Degrés étoilés

Consignes

Sauf mention contraire, les questions de ce sujet doivent être traitées dans l'ordre. Chaque question est caractérisée par un ou plusieurs "types", signalés par un pictogramme :

- Les questions marquées avec ⌨ nécessitent d'écrire un programme dans le langage demandé. **Le code produit doit compiler, s'exécuter correctement et être testé même lorsque l'énoncé ne le demande pas explicitement.** La clarté du code est également importante : il faut pouvoir en expliquer le fonctionnement à l'examinateur ou l'examinatrice à sa demande.
- Les questions marquées 👆 sont à traiter à l'oral. Elles peuvent néanmoins être préparées en amont, y compris à l'aide d'un support écrit si vous en avez besoin.
- Les questions marquées ✎ sont aussi à traiter à l'oral mais à l'aide d'un support écrit. Il est demandé pour ces questions d'écrire la réponse ou les grandes lignes de votre raisonnement sur une feuille et de s'appuyer dessus lors de l'explication à l'examinateur ou examinatrice.

Lors du passage de l'examinateur ou de l'examinatrice, vous devez présenter les réponses apportées aux questions traitées depuis le dernier passage. Il est toujours possible de solliciter un passage en levant la main, que ce soit pour vérifier une solution, discuter d'une idée ou demander de l'aide. Une fois que votre appel a été vu, vous pouvez aborder les questions suivantes en attendant.

Le sujet demande de compléter le code compagnon Calcul_degres.c. Ce dernier fournit du code qui sera présenté au fil de l'énoncé. Une partie du code ne pourra être décommentée qu'à partir de la question 2.

- - - - - - -

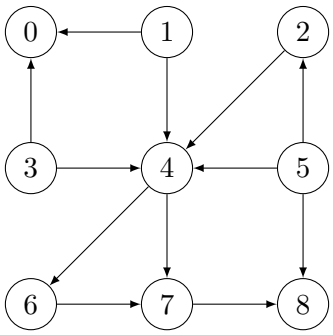
Dans ce sujet, on ne manipule que des graphes orientés. Si s est un sommet d'un tel graphe G, on note $d^+(s)$ son degré sortant et $\mathcal{A}(s)$ l'ensemble des sommets de G accessibles depuis le sommet s. On définit par ailleurs le *degré étoilé* d'un sommet s, noté $d^*(s)$, comme étant le maximum des degrés sortants des sommets accessibles depuis s, c'est-à-dire :

$$d^*(s) = \max\{d^+(s') \mid s' \in \mathcal{A}(s)\}$$

L'objectif du sujet est de concevoir un algorithme qui permette de répondre au problème $\mathcal{P}$ suivant :

$$\mathcal{P} : \begin{cases} \textbf{Entrée :} \text{ Un graphe } G \text{ orienté.} \\ \textbf{Sortie :} \text{ Les degrés étoilés de tous les sommets de } G. \end{cases}$$

Le graphe G_{ex} suivant servira aux exemples et aux tests du code. Il est représenté par la variable `g_ex` définie dans le `main` du code compagnon :



1. 👆 Déterminer le degré étoilé de 5 puis celui de 2 dans le graphe G_{ex}.

Dans la suite, on suppose toujours que les sommets des graphes sont numérotés consécutivement à partir de zéro. On choisit dans un premier temps de représenter un graphe orienté G via cette structure :

```
struct graphe {
  int n;
  int degres[100];
  int voisins[100][10];
};
```

L'entier `n` correspond au nombre de sommets du graphe. Pour $0 \leq$ `s` $<$ `n`, la case `degres[s]` contient le degré sortant d^+(`s`). Enfin, 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 dans un ordre quelconque.

2. 👆 ⌨ Vu la structure choisie, combien de sommets un graphe peut-il avoir au maximum ? Combien d'arcs ? Dans le code compagnon, modifier cette structure afin de lever ces restrictions.

Le code fournit des fonctions `creer_graphe` et `ajouter_arc` adaptées à cette nouvelle structure et permettant de construire des graphes. Par ailleurs, un sous-ensemble $S' \subset [\![0, n-1]\!]$ sera représenté par un tableau de taille n contenant `true` en case i si et seulement si i est dans l'ensemble S'. Les fonctions `afficher_graphe` et `afficher_ensemble` permettent d'afficher les objets correspondants.

3. ⌨ Écrire une fonction `void liberer_graphe(graphe* g)` permettant de libérer la mémoire allouée pour un graphe par la fonction `creer_graphe`. La tester sur `g_ex`.

4. ⌨ Écrire une fonction `int degre_max(graphe* g, bool* partie)`. Elle prend en entrée un graphe $G = (S, A)$ et un sous-ensemble $S' \subset S$ représenté par `partie` et renvoie $\max\{d^+(s) \mid s \in S'\}$.

5. ⌨ Écrire une fonction `bool* accessibles(graphe* g, int s)` qui prend en entrée un graphe G, un sommet s de G et qui renvoie le sous-ensemble de sommets $\mathcal{A}(s)$.

6. ⌨ 👆 En déduire une fonction `int degre_etoile_naif(graphe* g, int s)` qui calcule $d^*(s)$ pour le sommet s donné. Quelle est la complexité de cette fonction ? En déduire un algorithme pour résoudre le problème $\mathcal{P}$ et indiquer sa complexité.

Dans la suite, on cherche à obtenir un algorithme de meilleure complexité. Pour ce faire, on se restreint momentanément à la résolution de $\mathcal{P}$ dans le cas de graphes acycliques. On observe par ailleurs $(\star)$:

$$d^*(s) = \begin{cases} 0 \text{ si } s \text{ est de degré sortant nul} \\ \max(d^+(s), \max\{d^*(u) \mid u \text{ est voisin de } s\}) \text{ sinon} \end{cases}$$

7. ✏ Linéariser le graphe G_{ex}, c'est-à-dire représenter ses sommets sur une même ligne dans l'ordre donné par un tri topologique, tous les arcs allant de gauche à droite.

8. ✏ En exploitant la relation $(\star)$, concevoir un algorithme qui résout $\mathcal{P}$ en temps linéaire en la taille du graphe en entrée si on suppose qu'il est acyclique. Quelle est la stratégie algorithmique utilisée ?

9. 👆 Expliquer brièvement comment adapter l'algorithme précédent au cas où le graphe n'est plus supposé acyclique. Cela détériore-t-il sa complexité ?

10. ⌨ 👆 Implémenter l'algorithme décrit à la question 8. On pourra décomposer le travail en plusieurs fonctions. Solliciter l'examinateur ou examinatrice pour exposer votre plan ou demander de l'aide avant de commencer à coder.