TD HLIN612 Calculabilité/Complexité

Année 2020-21

Version 1.3

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

Université de Montpellier
Licence HLIN612

13 janvier 2021
Durée: 13,5h

Calculabilité/Complexité
TD – Séance n^o 1

---

1 Calculabilité

1.1 Variations sur le codage

Exercice 1 – Codage de couples d'entiers

Soit $Rang : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ 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. Calculer $Rang(4, 5)$. Donner le couple pour lequel la valeur du codage est 8.

Exercice 2 – 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 3 – 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 d'abord et à égalité de longueur l'ordre lexicographique ?

Exercice 4 – Codage (suite)

Proposer un codage pour les nombres rationnels.

Exercice 5 – Codage (suite)

On ordonne les listes de la façon suivante :

$$\sigma(l) = \text{ somme des entiers de la liste} + \text{longueur de la liste}$$

Puis à valeur de σ égale on ordonne dans l'ordre lexicographique.

On note U_k l'ensemble des liste l telles que $\sigma(l) = k$ et $u_k = |U_k|$.

1. Donner les ensembles $U_i, i = 0, \ldots, 4$.

2. Montrer que $u_k = 2^k, \forall k \geq 1$.
3. Quelle est la première liste de $U_k, \forall k \in \mathbb{N}^*$ et la dernière ?
4. Donner la fonction de codage en version itérative et récursive (resp. décodage).

Exercice 6 – Codage (suite et fin)

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

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

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

1. Donner quelques éléments de $A_i, \forall i \in \{1, 2, 3, 4, 5, 6\}$.
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 $A_4 \cup A_6$.

1.2 Dénombrabilité

Exercice 7 – 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 8 – Diagonalisation (pas prioritaire)

On considère l'ensemble U des suites $(u_n)_{n \in \mathbb{N}}$ à la valeurs dans $\{0, 1\}$,c'est à dire $\forall n \in \mathbb{N}$; montrer que U n'est pas dénombrable.

Exercice 9 – 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 10 – 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 11 – 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 de E dans $P(E)$.

Exercice 12 – Une preuve incorrecte

Nous considérons la fonction suivante donné par l'algorithme 1 :

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 13 – 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 14 – Calculabilité

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

Exercice 15 – 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 16 – 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 17 – 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 18 – 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 19 – 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 20 – 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 21 – Calculabilité (pas prioritaire

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

Exercice 22 – 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 23 – Calculabilité (pas prioritaire

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

Exercice 24 – Concept de la réduction

Pour deux sous-ensembles A et B de $\mathbb{N}$, on dit que A se réduit à B (ce qu'on note $A \propto B$) si il existe une fonction totale (récursive) totale f telle que pour tout $x \in \mathbb{N}$, $x \in A$ ssi $f(x) \in B$.

1. Montrer que si B est décidable et $A \propto B$, alors A est décidable.
2. Montrer que si B est récursivement énumérable et $A \propto B$, alors A est récursivement énumérable .

Exercice 25 – Concept de la réduction (suite)

Pour réduire un problème A à un problème B, il suffit de montrer que la résolution de B permet de résoudre A à condition qu'une solution à B soit disponible.

Pour illustrer, supposons que A est le problème suivant : $A(n) =$ le plus petit nombre premier qui est plus grand n, et B le problème de décision $B = \{n | n$ is prime $\}$.

1. Donner pour quelques valeurs de n la valeur de $A(n)$.
2. Proposer une réduction du problème A au problème B.

Exercice 26 – Réduction

Montrer que les deux problèmes Problème du carré d'un entier et Problème de la multiplication se réduisent l'un à l'autre.

L'addition, la soustraction et la division soient des opérations autorisées.

Problème de la multiplication
Entrée : Soient $a \in \mathbb{N}$ et $b \in \mathbb{N}$.
Question : Peut-on multiplier a et b ?

Problème du carré d'un entier
Entrée : Soit $a \in \mathbb{N}$.
Question : Peut-on élevé au carré a ?

2 Complexité

Exercice 27 – Une certaine idée de la complexité

Soit la fonction C suivante :

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

1. 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) ?
2. Si ph s'exécute en temps polynomial, à quelle condition le calcul de pf se fait-il en temps polynomial ?
3. Si les hypothèses de la question précédente est vérifiée que peut-on en déduire si la fonction h calculée par ph se calcule en temps polynomial ?
4. 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 ?
5. 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 ?
6. 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 ?

Exercice 28 – Certificat

Si pour un problème Π vous avez un certificat polynomial pour une réponse positive et un certificat polynomial pour une réponse négative. Que pouvez-vous conclure ? Justifiez votre réponse On connait un algorithme simple en $O(\sqrt{n})$ pour savoir si un nombre n est premier. Peut-on en déduire que savoir si un nombre est premier.

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 29 – 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 30 – 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 31 – 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 32 – Autour de Satisfaisabilité

Non égal Satisfaisabilité (NAESAT)

Entrée : Etant donnée 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 33 – 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 34 – Problème de la coloration

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

Exercice 35 – 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 un 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 36 – 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 37 – Recouvrement de sommets (suite)

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

Exercice 38 – Réductions autour de Hamiltonisme (pas prioritaire)

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 39 – 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 40 – 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 41 – 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 pair)

   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 $\bar{I}$ tel que $\sum_{i \in I} a_i = \sum_{i \in \bar{I}} a_i$, avec un entier entre a_{2j-1} et a_{2j} appartenant à I ?

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 $3P$.

   Question : Est-il possible de les partager en trois sous-ensembles $I_1,\ I_2$ et I_3 de $\{1, \ldots, n\}$ de même poids total P ?

Exercice 42 – 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 ?

2. Le problème du sac à dos :

Nous considérons le problème du sac à dos sans répétition, c'est à dire les objets seront pris au plus une fois. Pour cela considérons, un tableau K à deux dimensions tel que $K[j, w]$ représente la valeur maximale que l'on peut stocker dans un sac de capacité w avec des objets $1, \ldots, j$.

(a) Donner les formules ;

(b) Illustrer le principe avec les données suivantes : $(w_1, v_1) = (1, 1); (w_2, v_2) = (2, 6); (w_3, v_3) = (5, 18); (w_4, v_4) = (6, 22); (w_5, v_5) = (7, 24)$ et $W = 12$.

(c) Comment retrouver la solution à partir du tableau ?

(d) Donner la complexité en temps et en mémoire