Composition d'informatique n°4

Durée : 4 heures

L'utilisation de la calculatrice **n'est pas autorisée** pour cette épreuve.

⋆ ⋆ ⋆

Ensemble dominant

La notion d'ensemble dominant dans un graphe possède de nombreuses applications, allant de l'élection de représentants, à la répartition d'arrêts de bus, jusqu'à la résilience de réseaux de communication.

1 Introduction

Dans l'ensemble du sujet, on étudie des graphes non orientés, non pondérés et sans boucle, et sans précision contraire, ces caractéristiques seront supposées vérifiées sans qu'elle ne soit rappelées systématiquement.

Pour $G = (S, A)$ un graphe et $s \in S$, on appelle **voisinage de** s, noté $V(s)$, l'ensemble des voisins de s, c'est-à-dire l'ensemble des sommets $t \in S$ tels que $\{s, t\} \in A$.

On appelle **voisinage dominé par** s, noté $V[s]$, l'ensemble $V(s) \cup \{s\}$. Pour $X \subseteq S$ un ensemble de sommets, on appelle **voisinage dominé par** X, noté $V[X]$, l'ensemble $V[X] = \bigcup_{s \in X} V[x]$. Si $s \in X$, on dit aussi que s est **dominé par** X.

Une partie $X \subseteq S$ est appelée **ensemble dominant** G si $V[X] = S$, c'est-à-dire si tout sommet de S est soit dans X, soit possède un voisin dans X. La figure 1 donne un exemple d'un graphe G_0 et d'un ensemble dominant $X_0 = \{1, 3, 4\}$.

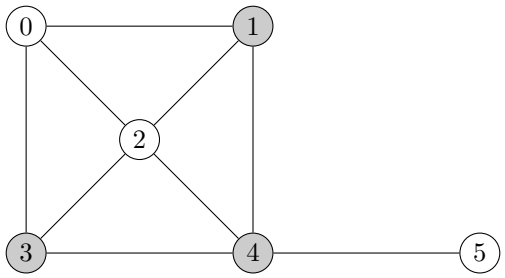

FIGURE 1 – Le graphe G_0 et un ensemble dominant X_0 (correspondant aux sommets grisés).

On appelle **dominance de** G, notée $\gamma(G)$, le cardinal minimal d'un ensemble dominant G.

Question 1 Déterminer, en justifiant, la dominance du graphe G_0 représenté figure 1.

Pour $G = (S, A)$ un graphe quelconque, on note $\delta(G)$ le degré minimal d'un sommet de G et $\Delta(G)$ le degré maximal d'un sommet de G.

Question 2 Montrer que pour $G = (S, A)$ un graphe quelconque, $\gamma(G) \geqslant \dfrac{|S|}{\Delta(G) + 1}$.

Question 3 Déterminer, en justifiant, la dominance des graphes suivants en fonction de n :

1. K_n, le graphe complet d'ordre n ;
2. C_n, le cycle de taille n.

Une partie $X \subseteq S$ est appelée **stable de** G si les sommets de X sont deux à deux non voisins. Un stable X est dit **maximal pour l'inclusion** s'il n'existe pas de stable Y tel que $X \subseteq Y$ et $|X| < |Y|$.

Question 4 Montrer qu'un stable maximal pour l'inclusion est un ensemble dominant.

Question 5 Donner, en justifiant, un exemple de graphe G tel qu'aucun ensemble dominant G de cardinal minimal n'est un stable.

On représente un graphe en OCaml comme un tableau de listes de sommets.

```
type graphe = int list array
```

de telle sorte que si `g` représente le graphe $G = (S = [\![0, n-1]\!], A)$, alors `g` est un tableau de taille n, et pour $s \in S$, `g.(s)` est la liste des voisins de s dans un ordre arbitraire.

Un ensemble X de sommets de S est représenté comme un tableau `x` de booléens de taille n tel que pour $s \in S$, `x.(s)` est égal à `true` si et seulement si $s \in X$.

Question 6 Écrire une fonction `cardinal (x: bool array) : int` qui calcule le cardinal d'un ensemble X.

Question 7 Écrire une fonction `domines (g: graphe) (x: bool array) : bool array` qui prend en argument un graphe $G = (S, A)$ et une partie $X \subseteq S$ et renvoie l'ensemble des sommets dominés par X.

Question 8 En déduire une fonction `est_dominant (g: graphe) (x: bool array) : bool` qui prend en argument un graphe $G = (S, A)$ et une partie $X \subseteq S$ et détermine si X est un ensemble dominant G. Cette fonction devra avoir une complexité en $\mathcal{O}(|S| + |A|)$ et on demande de justifier cette complexité.

On souhaite calculer un ensemble dominant $G = (S = [\![0, n-1]\!], A)$ de cardinal minimal. Pour ce faire, on propose une approche par Retour sur trace. On considère que pour une solution partielle $\widetilde{X}$, si X est une solution totale qui complète $\widetilde{X}$, il existe un indice $i \in [\![0, n]\!]$ tel que :

- pour $s < i$, $s \in X \Leftrightarrow s \in \widetilde{X}$;
- pour $s \geqslant i$, on n'a pas encore déterminé si $s \in X$.

Ainsi, l'arbre des possibilités est un arbre binaire complet de hauteur n, et les deux enfants de chaque nœud de profondeur i, correspondent aux cas où on ajoute ou non le sommet i à la solution partielle.

Question 9 Écrire une fonction `dominant_min (g: graphe) : bool array` qui prend en argument un graphe G et renvoie un ensemble dominant G de cardinal minimal selon un algorithme de type Retour sur trace.

Question 10 Déterminer la complexité de `dominant_min` pour un graphe $G = (S, A)$ en fonction de $|S|$ et $|A|$. Jusqu'à quel ordre de graphe (à 5 près) peut-on appliquer cet algorithme sur un ordinateur de bureau et avoir un résultat en moins de 10 minutes ?

Question 11 Proposer une heuristique d'évaluation et de branchement qui pourraient être utilisées pour adapter l'algorithme de Retour sur trace en un algorithme de type *Branch and Bound*. On justifiera que l'heuristique d'évaluation est admissible. On ne demande pas d'implémenter cette solution.

2 Ensemble dominant, arbres et forêts

Soit $G = (S, A)$ un graphe et $s \in S$. On note $\deg(s)$ le degré de s, c'est-à-dire son nombre de voisins. On dit que s est **une feuille** si $\deg(s) = 1$. On dit qu'il est **isolé** si $\deg(s) = 0$.

Question 12 Montrer si $G = (S, A)$ est connexe et $|S| > 1$, alors il existe une partie $X \subseteq S$ tel que X et $S \setminus X$ sont des ensembles dominant G.

Indication : on pourra considérer un arbre couvrant G.

Question 13 En déduire que si $G = (S, A)$ ne possède pas de sommet isolé, alors $\gamma(G) \leqslant \frac{|S|}{2}$.

Question 14 Soit $G = (S, A)$ un arbre tel que $|S| > 2$. Montrer qu'il existe un ensemble X dominant G de cardinal minimal tel que X ne contient aucune feuille de G.

Pour $G = (S, A)$ un graphe, $X \subseteq S$ et $s \in S$, on appelle **portée de s selon X**, notée $p_X(s)$, le nombre de sommets dominés par s et pas par X, c'est-à-dire le cardinal de $V[s] \setminus V[X]$.

On cherche à mettre en œuvre un algorithme pour calculer efficacement un ensemble dominant un arbre de cardinal minimal. Pour cela, on considère l'algorithme suivant :

Entrée : Arbre $G = (S, A)$.
Début algorithme
  Poser $X = \emptyset$.
  Tant que il reste des sommets non dominés par X **Faire**
    Poser s un sommet non dominé par X de portée minimale.
    Si $p_X(s) = 1$ **Alors**
      $X \leftarrow X \cup \{s\}$.
    Sinon
      Poser t un voisin de s non dominé par X.
      $X \leftarrow X \cup \{t\}$.
  Renvoyer X.

Question 15 L'algorithme précédent appliqué à un arbre G renvoie-t-il un ensemble dominant G ? Renvoie-t-il un ensemble dominant G de cardinal minimal ? Justifier.

On s'intéresse maintenant à une approche de programmation dynamique. Pour simplifier, on souhaite calculer uniquement $\gamma(G)$, et pas un ensemble dominant G de cardinal $\gamma(G)$.

Soit $r \in S$ un sommet arbitraire. On note T l'arbre correspondant à G enraciné en r. Pour $s \in S$, on note T_s le sous-arbre de T enraciné en s.

Pour $s \in S$, on note $\gamma(s)$, $\gamma^+(s)$ et $\gamma^?(s)$ le cardinal minimal d'un ensemble de sommets de T_s :

- dominant T_s pour $\gamma(s)$;
- dominant T_s et contenant s pour $\gamma^+(s)$;
- dominant T_s, sauf éventuellement s, pour $\gamma^?(s)$.

Question 16 Déterminer les valeurs de $\gamma(s)$, $\gamma^+(s)$ et $\gamma^?(s)$ si T_s est un arbre réduit à une feuille.

On suppose que $s \in S$ n'est pas une feuille de T et on note $s_1, \ldots, s_k$ les enfants de s dans T.

Question 17 Montrer les formules de récurrence suivantes :

1. $\gamma^+(s) = 1 + \sum_{i=1}^{k} \gamma^?(s_i)$;

2. $\gamma^?(s) = \min(\gamma^+(s), \sum_{i=1}^{k} \gamma(s_i))$;

3. $\gamma(s) = \min \left\{ \begin{array}{l} \gamma^+(s) \\ \min\limits_{i=1}^{k} \left(\gamma^+(s_i) + \sum\limits_{j=1, j\neq i}^{k} \gamma(s_i) \right) \end{array} \right.$

Question 18 Écrire une fonction `enracine (g: graphe) : graphe` qui prend en argument un graphe $G = (S, A)$ supposé être un arbre et renvoie un graphe **orienté** T correpondant à l'arbre G enraciné en 0.

Question 19 En déduire une fonction `gamma_arbre (g: graphe) : int` qui prend en argument un arbre G et calcule $\gamma(G)$ en temps polynomial selon le principe décrit précédemment. Des points supplémentaires seront accordés si la complexité est linéaire en la taille du graphe.

Cette question de programmation est longue et pourra être sautée en première passe.

3 Un problème difficile

Dans cette partie, on souhaite montrer que le problème du calcul de l'ensemble dominant de cardinal minimal est difficile dans le cas général.

Plus précisément, on s'intéresse au problème de décision `DOMINANT` :

* **Instance** : un graphe $G = (S, A)$ et un entier naturel k.
* **Question** : est-ce qu'il existe un ensemble dominant G de cardinal $\leqslant k$?

Pour $G = (S, A)$ et $X \subseteq S$, on dit que X est une **couverture des arêtes par les sommets** si et seulement si pour tout $a = \{s, t\} \in A$, alors $s \in X$ ou $t \in X$.

On admet que le problème `COUVERTURE PAR SOMMETS (CPS)` :

* **Instance** : un graphe $G = (S, A)$ et un entier naturel k.
* **Question** : est-ce qu'il existe une couverture des arêtes de G par les sommets de cardinal $\leqslant k$?

est NP-complet.

Pour prouver la NP-difficulté de `DOMINANT`, Sacha propose la preuve suivante :

> « On montre que `CPS` $\leqslant_{\mathsf{m}}^{\mathsf{p}}$ `DOMINANT` : à partir d'un graphe $G = (S, A)$, on construit le graphe $G' = (S', A)$, où $S' = \{s \in S \mid \deg(s) > 0\}$ est l'ensemble des sommets non isolés. Alors :
>
> – si X est une couverture de G, pour tout $s \in S'$, s possède un voisin t (car s n'est pas isolé), et l'arête $\{s, t\}$ est couverte par X, donc soit s, soit son voisin t est dans X. On en déduit que $X \cap S'$ est un ensemble dominant G' ;
>
> – si X est un ensemble dominant G', alors pour toute arête $a = \{s, t\} \in A$, comme X est dominant, soit $s \in X$, soit son voisin $t \in X$. On en déduit que l'arête a est couverte par X donc X est une couverture de G.
>
> Comme le graphe G' se construit en temps polynomial, on en déduit que `CPS` $\leqslant_{\mathsf{m}}^{\mathsf{p}}$ `DOMINANT`, donc `DOMINANT` est NP-difficile. »

Question 20 Expliquer quel est le problème dans le raisonnement de Sacha.

Pour $G = (S, A)$ un graphe, on pose $G' = (S', A')$ défini par :

– $S' = \{s \in S \mid \deg(s) > 0\} \cup \{s_a \mid a \in A\}$: on supprime les sommets isolés, et on ajoute un nouveau sommet par arête dans G ;

– $A' = A \cup \{\{s, s_a\} \mid s \in a\}$: on garde les arêtes déjà existantes dans G, et on relie chaque nouveau sommet associé à une arête à ses deux extrémités dans G.

Question 21 Représenter graphiquement le graphe G'_1 construit à partir du graphe G_1 donné en figure 2.

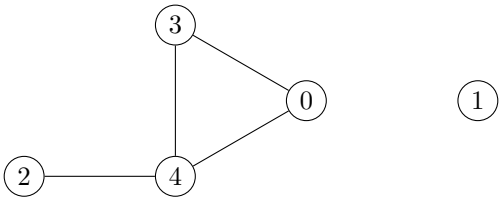

FIGURE 2 – Le graphe G_1.

Question 22 En utilisant la construction décrite précédemment, montrer que CPS $\leqslant_{\mathsf{m}}^{\mathsf{p}}$ DOMINANT.

Question 23 En déduire que DOMINANT est NP-complet.

4 Des algorithmes efficaces non optimaux

La section précédente montre qu'il est difficile d'espérer trouver des algorithmes efficaces permettant de trouver un ensemble dominant de cardinal minimal dans un cas général. Dans cette partie, on se propose d'étudier des alternatives permettant de trouver des ensembles dominants, mais pas nécessairement de cardinaux minimaux.

Pour $0 \leqslant p \leqslant 1$, on considère l'algorithme suivant :

Entrée : Graphe $G = (S, A)$.
Début algorithme
  Poser $X = \emptyset$.
  Pour $s \in S$ **Faire**
    Ajouter s à X avec probabilité p.
  Renvoyer X.

Pour les deux questions suivantes, on suppose que X est l'ensemble renvoyé par l'algorithme précédent sur un graphe $G = (S, A)$. On note $Y = S \setminus V[X]$.

Question 24 Montrer que $\mathbb{E}(|Y|) \leqslant |S| \times (1-p)^{\delta(G)+1}$.

Question 25 En déduire que $\gamma(G) \leqslant |S|(p + (1-p)^{\delta(G)+1})$.

Choisir $p = \frac{\ln(\delta(G)+1)}{\delta(G)+1}$ permet de montrer que dans le cas général, $\gamma(G) \leqslant \dfrac{|S|}{\delta(G)+1}(\ln(\delta(G)+1)+1)$.

Pour la suite, on souhaite réaliser des algorithmes déterministes efficaces, qui ne sont pas nécessairement optimaux.

On considère dans un premier temps l'algorithme glouton suivant :

Fonction `Glouton_naïf`$(G = (S, A))$
  Poser $X = \emptyset$.
  Tant que $V[X] \neq S$ **Faire**
    Poser s un sommet non dominé par X d'indice minimal.
    $X \leftarrow X \cup \{s\}$.
  Renvoyer X.

Il est clair que cet algorithme renvoie bien un ensemble dominant le graphe en entrée.

On représente en C un graphe par le type :

```c
struct Graphe {
    int n;
    int* deg;
    int** adj;
};

typedef struct Graphe graphe;
```

de telle sorte que si $G = (S, A)$ est représenté par l'objet `G` de type `graphe`, alors :

- `G.n` $= |S|$ et $S = [\![0, |S| - 1]\!]$;
- pour $s \in S$, `G.deg[s]` est le degré du sommet s ;
- pour $s \in S$, `G.adj[s]` est un tableau de taille $\deg(s)$ contenant les voisins de s.

On représente toujours une partie de S par un tableau de booléens de taille $|S|$.

Question 26 Écrire une fonction `bool* glouton_naif(graphe G)` qui renvoie un ensemble dominant un graphe $G = (S, A)$ selon l'algorithme glouton précédent. La fonction devra avoir une complexité en $\mathcal{O}(|S|+|A|)$, mais on ne demande pas de la justifier.

Question 27 Montrer par un exemple que pour tout $n > 2$, il existe un graphe G d'ordre n tel que l'algorithme précédent renvoie un ensemble dominant G de cardinal $(n-1)\gamma(G)$.

On propose une amélioration de l'algorithme précédent :

Fonction `Glouton_mieux`$(G = (S, A))$
  Poser $X = \emptyset$.
  Tant que $V[X] \neq S$ **Faire**
    Poser s un sommet d'indice minimal parmi ceux maximisant la portée $p_X(s)$.
    $X \leftarrow X \cup \{s\}$.
  Renvoyer X.

Pour rappel, la portée d'un sommet selon un ensemble X a été définie en partie 2.

Question 28 Appliquer l'algorithme précédent au graphe G_2 de la figure 3. On indiquera les sommets choisis à chaque itération, ainsi que leur portée. L'ensemble dominant ainsi obtenu est-il de cardinal minimal ? Justifier.

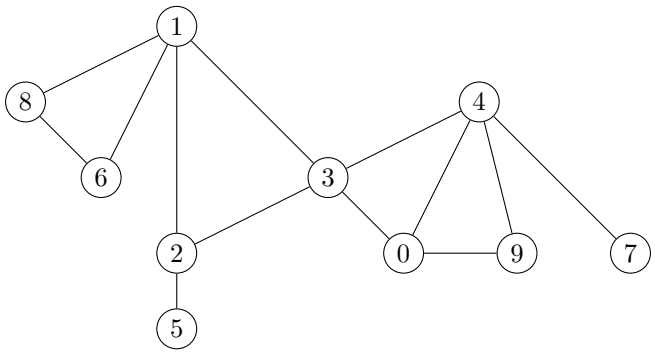

FIGURE 3 – Le graphe G_2.

Question 29 Écrire une fonction `int portee(graphe G, bool* VX, int s)` qui prend en argument un graphe $G = (S, A)$, une partie supposée égale à $V[X]$ (pour un certain $X \subseteq S$) et un sommet $s \in S$ et calcule et renvoie $p_X(S)$. Cette fonction devra avoir une complexité en $\mathcal{O}(\deg(s))$.

Question 30 En déduire une fonction `bool* glouton_mieux(graphe G)` qui renvoie un ensemble dominant un graphe $G = (S, A)$ selon l'algorithme glouton amélioré.

Question 31 Déterminer la complexité temporelle de la fonction `glouton_mieux`.

Dans les questions suivantes, on cherche à montrer que l'amélioration de l'algorithme glouton fournit une $(\ln \Delta(G) + 2)$-approximation du calcul de l'ensemble dominant de cardinal minimal pour un graphe G.

En considérant une exécution de l'algorithme sur un graphe $G = (S, A)$ qui renvoie l'ensemble dominant X, pour $s \in S$, on appelle **coût de** s, noté $c(s)$, la valeur $\frac{1}{p_X(t)}$, au moment où t est le sommet choisi à l'itération où s devient dominé par X, c'est-à-dire que t est le sommet de X qui domine s pour la première fois.

Question 32 Montrer qu'à la fin de l'algorithme, $\sum_{s \in S} c(s) = |X|$.

On note X^* un ensemble dominant G de cardinal $\gamma(G)$. On note également $s^* \in X^*$ et $\{s_1, \ldots, s_k\}$ les sommets de $V[s^*]$, numérotés dans l'ordre où ils ont été dominés lors de l'exécution de l'algorithme.

Question 33 Montrer que $c(s_1) \leqslant \dfrac{1}{\deg(s^*) + 1}$.

Question 34 Montrer que $\sum_{i=1}^{k} c(s_i) \leqslant H(\Delta(G) + 1)$, où $H(n) = \sum_{i=1}^{n} \frac{1}{i}$ est la somme partielle de la série harmonique.

Question 35 En déduire que l'algorithme glouton amélioré est bien une $(\ln \Delta(G) + 2)$-approximation du problème du calcul de l'ensemble dominant de cardinal minimal.

⋆ ⋆ ⋆