

Complexité

Examen - 3 heures

Les questions indiquées par une étoile () signalent des questions sans doute plus difficiles que les autres. Tout document et autre polycopié est interdit.*

Le sujet comporte 7 pages et est composé de 3 parties dans l'ensemble indépendantes. Il est ainsi possible de traiter chacune des parties séparément.

Il est possible d'admettre les résultats d'une question pour résoudre la suivante, il faut pour cela l'indiquer clairement dans la copie. Il n'est pas nécessaire de résoudre toutes les questions pour avoir la note maximale. La rédaction sera prise en compte dans la notation.

Introduction

On donne ici une liste de problèmes **NP**-complets. Il est admis que tous ces problèmes sont **NP**-complets, et on pourra donc utiliser ce fait dans la suite sans avoir à le prouver.

NOM : SAT
ENTREE : Une formule ϕ
QUESTION : Est-ce que ϕ est satisfaisable ?

NOM : 3SAT
ENTREE : Une formule ϕ avec exactement 3 littéraux par clause
QUESTION : Est-ce que ϕ est satisfaisable ?

NOM : PARTITION
ENTREE : Des entiers a_i
QUESTION : Est-ce qu'on peut partitionner les a_i en deux parties de même somme ?

NOM : 3COLOR
ENTREE : Un graphe G
QUESTION : Est-ce que G est 3-coloriable ?

NOM : CLIQUE
ENTREE : Un graphe G et un entier k
QUESTION : Est-ce que G a une clique de taille k
Une clique est un ensemble de sommets tous reliés deux à deux.

NOM : DIOPHANT
ENTREE : Des entiers a, b, c positifs
QUESTION : Est-ce qu'il existe des entiers positifs x et y tels que $ax^2 + by = c$?

NOM : HAMILTON
ENTREE : Un graphe G
QUESTION : Est-ce que G contient un cycle hamiltonien ?

Un cycle hamiltonien est un cycle qui passe par tous les sommets du graphe, et une et une seule fois par chacun des sommets du graphe.

1 Classes de complexité

Toutes les questions de cette partie admettent des réponses très courtes. En conséquence, toute réponse contenant plus de 100 mots sera considérée comme fausse.

Q 1) Rappeler la définition de \leq_m et de **NP**-complet.

Q 2) Montrer que si $L \leq_m 3SAT$ alors L est dans **NP**.

Réponse 2) Variantes :

- $3SAT$ est dans **NP**. L est plus facile qu'un problème **NP** donc est dans **NP**.
- $3SAT$ est dans **NP**. **NP** est stable par \leq_m donc $L \in \mathbf{NP}$.

On peut également refaire la démonstration du cours.

Q 3) Montrer que si $3SAT \leq_m L$ alors L est **NP**-dur.

Réponse 3) Variantes :

- L est plus dur que $3SAT$ qui est **NP**-dur donc L est **NP**-dur
- Soit $X \in \mathbf{NP}$. $X \leq 3SAT$ puisque $3SAT$ est **NP**-dur. Or $SAT \leq L$ donc $X \leq L$. On a donc montré que pour tout $X \in \mathbf{NP}$, $X \leq L$ donc L est **NP**-dur.

Q 4) Montrer que si $3SAT$ est **PSPACE**-dur, alors **NP** = **PSPACE**.

Réponse 4) Soit un problème L dans **PSPACE**. Comme $3SAT$ est par hypothèse **PSPACE**-dur, on a donc $L \leq_m 3SAT$. Mais alors par une question précédente, on a donc $L \in \mathbf{NP}$. On a montré que pour tout $L \in \mathbf{PSPACE}$, on a $L \in \mathbf{NP}$ de sorte que **PSPACE** \subset **NP**. L'autre inclusion étant déjà connue, on arrive à **PSPACE** = **NP**.

Q 5) Montrer que si $3SAT$ est **coNP**-dur, alors **NP** = **coNP**.

Réponse 5) En procédant de la même façon, on arrive à **coNP** \subset **NP**. donc **cocoNP** \subset **coNP** c'est à dire **NP** \subset **coNP**, donc **NP** = **coNP**

Q 6) Montrer que si les nombres sont donnés en unaire et non en binaire dans la définition de **PARTITION**, le problème se résout en temps polynomial.

Réponse 6) Soit $(a_i)_{1 \leq i \leq n}$ une instance de **PARTITION**.

Soit $K = \sum a_i$. Considérons le tableau T de taille $n \times K$ tel que $T[j, M] = 1$ si et seulement si on arrive à faire la somme M en utilisant certains des a_i pour $i \leq j$. On peut remplir le tableau par l'algorithme suivant :

- pour tout j , faire $T[j, 0] = 1$
- Pour j allant de 1 à n :
- Pour M allant de 1 à K : Si $T[j - 1, M] = 1$ OU $T[j - 1, M - a_j] = 1$
 - alors $T[j, M] = 1$
 - sinon $T[j, M] = 0$

(En effet, on peut faire la somme M en utilisant les j premiers objets si on arrive à faire la somme sans utiliser le j ème objet, ou si on arrive à faire $M - a_j$ en utilisant les objets de 1 à $j - 1$)

Cet algorithme est polynomial puisque T est de taille polynomial (c'est ici qu'on utilise l'hypothèse). Il suffit ensuite de savoir si $T[n, K/2] = 1$ pour répondre à **PARTITION**.

(Note : Cette méthode a été présentée en cours)

Le problème 4SAT est le suivant :

NOM : 4SAT
ENTREE : Une formule ϕ avec exactement 4 littéraux par clause
QUESTION : Est-ce que ϕ est satisfaisable?

Q 7) Montrer que 4SAT est **NP**-complet.

Le problème 4COLOR est le suivant :

NOM : 4COLOR
ENTREE : Un graphe G
QUESTION : Est-ce que G est 4-coloriable?

Q 8) Montrer que 4COLOR est **NP**-complet.

Réponse 8) 4COLOR est clairement dans **NP**, il suffit de deviner la couleur de chaque sommet et de vérifier que c'est bien une coloration du graphe.

Considérons la fonction ϕ qui transforme un graphe G en un graphe $\phi(G)$ en ajoutant un sommet à G et en reliant tous les sommets de G à ce nouveau sommet. ϕ se calcule clairement en espace logarithmique.

Il est clair que G est 3 coloriable si et seulement si $\phi(G)$ est 4-coloriable.

Donc $3COLOR \leq_m 4COLOR$. Comme 3COLOR est **NP**-dur, on en déduit que 4COLOR est **NP**-dur, et donc finalement que 4COLOR est **NP**-complet.

Le problème SUBGRAPH est le suivant :

NOM : SUBGRAPH
ENTREE : Un graphe G et un graphe H
QUESTION : Est-ce que G contient H comme sous-graphe?

H est un sous-graphe de G signifie que lorsqu'on restreint G à un certain sous-ensemble X de sommets, on reconnaît le graphe H .

Q 9) Montrer que SUBGRAPH est **NP**-complet.

Le problème UNIQUECLIQUE est le suivant :

NOM : UNIQUECLIQUE
ENTREE : Un graphe G , un entier k et une clique X de taille k
QUESTION : Est-ce que X est l'unique clique de G de taille k ?

Q 10) Montrer que UNIQUECLIQUE est **coNP**-complet.

Réponse 10) Il suffit de montrer que son complémentaire est **NP**-complet. Son complémentaire est le problème PLUSIEURSCLIQUES suivant :

NOM : PLSCLIQUE
ENTREE : Un graphe G , un entier k et une clique X de taille k
QUESTION : Est-ce que G a une autre clique que X de taille k ?

PLSCLIQUE est clairement dans **NP**, il suffit de deviner l'autre clique et de vérifier que c'est bien une clique.

On considère la fonction ϕ qui à (G, k) associe (G', k, X) où G' est l'union d'une nouvelle clique X de taille k et de G .

On voit que G a une clique de taille k si et seulement si G' a une autre clique de taille k que X .

On a donc $(G, k) \in CLIQUE \iff (G', k, X) \in PLSCLIQUE$, de sorte que $CLIQUE \leq_m PLSCLIQUE$. Comme $CLIQUE$ est **NP**-dur, $PLSCLIQUE$ est **NP**-dur, et donc **NP**-complet puisqu'il est dans **NP**.

Le problème MATH est le suivant :

NOM : MATH
 ENTREE : Des entiers $a_1 \dots a_n$
 QUESTION : Est-ce que $\int_0^{2\pi} \prod_i \cos(a_i t) dt \neq 0$?

Des gens qui font des maths ont remarqué que

$$\prod \cos(a_i t) = \frac{1}{2^n} \sum_I \cos\left(\sum_{i \in I} a_i t - \sum_{i \notin I} a_i t\right)$$

où I parcourt toutes les sous-ensembles de $[1 \dots n]$.

Et on rappelle que $\int_0^{2\pi} \cos(at) dt$ vaut 1 si $a = 0$ et 0 sinon.

Q 11) Montrer que MATH est **NP**-complet.

Réponse 11) MATH est exactement le problème PARTITION.

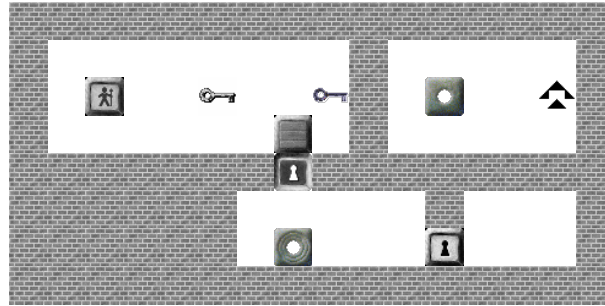
Yves a téléchargé, légalement, la première saison de sa série TV favorite. Chaque épisode fait entre 340 000 000 et 360 000 000 octets, et il y a 24 épisodes. La capacité totale prise actuellement sur le disque dur de Yves est actuellement exactement de 8 400 000 000 octets. Yves veut faire tenir tous ses épisodes sur deux DVD, chaque DVD ayant une capacité de 4 200 000 000 octets.

Q 12) Expliquez à Yves pourquoi son problème n'est a priori pas si facile, même si Yves a réussi à le résoudre en 2 secondes.

Réponse 12) Il s'agit de résoudre une instance de PARTITION. Comme PARTITION est **NP**-complet on pense que c'est n'est pas facile en général. Cependant, comme on est en présence d'un seul cas, fini, il est possible qu'une heuristique marche dans ce cas alors qu'elle ne marche pas dans le cas général.

2 Aventure

On s'intéresse ici à un jeu d'aventures spécialement conçu pour l'examen.
Voici un exemple typique :



Et sa représentation sous forme de tableau.

#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
#									#						#
#		@			k_2			k_1	#		A_1			FIN	#
#							B		#						#
#	#	#	#	#	#	#	d_1	#	#	#	#	#	#	#	#
#	#	#	#	#	#						#				#
#	#	#	#	#	#		T_1				d_2				#
#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#

Le but est d'amener le héros (représenté par @) vers l'arrivée (représenté par FIN). Le héros peut seulement se déplacer dans les quatres directions, et peut interagir avec les éléments du décor :

- Il ne peut pas traverser les murs (#) ;
- Il peut pousser les blocs (B), mais ne peut pousser qu'un bloc à la fois ;
- Il peut ramasser des clés k_i ;
- Il peut passer par une porte d_i s'il possède la clé k_i correspondante ;
- Il peut prendre un téléporteur T_i ce qui l'amène en A_i (le téléporteur ne marche que dans un sens : S'il est en A_i , il ne peut pas retourner en T_i).

Q 13) Résoudre le puzzle de l'exemple. On indiquera les directions à prendre.

Réponse 13) Facile.

Q 14) Montrer que savoir si un puzzle a une solution est dans **PSPACE**.

Réponse 14) On part d'une configuration du jeu de taille $n \times m$, et on veut savoir si le héros peut atteindre l'arrivée. Une configuration du jeu est en espace polynômial. Considérons l'algorithme suivant : A partir de la configuration d'origine, on choisit une direction pour le héros parmi les directions possibles, et on bouge le héros dans cette direction. On s'arrête si on est sur l'arrivée. Cet algorithme est un algorithme non déterministe en espace polynômial (puisque'il suffit de garder en mémoire uniquement la configuration) qui résout le problème. Le problème est donc dans **NPSPACE**. Comme **NPSPACE** = **PSPACE**, savoir si le puzzle a une solution est bien dans **PSPACE**.

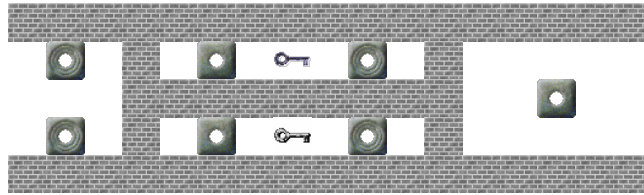
Q 15) Montrer que savoir si un puzzle a une solution est **NP-dur**. On réduira à partir de 3-SAT. Aide : Il y a au moins deux façons de faire, mais rien ne vous empêche d'en trouver une autre :

- On représente les variables x_i et \bar{x}_i par des clés, et les clauses correspondent à des portes spéciales qu'on ne peut passer que si on possède une des trois clés qui marchent sur la porte. Expliquer alors comment coder cette porte spéciale. Expliquer aussi comment s'arranger pour que si le héros ait pris la clé x_i , il n'ait pas pu prendre la clé \bar{x}_i .
- On représente chaque clause par une seule porte, et il faut s'arranger pour que choisir la variable x_i plutôt que sa négation corresponde à ramasser toutes les clés des clauses correspondant à cette variable. Là encore, expliquer comment s'arranger pour que le héros ne puisse pas choisir à la fois la variable x_i et sa négation.

Réponse 15) Suivons l'énoncé.

Soit ϕ une instance de 3SAT.

A chaque variable x de la formule, on construit le gadget suivant :



Et sa représentation sous forme de tableau.

#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
	T_1		#		A_1		k_x		T_3		#						
			#	#	#	#	#	#	#	#	#			A_3			
	T_2		#		A_2		$k_{\bar{x}}$		T_3		#						
#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#

On voit dans ce gadget que pour passer de la gauche à la droite, le héros doit prendre une des clés k_x ou $k_{\bar{x}}$ mais ne peut pas prendre les deux.

On associe maintenant à chaque clause $l_1 \vee l_2 \vee l_3$ le gadget suivant :



Et sa représentation sous forme de tableau.

#	#	#	
	d_{l_1}		
	d_{l_2}		
	d_{l_3}		
#	#	#	

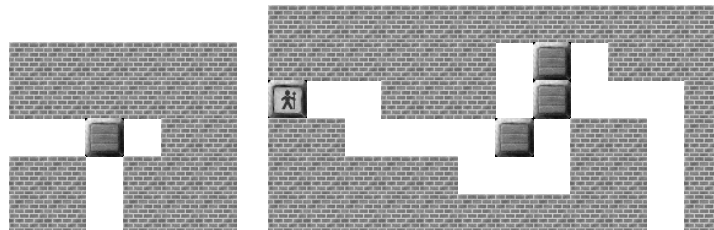
On remarque que dans ce gadget, le héros doit posséder une des clés correspondant à l_1 , l_2 ou l_3 pour pouvoir passer.

On considère maintenant le puzzle où on met le héros au début, le gadget correspondant à chacune des variables au début, les gadgets correspondant à chacune des clauses ensuite, puis l'arrivée. Pour que le héros arrive à la fin, il doit prendre une clé k_x ou $k_{\bar{x}}$, autrement dit choisir

pour chaque variable si elle est vraie ou fautive, puis il doit disposer, pour chaque clause, d'une clé correspondant à un littéral, sans quoi il ne peut pas passer. On en déduit donc que le héros arrive à l'arrivée si et seulement si il a choisi les bonnes valeurs pour chaque variable de sorte qu'il puisse passer chaque clause, donc que le héros peut atteindre l'arrivée si et seulement si la formule ϕ est satisfaisable.

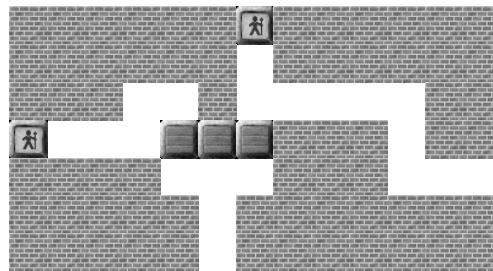
On a donc bien fait une réduction de *3SAT* à *PUZZLE*. Cette réduction est bien en espace logarithmique. Donc *PUZZLE* est **NP**-dur.

Q 16) On suppose maintenant que les téléporteurs marchent dans les deux directions. Après examen des gadget suivant, expliquer pourquoi le problème est toujours **NP**-dur.



Réponse 16) Grâce à ces gadgets, on peut transformer un téléporteur bidirectionnel en téléporteur unidirectionnel. En effet, le premier gadget est fait de telle façon qu'on ne peut passer dans le gadget que si on vient de la gauche (et non pas du bas). Le deuxième gadget est fait de telle façon que si on vient de la gauche, et qu'on veut passer à droite, on doit bouger les blocs de sorte qu'on ne puisse plus revenir (Y réfléchir, mais c'est facile à voir). En combinant ces gadgets avec les téléporteurs, on peut transformer les téléporteurs en téléporteurs qu'on ne peut utiliser qu'une fois, et que dans un sens, ce qui suffit pour faire la réduction de la question précédente.

Q 17) En examinant le gadget suivant, expliquer pourquoi le problème est toujours **NP**-dur si on enlève les clés et les portes (Il reste donc juste le téléporteur, les blocs, et l'arrivée).



*******Q 18)******* Montrer que le problème est toujours **NP**-dur si on a uniquement les blocs et l'arrivée.

(Il faut réussir à fabriquer un gadget qui permet de "croiser" des chemins. Bon courage)

3 Autoréductibilité

Les deux parties de ce problème sont indépendantes, mais utilisent toutes deux l'auto réductibilité de *SAT*, ce qu'on exprime dans la question suivante.

Soit ϕ une formule à n variables. Une instanciation partielle des variables est un ensemble de la forme $Y = \{x_1 = y_1, x_2 = y_2, \dots, x_k = y_k\}$. k est appelé la taille de Y , notée $|Y|$.

On note $\phi[Y]$ la formule $\phi[Y](x_{k+1} \dots x_n) = \phi(y_1 \dots y_k, x_{k+1} \dots x_n)$. Par exemple, si $\phi(x_1, x_2, x_3) = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_1)$ et $Y = \{x_1 = 1\}$, alors $\phi[Y](x_2, x_3) = (1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee 0)$

Q 19) Soit Y de taille k . Justifier rapidement que $\phi[Y]$ est satisfaisable si et seulement si $\phi[Y \cup \{x_{k+1} = 1\}]$ ou $\phi[Y \cup \{x_{k+1} = 0\}]$ est satisfaisable.

Q 20) Expliquer pourquoi le programme suivant résout SAT :

```
def sat(Y):
    k = |Y|
    Si k = n:
        calculer phi[Y].
        Si phi[Y] est vrai, répondre oui sinon répondre non.
    Sinon:
        calculer J = sat(Y union (x_{k+1}=0))
        Si J = oui, répondre oui.
        calculer K = sat(Y union (x_{k+1}=1)).
        Si K = oui, répondre oui.
        Sinon, répondre non.
```

3.1 P-sélectivité

A est dit p -selectif s'il existe une fonction en 2 variables f calculable en temps polynomial telle que

- Pour tout x, y , $f(x, y) = x$ ou $f(x, y) = y$.
- Si $x \in A$ ou $y \in A$, alors $f(x, y) \in A$.

Q 21) Montrer que les langages dans **P** sont p -selectifs.

Réponse 21) Commençons par préciser ce qu'est un langage p -sélectif. Il s'agit d'un langage L pour lequel on a un procédé (la fonction f) qui est capable de nous trouver, parmi deux mots dont l'un est dans L celui des deux qui est dans L .

Soit A un langage dans **P**. Considérons la fonction f qui sur l'entrée x, y , regarde si $x \in A$. Si c'est le cas, elle répond x , sinon elle répond y . On vérifie que la fonction f permet de montrer que A est p -sélectif.

Q 22) Montrer que si A est p -selectif, alors son complémentaire \bar{A} l'est aussi.

Réponse 22) Considérons la fonction g qui sur l'entrée x, y vaut x si $f(x, y) = y$ et vaut y si $f(x, y) = x$. La fonction g montre que \bar{A} est p -sélectif.

Q 23) Montrer que si A est p -sélectif, $A \times A \leq_m A$.

Réponse 23) Soit f qui témoigne que A est p -sélectif. Considérons la fonction g qui sur l'entrée x, y vaut x si $f(x, y) = y$ et vaut y si $f(x, y) = x$. On voit que $(x, y) \in A \times A \iff g(x, y) \in A$. De sorte que $A \times A \leq_m A$. (Note : techniquement le résultat est faux puisque g n'est pas a priori calculable en espace logarithmique (on sait juste qu'il est calculable en temps polynomial).

Si λ est un réel entre $1/2$ et 1 , on note L_λ l'ensemble des couples (p, q) d'entiers codés en binaires tels que $p/q < \lambda$.

Q 24) Montrer que L_λ est p -sélectif. (Attention : Dans la définition de p -sélectif, x doit donc être vu comme une paire $x = (p, q)$ et y de même comme une paire $y = (p', q')$)

Réponse 24) Soit f la fonction qui sur l'entrée $(p, q), (p', q')$ répond (p, q) si $p/q \leq p'/q'$ et répond (p', q') sinon. Cette fonction témoigne que L_λ est p -sélectif. (On remarque que f ne dépend pas de λ)

Si λ est un réel entre 0 et $1/2$, on note $B_\lambda = [0, \lambda] \cup [1/2, 1 - \lambda[$ et A_λ l'ensemble des couples (p, q) d'entiers tels que $p/q \in B_\lambda$.

Q 25) Montrer que A_λ est l'union de deux p -sélectifs.

Q 26) Montrer que si A_λ est p -sélectif, alors A_λ est dans **P**. (On remarquera que pour $x \in [0, 1] \setminus \{1/2\}$, $x \in B_\lambda \iff (1 - x) \notin B_\lambda$).

Réponse 26) Supposons que A_λ est p -sélectif et soit f la fonction qui en témoigne. Alors $(p, q) \in A_\lambda \iff f((p, q), (q - p, p)) = (p, q)$, ce qui nous donne un algorithme polynomial pour décider l'appartenance à A_λ .

(Dit autrement : On demande à f de choisir parmi x et $1 - x$. Si elle choisit x , c'est que $x \in B_\lambda$. En effet, si x n'y est pas, alors $1 - x$ y est, d'après la remarque. Réciproquement, si $x \in B_\lambda$, la fonction doit choisir x puisque $1 - x$ dans ce cas n'est pas dans B_λ)

Q 27) En déduire qu'il existe deux langages p -sélectifs dont l'union n'est pas p -sélectif.

Réponse 27) Il existe un langage A_λ qui n'est pas dans **P**. En effet, il existe un nombre dénombrable de langages dans **P**, et il y a un nombre non dénombrable de A_λ (si $\lambda \neq \beta$ on a bien $A_\lambda \neq A_\beta$). Un tel langage A_λ n'est pas p -sélectif d'après la question précédente, mais s'écrit comme l'union de deux p -sélectifs d'après une autre question.

Q 28) Montrer que si $3SAT$ est p -sélectif, alors **P** = **NP**.

Q 29) Montrer que si B est p -sélectif et $A \leq_m B$, alors A est p -sélectif.

Q 30) Déduire des deux questions précédentes que s'il existe un langage p -sélectif **NP**-complet alors **P** = **NP**.

3.2 Langages creux

On suppose connaître une fonction f telle que si $f(Y) = f(Y')$ alors $\phi[Y]$ est satisfaisable si et seulement si $\phi[Y']$ est satisfaisable.

On considère le programme suivant. T est un ensemble global à l'algorithme.

```
def sat(Y):
  Si f(Y) est dans la liste T, répondre non.
  k = |Y|
  Si k = n:
    calculer phi[Y].
    Si phi[Y] est vrai, répondre oui.
    Sinon, ajouter f(Y) à la liste T et répondre non.
```

Sinon:

```
calculer J = sat(Y union (x_{k+1}=0))
Si J = oui, répondre oui.
calculer K = sat(Y union (x_{k+1}=1)).
Si K = oui, répondre oui.
Sinon, ajouter f(Y) à la liste T et répondre non.
```

Lorsqu'on commence l'algorithme, la liste T est vide.

Q 31) Montrer que $\text{sat}(\emptyset)$ est vraie si et seulement si ϕ est satisfaisable. On vérifiera qu'à chaque étape de l'algorithme, si $f(Y)$ est dans T , alors $\phi[Y]$ n'est pas satisfaisable.

On note $\Delta(Y)$ la variation de taille de T pendant l'exécution de l'algorithme sur l'entrée Y

Q 32) Montrer par récurrence sur $m = n - |Y|$ que le nombre d'appels récursifs lors du calcul de $\text{sat}(Y)$ est inférieur à $2m(\Delta(Y) + 1)$ si $\text{sat}(Y)$ est vraie, et inférieur à $2m(\Delta(Y))$ sinon.

Q 33) En déduire que si f est à valeur dans un ensemble de cardinal K , alors la complexité de l'algorithme est polynomiale en K et n .

Un langage L est dit unaire si les seuls mots acceptés sont de la forme a^i , c'est à dire sont des mots utilisant une seule lettre.

Q 34) En utilisant les questions précédentes, montrer que s'il existe un langage unaire **NP**-complet, alors **P = NP**.

Un langage L est dit creux s'il existe un polynôme p tel que le nombre de mots de L acceptés de longueur n est inférieur à $p(n)$.

Q 35) En utilisant les questions précédentes, montrer que s'il existe un langage creux **coNP**-complet, alors **P = NP**.