Faculté des Sciences
Aix*Marseille Université

Année universitaire 2020/2021

Site : ☒ Luminy ☐ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ☐ Aix-Montperrin ☐ Aubagne-SATIS
Sujet de : ☒ 1er semestre ☐ 2ème semestre ☐ Session 2 Durée de l'épreuve : 1h30
Examen de : M1 Nom du diplôme : Master Informatique
Code du module : SINAU03L Libellé du module : Complexité
Calculatrices autorisées : NON Documents autorisés : NON

Préambule : Ce sujet contient 13 questions, ce qui peut sembler considérable pour une épreuve d'une durée de 1h30. Cela étant, certaines questions peuvent facilement être traitées en moins de 3 minutes (le temps de les lire avec quelques secondes seulement pour y répondre). Par ailleurs, il a été constaté lors de la correction du partiel que certaines copies étaient parfois très difficiles à déchiffrer. Aussi, il vous est demandé de veillez à rendre votre copie aussi claire et lisible que possible.

1 Quelques questions de cours

Chaque question peut posséder 0 ou plusieurs réponses possibles. Il vous est juste demandé de préciser "*OUI*" ou "*NON*", ou encore "*Je ne sais pas*", avec les numéros des différents items, ceci sans donner de justification. Mais attention, comme dans certains QCM, la notation prendra en compte (négativement) les réponses erronées. Vous êtes donc invités à ne donner que les réponses dont vous êtes sûrs.

Question 1. Soient A et B deux langages tels que A se réduit à B et B se réduit à A avec deux réductions many-one polynomiales. Supposons que A est **NP**-complet. Que peut-on déduire ?

1. B est **NP**-complet
2. $B \in \mathbf{NP}$
3. $A \in \mathbf{P}$
4. $A \notin \mathbf{P}$

Solution 1.

1. Vrai : B est **NP**-difficile puisque le problème **NP**-complet A s'y réduit. De plus $B \leq A$ avec $A \in \mathbf{NP}$; la classe **NP** étant close par les réductions many-one polynomiales, cela implique $B \in \mathbf{NP}$. Donc B est **NP**-complet.
2. Vrai : comme discuté dans le point précédent.
3. Faux : on n'a pas assez d'informations pour le déduire (cela impliquerait $\mathbf{P} = \mathbf{NP}$).
4. Faux : on n'a pas assez d'informations pour le déduire (cela impliquerait $\mathbf{P} \neq \mathbf{NP}$).

Question 2. Lesquelles des affirmations suivantes impliquent $\mathbf{P} = \mathbf{NP}$?

1. Il existe une machine de Turing déterministe qui reconnaît les mots palindromes en temps $O(n)$
2. Il existe une machine de Turing déterministe qui reconnaît le langage **SAT** en temps $O(2^n)$
3. Il existe une machine de Turing non déterministe qui reconnaît le langage **SAT** en temps $O(n)$
4. Aucune machine de Turing déterministe ne reconnaît un langage **NP**-complet en temps polynomial

Solution 2.

1. Faux : on sait déjà que le langage des mots palindromes appartient à **P**, il faudrait montrer qu'il est **NP**-complet pour que cela implique $\mathbf{P} = \mathbf{NP}$.
2. Faux : il faut une machine déterministe qui fonctionne en temps *polynomial* pour que cela implique $\mathbf{P} = \mathbf{NP}$.
3. Faux : il s'agit bien d'une machine qui fonctionne en temps polynomial, mais étant non-déterministe, cela n'implique pas $\mathbf{P} = \mathbf{NP}$.
4. Faux : cela impliquerait $\mathbf{P} \neq \mathbf{NP}$ et non pas $\mathbf{P} = \mathbf{NP}$.

2 Appartenance à la classe NP

Nous considérons ici le problème dit du "Sac à dos". Dans ce problème, on dispose d'un sac à dos dont le poids rempli ne peut excéder une certaine valeur P, d'une collection d'objets E dont on connaît le poids de chacun, à savoir un poids $p(e)$ pour un objet $e \in E$, et son utilité $u(e)$ pour la randonnée à réaliser. On veut savoir s'il est possible de remplir le sac avec une partie $F \subseteq E$ des objets disponibles, sans dépasser le poids limite P mais en assurant une utilité globale (somme de l'utilité de tous les objets emportés) supérieure ou égale à U. Le problème de décision correspondant s'énonce de la façon suivante :

> **Sac à dos**
> **Donnée :** Un ensemble de n éléments $E = \{e_1, e_2, \dots e_n\}$, une fonction p définie de E vers $\mathbb{N}$, donc telle que $\forall e \in E$, $p(e) \in \mathbb{N}$, une fonction u définie de E vers $\mathbb{N}$, donc telle que $\forall e \in E$, $u(e) \in \mathbb{N}$, un entier P et un entier U.
> **Question :** Existe-t-il un sous-ensemble $F \subseteq E$ tel que $\Sigma_{e \in F} p(e) \leq P$ et tel que $\Sigma_{e \in F} u(e) \geq U$?

Notez que les fonctions p et u peuvent facilement être représentées par des tableaux d'entiers indicés de 1 à n (ou de 0 à $n-1$ si vous préférez).

Question. Montrez que le problème **Sac à dos** appartient à **NP**. Notez que pour simplifier votre travail, nous ne vous demandons pas de rentrer dans le détail du codage binaire des nombres entiers et on supposera que les opérations arithmétiques sont exécutables en temps constant. Il vous est demandé de fournir un pseudo-code suffisamment détaillé de sorte à ce que vous puissiez fournir une évaluation précise de sa complexité. Vous avez le choix de recourir à l'une des deux techniques classiques vues en cours et en TD pour montrer l'appartenance de ce problème à **NP**.

Solution 1 (Technique basée sur un algorithme de résolution non-déterministe). Voici un algorithme non-déterministe qui résout le problème en temps polynomial et qui utilise un tableau X de booléens repésentant les objets remplissant le sac à dos :

```
 1.    procédure sac-à-dos(p, u, P, U)
 2.        n = longueur(p)
 3.        X = nouveau tableau de longueur n
 4.        pour i := 1 à n faire
 5.            X[i] := deviner(0, 1)
 6.        fin
 7.        p-total := 0
 8.        u-totale := 0
 9.        pour i := 1 à n faire
10.            si X[i] alors
11.                p-total := p-total + p[i]
12.                u-totale := u-totale + u[i]
13.            fin
14.        fin
15.        si p-total <= P et u-totale => U alors
16.            accepter
17.        sinon
18.            rejeter
19.        fin
20.    fin
```

Dans les lignes 3–6 on devine un tableau de booléens qui correspond au sous-ensemble X d'objets qu'on met dans le sac à dos. Puis (lignes 7–14) on calcule le poids total et l'utilité totale des objets choisis. Enfin (lignes 15-19) on vérifie que les contraintes du problème sont satisfaite : le poids des objets doit être $\leq P$ et l'utilité $\geq U$; si c'est bien le cas, on accepte, et sinon on rejette.

Les deux boucles de cet algorithme font n itérations et contiennent un nombre constant d'opérations, et le code en dehors des boucles effectue également un nombre constant d'opérations. Cela donne un temps de calcul total de $\Theta(n)$, qui est linéaire par rapport à la taille de l'entrée. Donc cela montre que le problème appartient à la classe **NP**

Solution 2 (Technique basée sur la mise en évidence d'un certificat polynomial). Un certificat polynomial pour le problème du sac à dos est l'ensemble des objets que l'on prend et qui vérifie les conditions sur le poids et sur l'utilité : ils s'agit du tableau X qu'on devine dans l'algorithme précédent, qui a une taille linéaire par rapport à l'entrée. Il faut maintenant s'assurer de l'existence d'un vérificateur déterministe polynomial pour ce certificat :

```
 1.    procédure vérificateur-sac-à-dos(p, u, P, U, X)
 2.        n = longueur(p)
 3.        p-total := 0
 4.        u-totale := 0
 5.        pour i := 1 à n faire
 6.            si X[i] alors
 7.                p-total := p-total + p[i]
 8.                u-totale := u-totale + u[i]
 9.            fin
10.        fin
11.        si p-total <= P et u-totale => U alors
12.            accepter
13.        sinon
14.            rejeter
15.        fin
16.    fin
```

Cela correspond à la portion déterministe de l'algorithme non-déterministe `sac-à-dos` : ici on n'a pas besoin de deviner l'ensemble X puisque il est fourni en entrée. L'analyse de complexité du vérificateur est la même que pour l'algorithme non-déterministe (seulement, il y a une boucle en moins), donc le temps de calcul sera $\Theta(n)$, voire linéaire. Comme ce certificat existe si et seulement si l'instance en entrée est positive, que sa taille est polynomiale, et qu'il peut être vérifié en temps polynomial sur modèle de calcul déterministe, il s'agit donc bien d'un certificat polynomial.

3 Réduction

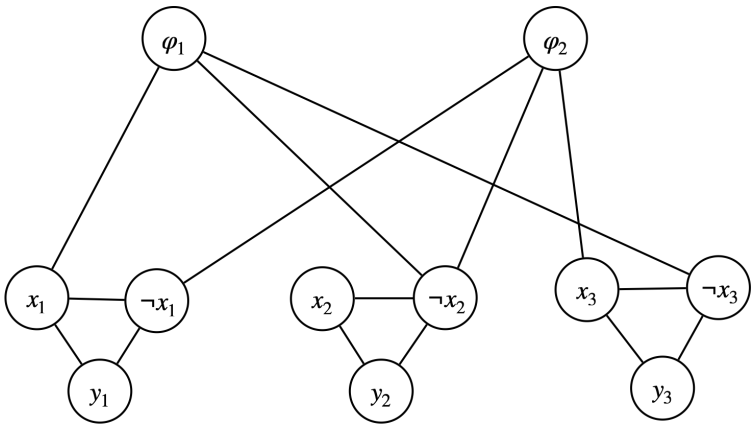
Nous rappelons les deux problèmes suivants :

> **3-SAT**
> **Donnée :** Une formule φ en forme normale conjonctive avec exactement trois littéraux par clause.
> **Question :** Existe-t-il une affectation des variables qui satisfait φ ?
>
> **Ensemble Dominant**
> **Donnée :** Un graphe non orienté $G = (S, A)$ et un entier naturel k.
> **Question :** Existe-t-il un sous-ensemble $D \subseteq S$ de k sommets tel que, pour chaque sommet $s \in S$, soit $s \in D$, soit il existe une arête $\{s, t\} \in A$ telle que $t \in D$ (c'est-à-dire, le sommet s est soit dans l'ensemble D, soit relié par une arête à un sommet qui est dans l'ensemble D) ?

On veut réduire le problème **3-SAT** à **Ensemble Dominant** avec une réduction *many-one* en temps polynomial. Nous donnons ci-dessous un exemple de transformation pour la formule φ définie avec $n = 3$ variables (x_1, x_2 et x_3) et $m = 2$ clauses (φ_1 et φ_2) telles que :

$$\varphi = \underbrace{(x_1 \vee \neg x_2 \vee \neg x_3)}_{\varphi_1} \wedge \underbrace{(\neg x_1 \vee \neg x_2 \vee x_3)}_{\varphi_2}$$

Cette formule φ se transforme ainsi en une instance (G, k) du problème **Ensemble Dominant**, où $k = 3$ et $G = (S, A)$ est le graphe suivant :



Question 1. Trouvez une affectation des variables x_1, x_2, x_3 de φ qui satisfait la formule et indiquez un ensemble de sommets de G qui représente cette affectation en montrant en quoi cet ensemble de sommet constitue un ensemble dominant de G de taille 3.

Solution 1. Par exemple, l'affectation $x_1 = 1, x_2 = 0, x_3 = 0$ satisfait la formule. Si on choisit l'ensemble de sommets $D = \{x_1, \neg x_2, \neg x_3\}$, c'est-à-dire l'ensemble correspondant aux littéraux de la formule que l'affectation rend vrais, il s'agit bien d'un ensemble dominant de taille 3. En effet, les sommets en dehors de D sont tous reliés à un sommet de D : $\neg x_1$ et y_1 sont reliés à $x_1 \in D$, les sommets x_2 et y_2 à $\neg x_2 \in D$, les sommets x_3 et y_3 à $\neg x_3 \in D$; en plus, φ_1 est relié à x_1 (ainsi qu'à $\neg x_2$ et $\neg x_3$) et φ_2 à $\neg x_2$.

Question 2. Trouvez un ensemble dominant de taille 3 dans G, différent de celui de la question précédente, et une affectation des variables x_1, x_2, x_3 de φ qui représente cet ensemble dominant.

Solution 2. Si on choisit l'ensemble $D = \{\neg x_1, x_2, \neg x_3\}$, il s'agit d'un ensemble dominant de taille 3, puisque les sommets en dehors de D $(x_1, y_1, \neg x_2, y_2, x_3, y_3, \varphi_1, \varphi_2)$ sont tous reliés à un sommet de D. Si on affecte la valeur vrai aux littéraux correspondants aux sommets dans D, on obtient l'affectation $x_1 = 0, x_2 = 1, x_3 = 0$, qui satisfait la formule.

Question 3. Il est possible de trouver des ensembles dominants de taille 3 dans G qui contiennent le sommet y_2. Donnez un tel ensemble et justifiez qu'il s'agit bien d'un ensemble dominant. À quoi correspond cet ensemble en termes d'affectation des variables de la formule φ ?

Solution 3. Un tel ensemble dominant est, par exemple, $D = \{x_1, y_2, x_3\}$. Cela représente le fait que les affectations avec $x_1 = 1$ et $x_3 = 1$ satisfont la formule, indépendemment de la valeur de x_2.

Question 4. Tout ensemble dominant de G doit contenir au moins 3 sommets. Indiquez parmi quels sommets de G il faut sélectionner ces 3 sommets et précisez pourquoi.

Solution 4. Soit D un ensemble dominant de G. Alors, pour chacun des triangles formés par des sommets du type $x_i, \neg x_i, y_i$ il est nécessaire qu'au moins l'un des trois sommets appartienne à D. Si ce n'était pas le cas, alors le sommet y_i ne serait ni dans D, ni adjacent à un sommet dans D, en contradiction avec le fait que D est dominant. Comme il y a 3 triangles de ce type (un pour chaque variable de la formule), l'ensemble D doit donc contenir au moins 3 sommets, chacun pris de l'un des triangles.

Question 5. Généralisez la transformation de la formule précédente φ en un graphe G à des formules **3-SAT** quelconques définies sur n variables et par m clauses, en décrivant (de forme précise, mais sans nécessairement écrire du pseudo-code) comment construire le graphe. Expliquez pourquoi cette transformation peut être calculée en temps polynomial.

Solution 5. Soit $\varphi = \varphi_1 \wedge \varphi_2 \cdots \varphi_m$ une formule de m clauses en forme normale conjonctive, avec trois littéraux par clause, définie sur les variables $x_1, x_2, \ldots, x_n$. Le graphe correspondant aura comme sommets

$$S = \{x_1, \neg x_1, y_1, x_2, \neg x_2, y_2 \ldots, x_n, \neg x_n, y_n, \varphi_1, \varphi_2, \ldots, \varphi_m\}$$

c'est-à-dire, trois sommets pour chaque variable x_i (la variable elle-même x_i, sa négation $\neg x_i$ et le sommet auxiliaire y_i), et un sommet pour chaque clause φ_j de la formule. On aura donc un total de $3n+m$ sommets.

Les arêtes de G relieront, pour chaque $i = 1, 2, \ldots, n$, les sommets $x_i, \neg x_i, y_i$ pour former les triangles de la question 4. En plus, il y aura une arête entre un sommet de la forme φ_j et un sommet de la forme x_i (ou $\neg x_i$) si la clause φ_j contient le littéral x_i (ou $\neg x_i$). On aura donc un total de $3n + 3m$ arêtes.

On peut construire la matrice d'adjacence de G en temps polynomial puisque elle a taille polynomiale $(3n+m) \times (3n+m)$ et qu'on peut calculer si chaque case contient 0 ou 1 en regardant les indices pour établir s'il s'agit de coordonnées correspondantes à l'une des arêtes des triangles $x_i, \neg x_i, y_i$ (dans ce cas la case contiendra 1), ou d'une arête entre deux sommets correspondants à une clause et un littéral (dans ce cas, la case contiendra 1 si le littéral appartient à la clause, ce qu'on peut vérifier en analysant la formule φ en entrée en temps polynomial).

Question 6. La transformation donnée dans la question précédente transforme une formule **3-SAT** quelconque en un graphe G. Il s'agit maintenant de montrer comment il est possible de transformer une formule **3-SAT** quelconque, donc une instance de **3-SAT**, en une instance de **Ensemble Dominant**, c'est-à-dire en un couple de la forme (G, k). Il faut donc préciser comment calculer la valeur de k (en justifiant que ce calcul est réalisable en temps polynomial).

Solution 6. La valeur de k est tout simplement le nombre de variables n, qu'on peut calculer en temps polynomial en analysant la description de la formule d'entrée.

Question 7. Expliquez pourquoi si une formule est satisfaisable, alors elle est transformée en un graphe qui admet un ensemble dominant de taille k.

Solution 7. Si la formule est satisfaite par une certaine affectation des variables, soit D l'ensemble que contient les sommets correspondants aux littéraux que cette affectation rend vrais (donc, pour chaque variable x_i, on aura $x_i \in D$ si $x_i = 1$ et $\neg x_i \in D$ si $x_i = 0$). Cet ensemble a taille $k = n$. Montrons qu'il s'agit d'un ensemble dominant.

Si $x_i = 1$, alors $x_i \in D$ et les sommets $\neg x_i$ et y_i sont donc adjacents sont adjacents à un sommet dans D. Si $x_i = 0$, alors $\neg x_i \in D$ et les sommets x_i et y_i sont donc adjacents sont adjacents à un sommet dans D. Il reste donc à vérifier que chaque sommet φ_j est adjacent à un sommet dans D. Si la formule est satisfaite par l'affectation, alors en particulier chaque clause a au moins un littéral que l'affectation rend vrai et qui est relié à φ_j dans le graphe G. Comme D contient exactement les littéraux vrais, en effet les sommets du type φ_j sont adjacents à des sommets de D, qui s'avère donc être un ensemble dominant.

Question 8. Expliquez pourquoi si une formule est transformée en un graphe qui admet un ensemble dominant de taille k, alors elle est satisfaisable.

Solution 8. Si le graphe admet un ensemble dominant D de taille k, qui est aussi le nombre n de variables de la formule, alors il contient, pour chaque $i = 1, 2, \ldots, n$, exactement un des sommets parmi $x_i, \neg x_i, y_i$ (question 4). Les sommets φ_j ne font donc pas partie de D ; ils doivent, par conséquent, être adjacents à au moins un sommet du type x_i ou $\neg x_i$ qui appartienne à D. Si on donne la valeur vrai aux variables $x_i \in D$ et faux aux variables x_i telles que $\neg x_i \in D$ (et n'importe quelle valeur aux variables x_i telles que $y_i \in D$), on aura donc au moins un littéral vrai par clause, et donc la formule sera satisfaite par cette affectation.

Question 9. À partir des réponses aux questions précédentes, que peut-on conclure concernant la transformation proposée ? Justifiez votre réponse.

Solution 9. La transformation proposée est calculable en temps polynomial et la formule d'entrée φ est satisfaisable (donc une instance positive de **3-SAT**) si et seulement si le graphe correspondant admet un ensemble dominant de taille $k = n$ (donc une instance positive d'**Ensemble Dominant**). Il s'agit donc d'une réduction many-one polynomiale du problème **3-SAT** au problème **Ensemble Dominant**.

Question 10. En sachant que **3-SAT** est **NP**-complet, et que **Ensemble Dominant** appartient à **NP**, comment peut-on classer le problème **Ensemble Dominant** en utilisant la transformation proposée ? Justifiez votre réponse.

Solution 10. La réduction depuis **3-SAT** montre qu'**Ensemble Dominant** est un problème **NP**-difficile. Étant, en plus, un problème dans **NP**, il s'agit donc d'un problème **NP**-complet.