TD N°3

Exercice 1 :

1) Prouver que NP ⊆ PSPACE
2) Décrire un certificat pour le problème de décision suivant pour montrer qu'il appartient à NP. Dites si vous pensez qu'il appartient également à P.

   DIV 3
   Entrée: un ensemble fini $A \subset Z+$.
   Question: existe-t-il un sous-ensemble $S \subseteq A$ tel que $s \in S$ soit divisible par trois?

Solution :

Preuve: Supposons que L ∈ NP alors il existe un polynôme, p (n), et une DTM M tel que

T_M (n) ≤ p (n) et sur toute entrée x ∈ Σ_0^* :

(i) si x ∈ L alors il existe un certificat y ∈ Σ_0^* tel que | y | ≤ p (| x |) et M accepte la chaîne d'entrée x y;

(ii) si x ≠ L alors pour toute chaîne y ∈ Σ_0^*, M rejette la chaîne d'entrée x y.

Nous formons une nouvelle DTM N qui à l'entrée x produit chaque chaîne possible y ∈ Σ_0^* de longueur au plus p (| x |) à son tour et imite le calcul de M sur la chaîne x y. Puisque M s'arrête toujours après au plus p (| x |) étapes, à chaque fois nous simulons le calcul de M sur x y, au plus 2p (| x |) + 1 carrés de bande sont nécessaires et ces carrés peuvent être réutilisés pour chaque y possible. Nous avons également besoin de quelques carrés de bande pour stocker x et le certificat actuel possible y à chaque étape afin que nous puissions redémarrer l'étape suivante du calcul avec la chaîne x z où z est le prochain certificat possible après y. Donc, au total, N utilisera l'espace O (p (| x |) + | x |). Si x ∈ L alors lorsque nous atteignons un bon certificat y tel que M accepterait x y, nous nous arrêtons N dans l'état γ_T. Si x ∉ L alors à aucun moment M n'accepterait x y et ainsi, après avoir essayé chaque certificat possible tour à tour, nous arrêtons N dans l'état γ_F. La DTM N décide clairement L dans l'espace polynomial. D'où L ∈ PSPACE et donc NP ⊆ PSPACE.

Exercice 2 :

Prouvez que QBF, défini ci-dessous, appartient à PSPACE.

QBF

Entrée: une formule booléenne quantifiée $F = (Q_1x_1)\ (Q_2x_2) \cdot \cdot \cdot (Q_{nxn})\ B\ (x_1,..., x_n)$,

où $B\ (x_1,..., x_n)$ est une expression booléenne dans les variables $x_1, . . . , x_n$ et chaque Q_i est un quantificateur ∀ ou ∃.

Question: F est-il vrai?

Exercice 3 :

1) Montrer que Si $A \leq_m B$ et $B \in P$ alors $A \in P$?
2) Montrer que si A et B sont des langages, $B \in NP$ et $A \leq_m B$ alors $A \in NP$?
3) Montrer que si A, B et C sont des langages, $A \leq_m B$ et $B \leq_m C$ alors $A \leq_m C$?

Solution :

1) Preuve : Si $A \leq_m B$ et $B \in P$ alors il existe deux DTM, M et N, avec les propriétés suivantes :

(i) M calcule une fonction f: $\Sigma_0^* \rightarrow \Sigma_0^*$ vérifiant $x \in A \Leftrightarrow f(x) \in B$;

(ii) il existe un polynôme p (n) tel que $T_M(n) \leq p(n)$;

(iii) N décide le langage B;

(iv) il existe un polynôme q (n) tel que $T_N(n) \leq q(n)$.

Nous construisons maintenant une machine du temps polynomial DTM qui décidera A. Étant donné une entrée $x \in \Sigma_0^n$ nous passant x comme entrée à M, pour obtenir f (x). On passe alors f (x) à N et on accepte ou on rejette selon que N accepte ou rejette f (x). Puisque M calcule une réduction de A à B et que N décide du langage B, notre nouvelle DTM décide certainement A. Pour voir qu'il s'exécute en temps polynomial, nous notons que le temps nécessaire pour calculer f (x) par M est au plus p(n). De plus $T_M(n) \leq p(n)$ implique que $|f(x)| \leq p(n) + n$. Ceci est dû au fait que la machine M commence avec une chaîne de longueur n sur sa bande et s'arrête après au plus p(n) étapes**,** donc lorsqu'elle s'arrête, elle ne peut pas avoir plus de p (n) + n carrés de bande non vierges.

Ainsi le temps pris par N sur l'entrée f (x) est au plus q (p (n) + n). Par conséquent, le temps de fonctionnement total de notre nouvelle machine est au plus p (n) + q (p (n) + n), qui est toujours polynomial en n, la taille d'entrée. D'où $A \in P$.

Exercice 4 :

1) Monter que SAT est NP-complet ?
2) Montrer que 3-SAT est NP-complet ?

Solution :

1) Avant de donner la preuve, nous prouvons le résultat suivant.

Théorème : Il existe des langages NP-complets.

Preuve: Le langage suivant est NP-complet.

BOUNDED HALTING (BH) (ARRÊT BORNÉ)

Entrée: p_M x 1^t, où p_M est une description d'une DTM M; 1^t est une chaîne de t unités

et x ∈ Σ_0^*.

Question: Existe-t-il un certificat y ∈ Σ_0^* tel que M accepte x y dans le temps borné par t?

BH appartient à NP car un certificat est simplement y ∈ Σ_0^* tel que la DTM M accepte x y en au plus t étapes.

Nous souhaitons maintenant montrer que tout langage L ∈ NP est polynomialement réductible en BH. Soit L ∈ NP et M un DTM pour L donné par la définition de NP, avec le polynôme correspondant p(n). Considérons maintenant la fonction f (x) = p_M x $1^{p(|x|)}$

Alors f ∈ FP, puisque p_M est indépendant de x (il ne dépend que du langage L); x peut être copié en temps linéaire et la chaîne $1^{p(|x|)}$ peut être écrite en temps O (p (| x |)).

De plus, x ∈ L si et seulement s'il existe un certificat y ∈ Σ_0^* tel que x y est accepté par M au temps p (| x |). Mais par définition de BH cela est vrai si et seulement si p_M x $1^{p(|x|)}$ ∈ BH. Donc x ∈ L ⇐⇒ f (x) ∈ BH. D'où L $\leq_m$ BH et donc BH est NP-complet.

Il s'agit d'un résultat théorique intéressant mais peu utile dans la pratique lorsque l'on essaie de trouver d'autres exemples de langages NP-complets. Pour ce faire, nous devons donner un exemple plus naturel d'un langage NP-complet. Pour cette raison, nous donnons maintenant une preuve du théorème de Cook.

Preuve du théorème (1) : Notons tout d'abord que SAT ∈ NP: un certificat succinct est une assignation de vérité satisfiable. Nous devons montrer que pour tout langage L ∈ NP nous avons L $\leq_m$ SAT Soit L ∈ NP alors, par définition de NP, il existe un DTM M et un polynôme p (n) tels que T(n) ≤ p (n) et sur toute entrée x ∈ Σ_0^* :-*

(i) si x ∈ L alors il existe y ∈ Σ_0^* tel que | y | ≤ p (| x |) et M accepte la chaîne d'entrée x y;

(ii) si x ∉ L alors pour toute chaîne y ∈ Σ_0^*, M rejette la chaîne d'entrée x y.

Notre réduction polynomiale de L à SAT prendra n'importe quelle entrée possible x ∈ Σ_0^* et construira une instance de SAT, disons S_x, telle que S_x est satisfiable si et seulement si x ∈ L. En utilisant la définition de NP, cela revient à dire que S_x est satisfiable si et seulement s'il existe un certificat y ∈ Σ_0^*, avec | y | ≤ p (| x |), tel que M accepte l'entrée x y.

Soit l'alphabet = $\{\sigma_0, \ldots, \sigma_l\}$ et l'ensemble des états soit Γ=$\{\gamma_0, \ldots, \gamma_m\}$. Nous supposerons que le symbole vide ∗ est σ_0, l'état initial est γ_0 et l'état d'acceptation est γ_1. Nous notons que si M accepte x y dans le temps p(n), pour un certain y ∈ Σ_0^*, alors les seuls carrés de bande qui ne peuvent jamais être balayés sont ceux à une distance au plus p(n) du carré de départ. En étiquetant les carrés de bande avec les nombres entiers de manière évidente, avec le carré de départ étiqueté par zéro, nous notons que seul le contenu des carrés de bande −p (n), . . ., p (n) peut jouer un rôle dans le calcul de M.

Pour x ∈ Σ_0^* , nous construisons S_x à partir de sept collections de clauses impliquant les variables suivantes :

$$\mathrm{sq}_{i,j,t}, \quad \mathrm{sc}_{i,t} \quad \text{and} \quad \mathrm{st}_{k,t}.$$

Nous considérons ces variables comme ayant les significations suivantes (lorsqu'elles sont vraies):

- $\mathrm{sq}_{i,j,t}$ – 'at time t square i contains symbol σ_j',
- $\mathrm{sc}_{i,t}$ – 'at time t the read-write head is scanning square i',
- $\mathrm{st}_{k,t}$ – 'at time t the machine is in state γ_k'.

Afin de construire les groupes de clauses, nous souhaitons assurer qu'exactement l'une d'une collection de variables, disons $z_1, \ldots, z_s$, est vrai. Nous utilisons la notation suivante pour montrer comment cela peut être réalisé dans CNF

$$\mathrm{Unique}(z_1, \ldots, z_s) = \left(\bigvee_{i=1}^{s} z_i\right) \wedge \left(\bigwedge_{1 \le i < j \le s} (\overline{z}_i \vee \overline{z}_j)\right)$$

Il est facile de voir que Unique (z_1,..., z_s) est vrai si et seulement si exactement une des variables z_1,. . . , z_s est vrai.

Les différents ensembles de clauses dans S_x garantissent que les différents aspects du calcul de M sont corrects.

(i) La tête de lecture-écriture ne peut pas être à deux endroits en même temps :
A tout moment t exactement un carré de bande est balayé par la tête de lecture-écriture :

$$\mathcal{C}_1 = \bigwedge_{t=0}^{p(n)} \mathrm{Unique}\left(\mathrm{sc}_{-p(n),t}, \ldots, \mathrm{sc}_{p(n),t}\right).$$

(ii) Chaque carré contient un symbole.

À tout moment t, chaque carré de ruban contient exactement un symbole :

$$\mathcal{C}_2 = \bigwedge_{i=-p(n)}^{p(n)} \bigwedge_{t=0}^{p(n)} \mathrm{Unique}(\mathrm{sq}_{i,0,t}, \ldots, \mathrm{sq}_{i,l,t}).$$

(iii) La machine est toujours dans un seul état.

A tout instant t la machine M est dans un seul état:

$$\mathcal{C}_3 = \bigwedge_{t=0}^{p(n)} \mathrm{Unique}(\mathrm{st}_{0,t}, \ldots, \mathrm{st}_{m,t}).$$

(iv) Le calcul démarre correctement.

Au temps t = 0 les carrés −p (n), . . ., −1 sont vides, les carrés 0, 1, . . ., n contient la chaîne $x = \sigma_{j0}\,\sigma_{j1} \cdots \sigma_{j_{n-1}}\,\sigma_{j_n}$ et les carrés n + 1 jusqu'à p (n) peut contenir n'importe quoi (puisque n'importe quelle chaîne dans ces carrés pourrait être un certificat possible $y \in \Sigma_0^*$). De plus, la position de départ de la tête de lecture-écriture est au carré zéro et l'état initial est γ_0:

$$\mathcal{C}_4 = \mathrm{sc}_{0,0} \wedge \mathrm{st}_{0,0} \wedge \bigwedge_{i=0}^{n} \mathrm{sq}_{i,j_i,0} \wedge \bigwedge_{i=-p(n)}^{-1} \mathrm{sq}_{i,0,0}.$$

(v) Le calcul se termine par l'acceptation.

A un moment t ≤ p (n) , M entre dans l'état d'acceptation γ_1:

$$\mathcal{C}_5 = \bigvee_{t=0}^{p(n)} \mathrm{st}_{1,t}.$$

(vi) Seul le symbole du carré courant peut changer.

Seul le symbole du carré courant au temps t peut être modifié au temps t + 1:

$$\mathcal{C}_6 = \bigwedge_{i=-p(n)}^{p(n)} \bigwedge_{j=0}^{l} \bigwedge_{t=0}^{p(n)} (\mathrm{sc}_{i,t} \vee \mathrm{sq}_{i,j,t} \vee \overline{\mathrm{sq}}_{i,j,t+1}) \wedge (\mathrm{sc}_{i,t} \vee \overline{\mathrm{sq}}_{i,j,t} \vee \mathrm{sq}_{i,j,t+1}).$$

(vii) La fonction de transition détermine le calcul.

Si à l'instant t la machine est dans l'état γ_k, la tête de lecture-écriture scanne le carré i, et ce carré contient le symbole σ_j alors

$$\delta(\gamma_k, \sigma_j) = (\gamma_p, \sigma_q, b)$$

décrit le nouvel état, le nouveau symbole à écrire dans le carré i et si la tête de lecture-écriture se déplace vers la gauche ou vers la droite. (Nous avons b = −1 s'il se déplace vers la gauche et b = 1 s'il se déplace vers la droite.)

$$\mathcal{C}_7 = \bigwedge_{i=-p(n)}^{p(n)} \bigwedge_{j=0}^{l} \bigwedge_{t=0}^{p(n)} \bigwedge_{k=0}^{m} (\overline{\mathrm{st}}_{k,t} \vee \overline{\mathrm{sc}}_{i,t} \vee \overline{\mathrm{sq}}_{i,j,t}) \vee (\mathrm{st}_{p,t+1} \wedge \mathrm{sq}_{i,q,t+1} \wedge \mathrm{sc}_{i+b,t+1}).$$

(Notez que pour simplifier nous n'avons pas écrit C_7 en CNF, il serait trivial de corriger cela.)

Il est facile de voir que si nous définissons S_x comme étant la formule booléenne donnée par la conjonction de toutes les collections de clauses ci-dessus, nous avons une instance de SAT. De plus, il n'est pas trop difficile de vérifier que la taille de S_x est polynomiale dans la taille d'entrée. (Vous pouvez vérifier que le nombre de variables dans S_x est O (p (n) ^{2}) et le nombre total de clauses est O (p (n) ^{3}).)

De plus S_x est clairement satisfiable si et seulement si M accepte x y, pour un certificat $y \in \Sigma_0^*$, en temps au plus p (n). (Étant donné une affectation satisfiable pour S_x, nous pouvons en fait lire un bon certificat : il sera décrit par les variables correspondant au contenu des carrés de bande n + 1 jusqu'à p (n) au temps t = 0.)

2) Preuve: Clairement 3-SAT ∈ NP puisqu'un certificat succinct est une assignation de vérité satisfiable. Ainsi, la preuve sera complète si nous montrons que SAT $\leq_m$ 3-SAT Nous le montrons en utilisant une méthode dite de remplacement local: nous prenons une instance f de SAT et la changeons localement pour donner une instance g (f) de 3-SAT tel que g (f) soit satisfiable si et seulement si f est satisfiable.

   Étant donné une instance de SAT

$$f(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} C_i,$$

nous laissons les clauses avec au plus trois littéraux inchangés. Considérons maintenant une clause Ci = ($z_1 \vee z_2 \vee \cdots \vee z_k$) avec au moins quatre littéraux, donc k ≥ 4. Introduire k - 3 nouvelles variables $y_1, \ldots, y_{k-3}$ et remplacer C_i par la conjonction de k - 2 nouvelles clauses contenant chacune trois littéraux

$$D_i = (z_1 \vee z_2 \vee y_1) \wedge (z_3 \vee \overline{y}_1 \vee y_2) \wedge (z_4 \vee \overline{y}_2 \vee y_3) \\ \wedge \cdots \wedge (z_{k-2} \vee \overline{y}_{k-4} \vee y_{k-3}) \wedge (z_{k-1} \vee z_k \vee \overline{y}_{k-3}).$$

Nous affirmons que:

(i) la restriction de toute assignation de vérité satisfaisable pour D_i à $z_1, z_2, \ldots, z_k$ est une affectation de vérité satisfaisable pour C_i.

(ii) Toute assignation de vérité satisfaisable C_i peut être étendue à une assignation de vérité satisfaisable pour D_i.

Si nous pouvons prouver ces deux affirmations alors nous aurons montré qu'il existe une fonction g: $\Sigma_0^* \rightarrow \Sigma_0^*$ satisfaisant f ∈ SAT si et seulement si g (f) ∈ 3-SAT. Les clauses de g (f) sont simplement les clauses de f qui contiennent moins de quatre littéraux avec les k - 2 clauses définies par D_i ci-dessus pour chaque clause C_i de f contenant plus de trois littéraux. Le fait que g appartienne à FP découle du fait qu'une clause avec k ≥ 4 littéraux est remplacée par une collection de k - 2 clauses contenant chacune 3 littéraux, d'où | g (f) | = O (| f |), qui est certainement polynomial dans | f |. Ainsi SAT $\leq_m$ 3-SAT et donc, 3-SAT est NP complet.

Nous devons maintenant prouver les deux affirmations. La première partie est facile. Prenez une assignation de vérité satisfiable pour D_i. Si (i) ne tient pas, alors chaque z_j doit être faux, mais alors $y_j = 1$ pour j = 1, . . ., k - 3 et donc la dernière clause de D_i n'est pas satisfiable. Cette contradiction prouve (i).

Pour voir que (ii) est également vrai, supposons que nous ayons une assignation de vérité satisfiable pour C_i, donc au moins un des z_j est vrai. Si $z_1 = 1$ ou $z_2 = 1$, définir chaque y_j égal

à 0 satisfait D_i. De même, si $z_{k-1} = 1$ ou $z_k = 1$, fixer chaque y_j égal à 1 satisfait D_i. On peut donc supposer que $k \geq 5$ et

$$l = \min\{j \mid z_j = 1\}$$

satisfait $3 \leq l \leq k - 2$. Fixer $y_j = 1$ pour $1 \leq j \leq l - 2$ et $y_j = 0$ pour $l - 1 \leq j \leq k - 3$ satisfait D_i. Par conséquent (ii) est également valable.

Exercice 5 :

1) Monter que CLIQUE est NP-complet ?
2) Monter que MAX CLIQUE est NP-dure ?
3) Soit #SAT la fonction, mappant les formules booléennes dans CNF à Z^+ défini par

   $$\#SAT(f) = |\{a \in \{0,1\}^n \mid f(a) = 1\}|.$$

   Montrez que #SAT est NP-dure ?
4) Montrer que Si L est NP-complet et L appartient à co-NP alors NP = co-NP ?
5) Monter que PRIMIER appartient à NP ∩ co-NP ?

Solution :

1) Preuve : Nous allons montrer que SAT $\leq_m$ CLIQUE. Étant donné une instance de SAT,

   $$f(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} C_i$$

   On construit un graphe G_f avec la propriété que G_f a une clique d'ordre m si et seulement si f est satisfiable.
   Les sommets de G f sont

   $$V(G_f) = \{(a, i) \mid a \text{ is a literal in clause } C_i\}.$$

   Les arêtes sont

   $$E(G_f) = \{\{(a, i), (b, j)\} \mid i \neq j \text{ and } a \neq \overline{b}\}.$$

   Le nombre de sommets dans $V(G_f)$ est simplement le nombre de littéraux dans f comptés en fonction du nombre de clauses dans lesquelles ils apparaissent, donc c'est O(| f |). Le nombre d'arêtes est alors au plus $O(|V|^2) = O(|f|^2)$. Il s'agit donc d'une construction polynomiale en temps. Il reste à montrer que cela donne une réduction de SAT à CLIQUE.
   Supposons que f est une instance satisfiable de SAT. Prenez une affectation de vérité satisfiable pour f et pour chaque clause, C_i , choisissez un littéral $a_i \in C_i$ tel que a_i soit vrai.
   Les sommets correspondants de $V(G_f)$ forment une clique d'ordre m dans G_f , puisque si l'on prend deux tels sommets (a_i , i) et (a_j , j) alors $i \neq j$ et a_i , a_j sont tous les deux vrais donc $a_i \neq a_j$.
   Inversement, supposons que G_f ait une clique d'ordre m, alors les sommets de la clique sont (a_1, 1), . . . , (a_m, m). Définir chaque a_i sur vrai donne une affectation de vérité satisfiable pour f , puisque chaque clause est maintenant satisfiable. Ceci est possible car chaque fois que nous définissons a_i comme étant vrai, nous savons que nous n'avons

jamais besoin de définir $\overline{a}_i$ comme étant également vrai, car sinon nous aurions une arête $\{(a_i, i), (\overline{a}_i, j)\}$ avec i ≠ j .

2) Supposons que nous ayons un oracle pour MAX CLIQUE, alors nous pourrions résoudre une instance du problème NP-complet CLIQUE en temps polynomial en utilisant l'algorithme simple suivant.

Entrée: un graphe G = (V, E) avec | V | = n et un entier k.

Sortie: vrai si et seulement si G a une clique d'ordre k.

Algorithme:

pour i = k à n

si MAX CLIQUE est vrai pour (G, i) alors la sortie est vraie

next i

sortie faux

Puisque cet algorithme effectue au plus n - k + 1 appels à l'oracle pour MAXCLIQUE et que chaque instance de MAX CLIQUE est essentiellement de la même taille que l'entrée, nous avons montré que CLIQUE $\leq_T$ MAX CLIQUE. Par conséquent, MAX CLIQUE est NP-hard (puisque CLIQUE est NP-complet).

3) A faire ?
4) Preuve: Pour deux langages A et B quelconques, il est facile de voir que si A $\leq_m$ B et B ∈ co-NP alors A ∈ co-NP. Supposons maintenant que L soit NP-complet et L ∈ co-NP. Si A ∈ NP alors A $\leq_m$ L et donc A ∈ co-NP. Ainsi NP ⊆ co-NP. Mais maintenant si A ∈ co-NP alors A^c ∈ NP ⊆ co-NP et donc A ∈ NP. D'où NP = co-NP.

5) Preuve : le fait que COMPOSÉ ∈ NP implique que PRIMIER ∈ co-NP nous devons donc montrer que PRIMIER ∈ NP.
Nous devons décrire un certificat succinct pour le fait qu'un entier n est premier. Si n est un nombre premier alors, il existe une racine primitive g mod n. Donc g satisfait $g^{n-1} = 1 \bmod n$ mais $g^d \neq 1 \bmod n$ pour tout diviseur propre d de n - 1. À l'inverse, supposons que $g \in \mathbb{Z}_n^*$ satisfait :
(i) $g^{n-1} = 1 \bmod n$ et
(ii) $g^d \neq 1 \bmod n$ pour tout diviseur propre d de n - 1,

On a : $\mathbb{Z}_n = \{a \mid 0 \leq a \leq n-1\}.$ et
$\mathbb{Z}_n^* = \{a \mid 1 \leq a \leq n-1 \text{ and } \gcd(a, n) = 1\}.$
et
$\phi : \mathbb{N} \to \mathbb{N}$
$\phi(n) = |\mathbb{Z}_n^*|$

if $n = p$ is prime then $\phi(p) = p - 1$.
If $g \in \mathbb{Z}_n^*$ then the *order* of g is

$$\text{ord}(g) = \min\{k \geq 1 \mid g^k = 1 \bmod n\}.$$

La condition (i) ci-dessus implique que ord (g) | (n - 1). De plus, la condition (ii) ci-dessus implique alors que ord (g) = n – 1 et ord (g) | φ (n) (théorie de groupe) et donc (n - 1) | Φ (n). Cela ne peut se produire que si Φ (n) = n - 1, auquel cas n est premier. Nous utiliserons ceci pour décrire un certificat succinct de la primalité d'un nombre n premier.

Nous n'aurons pas besoin d'un certificat pour la primalité de 2 puisque notre algorithme de vérification le reconnaîtra automatiquement. Soit C (n) le certificat pour un premier n ≥ 3,

C (n) sera composé :

(1) d'un g satisfaisant $g^{n-1} = 1 \bmod n$ mais $g^d \neq 1 \bmod n$ pour tout diviseur propre d de n - 1;

(2) une liste de nombres premiers $p_1 < p_2 < \cdots < p_r$ et d'exposants e_i tels que $n - 1 = \prod_{i=1}^{r} p_i^{e_i}$

(3) certificats C (p_2),. . . , C (p_r) pour la primalité des nombres premiers impairs

$p_2, p_3, \ldots, p_r$ (notez que $p_1 = 2$ puisque n est impair).

la condition (1) garantira que n est premier, nous devons donc décrire un algorithme de vérification du temps polynomial qui vérifiera que les conditions (1) - (3) sont réellement valables pour une entrée n particulière et un éventuel certificat C (n).

Afin de pouvoir vérifier (1) efficacement, nous utilisons la factorisation de n - 1 donnée en (2) avec le simple fait que si $a \in Z_n$ et qu'il existe un diviseur propre d de n - 1 tel que

$a^d = 1 \bmod n$ alors il y a un diviseur de n - 1 de la forme $d_i = (n - 1) / p_i$ tel que $a^{di} = 1 \bmod n$.

Nous pouvons maintenant décrire l'algorithme de vérification.

Algorithm *Prime certificate checking.*

Input: integer n and possible certificate $C(n)$.
Algorithm:
if $n = 2$ then output true
if $n - 1 \neq \prod_{i=1}^{r} p_i^{e_i}$ then output false
if $a^{n-1} \neq 1 \bmod n$ then output false
if $a^{(n-1)/2} = 1 \bmod n$ then output false
for $i = 2$ to r
  if $a^{(n-1)/p_i} = 1 \bmod n$ then output false
  if $C(p_i)$ is not a valid certificate for the primality of p_i
    then output false
next i
output true.

À ce stade, il devrait être clair que si n est premier, alors il existe un certificat C (n) que cet algorithme acceptera. Alors que si n est composé, quel que soit le certificat donné, l'algorithme le rejettera.

Afin de compléter la preuve, nous devons vérifier qu'il s'agit d'un algorithme de temps polynomial. Rappelez-vous que l'entrée est un entier n et que la taille d'entrée est donc O (log n). Notez que le nombre de facteurs premiers de n comptés en fonction de leur multiplicité est au plus log n car sinon leur produit serait supérieur à $2^{\log n} = n$. Ainsi, à l'exception possible de la ligne vérifiant le certificat C (p_i), chaque ligne de l'algorithme ci-dessus peut être exécutée en temps polynomial. Nous mesurerons le temps pris par cet algorithme par le nombre total de lignes de l'algorithme qui sont exécutées ; nous le désignons par f (n). (Notez que lorsqu'un certificat C (p_i) est vérifié, nous imaginons qu'une nouvelle version de l'algorithme démarre et comptons le nombre de lignes exécutées en conséquence.)

Notre algorithme «sait» que 2 est premier et n'a donc pas besoin de vérifier un certificat pour ce fait, il se termine après une seule ligne et donc f (2) = 1.

Maintenant, si n est un nombre premier impair, alors nous avons

Ligne1 +ligne2(sa valeur est calculée dans la 2eme condition de certificat)+ligne3+2(vérification de condition de primalité de deux formule de condition 1 de certificat)=5

r-1 pour le produit des r facteurs premiers condition 2 de certificat) +

r-1 pour la ligne 4+r-2(boucle for pour la ligne 6)) +

r-1 pour lire les certificats des facteurs premiers de la condition 3 de certificat =3(r-1)

$\sum_{i=2}^{r} f(p_i)$ pour la vérification de la validité de certificat des facteurs premiers dans la boucle for pour la ligne 7,donc :

$$f(n) = 5 + 3(r-1) + \sum_{i=2}^{r} f(p_i).$$

$$= 5 + \sum_{i=2}^{r} (f(p_i) + 3).$$

En posant g (n) = f (n) + 3, nous avons

$$g(n) = 8 + \sum_{i=2}^{r} g(p_i).$$

Nous utilisons maintenant l'induction sur n pour montrer que g (n) ≤ 8 log n. Ceci est vrai pour n = 2 puisque f (2) = 1 et donc g (2) = 4 <8. En supposant que cela est également vrai pour tous les nombres premiers p <n, nous avons

$$
\begin{aligned}
g(n) &\leq 8 + \sum_{i=2}^{r} 8 \log p_i \\
&= 8 + 8 \log \left(\prod_{i=2}^{r} p_i \right) \\
&\leq 8 \log((n-1)/2) + 8 \\
&= 8 \log(n-1) \\
&< 8 \log n.
\end{aligned}
$$

Donc f (n) ≤ 8 log n - 3 et donc l'algorithme de teste de primalité est un algorithme de vérification du temps polynomial, et donc PRIMIER ∈ NP.

Exemple : Certificat de primalité pour n = 103.

Un certificat pour 103 est

$$
\begin{aligned}
C(103) &= \{5, (2, 1), (3, 1), (17, 1), C(3), C(17)\} \\
C(3) &= \{2, (2, 1)\}, \quad C(17) = \{3, (2, 4)\}.
\end{aligned}
$$

Ceci est un certificat pour 103 puisque 5 est un racine primitif mod 103, $102 = 2^1 \times 3^1 \times 17^1$ et C (3), C (17) sont des certificats pour la primalité de 3, 17 respectivement. Le certificat pour 3 est C (3) puisque 2 est un racine primitif mod 3 et $2 = 2^1$. Enfin le certificat pour 17 est C (17) puisque 3 est un racine primitif mod 17 et $16 = 2^4$.

Exercice 6 :

Considérez le problème DE VOYAGEUR DE COMMERCE

Entrée: une liste de villes $c_1, \ldots, c_n$ et une matrice symétrique n × n d'entiers positifs donnant les distances entre chaque paire de villes.

Sortie: Le tour le plus court des villes, où un tour est un ordre des villes et la durée d'un tour est la somme des distances entre les villes consécutives (y compris la distance du dernier retour au premier).

En supposant que CYCLE HAMILTONIENNE est NP-complet, montrer que VOYAGEUR DE COMMERCE est NP-hard.

Où

CYCLE HAMILTONIENNE

Entrée: un graphe G.

Question: G est-il hamiltonien?

Exercice 7 :

1) Montrer que si $A \leq_T B$ et $B \leq_T C$ alors $A \leq_T C$.
2) On dit que deux langages sont Turing équivalents si elles sont Turing réductibles l'une à l'autre. Prouvez que deux langages NP-complets sont Turing équivalents.