

Information, Calcul et Communication

Compléments de cours

Examen 1 2018 Q7

On considère la machine de Turing dont la table de transition est :

	0	1	ϵ
1	(2, ϵ , +)	(3, ϵ , +)	(4, ϵ , -)
2	(2, 0, +)	(2, 1, +)	(4, 0, -)
3	(3, 1, +)	(3, 0, +)	(4, 0, +)

Quel est l'état de la bande lorsque la machine s'arrête, si elle a démarré avec sa tête de lecture positionnée comme suit :

$\dots \epsilon$	1	1	0	1	0	1	1	0	$\epsilon \dots$
	↑								

$\dots \epsilon$	0	1	0	1	0	0	1	0	$\epsilon \dots$
									↑

Examen 1 2018 Q8

Sachant que « 3-SAT » est le nom d'un problème de décision célèbre pour les informaticien(ne)s, connu pour être dans NP, que peut-on en dire ?

- ▶ « 3-SAT » n'a pas de solution (algorithme de résolution) :
faux (en fait, tous les problèmes de NP ont des solutions, mais on ne sait pas à ce jour s'ils ont des solutions *efficaces*)
- ▶ « 3-SAT » n'est pas dans P :
On ne sait pas !
- ▶ On connaît des algorithmes efficaces pour résoudre toute instance de « 3-SAT » :
faux (cf ci-dessus ; et car sinon 3-SAT serait dans P, donc, à ce jour, on ne *connait* pas de tels algorithmes ; mais il est vrai qu'on ne sait pas s'il n'en existe pas...)
- ▶ Toute « solution » (instance positive) de « 3-SAT » est facilement vérifiable :
vrai (c'est la définition de NP)

Examen 1 2018 Q9

Supposons que l'on sache qu'un problème de décision « X » n'est pas dans NP.
Que peut-on en dire ?

- ▶ « X » est dans P : **faux**
- ▶ « X » est indécidable : **pas forcément**
- ▶ « X » est décidable et vérifier qu'une « solution » (instance positive) du problème « X » en est effectivement une prend un temps *au plus* polynomial par rapport à la taille de cette solution :
faux
- ▶ Soit « X » est indécidable, soit vérifier qu'une « solution » (instance positive) du problème « X » en est effectivement une prend un temps *plus que* polynomial par rapport à la taille de cette solution :
vrai

Examen 1 2018 Q10

Supposons que l'on connaisse un algorithme de complexité $4n\log(n) + 3n + 2$ permettant de résoudre un problème de décision « Y » portant sur des données de taille n .
Que peut-on en déduire ?

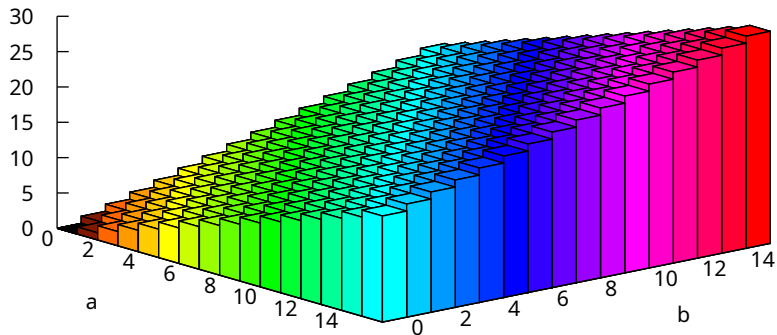
- ▶ « Y » n'est pas dans NP : faux
- ▶ « Y » est dans NP, mais pas dans P : faux
- ▶ « Y » est dans P : vrai

Leçon I.4 (Représentation de l'information) – Points clés

- ▶ nécessité d'une convention
on en a (vu cinq, mais) **retenu trois** :
 - ▶ entiers positifs (ou nul) : `unsigned int` (voir semaine 7 du cours de C++)
 - ▶ entiers relatifs : `int` (**complément à deux**)
 - ▶ décimaux, représentation à **virgule flottante** : `double`
- ▶ nombre de bits pour représenter K objets : $\lceil \log_2 K \rceil$
- ▶ domaine couvert
- ▶ précision (relative/absolue)
- ▶ comment utiliser les trois conventions ci-dessus

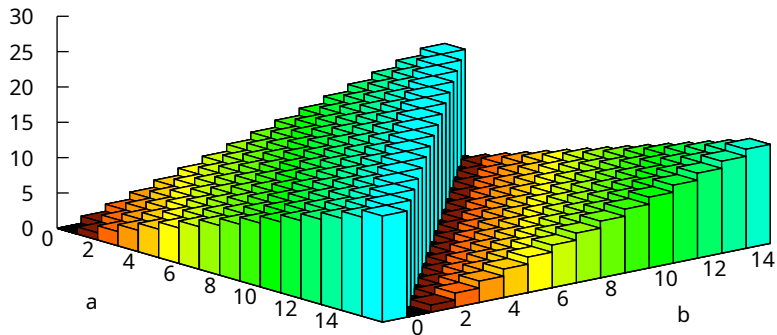
Leçon I.4 – Autre vue sur l'addition (sur 4 bits ici)

$\text{add}(a,b)$ en théorie



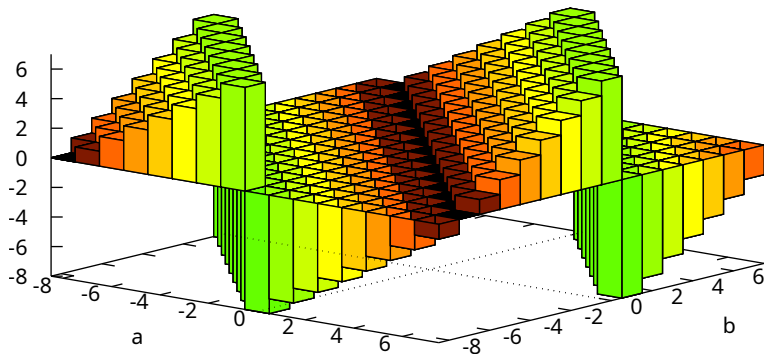
Leçon I.4 – Autre vue sur l'addition (sur 4 bits ici)

$\text{add}(a,b)$ sur 4 bits (non signé)



Leçon I.4 – Autre vue sur l'addition (sur 4 bits ici)

$\text{add}(a,b)$ sur 4 bits (signé complément à deux)



Leçon I.4 – Etude de cas 1

Que représente 10110101 ?

► **CONVENTION ???**

☞ faites avec les trois (dont : signe, exposant sur 3 bits et mantisse, dans cet ordre)

► entiers positifs :

$$128 + 32 + 16 + 4 + 1 = 181$$

► entiers négatifs :

opposé de 01001011 : -75

(on peut aussi faire $-(256 - 181)$)

► virgule flottante, signe (1 bit), exposant (3 bits), mantisse (4 bits) :

$$10110101 = 1 \quad 011 \quad 0101$$

$$= -2^{011} \times 1,0101$$

$$= -2^3 \times \left(1 + \frac{1}{4} + \frac{1}{16}\right)$$

$$= -(2^3 + 2^1 + 2^{-1})$$

$$= -10.5$$

Leçon I.4 ? – Etude de cas 2

Donnez une version récursive de :

écriture en binaire

entrée : $n \in \mathbb{N}$

sortie : *écriture binaire de n*

$L \leftarrow ()$ // *liste vide*

Répéter

Si n est pair

| $L \leftarrow 0 \oplus L$ // *ajouter 0 devant*

Sinon

| $L \leftarrow 1 \oplus L$ // *ajouter 1 devant*

| $n \leftarrow \lfloor \frac{n}{2} \rfloor$

Tant que $n > 0$

Sortir : L

Leçon I.2 – Etude de cas 2 – Solution

BinRec : écriture en binaire récursive

entrée : $n \in \mathbb{N}$

sortie : écriture binaire de n

$L \leftarrow ()$ // liste vide

Si $n > 1$

$L \leftarrow \text{BinRec}(\lfloor \frac{n}{2} \rfloor)$

Si n est pair

$L \leftarrow L \oplus 0$ // ajouter 0 à la fin

Sinon

$L \leftarrow L \oplus 1$ // ajouter 1 à la fin

Sortir : L

BinRec : écriture en binaire récursive

entrée : $n \in \mathbb{N}$

sortie : écriture binaire de n

$L \leftarrow ()$ // liste vide

Si $n > 1$

$L \leftarrow \text{BinRec}(\lfloor \frac{n}{2} \rfloor)$

Sortir : $L \oplus (n \bmod 2)$

BinRec : écriture en binaire récursive

entrée : $n \in \mathbb{N}$

sortie : écriture binaire de n

Si $n \leq 1$

 Sortir : (n)

Sortir : $\text{BinRec}(\lfloor \frac{n}{2} \rfloor) \oplus (n \bmod 2)$