Exercice B0005 *Nombre de sommets accessibles dans un graphe*

Cet énoncé est accompagné d'un ou plusieurs codes compagnons en C fournissant certaines des fonctions mentionnées dans l'énoncé : il sont à compléter en y implémentant les fonctions demandées.

La ligne de compilation `gcc -o main.exe -Wall *.c -lm` *vous permet de créer un exécutable* `main.exe` *à partir du ou des fichiers C fournis. Vous pouvez également utiliser l'utilitaire* `make`*. En ligne de commande, il suffit d'écrire* `make`*. Dans les deux cas, si la compilation réussit, le programme peut être exécuté avec la commande* `./main.exe`*.*

Il est possible d'activer davantage d'avertissements et un outil d'analyse de la gestion de la mémoire avec la ligne de compilation `gcc -o main.exe -g -Wall -Wextra -fsanitize=address *.c -lm` *ou en écrivant* `make safe`*. L'examinateur pourra vous demander de compiler avec ces options.*

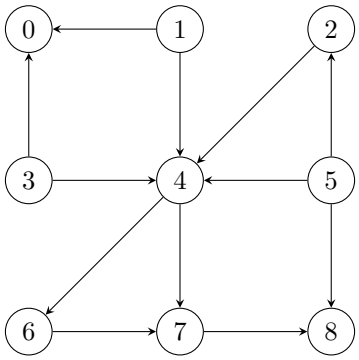
Si vous désirez forcer la compilation de tous les fichiers, vous pouvez au préalable nettoyer le répertoire en faisant `make clean` *et relancer une compilation.*

Dans cet exercice, tous les graphes seront orientés. On représente un graphe orienté $G = (S, A)$, avec $S = [\![0; n-1]\!]$ en C par la structure suivante.

```c
struct graph_s {
  int n;
  int degre[100];
  int voisins[100][10];
};
```

L'entier `n` correspond au nombre de sommets $|S|$ du graphe. On suppose que $n \leq 100$. Pour $0 \leq s < n$, la case `degre[s]` contient le degré sortant $d^+(s)$, c'est-à-dire le nombre de successeurs, appelés ici *voisins*, de s. On suppose que ce degré est toujours inférieur à 10. Pour $0 \leq s < n$, la case `voisins[s]` est un tableau contenant aux indices $0 \leq i < d^+(s)$ les voisins du sommet s. Il s'agit donc d'une représentation par listes d'adjacence, où les listes sont représentées par des tableaux en C.

Un programme en C vous est fourni dans lequel le graphe suivant est représenté par la variable `g_exemple`.



Pour $s \in S$, on note $\mathcal{A}(s)$ l'ensemble des sommets accessibles à partir de s. Pour $s \in S$, le maximum des degrés d'un sommet accessible à partir de s est noté $d^*(s) = \max\{d^+(s') \mid s' \in \mathcal{A}(s)\}$. Par exemple, pour le graphe ci-dessus, $\mathcal{A}(2) = \{2, 4, 6, 7, 8\}$, et $d^*(2) = 2$ car $d^+(4) = 2$. Dans cet exercice, on cherche à calculer $d^*(s)*$ pour chaque sommet $s \in S$.

On représente un sous-ensemble de sommets $S' \subseteq S$ par un tableau de booléens de taille n, contenant `true` à la case d'indice s' si $s' \in S'$ et `false` sinon.

1. Ecrire une fonction `int degre_max(graph* g, bool* partie)` qui calcule le degré maximal d'un sommet $s' \in S'$ dans un graphe $G = (S, A)$ pour une partie $S' \subseteq S$ représentée par S, c'est-à-dire qui calcule $\max\{d^+(s') \mid s' \in S'\}$.
2. Ecrire une fonction `bool* accessibles(graph* g, int s)` qui prend en paramètre un graphe et un sommet s et qui renvoie un (pointeur sur un) tableau de booléens de taille n représentant $\mathcal{A}(s)$. Une fonction `nb_accessibles` qui utilise votre fonction et un test pour l'exemple ci-dessus vous sont donnés dans le fichier à compléter.
3. Ecrire une fonction `int degre_etoile(graph* g, int s)` qui calcule $d^*(s)$ pour un graphe et un sommet passé en paramètre. Quelle est la complexité de votre approche ?
4. Linéariser le graphe donné en exemple ci-dessus, c'est-à-dire représenter ses sommets sur une même ligne dans l'ordre donné par un tri topologique, tous es arcs allant de gauche à droite.
5. Dans cette question, on suppose que le graphe $G = (S, A)$ est acyclique. Décrire un algorithme permettant de calculer tous les $d^*(s)$ pour $s \in S$ en $O(|S| + |A|)$.
6. On ne suppose plus le graphe acyclique. Décrire un algorithme permettant de calculer tous les $d^*(s)$ pour $s \in S$ en $O(|S| + |A|)$.