les graphes est celui du coloriage d'une carte du monde avec un minimum de couleurs possible. La Figure 2 montre une carte du monde (déjà coloriée !). Un problème de coloriage consiste à

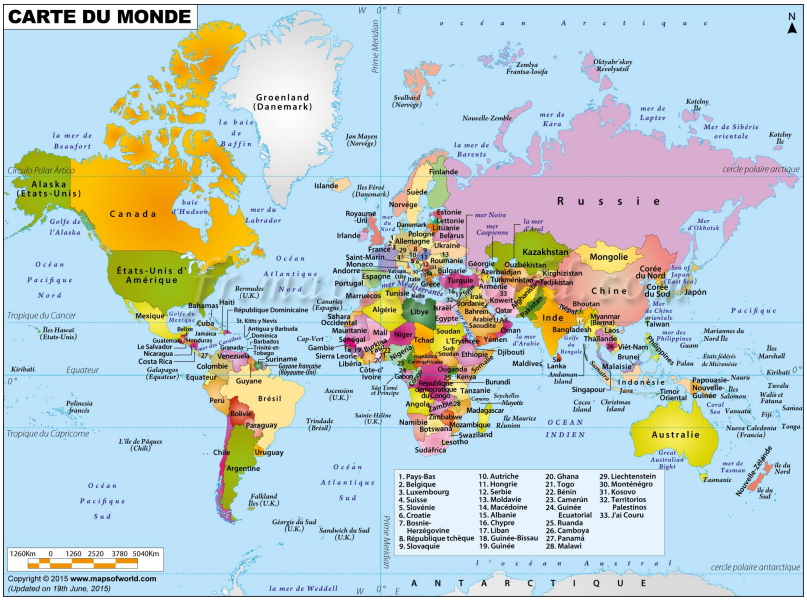

FIGURE 2 – Carte du monde.

affecter une couleur à chaque pays de telle sorte que chaque pays voisin est colorié avec une couleur différente. Jusque là le problème est facile (pourquoi ?). En revanche, le problème se complique si nous exigeons, en plus, de trouver le nombre minimum de couleurs pour réaliser se coloriage. Pour cet exemple, le graphe G associé au problème est le suivant :

- ☞ l'ensemble des sommets correspond au l'ensemble des pays,
- ☞ l'ensemble des arêtes E est le suivant : deux sommets (donc deux pays) u et v sont en relation si et seulement u et v ont une frontière commune.

Le problème du coloriage dont il était question ci-dessus, se réduit donc à colorier avec un minimum possible de couleur les sommets du graphe G de telle sorte que deux sommets partageant la même arrête [2] aient deux couleurs différentes.

Pour un graphe $G = \langle V, E \rangle$ donné, un sous-graphe H de G est un graphe tel que sont ensemble de sommets $V[H]$ et l'ensemble de ces arcs ou arêtes $E[H]$ satisfont les deux inclusions suivantes :

$$V[H] \subseteq V \ \ et \ E[H] \subseteq V.$$

Dans le cas ou $V[H]$ est égal à l'ensemble V et que l'ensemble $E[H]$ est un sous-ensemble distinct de E alors le graphe H est appelé graphe partiel de G. Pour tout sous-ensemble S de V le sous-graphe induit par S est un sous-graphe H tel que : $V[H]$ est égal à S et $E[H]$ est formé des arcs ou arêtes dont les deux extrémités appartiennent à l'ensemble S.

Liste des sujets

1. Plus difficile celui-ci !
2. Dans ce cas les deux sommets sont adjacents.

Sujet 1 : Arbres Couvrants

Soit $G = \langle V, E, \pi, \Delta \rangle$ un graphe non orienté, connexe et dont les arêtes sont valuées par la fonction $\pi : E \longrightarrow \mathbb{N}$ et les sommets par la fonction $\Delta : V \longrightarrow \mathbb{N}$. Pour toute arête $e \in E$, la valuation $\pi(e)$ est aussi appelée poids de l'arête e.

Un *arbre couvrant* de G est un sous-graphe partiel de G dont l'ensemble des sommets est V. Le poids d'un arbre est, par définition, la somme des poids de ses arêtes. Nous dirons qu'un arbre couvrant satisfait la valuation Δ si pour tout sommet $v \in V$ nous avons $d(v) \leq \Delta(v)$.

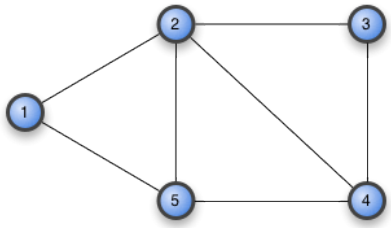

FIGURE 3 – Graphe G

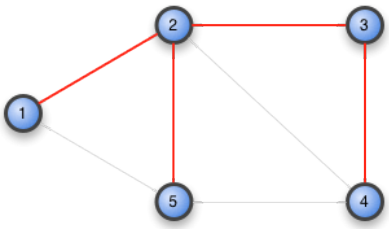

FIGURE 4 – Arbre $\mathcal{A}_1$

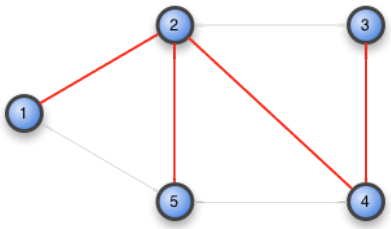

FIGURE 5 – Arbre $\mathcal{A}_2$

A titre d'exemple, la figure 3 représente un graphe G dont toutes les arêtes sont de poids 1. La figure 4 représente un arbre couvrant de G. Soit $\Delta = (1, 3, 1, 2, 2)$ une valuation des sommets du graphe G. L'arbre couvrant de la figure 5, contrairement à celui de la figure 4, satisfait cette valuation.

Cahier des charges

Ci-dessous ce qui vous est demandé. Les initiatives personnelles sont très bien appréciées.

- ☞ Concevoir un algorithme, le plus efficace possible, pour déterminer tous les arbres couvrants de G satisfaisant la valuation Δ et de qui sont de poids minimum.
- ☞ Mettre en oeuvre votre algorithme sur des instances générées aléatoirement.
- ☞ Rédiger un rapport devant contenir : (i) une explication du problème traité ; (ii) la solution algorithmique proposée ; (iii) les structures de données proposées ; Et éventuellement :
  - ❑ une analyse de complexité détaillée,
  - ❑ une analyser détaillée des expériences de calcul réalisées.

Sujet 2 : Numérotations minimales

Soit $G = \langle V, E \rangle$ un graphe non orienté et connexe. Une *numérotation* des sommets de G est une bijection $\eta : V \longrightarrow \{1, \ldots, n\}$ telle que $\eta(v)$ indique le numéro du sommet v de G.

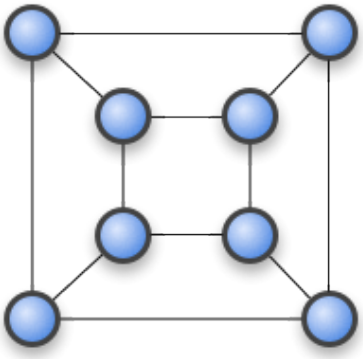

FIGURE 6 – Graphe Q_3

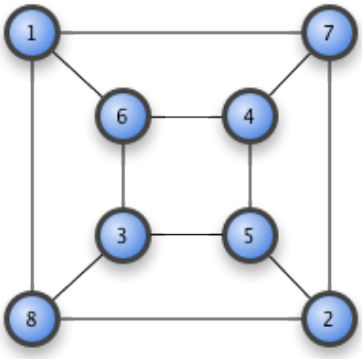

FIGURE 7 – Fonction η_1

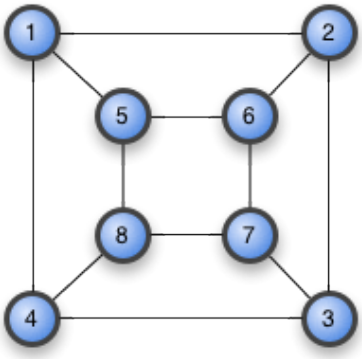

FIGURE 8 – Fonction η_2

A titre d'exemples, les figures 7 et 8 représentent deux numérotations possibles du graphe Q_3 de la figure 6. Nous nous intéresserons aux numérotations de *faible rang*.

Soit η une numérotation des sommets du graphe G. Le rang de la numérotation η est le nombre, noté $r_\eta(G)$, tel que

$$r_\eta(G) = \max\{|\eta(u) - \eta(v)| : (u, v) \in E\}.$$

Ainsi, le rang de la numérotation η_1 est 7 et celui de la numérotation η_2 est 4.

La *numérotation minimale* du graphe G est une numérotation η ayant le plus petit rang $r_\eta(G)$. Nous noterons $\eta(G)$ le rang d'une numérotation minimale de G. Montrer que $\eta(Q_3) = 4$.

Cahier des charges

Ci-dessous ce qui vous est demandé. Les initiatives personnelles sont très bien appréciées.

- ☞ Concevoir un algorithme, le plus efficace possible, pour déterminer toutes les numérotations minimales d'un graphe donné.
- ☞ Mettre en œuvre votre algorithme sur des instances générées aléatoirement.
- ☞ Rédiger un rapport devant contenir : (i) une explication du problème traité ; (ii) la solution algorithmique proposée ; (iii) les structures de données proposées ; Et éventuellement :
  - ❑ une analyse de complexité détaillée,
  - ❑ une analyser détaillée des expériences de calcul réalisées.

Sujet 3 : Ensemble dominants minimaux

Soit $G = \langle V, E \rangle$ un graphe non orienté et connexe. Un sous-ensemble S de sommets de G est dit *ensemble dominant* de G si tout sommet $v \in V$ est soit un élément de S soit il existe un somment de S adjacent à v.

A titre d'exemples, l'ensemble S_1 $\{1,3,5,7\}$ est un *ensemble dominant* du graphe de la figure 9 suivante.

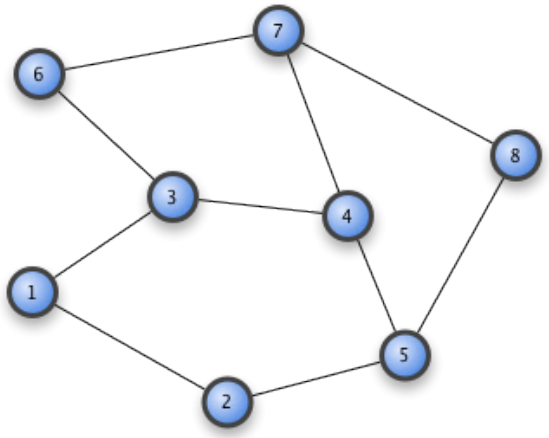

FIGURE 9 – Ensembles dominants

L'ensemble $S_2 = \{3,5,7\}$ est aussi un ensemble dominant du graphe G. Vous remarquerez, au passage, que l'ensemble V est aussi un ensemble dominant de G. L'ensemble $S_3 = \{3,5\}$ n'est pas un ensemble dominant du graphe de la figure 9 car le sommet 7, par exemple, ne lui est pas adjacent.

La taille d'un ensemble dominant est sa cardinalité, c'est-à-dire, le nombre de sommets dont il est composé. L'ensemble dominant de cardinalité minimale est qualifié d'*ensemble dominant minimum*. Dans le graphe de la figure 9 l'ensemble S_2 est dominant minimum. Cependant, il n'est pas unique car l'ensemble $S_4 = \{3,5,8\}$ est aussi dominant minimum.

Cahier des charges

Ci-dessous ce qui vous est demandé. Les initiatives personnelles sont très bien appréciées.

- ☞ Concevoir un algorithme, le plus efficace possible, pour déterminer tous les ensembles dominants minimaux d'un graphe donné.
- ☞ Mettre en œuvre votre algorithme sur des instances générées aléatoirement.
- ☞ Rédiger un rapport devant contenir : (i) une explication du problème traité ; (ii) la solution algorithmique proposée ; (iii) les structures de données proposées ; Et éventuellement :
  - ❑ une analyse de complexité détaillée,
  - ❑ une analyser détaillée des expériences de calcul réalisées.

Sujet 4 : Arbres minimaux

Soit $G = \langle V, E, w, c \rangle$ un graphe non orienté et connexe où : (i) V est l'ensemble des sommets de G ; (ii) E l'ensemble des arêtes de G, (iii) w une application de E dans $\mathbb{N}$, qui à chaque arête e du graphe G associe un poids w_e ; (iv) c est une application de V dans $\mathbb{N}$, qui à chaque sommet j du graphe associe un coût c_j. Enfin, soit T un sous-ensemble de sommets donnés.

L'objectif principal de ce projet est *d'énumérer* tous les *arbres minimaux* (voir ci-dessous) connectant tous les sommets de l'ensemble T. Nous considérerons la mesure suivante :

$$\sum_{e \in E[A]} w_e, \tag{1}$$

où, pour tout arbre $A = \langle V[A], E[A] \rangle$, $V[A]$ est l'ensemble de ses sommets et $E[A]$ celui de ses arêtes.

Pour fixer les idées, soit le graphe ci-dessous.

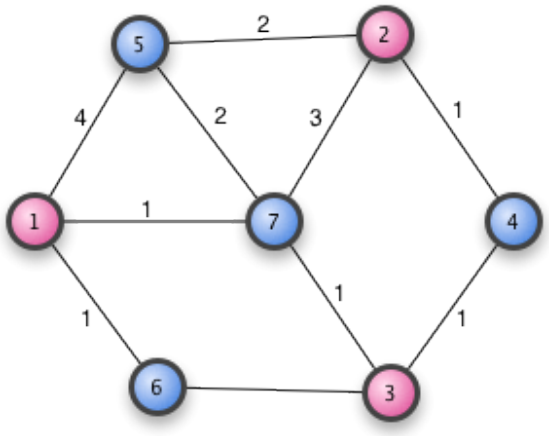

FIGURE 10 – Graphe G

Dans le graphe G ci-dessus la fonction w est donnée par les valeurs sur les arêtes de G ; la fonction c vaut 1 sur tous les sommets de G. Enfin, l'ensemble T est formé des sommets 1, 2 et 3. A titre d'exemple, l'arbre $A_1 = \langle \{1,2,3,7\}, \{(1,7), (7,2), (7,3)\} \rangle$ connecte bien tous les sommets de l'ensemble T. Ce qui n'est pas le cas de l'arbre $A_2 = \langle \{1,3,6\}, \{(1,6), (6,3)\} \rangle$. Le poids de l'arbre A_1 suivant les trois mesures (1), (**??**) et (**??**) est égal à 5, 4 et 9 respectivement.

Cahier des charges

Ci-dessous ce qui vous est demandé. Les initiatives personnelles sont très bien appréciées.

- ☞ Concevoir un algorithme, le plus efficace possible, pour déterminer tous les arbres minimaux d'un graphe donné.
- ☞ Mettre en oeuvre votre algorithme sur des instances générées aléatoirement.
- ☞ Rédiger un rapport devant contenir : (i) une explication du problème traité ; (ii) la solution algorithmique proposée ; (iii) les structures de données proposées ; Et éventuellement :
  - ❑ une analyse de complexité détaillée,
  - ❑ une analyser détaillée des expériences de calcul réalisées.

Sujet 5 : Multiplication de contraintes linéaires

Soit n un entier non nul donné. Une *contrainte* linéaire, dans $\mathbb{R}^n$, est une *expression* de la forme :

$$\sum_{j=1}^{n} a_j x_j \leq \beta, \tag{2}$$

où β et a_j, pour tout j appartenant à $\{1, \ldots, n\}$, sont des réels.

Un *monôme* de degré total d, noté F_d, est l'*expression* suivante :

$$F_d(A) = \prod_{j \in A} x_j,$$

où A est un sous-ensemble de $\{1, \ldots, n\}$ et $|A| = d$ (i.e., le nombre d'éléments de A est égal à d).

Une *substitution de variables* est un *ensemble de règles* indiquant comment remplacer un produit de variables en x par une nouvelle variable. Par exemple, dans le cas où $n = 2$, on peut définir la substitution suivante :

$$\begin{cases} x_1^2 = x_1, \\ x_2^2 = x_2, \\ x_1 x_2 = w. \end{cases} \tag{3}$$

On souhaite *implémenter* une multiplication formelle sur des contraintes linéaires, nous l'appellerons **prodmclin.** Les arguments de la fonction **prodmclin** sont une *contrainte linéaire*, un *monôme* d'un certain degré et une *substitution*. La valeur de retour de la fonction **prodmclin** est une contrainte linéaire.

Voici un exemple. Soit $n = 2$, et considérons la contrainte suivante :

$$x_1 + x_2 \leq 4. \tag{4}$$

Soit le monôme suivant (de degré total 1) :

$$F_1(\{1\}) = x_1. \tag{5}$$

Le produit de la contrainte (4) par le monôme (5) est la contrainte (non linéaire) suivante :

$$x_1^2 + x_1 x_2 \leq 4x_1.$$

En utilisant le substitution (3), nous obtenons la nouvelle contrainte suivante :

$$x_1 + w \leq 4x_1.$$

Après *simplification*, nous obtenons la contrainte équivalente suivante :

$$-3x_1 + w \leq 0.$$

Comme le montre l'exemple ci-dessous, la contrainte obtenue après *substitution* peut contenir, dans son expression, de nouvelles variables (par exemple, la variable w n'apparaîssait pas dans la définition de la contrainte (4)).

Cahier des charges

Ci-dessous ce qui vous est demandé. Les initiatives personnelles sont très bien appréciées.

- ☞ Concevoir un algorithme, le plus efficace possible, implémentant la fonction **prodmclin** décrite ci-dessus.
- ☞ Mettre en oeuvre votre algorithme sur des instances générées aléatoirement.
- ☞ Rédiger un rapport devant contenir : (i) une explication du problème traité ; (ii) la solution algorithmique proposée ; (iii) les structures de données proposées ; Et éventuellement :
  - ❑ une analyse de complexité détaillée,
  - ❑ une analyser détaillée des expériences de calcul réalisées.

Sujet 6 : Filtre de Bloom

Les filtres de Bloom sont des structures de données permettant de connaître rapidement si un élément appartient a un ensemble.

Avec ce type de structure il est possible d'avoir des faux positifs mais les faux négatifs sont impossible. C'est à dire que l'on peut seulement être sûr qu'un élément n'appartient pas un ensemble.

Les filtres de Bloom sont d'accès très rapide et sont des structures très compactes en mémoires. Ils permettent, par exemple, d'éviter des appels inutiles à de très grandes base de données.

Cahier des charges

Ci-dessous ce qui vous est demandé. Les initiatives personnelles sont très bien appréciées.

- ☞ Concevoir une implémentation d'un filtre Bloom.
- ☞ Appliquer votre filtre à la détection de faute d'orthographe dans un fichier texte. (Le fichier `http://www.pallier.org/ressources/dicofr/liste.de.mots.francais.frgut.txt` contient 336531 mots français)
- ☞ Essayer votre algorithme avec différents paramètres.
- ☞ Rédiger un rapport devant contenir : (i) une explication du problème traité ; (ii) la solution algorithmique proposée ; (iii) les structures de données proposées ; et éventuellement :
  - ❑ une analyse de complexité détaillée,
  - ❑ une analyser détaillée des expériences de calcul réalisées.

Sujet 7 : Tas Binomiaux

Un tas binomial est un ensemble d'arbre binomiaux disjoints. Un arbre binomial de rang i est définit récursivement de la façon suivante :

— B_0 est un arbre contenant un unique nœud.

— Si $i > 0$ B_i est construit à partir de 2 arbres binomiaux B_{i-1} en rajoutant l'un comme fils le plus à gauche de la racine de l'autre.

La figure 11 présente les arbres binomiaux pour i=0, 1, 2, et 3.

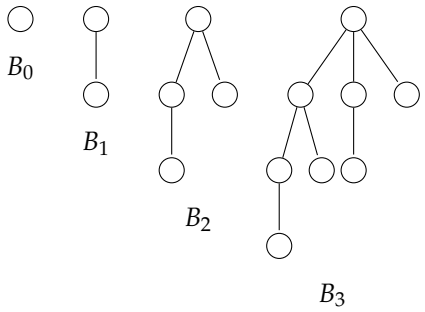

FIGURE 11 – Exemples d'arbres binomiaux

Un tas binomial vérifie alors les propriétés suivantes :

1. Dans chaque arbre binomial, la valeur de chaque nœuds est inférieure à la valeur de ses fils. (comme dans un tas)
2. Les arbres binomiaux de l'ensemble sont tous d'indices différents,
3. Les arbres binomiaux sont triés par ordre d'indices strictement croissants.

Les tas binomiaux sont utilisés pour implémenter des files de priorités lorsque l'opération d'union est nécessaire.

Cahier des charges

Ci-dessous ce qui vous est demandé. Les initiatives personnelles sont très bien appréciées.

☞ Concevoir une implémentation d'un tas binomial avec au moins les opérations suivantes :

1. Création d'un tas binomial
2. Extraction du minimum
3. Union de 2 tas
4. Insertion d'un élément
5. Suppression d'un élément

☞ Essayer vos algorithmes avec différents paramètres.

☞ Comparer la complexité des opérations au tas vu en cours.

☞ Rédiger un rapport devant contenir : (i) une explication du problème traité ; (ii) la solution algorithmique proposée ; (iii) les structures de données proposées ; et éventuellement :

❑ une analyse de complexité détaillée,

❑ une analyser détaillée des expériences de calcul réalisées.