

UNIVERSITÉ MONTPELLIER II  
MASTER 1ÈRE ANNÉE INFORMATIQUE



**December 19, 2013**  
**Rapport de Travaux Pratique**

## **TP d'Algo/Complexité/Calculabilité**

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

# 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.

$$\begin{aligned}s &\rightarrow 4 \rightarrow 5 \rightarrow t \quad F = 1 \\s &\rightarrow 3 \rightarrow 4 \rightarrow t \quad F = 1 \\s &\rightarrow 2 \rightarrow 3 \rightarrow t \quad F = 1\end{aligned}$$

Tous les arcs du graphe sont saturés donc on ne peut plus trouver de chemin améliorant et  $F_{\max} = 3$ .

2. Soit  $S_i$  l'ensemble des sommets de la coupe  $i$ ,  $\bar{S}_i$  l'ensemble des sommets du complémentaire de la coupe  $i$ ,  $Av_i$  l'ensemble des arcs avant de la coupe  $i$ , et  $Ar_i$  l'ensemble des arcs arrière de cette coupe.

$$\begin{aligned}S_1 &= \{1\}; \bar{S}_1 = \{2, 3, 4, 5, 6\} \\Av_1 &= \{1 \rightarrow 4; 1 \rightarrow 3; 1 \rightarrow 2\}; Ar_1 = \{\} \\S_2 &= \{1, 2\}; \bar{S}_2 = \{3, 4, 5, 6\} \\Av_2 &= \{1 \rightarrow 4; 1 \rightarrow 3; 2 \rightarrow 3\}; Ar_2 = \{\} \\S_3 &= \{1, 4\}; \bar{S}_3 = \{2, 3, 5, 6\} \\Av_3 &= \{1 \rightarrow 3; 1 \rightarrow 2; 4 \rightarrow 5; 4 \rightarrow 6\}; Ar_3 = \{3 \rightarrow 4\} \\S_4 &= \{1, 2, 3\}; \bar{S}_4 = \{4, 5, 6\} \\Av_4 &= \{1 \rightarrow 4; 3 \rightarrow 4; 3 \rightarrow 6\}; Ar_4 = \{\} \\S_5 &= \{1, 2, 4\}; \bar{S}_5 = \{3, 5, 6\} \\Av_5 &= \{1 \rightarrow 3; 2 \rightarrow 3; 4 \rightarrow 5; 4 \rightarrow 6\}; Ar_5 = \{3 \rightarrow 4\} \\S_6 &= \{1, 4, 5\}; \bar{S}_6 = \{2, 3, 6\} \\Av_6 &= \{1 \rightarrow 2; 1 \rightarrow 3; 4 \rightarrow 6; 5 \rightarrow 6\}; Ar_6 = \{3 \rightarrow 4\} \\S_7 &= \{1, 2, 3, 4\}; \bar{S}_7 = \{5, 6\} \\Av_7 &= \{4 \rightarrow 5; 4 \rightarrow 6; 3 \rightarrow 6\}; Ar_7 = \{\} \\S_8 &= \{1, 2, 3, 4, 5\}; \bar{S}_8 = \{6\} \\Av_8 &= \{3 \rightarrow 6; 4 \rightarrow 6; 5 \rightarrow 6\}; Ar_8 = \{\}\end{aligned}$$

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

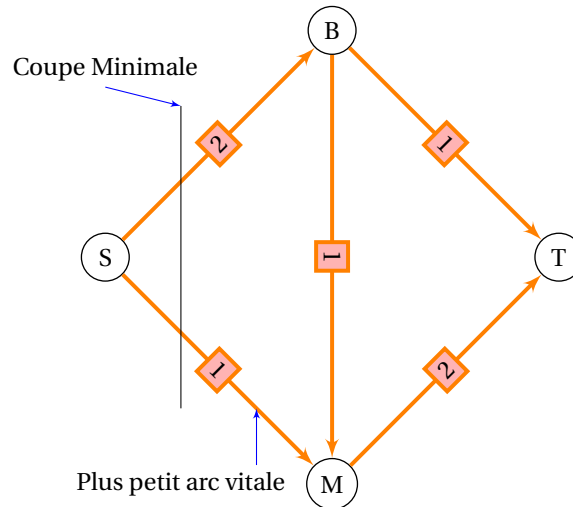
Soit un graphe  $G=(V,E)$ ;

avec  $\forall i, j \in V$ ,

et  $f(i,j)$ -est la fonction représentant le flot de l'arc ou  $\forall (i, j) \in E$ ;

1. Vrai, car si on enlève un arc  $(i,j)$ , avec la  $f(i,j)=0$ , alors on n'influence pas la valeur du flot maximal.  
Ce qui correspond à la définition 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'après la définition de plus petit arc vital.
3. Faux, voilà un contre exemple.



### 1.1.3 Exercice 3

1. Pour trouver une double étoile de coût 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 départ et on enlève un sommet aléatoire sur ce graphe, ce qui nous donne  $G'$ . Ensuite on va chercher un couplage min dans  $G'$ , après pour chaque couple de notre couplage  $(i,j)$  on ajoute à  $G$  l'arête  $\min[(x,i),(x,j)]$ .  
Donc on applique les instructions montrées plus haut à tous les sommets de notre graphe initial  $G$ . et à la fin on choisit le couplage de plus petit coût.

## 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 variables propositionnelles (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'il existe une fonction  $f$  telle que  $x \in D(P) \iff f(x) \in D(P')$
- (c) 3-SAT est un cas particulier de SAT, or  $SAT \in NP$ .  
Donc  $3-SAT \in NP$   
 $SAT \in NP$ -Complet  
Nous allons chercher à réduire un problème SAT à un problème 3-SAT :  
Soit  $P$  une instance du problème SAT.  
 $\rightarrow$  Soit  $U = (l_1 \vee l_2 \vee l_3 \dots \vee l_k)$  une clause de taille  $k > 3$   
On la divise en 2 clauses :  $>$  une clause de taille  $\lfloor k/2 \rfloor + 1$  en complétant par une variable  $x \notin U$   $>$  une clause de taille  $\lfloor k/2 \rfloor + 1$  en complétant par  $\bar{x}$  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  $\bar{x}$  on obtient :  $U_1 = l_1 \vee l_2 \vee x$  et  $U_2 = l_1 \vee l_2 \vee \bar{x}$

→ 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

$$(l_1 \vee l_2 \vee l_3 \vee l_4) \left\{ \begin{array}{l} (l_1 \vee l_2 \vee u) \\ (l_3 \vee l_4 \vee \bar{u}) \end{array} \right.$$

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

$$n_4 \rightarrow 2n_4$$

On ajoute ainsi une variable pour chaque clause

$$4n_4 \rightarrow 5n_4$$

→ Clause de taille 5 :

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

$$(l_1 \vee l_2 \vee l_3 \vee l_4 \vee l_5) \left\{ \begin{array}{l} (l_1 \vee l_2 \vee u) \\ (l_3 \vee l_4 \vee l_5 \vee \bar{u}) \end{array} \right.$$

$$(l_3 \vee l_4 \vee l_5 \vee \bar{u}) \left\{ \begin{array}{l} (l_3 \vee l_4 \vee y) \\ (l_5 \vee \bar{u} \vee \bar{y}) \end{array} \right.$$

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

$$n_5 \rightarrow 3n_5$$

On ajoute ainsi 2 variables pour chaque clause

$$5n_5 \rightarrow 7n_5$$

→ Clause de taille 2 :

Soit  $(l_1 \vee l_2)$  une clause de taille 2

$$(l_1 \vee l_2) \left\{ \begin{array}{l} (l_1 \vee l_2 \vee u) \\ (l_1 \vee l_2 \vee \bar{u}) \end{array} \right.$$

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

$$n_2 \rightarrow 2n_2$$

On ajoute ainsi 1 variables pour chaque clause

$$2n_2 \rightarrow 3n_2$$

→ Clause de taille 1 :

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

$$n_1 \rightarrow n_1$$

On ajoute ainsi 1 variables pour chaque clause

$$n_1 \rightarrow 3n_1$$

On peut donc conclure :

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

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

En appliquant la réduction, on obtient :

$$(l_1 \vee l_2 \vee l_3) \left\{ \begin{array}{l} (l_1 \vee u) \\ (l_2 \vee l_3 \vee \overline{u}) \end{array} \right\}$$

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 évidemment partie de la classe NP.

En effet, on vérifie aisément, 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  $L \subset V$ .

On choisit 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 même 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 évidemment 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  $L \subset V$ .

On choisit 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 dans  $L$ .

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 évidemment 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  $\forall (u, v) \in E$ , 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  $\exists S$  un ensemble dominant de taille  $k$  dans  $G'$  et inversement.

6. Balanced 3-SAT est un cas particulier de Balanced SAT, or Balanced SAT  $\in$  NP.  
Donc Balanced 3-SAT  $\in$  NP

Balanced SAT  $\in$  NP-Complet

On réduit balanced SAT à balanced 3-SAT de la même 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 énumérer les couples d'entiers?
2. Donner les fonctions de codage et de décodage  $f_1 \rightarrow x$  et  $f_2 \rightarrow y$
3. Montrer que l'on peut coder les triplets. Généraliser aux k-uplets.
4. Pensez-vous que l'on peut coder les éléments de l'intervalle  $[0,1]$ . Justifier.

1. Soit  $(x, y) \in \mathbb{N}^* \times \mathbb{N}$ , 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 raisonnement pour les k-uplets, ainsi on a

$$k(x_1, x_2, \dots, x_k) = c(x_1, c(x_2, \dots, c(x_{k-1}, x_k)))$$

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 définit la suite  $S$  telle que tout éléments de  $[0,1]$  soit élément de la suite  $S$ . Et on définit un réel  $r$  tel que la partie entière est égal à 0 et que chaque décimal en position  $n$  est égal à  $s_n(n)^1 + 1$  si  $s_n(n)$  est différent de 9 et  $s_n(n) - 1$  si  $s_n(n)$  est égal à 9.

Par construction,  $r$  n'est pas dans  $S$  sinon on aurait un  $S_n$  tel que

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

ou

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

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  $SUC \in \mathbb{N}^1$

$$O() = 0$$

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

$$SUC(x_1) = x_1 + 1$$

Soit la fonction qu'on utilise pour la récursion primitive:

$g \in \mathbb{N}^1$

$$g() = SUC(O())$$

Soit la fonction recursive primitive:

$f \in \mathbb{N}^1$

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

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

2. yooooooooo 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:

$Mult \in \mathbb{N}^2$

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

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

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

---

<sup>1</sup>la nème décimal du nème élément de  $S$



- (c) Soit la fonction puissance défini ainsi:

$$X^Y \in \mathbb{N}^2$$

$$X^Y(x, 0) = \text{Suc}(0) = 1$$

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

- (d) Soit la fonction prédécesseurs tel que:

$$\text{Pred} \in \mathbb{N}^1$$

$$\text{Pred}(0) = 0 = 0$$

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

- (e) Soit la fonction soustraction tel que:

$$X - Y \in \mathbb{N}^2$$

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

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

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

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

$$\text{sg}(0) = 0 = 0$$

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

- (g) Soit la fonction  $X > Y$  tel que :

$$X > Y \in \mathbb{N}^2$$

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

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

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

Soit la fonction  $X \geq Y$  tel que :

$$X \geq Y \in \mathbb{N}^2$$

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

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

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

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

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) Faisons une preuve par récurrence

$$A_0(n) = \text{Suc}(n) = n + 1$$

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 récurrence, 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 récurrence, 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  $A_{m+1}(n) > n$

Maintenant, 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 récurrence:

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

Ainsi

$$A_{m+1}(n) \geq n + 1$$

Donc

$$A_{m+1}(n+1) > n+1$$

On peut 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  $A_m(A_{m+1}(n-1)) \geq A_m(n)$

On peut 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 récurrence, 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.