1 Réduction de CNF-SAT à chemin hamiltonien

Un chemin de s à t dans un graphe G est dit hamiltonien si le chemin passe exactement une fois dans chaque nœud du graphe. On définit alors le problème $HAMPATH$ de la façon suivante :

$$HAMPATH = \{(G, s, t) \mid \text{il existe un chemin hamiltonien de } s \text{ à } t \text{ dans le graphe orienté } G\}$$

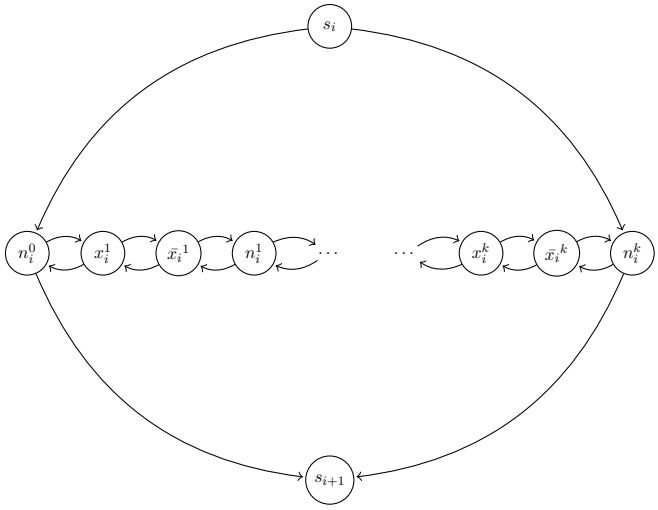
Question 1. Montrer que $HAMPATH$ est $\mathcal{NP}$-complet.

Solution 1.
Considérons le graphe avec $3l + 2$ nœuds par variable i ($v_0^i \dots v_{3k}^i$) et s_i. On relit bilatéralement les v_j^i à v_{j-1}^i et v_{j+1}^i quand ils existent. On ajoute $s_i \to v_0^i$, $v_0^i \to s_{i+1}$, $s_i \to v_{3k}$ et $v_{3k} \to s_{i+1}$. Pour chaque clause j on ajoute un nœud C_j et on branche pour chaque x_i de la clause les arcs $v_{3j+1}^i \to C_j$ et $C_j \to v_{3j+2}^i$ tandis que pour chaque $\bar{x}_i$ dans C_j on ajoute $v_{3j+2}^i \to C_j$ et $v_{3j+1}^i \to C_j$. La réduction polynomiale consiste maintenant à chercher un chemin hamiltonien de s_1 à s_{k+1}.
Soit un chemin hamiltonien de s_1 à s_{k+1}. Nous allons montrer par induction sur i puis j que les nœuds v_j^i apparaissent dans le chemin par i croissant et que pour un i donné les v_j^i apparaissent par j monotone (soit croissant soit décroissant) et que premier nœud visité après la couche i est s_{i+1}.
Supposons la propriété vraie pour tout $j < i$ avec $i \in \mathbb{N}^*$ (trivialement vraie pour $i = 0$ puisque s_1 est le départ du chemin). s_i n'est suivie que de v_0^i et v_{3k}^i. Les deux cas sont symétriques (mais l'un implique de traiter la couche par j croissant l'autre décroissant). Supposons s_i est suivi de v_0^i. Le nœud suivant de v_0^i ne peut pas être s_{i+1} car v_1^i n'est relié qu'à deux nœuds dont v_0^i. Donc le nœud suivant v_0^i est v_1^i. Montrons maintenant la propriété de croissance par induction sur j :

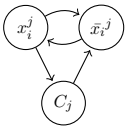
- Elle est vraie pour $j = 1$.
- Soit $1 \leq j < 3k$ et supposons la propriété vraie pour j. Si j est un multiple de 3 alors comme v_j^i n'est reliée qu'à v_{j+1}^i et v_{j-1}^i (qui est visité par induction) le successeur de v_j^i ne peut être que v_{j+1}^i et donc la propriété est vérifiée pour $j + 1$.
- Si $j = 3n + 1$ alors v_j^i ne peut être suivi que de C_n et de v_{j+1}^i (par induction v_{j-1}^i est déjà visité). Si le nœud suivant est v_{j+1}^i la propriété est obtenue. Sinon nous avons dans le chemin $v_{3n+1}^i \to C_n$. Si le nœud suivant C_n n'est pas v_{j+1}^i alors v_{3n+2}^i ne sera plus relié qu'à un unique nœud non visité v_{3n+3}^i et donc ne pourra pas être pris par un chemin hamiltonien. Donc le nœud suivant C_n est forcément v_{j+1}^i.
- Si $j = 3n + 2$ alors les successeurs possibles de v_j^i sont v_{j+1}^i (auquel cas la propriété est claire) et C_n (quand $\bar{x}_i$ apparaît dans C_n). Dans le cas où v_j^i est suivi de C_n alors v_{j+1}^i n'est plus relié qu'à un unique nœud et donc le chemin hamiltonien ne pourra plus être construit, donc v_j^i ne peut être suivi que de v_{j+1}^i.

Dans le cas où v_{3k}^i est le successeur de s_i alors on prouve la même chose dans l'autre sens, les nœuds v_j^i peuvent suivis que de v_{j-1}^i sauf dans le cas où $j = 3n + 2$ auquel cas on peut avoir $v_j^i \to C_{\lfloor j/3 \rfloor} \to v_{j-1}^i$ (dans le cas $j = 3n + 1$ ce n'est pas possible car v_{j-1}^i ne serait plus relié qu'à un unique nœud).
Dans tous les cas les v_j^i sont parcourus en j monotone et en arrivant au bout le seul successeur possible est s_{i+1}.
Nous voulons montrer que si l'on a un chemin hamiltonien alors nous avons une solution à la formule CNF. Pour cela on fixe $x_i = \top$ ssi les v_j^i sont parcourus en ordre croissant. Par la façon dont nous avons encodé les clauses, comme tous les nœuds C_j sont visités nous avons $v_{3j+1}^i \to C_j \to v_{3j+2}^i$ quand $x_i = \top$ et x_i apparaît dans C_j et inversement $v_{3j+2}^i \to C_j \to v_{3j+1}^i$ quand $x_i = \bot$ et $\bar{x}_i$ apparaît dans C_j. Donc la formule est bien satisfaite par l'assignement donné.
Montrons maintenant que si l'on a un assignement alors il existe une solution au chemin hamiltonien. Pour cela, on choisit pour chaque clause un de ses représentants qui la satisfait et le chemin est de parcourir chaque couche dans l'ordre croissant quand $x_i = \top$ et décroissant quand $x_i = \bot$. Si x_i est le littéral qui satisfait C_j alors on prend le chemin $v_{3j+1}^i \to C_j \to v_{3j+2}^i$ sinon $v_{3j+1}^i \to v_{3j+2}^i$. Dans le cas où $\bar{x}_i$ satisfait C_j alors on prend le chemin $v_{3j+2}^i \to C_j \to v_{3j+1}^i$ sinon $v_{3j+2}^i \to v_{3j+1}^i$. On voit que tous les v_j^i, les s_i aussi et les C_j sont visités une unique fois.

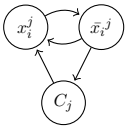
Suggestion : partir d'une formule CNF-SAT à k clauses $(C_1, \dots, C_k)$ et l variables $(x_1, \dots, x_l)$ et construire un graphe qui contient (pour i une variable et j une clause) : trois nœuds x_i^j, $\bar{x}_i^j$ et n_j^i ; deux nœuds s_i et n_i^0 ; un nœud C_j plus un nœud s_{l+1} et utilise les gadgets suivants :



(a) Gadget de variable



(b) Gadget pour x_i utilisé dans C_j



(c) Gadget pour $\bar{x}_i$ utilisé dans C_j

Question 2. Montrer que le problème du chemin hamiltonien reste $\mathcal{NP}$-complet pour les graphes non orientés.

Solution 2. Étant donné une instance du problème du chemin hamiltonien sur le graphe $G = (V, E)$ on triple chaque nœud n en n_{in}, n_{out} et n_{mid}. Pour chaque nœud on ajoute $n_{in} \to n_{mid} \to n_{out}$ et chaque arc (u, v) est transformé en $u_{out} \to v_{in}$. Enfin on enlève les nœuds s_{in}, s_{mid}, t_{out} et t_{mid}.
Chaque chemin dans G de s à t est clairement transposable en un chemin dans G' de s_{out} à t_{in}.
Soit un chemin $n_1, \dots n_l$ de G' de s_{out} à t_{in}. Nous allons montrer que le chemin $n_1, \dots, n_k$ est une succession de nœuds n_{in}, n_{mid}, n_{out} (le *in*, *mid*, *out* du même nœud) où n_{out} précède n'_{in} quand $n \to n'$ dans par induction sur k sauf le premier qui vaut s_{out} et le dernier qui vaut t_{in}.
Pour $k = 1$, on a $n_1 = s_{out}$ donc c'est clair .
Pour $k < l$ si on suppose que $v_1, \dots v_k$ est une alternance *in*, *mid*, *out*, alors si on a $v_k = n_mid$ c'est que l'on vient de n_{in} et que le seul nœud suivant est n_{out}. Si $v_k = n_in$ alors le noeud n_{mid} n'est pas encore visité (car la seule manière de rentrer dans n_{mid} c'est par n_{in} ou n_{out}) et doit l'être immédiatement sinon il ne sera plus relié qu'à un nœud non visité (n_{out}). Si $v_k = v_{out}$ alors n_{in} et n_{mid} ont été visités et n_{out} n'est relié qu'à des nœuds n'_{in} tels que n' suit n dans G.
Au bout du compte, un chemin hamiltonien dans G' a la forme $s_{out}, n^1_{in}, n^1_{mid}, n^1_{out}, \dots, n^k_{in}, n^k_{mid}, n^k_{out}, t_{in}$ et donc l'on peut retrouver un chemin dans le graphe original $s, n^1, \dots, n^k, t$.

2 Le théorème de hiérarchie temporelle

Pour simplifier la programmation des machines, nous n'utiliserons ici que des machines de Turing déterministes à 2 bandes. On utilise un encodage raisonnable des MT que l'on note $\langle M \rangle$ pour une machine M. Par exemple, $\langle M \rangle$ code les états par des mots de $\{0,1\}^*$, l'alphabet commun des rubans par des mots de $\{\alpha, \beta\}^*$, une direction de changement des têtes par $\{<, >, \_\}^2$. Une machine est alors représentée par une suite de transitions pour chaque état. Pour l'état e on représente ça de la façon $e|a_1, b_1, c_1, d_1, e_1, f_1, \dots a_k, b_k, c_k, d_k, e_k, f_k|$ où les $e, (e_i)_i$ sont des états, les $(a_i)_i, (b_i)_i, (c_i)_i, (d_i)_i$ sont des lettres du ruban, les $(f_i)_i$ des directions. a correspond à la lettre lue sur le premier ruban, b sur le second ruban, c à celle écrite sur le premier ruban et d à celle sur le second.

Définition 1. *L'ensemble $TIME(f(n))$ correspond à l'ensemble des problèmes pour lesquels il existe une machine de Turing déterministe qui décide en moins de $f(n)$ étapes de calcul les entrées de taille n.*

Attention, de la même manière qu'on dit que le tri peut se faire en $O(n \times ln(n))$ en omettant de dire "sur une entrée de taille n", on écrit toujours $TIME(f(n))$ mais il faut bien évidemment lire $TIME(n \to f(n))$ car le n représente la taille de l'entrée.
Dans la suite de ce TD, si besoin, vous pouvez vous reposer sur le théorème d'accélération linéaire qui énonce que pour tout g et tout $\epsilon > 0$ **fixés**, $TIME(g(n)) \subseteq TIME(n + 2 + \epsilon g(n))$.

Définition 2. *Une fonction f est dite* constructible en temps *s'il existe une machine M qui prend en entrée un mot 1^n et produit la représentation binaire de $f(n)$ en temps $f(n)$.*

Le théorème de hiérarchie temporelle dit que si f est une fonction constructible en temps alors :

$$TIME\left(o\left(\frac{f(n)}{log f(n)}\right)\right) \subsetneq TIME(f(n))$$

Nous allons ici montrer une version amoindrie, $TIME(f(n)) \subsetneq TIME(f(2n+1)^3)$ avec $n^3 \leq f(n)$.
Pour f constructible en temps, on définit le langage $Halt_f = \{\langle M\rangle \# x \mid M \text{ accepte } x \text{ en } f(|x|) \text{ étapes}\}$.
Question 3. Justifier que $Halt_f \in TIME(f(n)^3)$.

Solution 3. Nous allons simuler une machine M en temps $f(n)^3$. Au départ le ruban 1 contient $\langle M\rangle \# x$. Notre pseudo code est :

Action	État de bande 1	État de bande 2
Départ	$\langle M\rangle \# x$	
Calculer $f(n)$	$\langle M\rangle \# x_1 \dots x_k \# f(n)$	
Mettre x dans 2	$\langle M\rangle \# f(n)$	$x_1 0 x_2 0 \dots x_k 0$
Mettre état de départ et les têtes dans la bande 1	$\langle M\rangle \# f(n) \# e_0, x_1, 0$	$t_1 t_2 x_2 0 \dots x_k 0$
Répéter	$\langle M\rangle \# k \# e, a, b$	w
Copier e, a, b à la fin de w	$\langle M\rangle \# k \# e, a, b$	$w \# e, a, b$
Trouver la transition correspond à e, a, b dans $\langle M\rangle$		
Stocker f correspond dans les états de la MT		
Écrire le e de la transition sur la bande 2	$\langle M\rangle \# k \# e, a, b$	$w \# e', a, b$
Écrire le e de la transition depuis la bande 2 sur 1	$\langle M\rangle \# k \# e', a, b$	$w \# e', a, b$
Trouver t_1 sur la bande 2		
Échange t_1 et a puis		
déplacer la tête et mettre ce qu'on lit à la place de a	$\langle M\rangle \# k \# e', a', b$	
Trouver t_2 sur la bande 2		
Échange t_2 et b puis		
déplacer la tête et mettre ce qu'on lit à la place de b	$\langle M\rangle \# k \# e', a', b'$	
Décrémenter k		
Si $k = 0$ on termine et accepte ssi e est acceptant dans $\langle M\rangle$		

Supposons par l'absurde que $Halt_f \in TIME\left(\frac{f(\lfloor n/2 \rfloor)}{2}\right)$. Soit K décidant $Halt_f$ en temps $f(\lfloor n/2 \rfloor)/2$, on peut alors construire D_K qui prend en entrée $\langle M\rangle$ le code d'une machine de Turing et lance K sur $\langle M\rangle \# \langle M\rangle$. D_K accepte quand K refuse et refuse quand K accepte.
Question 4. Justifier que D_K peut être construit de façon à ce que $D_K \in TIME(f(n))$.

Solution 4. Pour D_K il faut lire $\langle M\rangle$ puis l'écrire sur la bande 2, revenir au début de la bande 2, ajouter # sur la bande 1 et copier le contenu de 2 sur 1 puis remettre les têtes au début. Ensuite on lance K et on inverse le résultat.
Le temps de calcul est celui de K puis une action linéaire donc $O(n + f(\lfloor 2n + 1 \rfloor))$ où $n = |\langle M\rangle|$. Par le théorème on a bien ce que l'on veut.

Question 5. Justifier que D_K ne peut pas exister par un argument de diagonalisation.

Solution 5. L'acceptation de D_K lancée sur $\langle D_K \rangle$ est l'inverse de celle de K sur $\langle D_K \rangle \# \langle D_K \rangle$ qui est la même que celle de D_K sur $\langle D_K \rangle$ (car $D_K \in TIME(f(n))$). Mais donc D_K s'accepte si et seulement si elle se refuse. Donc D_K ne peut pas exister.

Question 6. En déduire que $TIME(f(\lfloor n/2 \rfloor)/2) \subsetneq TIME(f(n)^3)$.

Solution 6. D_K ne pouvant pas exister c'est donc que $Halt_f \notin TIME(f(\lfloor n/2 \rfloor)/2)$.

Question 7. En déduire que $P \subsetneq EXPTIME \subsetneq 2 - EXPTIME$.
On rappelle : $P = \bigcup_{k \in \mathbb{N}} TIME(n^k)$, $EXPTIME = \bigcup_{k \in \mathbb{N}} TIME(2^{n^k})$, $2-EXPTIME = \bigcup_{k \in \mathbb{N}} TIME(2^{2^{n^k}})$

Solution 7. $x \to 2^x$ est constructible en temps car on lit 1^n et on écrit $1^n \# 0$ et ensuite on répète (la bande est $1^n \#^k x$) aller avant la partie $\#^k$ et si c'est un 1 alors doubler x sinon s'arrêter.
Soit $f(x) = 2^x$ on a $Halt_f$ qui est dans $EXPTIME$ mais pas dans $TIME(2^{x/2})$ donc pas dans P. De même avec $f(x) = 2^{2^x}$ on a $EXPTIME$ strictement inclus dans $2 - EXPTIME$.

3 Un problème $EXPTIME$ complet

Question 8. Justifier que $f(x) = 2^x$ est constructible en temps. En déduire que $Halt_f \in EXPTIME$.

Solution 8. Voir plus haut.

Question 9. Soit $L \in EXPTIME$, donner une réduction polynomiale de L à $Halt_f$. En déduire que $Halt_f$ est $EXPTIME$ complet.

Solution 9. Si $L \in EXPTIME$ on a M et k tel que L est reconnu par M en temps 2^{n^k} donc étant donné un x on lance notre machine pour $Halt_f$ sur $\langle M \rangle \# x \# 1^{n^k}$.

4 Un problème $PSPACE$ complet

L'ensemble des formules booléennes quantifiées (QBF) est défini par induction :
- Une variable propositionnelle est une formule booléenne quantifiée ;
- Si ϕ est une formule booléenne quantifiée, alors $\neg\phi$ est une formule booléenne quantifiée ;
- Si ϕ et ψ sont deux formules booléennes quantifiées, alors $\phi \wedge \psi$ est une formule booléenne quantifiée ;
- Si ϕ est une formule booléenne quantifiée et p est une variable propositionnelle, alors $\forall p\phi$ et $\exists p\phi$ sont des formules booléennes quantifiées.

Les QBF sont dotées de leur valeurs de vérité usuelle. Le langage $TQBF$ est le langage des QBF valant vrai.

Question 10. Montrer que le problème $TBQF$ est $PSPACE$ complet.

Suggestion pour la partie $PSPACE$ difficile : partir d'un problème L qui est $PSPACE$. Montrer que L est décidé par une machine K pour laquelle il existe $P \in \mathbb{N}[X]$ tel que sur une entrée de taille n, K accepte ou refuse en temps $2^{P(n)}$ avec $P(n)$ de mémoire. Faire une réduction de L à $TQBF$ Encoder l'état de K et de la mémoire de K avec un $P(n)$ uplet de $\{0,1\}$ et donner la formule $\Phi(c_1, c_2, t)$ qui décrit si l'on peut passer de l'état c_1 à c_2 en temps 2^t (utiliser une exponentiation rapide).

Solution 10. Tester toutes les assignations met clairement le problème dans $PSPACE$ puisque qu'assignation a une taille linéaire et tester la formule peut alors se faire linéairement.
Maintenant on réduit n'importe quel problème reconnu par une machine M en espace n^k et temps 2^{n^k} en simulant M avec une formule. Nous supposons que M travaille sur un alphabet $\{0,1\}$ donc la machine M peut s'encoder avec $|M|$ variable pour déterminer l'état courant, n^k variable pour savoir où est la tête et n^k variable pour savoir le contenu de chaque case du ruban.

Déterminer si l'on peut passer d'un état (du ruban et de la machine) c_1 à un état c_2 (tous deux étant des $2n^k + |M|$-uplets de variables) se fait en une formule $\Psi(c_1, c_2)$. Maintenant pour savoir si l'on peut passer de c_1 à c_2 en 2^t étapes on calcule $\Phi(c_1, c_2, t)$ avec $\Phi(c_1, c_2, 0) = \Psi(c_1, c_2)$ et

$$\Phi(c_1, c_2, t+1) = \exists m \forall (c_3, c_4)((c_3 \neq c_1 \vee c_4 \neq m) \wedge (c_4 \neq c_3 \vee c_3 \neq m)) \vee \Phi(c_3, c_3, t)$$

Les c_3, c_4 sont ici des $2n^k + |M|$-uplets de variables. L'inégalité entre deux l-uplets $v_1, \dots v_l$ et $w_1, \dots, w_l$ est réalisée par la formule : $(v_1 \wedge \neg w_1) \vee (\neg v_1 \wedge w_1) \vee \dots \vee (v_l \wedge \neg w_l) \vee (\neg v_l \wedge w_l)$.