TD 02 – Master théorème, logique propositionnelle, nombre d'or

Exercice 1. *Multiplication de Karatsuba*

L'addition et la multitplications de grands nombres ne sont pas réalisées en temps constant. Étant donnés deux nombres à n chiffres, on apprend à l'école primaire un algorithme pour les multiplier en temps $\Theta(n^2)$, qui utilise un algorithme pour les additionner en temps $\Theta(n)$.
Dans cet exercice, nous allons tâcher de multiplier avec une meilleure complexité asymptotique. On voit les nombres en base 10 comme des mots sur l'alphabet $\{0, 1, 2, \ldots, 9\}$.

1. Que sont w et x tels que le nombre u se décompose en $u = 10^s w + x$ avec $0 \leq s \leq n-1$ et $x < 10^s$?
2. Quelle est la complexité de multiplier par 10^s un nombre w donné en base 10 ?

Dans la suite de cet exercice, multiplier par 10^s sera réalisé avec cette complexité.

3. Pour quelle valeur de s les tailles de w et x sont-elles égales ?

Dans la suite de cet exercice, s aura cette valeur.

4. Utiliser les expressions $u = 10^s w + x$ et $v = 10^s y + z$ pour décomposer la multiplication uv en une somme de 4 multiplications sur des entiers à s chiffres.
5. Combien d'autres opérations sont nécessaires pour calculer uv avec cette formule ? Quelle est leur complexité ?
6. En déduire un algorithme de type diviser-pour-régner, permettant de multiplier deux entiers u, v à n chiffres, et évaluer sa complexité à l'aide du Master théorème.
7. Expliquer comment calculer uv en temps linéaire à partir des trois valeurs :
$$(w + x)(y + z) \qquad wy \qquad xz$$
8. En déduire un nouvel algorithme de type diviser-pour-régner réalisant seulement trois appels récursifs, et évaluer sa complexité à l'aide du Master théorème.

Exercice 2. *Rappels de logique propositionnelle*

Une formule propositionnelle est en forme normale conjonctive (FNC) lorsqu'elle est une conjonction de disjonctions de littéraux. Un littéral est une variable ou sa négation, et une clause est une disjonction de littéraux. Donc une FNC est une conjonction de clauses.
Les variables seront en général notées $x_1, x_2, x_3, \ldots$ ou $a, b, c, \ldots$ et leur négation $\bar{a}$ ou $\neg a$.

1. Donner une formule en FNC sur 4 variables, avec 4 clauses de tailles respectives 1, 2, 3, 4. Est-elle satisfaisable ? Si oui, donner un modèle.
2. On considère qu'une clause ne contient jamais une variable et sa négation. Pourquoi ?
3. Donne une formule en FNC sur 2 variables, avec des clauses binaires (de taille 2), qui n'est pas satisfaisable. Même question sur 3 variables avec des clauses ternaires (de taille 3).
4. Est-il possible de définir une FNC non-vide qui soit une tautologie ?
5. Donne une FNC sur 3 variables, avec des clauses de taille 3, qui admet exactement 4 modèles. Est-ce possible avec une ou plusieurs clauses de taille 2 ? de taille 1 ?

Nous allons modéliser le principe des tiroirs (*pigeonhole principle*) comme une formule CNF. Ce principe affirme que si n chaussettes sont rangées dans m tiroirs et que $n > m$, alors il y a un

tiroir qui contient au mons deux chaussettes. En termes mathématiques, si E et F sont deux ensembles finis et que $|E| > |F|$, alors il n'existe pas d'application injective de E dans F.
Pour cette modélisation, nous utiliserons mn variables de la forme $p_{i,j}$, indiquant si la chaussette $i \in \{1, \ldots, n\}$ se trouve dans le tiroir $j \in \{1, \ldots, m\}$ ($p_{i,j} = \top$) ou non ($p_{i,j} = \bot$). Pour n et m donnés, nous souhaitons construire une formule modélisant les rangements des n chaussettes dans les m tiroirs, qui vérifient que :

— chaque chaussette est rangée dans un tiroir, et
— chaque tiroir contient au plus une chaussette.

6. Donner une formule en FNC pour $n = 2$ et $m = 2$. Est-elle satisfaisable ?

7. Donner une formule en FNC pour $n = 3$ et $m = 2$. Est-elle satisfaisable ?

8. Généraliser la formule précédente pour tous $n, m \in \mathbb{N}_+$, à l'aide des notations $\bigwedge_{i=1}^{n}$ et $\bigvee_{i=1}^{n}$.

9. Appliquer l'algorithme de Quine (backtracking) sur la formule pour $n = 3$ et $m = 2$.

10. Appliquer l'algorithme DPLL (avec propagation unitaire, élimination des littéraux purs, et l'heuristique de branchement de votre choix) sur cette même formule.

Exercice 3. *Fibonacci et nombre d'or*

Appliquons la méthode de résolution d'une équation de récurrence linéaire homogène, afin de déterminer la complexité de l'algorithme récursif suivant de calcul de la suite de Fibonacci.

```
let rec fibonacci n = match n with
| 0 -> 0
| 1 -> 1
| _ -> (fibonacci (n-1)) + (fibonacci (n-2));;
```

1. Quelle équation de récurrence exprime la complexité $T(n)$ de cet algorithme ?

2. Quelles sont les deux conditions initiales données par l'algorithme ?

3. Quelle est l'équation caractéristique de cette récurrence ?
(Il faut supposer $T(n) = x^n$ puis divier par x^{n-2}.)

4. Quelles sont les deux solutions y et z de cette équation ?

Toute combinaison linéaire de ces deux solutions $f(n) = y^n$ et $g(n) = z^n$, c'est-à-dire toute fonction $h(n) = sf(n) + tg(n)$ avec $s, t \in \mathbb{R}$, est également solution de la récurrence sur $T(n)$.

5. Démontrer l'affirmation ci-dessus.

6. Avec les conditions initiales, obtenir deux équations permettant de déterminer les deux inconnues s et t dans l'expression de $h(n)$.

7. Quelle est la complexité asymptotique de l'algorithme `fibonacci` ?