TP d'Algo/Complexité/Calculabilité

CIMBE Pierre-Alexandre
LAGNIEZ Jean-Marc
LESNYAK Viktor
RAFIK Ahmed

December 18, 2013

1 Partie théorique

1.1 Partie algorithmique

1.1.1 Exercice 1

1. Le nombre de s-t chemins d'arc disjoint est de 3.
   On trouve ce résultat en appliquant l'algorithme de Ford-Fulkerson.

   s -> 4 -> 5 -> t F = 1
   s -> 3 -> 4 -> t F = 1
   s -> 2 -> 3 -> t F = 1

   Tous les arcs du graphe sont saturés donc on ne peut plus trouver de chemin améliorant et Fmax = 3.

2. Soit Si l'ensemble des sommets de la coupe i, $\overline{S}i$ l'ensemble des sommets du complémentaire de la coupe i, Avi l'ensemble des arcs avants de la coupe i, et Ari l'ensemble des arcs arrière de cette coupe.

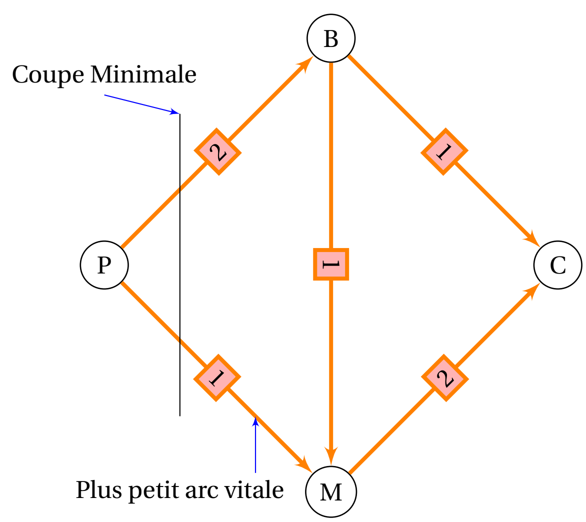
   S1 = {1} ; $\overline{S}$1 = {2, 3, 4, 5, 6}
   Av1 = {1->4 ; 1->3 ; 1->2} ; Ar1 = {}
   S2 = {1, 2} ; $\overline{S}$2 = {3, 4, 5, 6}
   Av2 = {1->4 ; 1->3 ; 2->3} ; Ar2 = {}
   S3 = {1, 4} ; $\overline{S}$3 = {2, 3, 5, 6}
   Av3 = {1->3 ; 1->2 ; 4->5 ; 4->6} ; Ar3 = {3->4}
   S4 = {1, 2, 3} ; $\overline{S}$4 = {4, 5, 6}
   Av4 = {1->4 ; 3->4 ; 3->6} ; Ar4 = {}
   S5 = {1, 2, 4} ; $\overline{S}$5 = {3, 5, 6}
   Av5 = {1->3 ; 2->3 ; 4->5 ; 4->6} ; Ar5 = {3->4}
   S6 = {1, 4, 5} ; $\overline{S}$6 = {2, 3, 6}
   Av6 = {1->2 ; 1->3 ; 4->6 ; 5->6} ; Ar6 = {3->4}
   S7 = {1, 2, 3, 4} ; $\overline{S}$7 = {5, 6}
   Av7 = {4->5 ; 4->6 ; 3->6} ; Ar6 = {}
   S8 = {1, 2, 3, 4, 5} ; $\overline{S}$8 = {6}
   Av6 = {3->6 ; 4->6 ; 5->6} ; Ar6 = {}

3. Nombre minimum d'arc sortant dans une s-t coupe = 3
   Nombre maximum de chemin d'arc disjoint = 3

1.1.2 Exercice 2

Soit un graphe G=(V,E);
avec $\forall i, j \in \mathrm{V}$**,**
et f(i,j)-est la fonction representant le flot de l'arc ou $\forall (i, j) \in \mathrm{E}$**;**

1. Vrai, car si on enleve un arc (i,j), avec la f(i,j)=0,
   alors on n'influence pas la valeur du flot maximal.
   Ce qui correspondt a la definition de plus petit
   arc vital.
2. Vrai, car dans un flot maximum un arc (i,j) avec le f(i,j)
   minimum est le plus petit arc vital d'apres la
   definition de plus petit arc vital.
3. Faux, voila un contre exemple.



1.1.3 Exercice 3

1. Pour trouver une double etoile de cout min dans le Graphe, on va créer une instance G' qui est un sous-graphe de G_{2n+1}.
   Pour construire cette instance G', on prends notre graphe de depart et on enleve un sommet aleatoire sur ce graph, ce qui nous donne G'. Ensuite on va chercher un couplage min dans G', apres pour chaque couple de notre couplage (i,j) on ajoute à G l'arete min[(x,i),(x,j)].
   Donc on applique les instruction montré plus haut a tous les sommets de notre graphe initialle G. et a la fin on choisi le couplage de plus petit coup.

1.2 Partie complexité

1.2.1 Exercice 5

1. (a) SAT : Un problème SAT est un problème de décision visant à montrer l'existence d'une interprétation satisfaisant un ensemble de variable propositionnelle (Formule logique CNF)
3-SAT : Cas particulier du problème SAT dans lequel les clauses sont toutes de taille 3.

(b) On dit qu'il existe une réduction d'un problème P à un problème P' s'ilf existe une fonction f telle que x ∈ D(P) <=> f(x) ∈ D(P')

(c) 3-SAT est un cas particulier de SAT, or SAT ∈ NP.
Donc 3-SAT ∈ NP
SAT ∈ NP-Complet
Nous allons chercher à réduire un problème SAT à un problème 3-SAT :
Soit P une instance du problème SAT.
↠Soit U=($l_1 \vee l_2 \vee l_3 ...... \vee l_k$) une clause de taille k > 3
On la divise en 2 clause : ≻ une clause de taille ⌊k/2⌋+1 en complétant par une variable x∉U ≻ une clause de taille ⌈k/2⌉+1 en complétant par $\overline{x}$ le complémentaire de x.
On applique ce principe récursivement jusqu'à obtenir des clauses de taille 3
↠Soit U=$l_1 \vee l_2$ une clause de taille 2
On clone la clause U pour avoir 2 clauses U_1 et U_2 auxquelles on ajoute respectivement une variable x et son complémentaire $\overline{x}$ on obtient : $U_1 = l_1 \vee l_2$vx $U_2 = l_1 \vee l_2 \vee \overline{x}$
↠Soit U=l_1 une clause de taille 1 On force l_1 à vrai et on retire les clauses unitaire.
On obtient ainsi un problème P' de type 3-SAT.
Donc 3-SAT est NP-Complet.

(d) Détaillons chaque cas concernant chaque type de clause : ↠Clause de taille 4 :
Soit ($l_1 \vee l_2 \vee l_3 \vee l_4$) une clause de taille 4

$$(l_1 \vee l_2 \vee l_3 \vee l_4) \begin{cases} (l_1 \vee l_2 \vee u) \\ (l_3 \vee l_4 \vee \overline{u}) \end{cases}$$

Chaque clause contenant 4 littéraux est ainsi divisée en 2 clauses de 3 littéraux

$$n_4 \text{ -> } 2n_4$$

On ajoute ainsi une variable pour chaque clause

$$4n_4 \text{ -> } 5n_4$$

↠Clause de taille 5 :
Soit ($l_1 \vee l_2 \vee l_3 \vee l_4 \vee l_5$) une clause de taille 5

$$(l_1 \vee l_2 \vee l_3 \vee l_4 \vee l_5) \begin{cases} (l_1 \vee l_2 \vee u) \\ (l_3 \vee l_4 \vee l_5 \vee \overline{u}) \end{cases}$$

$$(l_3 \vee l_4 \vee l_5 \vee \overline{u}) \begin{cases} (l_3 \vee l_4 \vee y) \\ (l_5 \vee \overline{u} \vee \overline{y}) \end{cases}$$

Chaque clause contenant 5 littéraux est ainsi divisée en 3 clauses de 3 littéraux

$$n_5 \text{ -> } 3n_5$$

On ajoute ainsi 2 variables pour chaque clause

$$5n_5 \text{ -> } 7n_5$$

↠<u>Clause de taille 2 :</u>
Soit $(l_1 \vee l_2)$ une clause de taille 2

$$(l_1 \vee l_2) \begin{cases} (l_1 \vee l_2 \vee u) \\ (l_1 \vee l_2 \vee \overline{u}) \end{cases}$$

Chaque clause contenant 2 littéraux est transformée en 2 clauses de 3 littéraux

$$n_2 \text{ -> } 2n_2$$

On ajoute ainsi 1 variables pour chaque clause

$$2n_2 \text{ -> } 3n_2$$

↠<u>Clause de taille 1 :</u>
Soit (l_1) une clause de taille 1

$$(l_1) \{ \ (l_1 \vee u \vee \overline{u})$$

Chaque clause contenant 1 littéraux est transformée en une clause de 3 littéraux

$$n_1 \text{ -> } n_1$$

On ajoute ainsi 1 variables pour chaque clause

$$n_1 \text{ -> } 3n_1$$

On peut donc conclure :

$$\text{Nombre de clauses} = n_1 + 3n_2 + n_3 + 2n_4 + 3n_5$$
$$\text{Nombre de variables} = 3n_1 + 3n_2 + 3n_3 + 5n_4 + 7n_5$$

2. Soit U = $(l_1 \vee l_2 \vee l_3)$ une clause de taille 3

   En appliquant la réduction, on obtient :

$$(l_1 \vee l_2 \vee l_3) \begin{cases} (l_1 \vee u) \\ (l_2 \vee l_3 \vee \overline{u}) \end{cases}$$

   On obtient ainsi une nouvelle clause de taille 3.
   Il est donc impossible de réduire un problème 3-SAT à un problème 2-SAT de cette manière.

3.

4. (a) Le problème étudié fait évidement partie de la classe NP.
   En effet, on vérifie aisement, à partir d'un ensemble L de sommets, que L est l'ensemble des feuilles d'un arbre couvrant de G.

   Réduction : Soit un graphe G = (V,E) et un ensemble L ⊂ V.
   On choisis arbitrairement 2 sommets x et y dans L.
   On prend le sous graphe G' = (V',E') tel que V'=V-L-x,y

   Si il existe un chemin Hamiltonien entre x et y dans G' alors il existe un arbre couvrant dont l'ensemble des feuilles est L.

En effet, le chemin hamiltonien relie tous les sommets de G'. En y rajoutant les sommets restant L on obtient un arbre couvrant de G.

De la meme manière, si il existe un arbre couvrant dans G, on peut lui retirer les sommets de L-x,y. Ainsi il nous reste un chemin hamiltonien de x à y dans G'.

Notre problème est donc NP-Complet.

(b) Le problème étudié fait évidement partie de la classe NP.
En effet, on vérifie aisement, à partir d'un ensemble L de sommets, que l'ensemble des feuilles d'un arbre couvrant de G est contenu dans L.

Réduction : Soit un graphe G = (V,E) et un ensemble L ⊂ V.
On choisis arbitrairement 2 sommets x et y dans L.
On prend le sous graphe G' = (V',E') tel que V'=V-L-x,y

S'il existe un chemin Hamiltonien entre x et y dans G' alors il existe un arbre couvrant dont l'ensemble des feuilles est contenu L.

S'il existe un arbre couvrant dans G, on peut lui retirer les sommets de L-x,y.
Ainsi il nous reste un chemin hamiltonien de x à y dans G'.

Notre problème est donc NP-Complet.

(c)

5. Le problème Connected dominating fait évidement partie de la classe NP.
En effet, on vérifie aisement, à partir d'un ensemble S, que chaque sommet x de G est soit dans S, soit un voisin d'un sommet de S.

Réduction : Soit un graphe G(E,V) connexe.
On crée un nouveau graphe G'(E',V') tel que ∀ (u,v)∈ E, on ajoute dans G' un nouveau sommet x relié à u et v.

Un vertex cover de G est alors un ensemble dominant de G'.
Donc, si on trouve un vertex cover de taille k dans G, alors ∃ S un ensemble dominant de taille k dans G' et inversement.

6. Balanced 3-SAT ∈ NP
On réduit balanced SAT à balanced 3-SAT de la meme manière que nous avons réduit SAT à 3-SAT.
Nous savons que balanced SAT est NP-Complet donc balanced 3-SAT est NP-Complet.

1.3 Partie calculabilité

1.3.1 Exercice 7

1. **Comment enumérer les couples d'entiers?**
2. **Donner les fonctions de codage et de décodage f1 → x et f2 → y**
3. **Montrer que l'on peut coder les triplets. Géneraliser aux k-uplets.**
4. **Pensez-vous que l'on peut coder les éléments de l'intervalle [0,1]. Justifier.**

1. Soit (x,y) ∈ ℕ * ℕ, alors faire x + y et trié par ordre lexicographique
2. La fonction de codage est :
$$z = \frac{(x+y)(x+y+1)}{2} + y$$
Pour les fonction de décodage, posons t tel que
$$t = x + y$$
On va prendre t tel que si t augmente de 1 alors
$$\frac{t(t+1)}{2} > z$$
sinon on a
$$\frac{t(t+1)}{2} \leq z$$
La fonction de décodage de y est:
$$z = \frac{t(t+1)}{2} + y$$
$$y = z - \frac{t(t+1)}{2}$$
La fonction de décodage de x est:
$$x = t - y$$
$$x = -z + t + \frac{t(t+1)}{2}$$
$$x = -z + \frac{t(t+3)}{2}$$
3. Pour coder les triplets, il suffit de coder deux entier et coder le résultat et le dernier entier.
$$h(x,y,z) = c(x, c(y,z))$$
On peut repeter se raisonement pour les k-uplets, ainsi on a
$$k(x1, x2...xk) = c(x1, c(x2, ...c(xk-1, xk))$$

4. On ne peut pas coder les éléments de l'intervalle [0,1] car l'ensemble n'est pas dénombrable. On utilise la diagonal de cantor sur cette ensemble.
   Supposons que l'on puisse numeroter $\mathbb{N} \rightarrow [0,1]$ et on en définie la suite S telle que tout éléments de [0,1] soit élément de la suite S. Et on définie un réel r tel que la partie entière est égal à 0 et que chaque décimal en position n est égal à sn(n)[1]+1 si sn(n) est différent de 9 et sn(n)-1 si sn(n) est égal à 9.
   Par construction, r n'est pas dans S sinon on aurait un Sn tel que

$$Sn(n) = r(n) = Sn(n) + 1$$

   ou

$$Sn(n) = r(n) = Sn(n) - 1$$

   C'est absurbe, ainsi ce n'est pas dénombrable.

1.3.2 Exercice 8

1. Les fonctions primitives récursives sont toutes les fonctions que l'on peut construire à partir des fonctions de base pas composition et récursion primitive.

   Exemple

   Soit les fonctions primitives:
   $O \in \mathbb{N}^0$, $\pi_i^k \in \mathbb{N}^k$ et SUC $\mathbb{N}^1$

$$O() = 0$$

$$\pi_i^k(x_1, x_2..., x_k) = x_i$$

$$\mathrm{SUC}(x_1) = x_1 + 1$$

   Soit la fonction qu'on utilise pour la récursion primitive:
   $g \in \mathbb{N}^1$

$$g() = \mathrm{SUC}(O())$$

   Soit la fonction recursive primitive:
   $f \in \mathbb{N}^1$

$$f(0) = g()$$

$$f(\mathrm{SUC}(n)) = \pi_1^2(f(n), n)$$

2. yoloooooooo je ne sais pas

3. (a) Soit la fonction somme défini ainsi:
   Sum $\in \mathbb{N}^2$

$$Sum(0, y) = \pi_1^1(y) = y$$

$$Sum(Suc(x), y) = \pi_2^3(x, Sum(x, y), y)$$

   (b) Soit la fonction Mult défini ainsi:
   Mult $\in \mathbb{N}^2$

$$Mult(O, y) = 0() = 0$$

$$Mult(1, y) = \pi_1^1(y) = y$$

$$Mult(Suc(x), y) = \pi_2^3(x, Sum(Mult(x, y), y), y)$$

[1] la nème décimal du nème élément de S

(c) Soit la fonction puissance défini aisni:
$X^Y \in \mathbb{N}^2$

$$X^Y(x,0) = Suc(0()) = 1$$

$$X^Y(x,Suc(y)) = \pi_2^3(x, Mult(X^Y(x,y),x),y)$$

(d) Soit la fonction prédecesseurs tel que:
$Pred \in \mathbb{N}^1$

$$Pred(0) = O() = 0$$

$$Pred(Suc(x)) = \pi_1^2(x, Pred(x))$$

(e) Soit la fonction soustraction tel que:
$X\text{-}Y \in \mathbb{N}^2$

$$X - Y(0,y) = 0() = 0$$

$$X - Y(x,0) = \pi_1^1(x) = x$$

$$X - Y(x,y) = \pi_2^3(x, X - Y(Pred(x), Pred(y)), y))$$

(f) Soit la fonction sg tel que: $sg \in \mathbb{N}^1$

$$sg(0) = 0() = 0$$

$$sg(Suc(x)) = \pi_1^2(1, Suc(x))$$

(g) Soit la fonction $X > Y$ tel que :
$X>Y \in \mathbb{N}^2$

$$X > Y(0,y) = 0$$

$$X > Y(x,0) = 1$$

$$X > Y(x,y) = \pi_2^3(x, X > Y(Pred(x), Pred(y)), y)$$

Soit la fonction $X \geq Y$ tel que :
$X \geq Y \in \mathbb{N}^2$

$$X \geq Y(0,0) = 1$$

$$X \geq Y(0,y) = 0$$

$$X \geq Y(x,0) = 1$$

$$X \geq Y(x,y) = \pi_2^3(x, X > Y(Pred(x), Pred(y)), y)$$

4. (a) Voici la fonction d'Ackerman pour $0 \leq m \leq 3$ et $0 \leq n \leq 4$

m/n	0	1	2	3	4
0	1	2	3	4	5
1	2	3	4	5	6
2	3	5	7	9	11
3	5	13	28	58	118

(b) Fesons une preuve par récurrence

$$A_0(n) = Suc(n) = n + 1$$

Hypothèse: $A_m(n)$ est primitive récursive
Montrons que $A_{m+1}(n)$ est primitive récursive

Si n = 0, on a que $A_{m+1}(n) = A_m(1)$. D'après l'hypothèse de réccurence, on a que $A_m(n)$ est primitive récursive. Donc $A_{m+1}(n)$ est primitif recursive

Si n > 0, on a que $A_{m+1}(n) = A_m(A_{m+1}(n))$.
Posons n' = $A_{m+1}(n)$. Donc on a $A_m(n')$. D'après l'hypothèse de réccurence, on a que $A_m(n)$ est primitive récursive pour tous $n \in \mathbb{N}$. Donc $A_{m+1}(n)$ est primitif recursive.

(c) Fesons une preuve par récurrence

$$A_0(n) = n+1$$

n+1 > n donc c'est vrai au premier rang
Hypothèse: $A_m(n) > n$
Montrons que $A_{m+1}(n) > n$

Maitenant, on applique une récurrence sur n
n = 0 : $A_{m+1}(1) > 1 > 0$ Hypothèse: $A_{m+1}(n) > n$
Montrons que: $A_{m+1}(n+1) > n+1$
On utilise les deux hypothèse de réccurence:

$$Am+1(n+1) = Am(Am+1(n)) > Am+1(n) > n$$

Ainsi

$$Am+1(n) \geq n+1$$

Donc

$$Am+1(n+1) > n+1$$

On peux donc conclure que $A_m(n) > n$

(d) Il faut montrer que $A_{m+1}(n) - A_m(n) \geq 0$
Fesons une preuve par récurrence sur m

$$A_0(n+1) - A_0(n) = n+1-n = 1$$

Hypothèse: $A_m(n+1) - A_m(n) > 0$
Montrons que $A_{m+1}(n+1) - A_{m+1}(n) > 0$
$A_{m+1}(n+1) = A_m(A_{m+1}(n)) > A_m(n)$
On peut conclure que
$A_m(n+1) - A_m(n) > 0$

(e) Pour n = 0: $A_{m+1}(0) = A_m(1)$. De plus, d'après la question précédente, on a que $A_m(1) > A_m(0)$

Pour n > 0: $A_{m+1}(n) = A_m(A_{m+1}(n-1))$. De plus on a que $A_m(n-1) > n - 1 \rightarrow A_m(n-1) \geq n$
Comme la fonction est strictement croissante, on a que $A_m(A_m(n-1)) \geq A_m(n)$

On peux en conclure que
$A_{m+1}(n) = A_m(A_{m+1}(n-1)) \geq A_m(n)$

(f) D'après les question précédente, on a montré que $A_{m+1}(n) \geq A_m(n)$ et que $A_m(n+1) > A_m(n)$. Ceci prouve que A_m^k est strictement croissante.

(g) Fesons une preuve pas récurrence sur k.
Au cas de base, on a bien $A_{m+1}(n) \geq A_m(n)$
Hypothèse: $A_{m+1}(n + k) \geq A_m^k(n)$
Montrons que : $A_{m+1}(n + k + 1) \geq A_m^{k+1}(n)$
D'après l'hypothèse de réccurence, on a

$$A_m^{k+1} = A_m(A_m^k(n)) \leq A_m(A_{m+1}(n+k))$$

De plus:

$$A_{m+1}(n+k+1) = A_m(A_{m+1}(n+k))$$

On peut conclure que:

$$A_m^{k+1} = A_m(A_m^k(n)) \leq A_{m+1}(n+k+1)$$

(h) Fesons une preuve par l'absurbe, soit la fonction d'Ackermann primitive récursive.

Sois la fonction

$$f : \mathbb{N} \rightarrow \mathbb{N} : n \rightarrow A(n, 2n)$$

Comme la fonction d'Ackerman est primitive récursive alors f est primitive récursive.