TD HLIN612 Calculabilité/Complexité

Année 2018-19

Version 1.1

Université de Montpellier
Place Eugène Bataillon
34095 Montpellier Cedex 5

Rodolphe Giroudeau et Jean-Claude König
161, rue Ada
34392 MONTPELLIER Cedex 5
TEL : 04-67-41-85-40
Mail : $\{rgirou, konig\}$@lirmm.fr

Calculabilité/Complexité
TD – Séance n^o 1

---

1 Rappel

Exercice 1 – Une certaine idée de la complexité

Soit la fonction C suivante :

```
int pf(int x)
    {
    int y=pg(x)
    return ph(y)
    }
```

1. Exprimer la complexité de pf en fonction de la complexité de ph, celle de pg et de la fonction g, calculé par pg.
2. Quelle est la complexité du calcul de pf si pg est de complexité $O(n^4)$, ph de complexité linéaire et si $g(n) < n^2$ (g étant la fonction calculée par pg) ?
3. Si $pg(n)$ s'exécute en temps exponentiel (par exemple en $O(2^n)$), que peut-on en déduire sur l'existence d'une procédure C qui calcule f (la fonction calculée par le procédure pf) en temps polynomial ?
4. A quelle condition le calcul de pf se fait-il en temps polynomial ?
5. Soit le calcul de Fibonacci en utilisant directement la formule de récurrence :$f_0 = 1, f_1 = 1, n > 1, f_n = f_{n-1} + f_{n-2}$. Montrons que le nombre d'additions nécessaires pour faire le calcul est compris entre $\sqrt{2}^n$ et 2^n ?
6. Comment améliorer pour que ce nombre soit en $O(n)$? Peut-on en déduire qu'il existe un algorithme qui calcule f_n avec un nombre d'additions polynomial par rapport à la taille de la donnée ? Pourquoi ?
7. Questions difficiles : comment calculer f_n avec un nombre d'additions et de multiplications polynomial par rapport à la taille de la donnée ? Peut-on trouver un algorithme qui s'exécute en temps polynomial par rapport à la taille de la donnée ?

2 Calculabilité

Exercice 2 – Codage de couples d'entiers

Soit $Rang : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ tel que $Rang(x, y) = \frac{(x+y)(x+y+1)}{2} + x$

1. Donner une version récursive de la fonction $Rang$.

2. Donner la fonction inverse.
3. Appliquer les fonctions sur quelques exemples.

Exercice 3 – Codage de triplets

Soit c la fonction de codage pour les couples d'entiers vue en cours.

1. Soit h la fonction de codage pour les triplets définie par $h(x, y, z) = c(c(x, y), z)$. Quel est le doublet codé par 67 ? Quel est le triplet codé par 67 ?
2. le couple (z, t) succède au couple (x, y) si $c(z, t) = c(x, y) + 1$. Ecrire la fonction successeur qui prend en paramètre un couple et retourne le couple successeur.

Exercice 4 – Codage (suite)

Pour coder les listes d'entiers peut-on :

1. Faire la somme des entiers de la liste, et à somme égale prendre l'ordre lexicographique ?
2. Faire comme pour les mots : prendre les listes les plus courtes et à égalité de longueur l'ordre lexicographique ?

Exercice 5 – Codage (suite et fin)

Soit la fonction f suivante de $\mathbb{N}^* \to \mathbb{N}$:

$$\begin{array}{rcl} f(n) & = & k \text{ si } n = 2^k \\ f(n) & = & f(n/2) \text{ si } n \text{ pair et n'est pas une puissance de 2} \\ f(n) & = & f((3n+1) \text{ sinon} \end{array}$$

Nous appelons $A_i = \{x | f(x) = i\}$.

1. Donner quelques éléments de A_4.
2. Donner un algorithme qui prend i en paramètre et qui affiche tous les éléments de A_i.
3. Donner un algorithme qui affiche $A_1 \cup A_2$.
4. Donner un algorithme qui affiche $A_4 \cup A_6$.

Exercice 6 – Diagonalisation

1. Soit une suite quelconque d'ensembles $E_i \subset \mathbb{N}$. Construire un ensemble qui n'appartient pas à cette suite (en vous inspirant de la diagonalisation).
2. Que pouvons-nous conclure sur l'ensemble des sous-ensembles de $\mathbb{N}$?

Exercice 7 – Paradoxe

Montrer que les problèmes suivants engendrent un paradoxe

1. Le conseil municipal d'un village vote un arrêté municipal qui enjoint à son barbier (masculin) de raser tous les habitants masculins du village qui ne se rasent pas eux-mêmes et seulement ceux-ci.

2. Un crocodile s'empare d'un bébé et dit à la mère : « si tu devines ce que je vais faire, je te rends le bébé, sinon je le dévore. »
   En supposant que le crocodile tienne parole, que doit dire la mère pour que le crocodile rende l'enfant à sa mère ?
   Une réponse usuelle de la mère est : « Tu vas le dévorer ! »

Exercice 8 – Ensemble fini/infini

Un ensemble est fini si on ne peut pas le mettre en bijection avec une partie stricte de lui-même. Il est infini sinon.

Montrer que l'ensemble des entiers est infini.

Exercice 9 – Taille des ensembles

Soit E un ensemble, et soit $\mathcal{P}(E)$ l'ensemble des parties de E. On a $|E| < |\mathcal{P}(E)|$.

Pour montrer ceci, on suppose qu'il existe une bijection.

Exercice 10 – Une preuve incorrecte

Nous considérons la fonction suivante :

Algorithm 1 La fonction de Collatz

```
while n ≠ 1 do
  if n = 0 mod 2 then
    n := n/2
  else
    n := 3 × n + 1
  end if
end while
```

Actuellement nous ne savons pas si cette fonction termine $\forall n$.

Est-ce que vous êtes d'accord avec la preuve suivante :

« Si le problème de l'arrêt était décidable il suffirait de l'appliquer à ce programme pour savoir si son exécution s'arrête. Or, on ne sait pas si son exécution s'arrête. D'où la contradiction »

Exercice 11 – Calculabilité

Soit $f : \mathbb{N} \to \{0, 1\}$ une fonction totale non calculable.

1. Rappeler la définition d'une fonction totale et d'une fonction non calculable.
2. Construire une fonction $g : \mathbb{N} \to \mathbb{N}$ totale, croissante et non calculable à partir de f.

Exercice 12 – Calculabilité

Montrer que l'inverse d'une fonction f calculable et bijective est calculable.

Exercice 13 – Calculabilité

Montrer qu'une fonction f totale $\mathbb{N} \to \mathbb{N}$ est calculable si et seulement si son graphe

$$G = \{(x, f(x) | x \in \mathbb{N}\}$$

est décidable.

Exercice 14 – Calculabilité

Soient E un ensemble et ϕ une fonction telle que $\phi(n)$ est égale au nombre d'éléments de E strictement inférieur à n.

1. Montrer que ϕ est calculable si et seulement si E est décidable.

Exercice 15 – Calculabilité

En vous inspirant du théorème de Rice, donnez le prédicat (indécidable) et la fonction contradictoire qui prouve par l'absurde le résultat d'indécidabilité pour chacun des exemples suivants : on ne peut décider si une procédure calcule

1. une fonction totale
2. une fonction injective
3. une fonction croissante
4. une fonction à valeurs bornées

Exercice 16 – Calculabilité

Soit E l'ensemble $val(f)$ où f est calculable partielle.

1. Montrer que E est récursivement énumérable (inspirez-vous du fait que l'arrêt en t unités de temps est décidable)

Exercice 17 – Calculabilité

Soit f une fonction calculable, un ensemble B et son image réciproque par f, A :

$$A = f^{-1}(B) = \{x | f(x) \in B\}$$

1. Rappeler la définition d'un ensemble décidable et d'un ensemble récursivement énumérable.
2. A-t'-on B décidable implique A décidable ?
3. A-t'-on B récursivement énumérable implique A récursivement énumérable ?

Exercice 18 – Calculabilité

1. Montrer qu'un ensemble énuméré par une fonction calculable strictement croissante f est décidable.
2. En déduire que tout ensemble récursivement énumérable non décidable contient un sous-ensemble infini et décidable.

Exercice 19 – Calculabilité

1. Montrer que tout ensemble récursivement énumérable peut-être énuméré par une fonction sans répétition.

Exercice 20 – Calculabilité

Soient A et B deux ensembles décidables :

1. Est-on sûr que le complémentaire de A est décidable ?

2. Est-on sûr que l'union de A et B est décidable ?
3. Est-on sûr que l'intersection de A et B est décidable ?
4. Même question en remplaçant décidables par récursivement énumérables.

Exercice 21 – Calculabilité

1. soit A un ensemble décidable de couples d'entiers. Montrer que la projection de A à savoir $E = \{x | \exists y \text{ tel que } (x, y) \in A\}$ est récursivement énumérable.
2. Montrer que réciproquement tout ensemble récursivement énumérable est la projection d'un ensemble décidable.

3 Complexité

Exercice 22 – Certificat

On peut savoir si n peut s'écrire comme le produit de deux nombres premiers et on connait un algorithme en $O(\sqrt{n})$. Peut-on en déduire que ce problème est dans P ? Quel serait l'impact si ce problème était dans P ?

Exercice 23 – Puissance de calcul

Tous les 4 ans la puissance des machines est multipliée environ par 8. Vous avez deux algorithmes A et B l'un dont le temps d'exécution est proportionnel à n^3 et l'autre dont le temps d'exécution est proportionnel à 2^n. Avec les deux algorithmes vous traitiez un problème de taille $n = 10$ en $1s$, il y a 40 ans. Quelle est la taille des problèmes que vous êtes capables de traiter aujourd'hui avec chacun des deux algorithmes en $1s$?.

Exercice 24 – Optimisation versus décision

Soit le problème du stable de taille k :

STABLE (Stable)
Entrée : Un graphe non orienté $G = (X, E)$
Question : Existe-t'il un stable (c'est à dire un sous-ensemble de sommets tel que deux sommets de ce sous-ensemble ne soient jamais reliés par une arête) de taille k

et sa version optimisation

MAX STABLE (MaxStable)
Entrée : Un graphe non orienté $G = (X, E)$
Question : Trouver un stable (c'est à dire un sous-ensemble de sommets tel que deux sommets de ce sous-ensemble ne soient jamais reliés par une arête) de taille maximum

1. Montrer que S'il existe un algorithme polynomial qui résout le problème de stabilité maximum alors la version décisionnelle est résoluble, lui aussi, en temps polynomial.
2. Montrer que s'il existe un algorithme qui résout le problème de stable de taille k en temps polynomial alors le problème de stabilité maximum est résoluble, lui aussi, en temps polynomial.

Exercice 25 – 2-SATISFAISABILITÉ

1. Montrer en calculant le nombre de clauses créées et le nombre de variables ajoutées que la réduction de SATISFAISABILITÉ à 3-SATISFAISABILITÉ vue en cours est bien polynomiale.
2. Montrer que 2-SATISFAISABILITÉ peut-être résolu est temps polynomial.

Exercice 26 – Autour de SATISFAISABILITÉ

NON ÉGAL SATISFAISABILITÉ (NAESAT)
Entrée : Etant donné une formule conjonctive ϕ sur n variables et m clauses
Question : Existe-t-'il une affectation de valeurs de vérité aux variables qui satisfasse ϕ tel que chaque clause à une littéral à vrai et un à faux ?

Montrer que NON ÉGAL SATISFAISABILITÉ est $\mathcal{NP}-complet$. La preuve se fera à partir de SATISFAISABILITÉ

Exercice 27 – Autour de SATISFAISABILITÉ (suite)

COUPE MAXIMUM (CUT)
Entrée : Soit $G = (V, E)$ un graphe non orienté, $k \in \mathbb{N}$
Question : Existe t'il une partition de sommets en deux sous-ensembles V_1 et V_2 tel que le nombre d'arêtes entre V_1 et V_2 est k ?

Réduire NON ÉGAL 3-SATISFAISABILITÉ à COUPE MAXIMUM. Conclure.

Remarque : vous verrez en master que coupure min est polynomiale même si les arêtes ont des poids.

Exercice 28 – Problème de la coloration

Montrer que 3-coloriable est $\mathcal{NP}$-complet (réduction à partir de 3-SATISFAISABILITÉ).

Exercice 29 – VOYAGEUR DE COMMERCE

VOYAGEUR DE COMMERCE (TSP)
Entrée : Un ensemble de m villes X, un ensemble de routes entre les villes E. Une fonction de coût $v : E \to \mathbb{N}$ où $v(x, y)$ est le coût de déplacement de x à y, $k \in \mathbb{N}$.
Question : Existe-il aucun cycle Hamiltonien de distance inférieure ou égale à k ?

Montrer que VOYAGEUR DE COMMERCE est $\mathcal{NP}$-complet. (La preuve se fait à partir de CYCLE HAMILTONIEN). Qu'en est t'il si on autorise l'inégalité triangulaire $\forall i, j, k, c_{ik} \leq c_{ij} + c_{jk}$?

Exercice 30 – RECOUVREMENT DE SOMMETS

On veut montrer que le problème RECOUVREMENT DE SOMMETS est $\mathcal{NP}$-complet. La preuve se fera à partir de 3-SATISFAISABILITÉ .

Aide pour la transformation polynomiale : Considérons les variables $x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n$ et n arêtes $(x_i, \bar{x}_i), \forall i = 1, \ldots, n$. Nous considérons m triangles constitués des littéraux. Pour une clause C_i nous notons $c_{i_1}, c_{i_2}, c_{i_3}$ et nous relions le sommet x_i à un sommet d'un triangle noté $C_{j_k}, k = 1, 2, 3$ si la variable x_i apparaît dans la clause C_j à la position k. littéraux

Exercice 31 – RECOUVREMENT DE SOMMETS (suite)

Montrer que le RECOUVREMENT DE SOMMETS reste $\mathcal{NP}$-complet même si tous les sommets sont de degrés pairs.

Exercice 32 – Réductions autour de Hamiltonisme

Montrer que les problèmes sont tous NP-complets si l'un d'eux l'est :

1. CYCLE HAMILTONIEN dans un graphe non orienté
2. CHAÎNE HAMILTONIENNE dans un graphe non orienté
3. CIRCUIT HAMILTONIEN dans un graphe orienté
4. CHEMIN HAMILTONIEN dans un graphe orienté
5. CYCLE HAMILTONIEN dans un graphe biparti non orienté
6. CHAÎNE HAMILTONIENNE dans un graphe biparti non orienté

Exercice 33 – Mètre du charpentier

Montrer que le problème du mètre de charpentier est un problème $\mathcal{NP}$-complet

MÈTRE DU CHARPENTIER (MC)
Entrée : La longueur de l'étui L et des segments l_i (i de 1 à n).
Question : Peut-on plier le mètre pour qu'il rentre dans l'étui ?

Exercice 34 – Algorithmes pseudo-polynomiaux

Donner deux algorithmes pseudo-polynomiaux pour résoudre la problème du sac-à-dos dont le temps d'exécution est proportionnel au produit d'un polynôme en n (nombre d'objet) et au volume du sac à dos (pour l'un des algorithme) et au poids de l'objet le plus lourd (pour l'autre).

Exercice 35 – Autour du problème de la 2-PARTITION

Nous rappelons que le problème suivant est NP-complet.

2-PARTITION (Partition)
Entrée : Etant donnés n objets a_i $(1 \leq i \leq n)$ de poids entiers $p(a_1),\ p(a_2), \ldots,\ p(a_n)$ de somme $2P$.
Question : Est-il possible de les partager en deux sous-ensembles de même poids total P ?

Montrer que les problèmes suivants sont NP-complets.

1. 2-PARTITION À VALEURS PAIRES (Partition à valeurs paires)
   Entrée : Etant donnés n objets a_i $(1 \leq i \leq n)$ de poids entiers à valeurs paires $p(a_1),\ p(a_2), \ldots,\ p(a_n)$ de somme $2P$.
   Question : Est-il possible de les partager en deux sous-ensembles de même poids total P ?
2. 2-PARTITION AVEC NOMBRE PAIRE (Partition avec nombre paire)
   Entrée : Etant donnés $2n$ objets a_i $(1 \leq i \leq 2n)$ de poids entiers $p(a_1),\ p(a_2), \ldots,\ p(a_{2n})$ de somme $2P$.
   Question : Est-il possible de les partager en deux sous-ensembles de même poids total P ?
3. 2-PARTITION ÉQUILIBRÉ (Partition équilibré)
   Entrée : Etant donnés $2n$ objets a_i $(1 \leq i \leq 2n)$ de poids entiers $p(a_1),\ p(a_2), \ldots,\ p(a_{2n})$ de somme $2P$.
   Question : Est-il possible de les partager en deux sous-ensembles I et $\bar{I}$ de même poids total P tel que $|I| = n$?

4. 2-Partition (Partition impair/paire)
   Entrée : Etant donnés $2n$ objets a_i $(1 \leq i \leq 2n)$ de poids entiers $p(a_1),\ p(a_2), \ldots,\ p(a_{2n})$ de somme $2P$.
   Question : Est-il possible de les partager en deux sous-ensembles I et $\bar{I}$ de même poids total P tel que $|I| = n$?
5. 3-Partition (Partition à trois)
   Entrée : Etant donnés n objets a_i $(1 \leq i \leq n)$ de poids entiers à valeurs paires $p(a_1),\ p(a_2), \ldots,\ p(a_n)$ de somme $2P$.
   Question : Est-il possible de les partager en deux sous-ensembles de même poids total P ?

Exercice 36 – Programmation dynamique : algorithme pseudo-polynomial

1. Sur le problème de la partition :

   2-Partition (Partition)
   Entrée : Etant donnés n objets a_i $(1 \leq i \leq n)$ de poids entiers $p(a_1),\ p(a_2), \ldots,\ p(a_n)$ de somme $2P$.
   Question : Est-il possible de les partager en deux sous-ensembles de même poids total P ?

   (a) Nous allons plonger le problème dans une classe de problèmes dépendant de paramètres et liés par une relation de récurrence. On considère deux entiers i et j avec $1 \leq i \leq n$ et $0 \leq j \leq P$, et l'expression booléenne $T(i,j)$: « étant donnés les i premiers éléments de la famille, il existe un sous-ensemble de ces i éléments de poids j ». On remplit alors ligne par ligne un tableau A, qui contient les valeurs de T dont les colonnes sont indicées par j et les lignes par i.

      i. Donner la formule qui lie la ligne i et $i-1$ et $p(a_i)$.
      ii. Illustrer ce principe avec les données suivantes : $n = 6,\ p(a_1) = 5, p(a_2) = 9,\ p(a_3) = 3,\ p(a_4) = 8,\ p(a_5) = 2,\ p(a_6) = 5$.
      iii. Comment avec le tableau rempli obtient-on les éléments de la partition ?

   (b) Donner la complexité de cet algorithme ?
   (c) **Facultatif** Faire des jeux d'essais.

2. Le problème du sac à dos :
   Le problème du sac à dos est une généralisation du problème de la partition, on peut s'inspirer de ce qui a été fait plus haut. Soit (P) le problème du sac à dos (généralisé) qui s'écrit sous la forme

$$\begin{cases} \max z = \sum_{j=1}^{n} u_j.x_j \\ \sum_{j=1}^{n} v_j.x_j \leq V \\ x_j \text{ entier pour tout } j \end{cases}$$

   Nous supposerons que tous les coefficients u_j et v_j sont des entiers strictement positifs. On peut considérer (P) comme un cas particulier de la famille des problèmes $(P_{k,v})$ définis pour $1 \leq k \leq n$ et $0 \leq v \leq V$ par :

$$\begin{cases} \max z = \sum_{j=1}^{k} x_j.u_j \\ \sum_{j=1}^{n} v_j.x_j \leq v \\ x_j \text{ entier pour tout } j \text{ compris entre 1 et } k \end{cases}$$

   On a effet $(P) = P_{n,V}$. On a ainsi réalisé le plongement ; il nous reste à exhiber les relations de récurrence permettant de résoudre les problèmes $(P_{k,v})$ de proche en proche. Appelons $z_{k,v}$ le maximum de $(P_{k,v})$. En posant $z_{0,v} = 0$ pour tout v compris entre 0 et V, il vient

$$z_{k,v} = \max_{x_k \in X_k} \{z_{k-1,v-v_k x_k} + u_k x_k\}$$

   en posant

$$X_k = \{x_k \text{ entier tel que } 0 \leq x_k \leq v/v_k\}$$

   (a) Justifier les formules proposées ci-dessus.
   (b) Illustrer le principe sur un exemple de votre choix.
   (c) Quelle est la complexité de cet algorithme ?

3. Le problème du voyageur de commerce :
   Instance Etant donné un graphe complet valué à n sommets (à chaque arête est associée une longueur),
   Question : on cherche un cycle Hamiltonien de longueur minimum de ce graphe, un cycle Hamiltonien étant un cycle qui passe une fois et une seule fois par chaque sommet et la longueur d'un cycle étant la somme des longueurs des arêtes le constituant.

   (a) Soit une instance du problème du voyageur de commerce, c'est-à-dire la donnée d'une matrice $n \times n$ des poids P d'un graphe $G = (X, E)$ à n sommets, supposés numérotés de 0 à $n-1$. Pour toute partie S de X contenant le sommet 0, et tout sommet i non dans cette partie, on considère le problème suivant : déterminer une plus courte chaîne du sommet 0 au sommet i passant une fois et une seule fois par tout sommet de S et n'utilisant pas de sommet non dans S en dehors de i : appelons $C(S,i)$ la longueur d'une telle chaîne. Si S ne contient que le sommet 0, on voit qu'on a, pour tout $i \neq 0$; $C(S,i) = P(0,i)$ ce qui nous sera utile pour initialiser la récurrence. Sinon, on a

$$C(S,i) = \min_{k \in S-\{0\}} \{C(S-\{k\},k) + P(k,i)\} \text{ pour } i \notin S$$

   On a donc établi une relation de récurrence sur la taille de S liant les $C(S,i)$. Quant à notre problème lui-même, il suffit pour le résoudre de déterminer la valeur de $\min_{i \in X-\{0\}} \{C(X-\{i\},i) + P(i,0)\}$. Nous avons bien ici appliqué les principes de la méthode, en plongeant le problème dans une classe plus générale que nous résolvons en utilisant la relation de récurrence.

   (a) Illustrer la méthode en utilisant la figure 1.
   (b) Donner la complexité de la méthode. Pour cela évaluer le coût si $|S| = l$. Quel est l'intervalle pour l ?

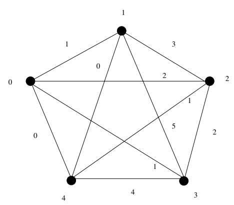

FIGURE 1 – *Un graphe complet à* 5 *sommets*