Université de Bordeaux — Collège Sciences et technologies masters

Année universitaire 2020/21, session 1 d'automne décembre 2020
Mention : master informatique, code UE 4TIN704U
Intitulé de l'épreuve : examen Calculabilité et complexité
Date : 14/12/2020 Heure : 9h00 Durée : 3h
Document autorisé : 1 feuille A4 recto-verso

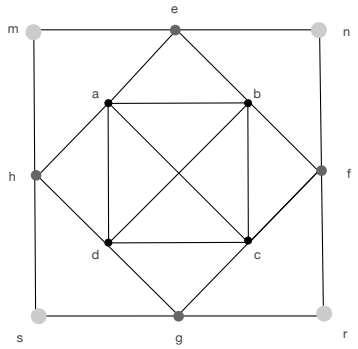
Le barème est indicatif. On attachera une grande importance à la clarté et à la concision des justifications.

Question 1 (16 points)

Dans un graphe non-orienté $G = (V, E)$, une *clique* $C \subseteq V$ est un ensemble de sommets qui sont reliés deux à deux : $uv \in E$ pour tous $u, v \in C$, $u \neq v$. La distance $d(v, C)$ d'un sommet v par rapport à $C \subseteq V$ est la longueur minimale d'un chemin entre C et v, donc le plus petit ℓ tel que il existe un chemin $v_0, v_1, \ldots, v_\ell = v$ avec $v_0 \in C$. On notera cette distance par $d(v, C)$. (Remarque : $v \in C$ si et seulement si $d(v, C) = 0$.)

Clique2 est le problème suivant :

Clique2
Entrée : Un graphe non-orienté $G = (V, E)$ et un entier k.
Sortie : Existe-t-il une clique $C \subseteq V$ de taille k telle que $d(v, C) \leq 2$ pour tout sommet v ?



Dans l'exemple à côté, les sommets a, b, c, d du centre forment une clique C de taille 4, et tous les sommets sont à distance au plus deux de C : les sommets e, f, g, h sont à distance 1, et m, n, r, s à distance 2.

1. (3 p.) Ecrivez un vérificateur polynomial du problème Clique2, en précisant bien quel est le certificat d'une instance positive quelconque et quelle est l'entrée du vérificateur.
   (a) Certificat : un sous-ensemble de sommets $C \subseteq V$
   (Rq : la réponse « clique » n'est pas correcte. C'est au vérificateur de vérifier *toutes* les propriétés de C.)
   (b) Le vérificateur prend en entrée G, k, C et vérifie que C est une clique, $|C| = k$ et que $d(v, C) \leq 2$ pour tout sommet v. On suppose que $V = \{1, \ldots, n\}$, C est décrit par un tableau $C[1 \ldots n]$ et $M[1 \ldots n]$ est un autre tableau pour marquer les noeuds visités :

```
for   v = 1 to n {
   if (C[v]==1) cpt++ ;
 };
if (cpt != k) return(0);
for u = 1 to n
   for v = 1 to n
   if (C[u]==1 & C[v]==1 & (u,v) notin E) return(0);
                                        // C est une clique
for v = 1 to n
if (C[v]== 1) {
       M[v] := 1;
        forall (w in Adj[v]) {
            M[w] := 1;
             forall (z in Adj[w]) M[z] := 1;
        }
 }
for v = 1 to n
    if (M[v]== 0) return(0);
return(1)
```

Quelle est la taille du certificat et quel est le temps de calcul du vérificateur (en fonction de la taille du graphe) ? $|C| \leq n$ et le temps de calcul est : $O((n+m)*n)$ (en supposant qu'on a des listes d'adjacence, et $m = |E|$).

2. (1 p.) Que déduisez-vous de la question précédente sur la complexité du problème Clique2 ? Clique2 appartient à la classe NP.
3. (2 p.) Ecrivez un algorithme qui calcule une solution pour le problème Clique2 (s'il en existe). Quelle est sa complexité (en fonction de la taille du graphe) ?
   On génère tous les sous-ensembles C de V de taille k, et on lance le vérificateur sur G, k, C. On accepte l'entrée G, k si un des appels retourne 1. La complexité est exponentielle : $2^k \cdot O(n^2 + nm) \leq 2^n \cdot O(n^2 + nm)$.
4. (1 p.) Quelle est la différence entre le vérificateur (question 1) et l'algorithme (question 3) pour le problème Clique2 ? Le vérificateur vérifie qu'un certificat représente bien une solution, tandis que l'algorithme cherche lui-même une solution.
5. (3 p.) Donnez une réduction polynomiale de Clique2 vers SAT. Vous utiliserez les variables $x_{v,i}$, pour signifier que le sommet v est le $i$ème sommet de la clique C.
   La formule doit être en CNF, et vous devez expliquer les sous-formules dont elle est composée. La formule $phi(G, k)$ est une conjonction des clauses suivantes :
   (a) Chaque position dans C est occupée par un et un seul sommet :
   $$\wedge_{i=1}^{k} \left((\vee_{v \in V} \; x_{v,i}) \wedge \wedge_{u \neq v} (\neg x_{u,i} \vee \neg x_{v,i}) \right)$$
   (b) Aucun sommet apparait à deux positions différentes :
   $$\wedge_{v \in V, i \neq j} (\neg x_{v,i} \vee \neg x_{v,j})$$
   (c) C est clique (il n'existe pas deux sommets non-reliés qui appartiennent tous les deux à C :
   $$\wedge_{i \neq j} \wedge_{u \neq v, (u,v) \notin E} (\neg x_{u,i} \vee \neg x_{v,j})$$

(d) Tout sommet est à distance ≤ 2 de C :

$$\wedge_{v \in V} \big(\vee_{i=1}^{k} \ (x_{v,i} \vee \vee_{w,(v,w)\in E} (x_{w,i} \vee \vee_{z,(w,z)\in E} x_{z,i})) \big)$$

6. (2 p.) Justifiez bien qu'il s'agit d'une réduction à la question 5, en montrant comment associer à chaque solution du problème Clique2 une solution pour SAT, et inversement.

   Pour chaque instance positive G, k de Clique2 il existe C avec les propriétés demandées. On ordonne les sommets de C : $v_1, \ldots, v_k$. La valuation qui met $x_{u,i}$ à vrai ssi $u = v_i$ satisfait toutes les clauses de la formule $\phi(G, k)$.

   Dans l'autre sens, si on a une valuation qui rend $\phi(G, k)$ vraie on trouve k sommets $v_1, \ldots, v_k$: v_i est l'unique (!) sommet t.q. $x_{v,i}$ est vraie (sous-formule (a)). Ces k sommets (sous-formule (b)) forment une clique (sous-formule (c)) et tout sommet est à distance ≤ 2 de C (sous-formule (d)).

7. (1 p.) Combien de clauses comporte votre formule de la question 5 ? Justifiez que la réduction est polynomiale.

   On a $O(k^2 n^2)$ clauses. La formule $\phi(G, k)$ se calcule en temps polynomial et G, k est instance positive de Clique2 ssi $\phi(G, k)$ est instance positive de SAT.

8. (1 p.) Que déduisez-vous de la réduction vers SAT pour la complexité du problème Clique2 ? Justifiez votre réponse avec les notions que vous avez apprises en cours ou en TD.

   Clique2 appartient à NP, car tout problème qui se réduit de facon polynomiale vers SAT appartient à NP.

9. (2 p.) Dans cette question vous allez montrer une réduction polynomiale du problème Clique (vu en cours) au problème Clique2.

   Soit $G = (V, E)$ un graphe non-orienté et k un entier. On veut savoir si cette instance est positive pour Clique, c'est-à-dire si G possède une clique de taille k.

   On définit le graphe $G' = (V', E')$ en rajoutant à G un sommet $v_0 \notin V$ qui est relié à tous les sommets de G :

$$V' = V \cup \{v_0\}, \qquad E' = E \cup \{v_0 v \mid v \in V\}$$

   (a) Montrez que (G, k) est instance positive de Clique si et seulement si $(G', k+1)$ est instance positive de Clique2.

   (G, k) est instance positive de Clique implique qu'il y a une clique C de taille k dans G. Alors $C \cup \{v_0\}$ est une clique de taille $k+1$ dans G', et tout sommet est à distance 0 ou 1 de v_0.

   Dans l'autre sens, si C' est une clique de taille $k+1$ dans G', alors soit C' est deja clique dans G, ou $v_0 \in C'$ et $C' \setminus \{v_0\}$ est une clique de taille k dans G.

   (b) Que déduisez-vous de cette réduction de Clique vers Clique2 pour la complexité du problème Clique2 ? Justifiez votre réponse avec les notions que vous avez apprises en cours ou en TD.

   Clique2 est NP-difficile, car Clique est NP-difficile : on sait que si $A \leq_P B$ et A est NP-difficile, alors B est NP-difficile aussi (par transitivité de $\leq_P$).

Question 2 (4 points)

Écrivez un programme **LOOP** qui prend en entrée un entier $n \in \mathbb{N}$ et retourne 1 si $n > 0$ et tous les chiffres dans la représentation décimale de n sont les mêmes, et 0 sinon. Par exemple, si $n = 22222$ ou $n = 555$, alors le programme retourne 1. Par contre, si $n = 0$, $n = 42$ ou $n = 1119$, le programme retourne 0.

Indication : Vous pouvez réutiliser directement les fonctions arithmétiques vues en TD (addition, multiplication, division, modulo,...) dans votre programme. Un appel de ADD(x, y) par exemple, calcule la somme $x + y$ et la met dans la variable *res*. On suppose qu'il n'y a pas d'effet de bord, en particulier, l'appel ne modifie pas les arguments. Il faut bien-sûr respecter la syntaxe des programmes **LOOP** (en particulier, il n'y a pas d'instruction *break*).

On va utiliser les instructions IF (x=0) THEN P ELSE Q, DIV, MOD et la différence tronquée DiffTr (qu'on ne redéfinit pas ici, voir cours/TD) :

```
res := 1;
cont := 0;      // cont = 1 joue le role du break pour loop
IF (n = 0) THEN res := 0; cont := 1 ELSE a = n MOD 10;
LOOP (n) DO
  IF (cont = 0) THEN
     n := n DIV 10;
     IF (n=0) THEN cont := 1
     ELSE b := DiffTr(n MOD 10,a) + DiffTr(a, n MOD 10);
     IF (b=0) THEN a := n MOD 10 ELSE res := 0; cont := 1;
  ELSE skip;
```