Clôture transitive

Ce sujet est directement adapté d'un problème posé au concours Mines-Ponts en 2014, reformulé de manière parfaitement équivalente en termes de graphes, et traduit en C.

On considère un graphe orienté $G := (V, E)$, et l'on note $n = |V|$ son nombre de sommets. On se fixe une numérotation $x_0, \dots, x_{n-1}$ des sommets.

— Pour $x, y \in V$, on note $x \Longrightarrow_k y$ lorsqu'il existe $k+1$ sommets $y_0, \dots, y_k$ tels que :
  — $y_0 = x$
  — $y_k = y$
  — $(y_i, y_{i+1}) \in E$ pour tout $i \in [\![0, k-1]\!]$.
  On a donc $x \Longrightarrow_0 y$ si et seulement si $x = y$.
— On suppose que pour tout sommet x, la *boucle* (x, x) fait partie de l'ensemble E des arcs. On a donc également $x \Longrightarrow_1 x$ pour tout sommet x.
— On note $x \Longrightarrow y$ s'il existe un entier k tel que $x \Longrightarrow_k y$.

On se référera aux deux graphes ci-dessous dans le problème :

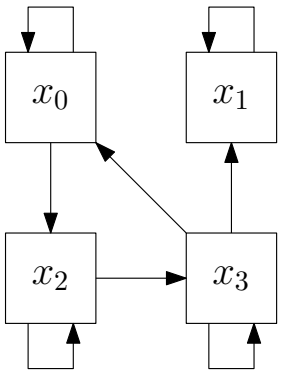

FIGURE 1 – Le graphe G_1.

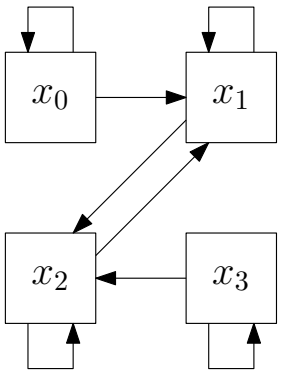

FIGURE 2 – Le graphe G_2.

1. Soit $x, y \in V$ et $h \leqslant k$ deux entiers naturels. Montrer que si $x \Longrightarrow_h y$, alors $x \Longrightarrow_k y$.
2. Soit $x, y \in V$. Montrer que l'on a $x \Longrightarrow y$ si et seulement si $x \Longrightarrow_{n-1} y$.

Une *matrice booléenne* est une matrice dont les coefficients sont à valeurs dans $\{\text{vrai}, \text{faux}\}$. Le produit de deux matrices booléennes s'obtient selon la formule habituelle du produit matriciel, en prenant pour somme de deux valeurs booléennes le « ou logique » (noté $\vee$) et pour produit de deux valeurs booléennes le « et logique » (noté $\wedge$). Dans toute la suite, le produit matriciel, et les puissances d'une matrice, sont à comprendre avec cette définition. On a par exemple

$$\begin{pmatrix} \text{vrai} & \text{vrai} \\ \text{faux} & \text{vrai} \end{pmatrix} \cdot \begin{pmatrix} \text{faux} & \text{vrai} \\ \text{vrai} & \text{faux} \end{pmatrix} = \begin{pmatrix} \text{vrai} & \text{vrai} \\ \text{vrai} & \text{faux} \end{pmatrix}$$

En C, une matrice booléenne M de dimensions (n, p) sera représentée par un `m` de type `bool **`. `m` pointera vers un bloc alloué de n pointeurs de type `bool *`, `m[i]` étant un pointeur vers un bloc alloué de p booléens correspondant à la ligne i de la matrice, les lignes étant numérotées de 0 à $n-1$.

3. Écrire une fonction `matrix_new` renvoyant une nouvelle matrice de la taille spécifiée, initialisée à `false`.

```
bool **matrix_new(int n, int p);
```

4. Écrire une fonction `matrix_free` qui libère toute la mémoire associée à une matrice. On pourra supposer que la matrice a été créée par un appel à `matrix_new` et qu'on a ensuite pu modifier ses coefficients, mais pas les pointeurs. Seul le nombre de lignes est pris en paramètre, le nombre de colonnes étant superflu.

```
void matrix_free(bool **m, int n);
```

5. Écrire une fonction `identity` renvoyant la matrice carrée de taille n contenant des `true` sur la diagonale et des `false` ailleurs.

```
bool **identity(int n);
```

6. Écrire une fonction `product` renvoyant le produit des deux matrices passées en argument.

```
bool **product(bool **a, bool **b, int n, int p, int q);
```

On pourra supposer sans le vérifier que `a` est de dimensions (n, p), `b` de dimensions (p, q) et que tous ces entiers sont strictement positifs.

On associe à un graphe G sa matrice d'adjacence, vue comme une matrice booléenne, que l'on notera A. On a donc $A[i, j]$ qui vaut vrai si et seulement si $x_i \Longrightarrow_1 x_j$. On rappelle que $x_i \Longrightarrow_1 x_i$ pour tout i.

7. Montrer que pour tous $i, j \in [\![0, n-1]\!]$ et pour tout $k > 0$, on a $A^k[i, j] =$ vrai si et seulement si $x_i \Longrightarrow_k x_j$.
8. Montrer que pour $k \geqslant n-1$, on a $A^k = A^{n-1}$.

On appelle *clôture transitive* de la matrice A la matrice $\mathrm{CT}(A) = A^{n-1}$.

9. Écrire une fonction `closure` qui prend en entrée une matrice carrée A de taille n et renvoie sa clôture transivite, calculée à l'aide de multiplications de matrices.

```
bool **closure(bool **a, int n);
```

10. Déterminer la complexité de la fonction `closure`.
11. Écrire une fonction `accessible` qui prend en entrée une matrice carrée A de taille n et un entier $i \in [\![0, n-1]\!]$ et renvoie un pointeur `arr` vers un bloc alloué de n booléens tel que `arr[j]` soit égal à `true` si $x_i \Longrightarrow x_j$, `false` sinon.

```
bool *accessible(bool **a, int n, int i);
```

On exige une complexité temporelle en $\mathrm{O}(n^2)$, que l'on justifiera.

12. Écrire une nouvelle version de la fonction `closure` ayant une complexité temporelle en $\mathrm{O}(n^3)$.

```
bool **closure(bool **a, int n);
```

On dit qu'un sommet x du graphe est un *axiome* s'il possède la propriété suivante : pour tout sommet y tel que $y \Longrightarrow x$, on a $x \Longrightarrow y$.

13. Donner tous les axiomes du graphe G_1.
14. Donner tous les axiomes du graphe G_2.
15. Écrire une fonction `is_axiom` qui prend en entrée la matrice $B \coloneqq \mathrm{CT}(A)$ et un sommet i, et indique si ce sommet est un axiome.

```
bool is_axiom(bool **b, int n, int i);
```

On appelle *suite unidirectionnelle de sommets* une suite finie $y_0, \ldots, y_h$ de sommets vérifiant :
- Pour tout $i \in [\![0, h-1]\!]$, $y_i \Longrightarrow y_{i+1}$
- Pour tout $i \in [\![0, h-1]\!]$, $y_{i+1} \not\Longrightarrow y_i$ (y_i n'est pas accessible depuis y_i).

16. Montrer que les sommets d'une suite unidirectionnelle sont deux à deux distincts.
17. Soit y un sommet. Montrer qu'il existe un axiome x tel que $x \Longrightarrow y$.
18. Donner les composantes fortement connexes du graphe G_2.
19. On considère une composante fortement connexe C contenant un axiome. Montrer que tous les sommets de C sont des axiomes.

On dit qu'une composante fortement connexe est une *composante source* si elle contient un axiome, ce qui revient à dire que tous ses éléments sont des axiomes, d'après la question précédente. On dit qu'une partie X de V est un *système d'axiomes* si, pour tout sommet $y \in V$, il existe un sommet $x \in X$ tel que $x \Longrightarrow y$.

20. Montrer qu'on obtient un système d'axiomes de cardinal minimum en choisissant un et un seul sommet dans chacune des composantes source.

21. Écrire une fonction `axiom_system` prenant en entrée la matrice $B \coloneqq \mathrm{CT}(A)$ et renvoyant un système d'axiome de cardinal minimum. On renverra ce système sous forme d'un bloc alloué `s` de n booléens, tel que `s[i]` soit vrai si et seulement si le sommet i fait partie du système d'axiomes.

```
bool *axiom_system(bool **b, int n);
```