

Informatique Générale

Histoires d'algorithmes et de machines

des cailloux aux circuits intégrés

Jacques Farré

Jacques.Farre@unice.fr





Origine des algorithmes

- Algorithme : énoncé, dans un langage bien **défini**, d'une **suite d'opérations** permettant de résoudre par calcul un problème en un **nombre fini** d'étapes
- Du nom du mathématicien, géographe et astronome perse al-Khuwārizmī (9^e siècle)
 - le mot algèbre provient de son livre sur la résolution des équations du second degré *Kitāb al-jabr wa'l-muqābalah (la transposition et la réduction)*, publié en 825
 - au 12^e siècle, le moine Adelard de Bath introduit le terme latin de algorismus (par référence à al-Khuwārizmī) ; ce mot donne algorithme en français au 16^e siècle
- En fait, les algorithmes remontent à l'aube de la civilisation (Sumer, Babylone)



statue de al-Khuwārizmī
à Université de Téhéran

Retour sur le système de numération babylonien

- Les babyloniens comptaient en base 60, avec des « chiffres » de 1 à 59 représentés par des symboles valant 1 () et 10 ()
 - 728 ($12 \times 60 + 8$) s'écrivait  (**qu'on notera 12;8**)
- Petits problèmes : l'absence du "zéro" (un espace plus grand) entre 2 chiffres ?), et de virgule décimale
- Donc  pouvait aussi représenter
 - 43680 ($12 \times 60^2 + 8 \times 60$),
 - ou $12 + 8/60$,
 - ou tout autre nombre de la forme $12 \times 60^n + 8 \times 60^{n-1}$ (seul le contexte permettait de déterminer l'ordre de grandeur du nombre)

Un algorithme babylonien

le problème à résoudre



- De nombreuses tablettes ont été retrouvées qui exposaient des problèmes et donnaient leur solution, par exemple (tablette exposée au Musée du Louvre à Paris) :
 - [Pour un rectangle, la somme] de la longueur et de la largeur est égale à la surface ; la longueur [L] est connue, quelle est la largeur [x] ?*
 - Donc, en algèbre moderne $L + x = L \times x$, par conséquent **$x = L / (L-1)$** (car $L = L \times x - x = (L-1) \times x$)
 - Il faut faire une division, ce qu'ils ne savaient pas faire
 - $a/b = a \times 1/b$, on peut faire une multiplication si on connaît $1/b$
 - Les babyloniens avaient des *tables d'inverses*, par exemple :
 - inverse de 2 : 30 $(1/2=30/60)$
 - inverse de 3 : 20 $(1/3=20/60)$
 - inverse de 9 : 6;40 $(1/9=6/60+40/3600)$ car $1/9=(1/3)^2 \Rightarrow$
 $(20/60)^2=400/3600=(360+40)/3600=6/60+40/3600$

Un algorithme babylonien

la solution proposée

- La solution pour calculer $x = L / (L-1)$ est la suivante:

telle que donnée sur la tablette	correspondance en langage informatique
<i>fais 2 copies du paramètre (c'est à dire la longueur L) retire 1 de l'une d'entre elles forme son inverse et multiplie par l'autre copie telle est la solution</i>	$L1 := L2 := L$ $L1 := L1 - 1$ $l := 1 / L1$ $x := l * L2$ afficher x

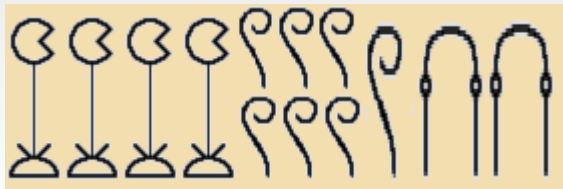
- C'est de l'algèbre avant la lettre !
- Mais la division comme dans un ordinateur repose sur une autre méthode, qu'on trouve déjà chez les égyptiens et qu'on verra un peu plus tard

Un algorithme babylonien application numérique

- Soit $L = 3$; Donc $L - 1 = 2$ et $x = 3/2$
- Pour les babyloniens :
Inverse de 2 = 30 (en fait 30/60, soit $\frac{1}{2}$)
Donc, $L \times 1/(L - 1) = 3 \times 30 = 90 = 60 + 30$
Par conséquent $x = 1;30$ (soit $1 + \frac{1}{2}$ dans notre système décimal
puisque'on a multiplié en fait par 30/60 : *il faut rétablir l'exposant*)
 - Vérifications :
 - $L + x = 3 + 1;30 = 4;30$ (soit $4 + \frac{1}{2}$ dans notre système décimal)
 - $L \times x = 3 \times 1;30 = 3;90 = 3;(60+30) = 4;30$
- Pour nous, c'est pareil à la base près :
Inverse de 2 = 5 (5/10 en fait)
Donc, $L \times 1/(L - 1) = 3 \times 5 = 15 = 10+5 = 1;5$ (donc 1,5 dans
notre système puisque'on a multiplié en fait par 5/10)
 - $L + x = 3 + 1;5 = 4;5$
 - $L \times x = 3 \times 1;5 = 3;15 = 3;(10+5) = 4;5$

Les premiers algorithmes connus (Égypte antique)

- Les anciens égyptiens avaient un système *additif* basé sur le décimal : ils avaient un hiéroglyphe pour chaque puissance de 10 jusqu'à 10^6 (considéré comme l'infini) :



4 fleurs de lotus (1000), 7 cordes (100),
2 anses de panier (10), donc 4720

- pour l'addition, c'est facile : on regroupe les mêmes symboles, et chaque fois qu'on en a 10, on les remplace par le symbole de la puissance de 10 supérieure
- Une machine mécanique additionne de la même manière : des roues avec 10 dents correspondent à chaque symbole égyptien
 - Quand une roue tourne, un cran situé entre la dent 9 et la dent 0 fait avancer la roue à sa gauche (c'est la retenue)

http://pagesperso-orange.fr/therese.eveilleau/pages/truc_mat/textes/pascaline.htm
donne une simulation de la *Pascaline* (comme déjà vu en TP)

Les premiers algorithmes connus (addition, Egypte antique)

On a donc l'algorithme suivant pour additionner A et B :

r := 0 (la retenue)

*pour chaque «chiffre» c dans {1,10,100...} tant que $\exists c' \geq c$ dans A ou B
soit n_A et n_B le nombre de ces chiffres respectivement dans A et B*

si $n_A + n_B + r < 10$

écrire $n_A + n_B + r$ chiffres c, et $r := 0$

sinon

écrire $n_A + n_B + r - 10$ chiffres c, et $r := 1$

fin pour

si $r = 1$

écrire le chiffre immédiatement supérieur au dernier c traité

C'est identique dans n'importe quelle base (et donc en base 2) en remplaçant les puissances de 10 ($10^0, 10^1, 10^2 \dots$) par $B^0, B^1, B^2 \dots$

Et pour la soustraction A - B ? Algorithme laissé en exercice

L'additionneur binaire

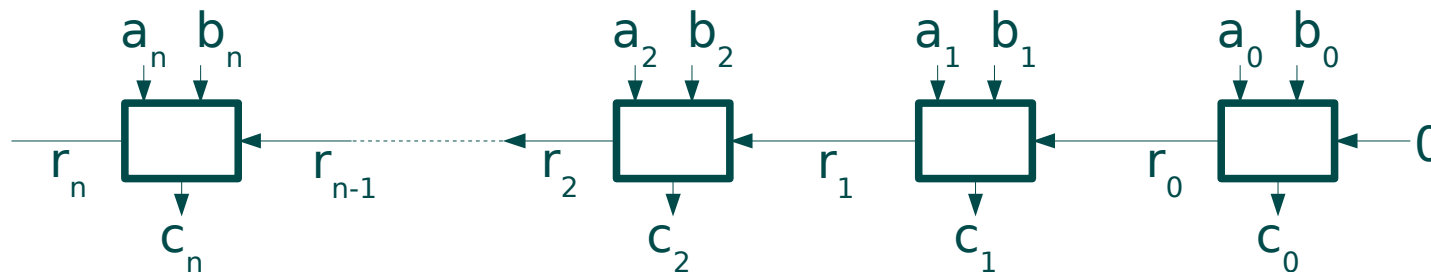
- Pour une machine calculant en binaire, il suffit d'utiliser des circuits nommés additionneurs tels que $a + b + r' = 2r + s$



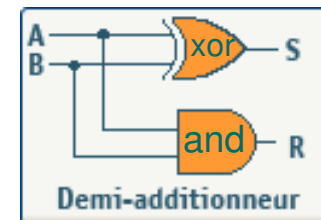
a	b	r'	s	r
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1

a	b	r'	s	r
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

- Il suffit ensuite de combiner ces additionneurs

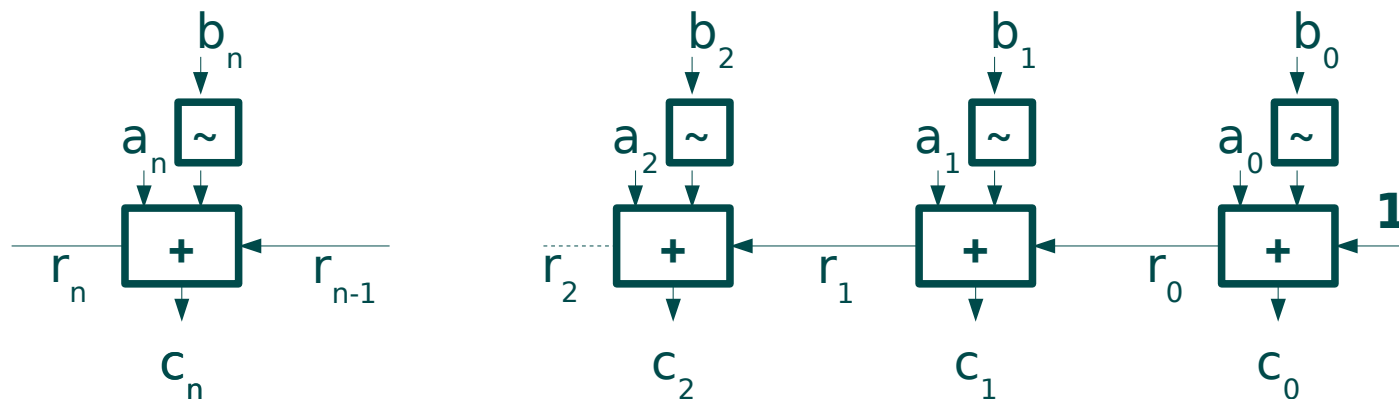


Vous verrez en électronique comment réaliser ce circuit comme une suite de $\frac{1}{2}$ additionneurs



La soustraction binaire signée

- L'avantage de la complémentation à 2 pour les nombres négatifs est qu'on peut additionner tous les bits, y compris le bit de signe, et qu'on obtient le bon résultat
- Ce qui permet d'utiliser un additionneur pour faire une soustraction : $22-17 = 22+(-17) = 22+(\sim 17)+1$
(\sim représente l'inversion des bits du nombre)



Un peu de réflexion sur l'arithmétique évite bien des complications du point de vue électronique !

Nombres binaires signés et machine de Pascal

- Vous avez vu en TP que pour effectuer la soustraction $A - B$ avec la Pascaline, il fallait calculer le complément C de A à 9, faire calculer à la machine $D=C+B$, et que le résultat R de la soustraction était le complément de D à 9
 - Par exemple pour $123 - 17 (=106)$
calculer $(999-123) + 17 = 893$, et le résultat = $999-893$
- Pour faire la complémentation à 9, les dents des roues portent deux chiffres : c de 0 à 9, et $c'=9-c$
 - pour afficher 17, on utilise les chiffres c ; pour afficher $999-123$, on affiche 123 avec les chiffres c' , qui correspondent donc à 876 en chiffres
 - on additionne 17, donc $876+17 = 893$ sur les chiffres c
Pour lire le résultat, on regarde les chiffres c' (106)
 - donc ici aussi, soustraction = complémentation + addition



Les premiers algorithmes connus (multiplication, Egypte antique)

- Les égyptiens ne savaient que doubler ou diviser par 2 ; comment multiplier 2 nombres (autre que multiplier par 10, où il suffit de remplacer chaque symbole par celui de la puissance de 10 supérieure) ?
 - Ils connaissaient les puissances de 2 (et implicitement la décomposition d'un nombre en binaire) ; par exemple pour multiplier 45 par 12

i	$n=2^i$	$12 \times n$	
0	1	12	$32 \leq 45 < 64$, donc compter 384
1	2	24	$45 - 32 = 13$, $8 \leq 13 < 16$, compter 96
2	4	48	$4 \leq 13 - 8 < 8$, compter 48
3	8	96	$1 \leq 5 - 4 < 2$, compter 12
4	16	192	donc $45 \times 12 = 384 + 96 + 48 + 12 = 540$
5	32	384	
6	64	768	ce qui revient à calculer $12 \times (2^0 + 2^2 + 2^3 + 2^5)$

Modernité de la multiplication égyptienne

- La multiplication égyptienne est en fait très moderne : $r = a \times b$ peut être défini comme
 - $r = 0$ si $b = 0$,
 - $r = 2a \times p$ si $b = 2p$,
 - $r = 2a \times p + a$ si $b = 2p + 1$
 - Et on recommence pour $a' \times p$ ($a' = 2a$), jusqu'à ce que p soit nul, en cumulant les a' quand p est impair

- D'où l'algorithme :

produit (a, b) \rightarrow r

$r := 0$

tant que b > 0 *faire*

si b impair *alors* $r := r + a$

$a := 2 \times a$

$b := b \div 2$ (*quotient de la division entière*)

pour a=12, b=45		
a	b	r
12	45	0
24	22	12
48	11	
96	5	12+48
192	2	12+48+96
384	1	
768	0	12+48+96+384

- Ce qui revient, pour 12×45 , à calculer comme les égyptiens $12 + 12 \times 2^2 + 12 \times 2^3 + 12 \times 2^5$

Multiplication en binaire

- La table de multiplication en binaire est très simple :
 $0 \times 0 = 0 \times 1 = 1 \times 0 = 0$ $1 \times 1 = 1$ (= porte AND)

- Par exemple 6×5 :

notez que décaler à gauche revient à multiplier par 2 (ça ne vous rappelle pas les égyptiens ?)

$$\begin{array}{r}
 \times \\
 \\
 \\
 \\
 \hline
 6 \times 1 \times 2^0 \\
 6 \times 0 \times 2^1 \\
 6 \times 1 \times 2^4 \\
 \hline
 2^4 + 2^3 + 2^2 + 2 = 30
 \end{array}$$

- Repose simplement sur le fait que, pour $a \times b$
 $a \times (2^n b_n + 2^{n-1} b_{n-1} + \dots + 2b_1 + b_0) = a.b_0 + 2a.b_1 + \dots + 2^{n-1} a.b_{n-1} + 2^n a.b_n$
notez que si a et b occupent chacun n bits, le résultat peut occuper 2n bits
- Le circuit machine (basique) utilise des AND, des additionneurs et des décaleurs, assemblés en cascade

Les premiers algorithmes connus (division, Egypte antique)

- Pour la division $540 \div 12$, il suffisait aux égyptiens de procéder à l'inverse de la multiplication, en utilisant la même table :

1	12
2	24
4	48
8	96
16	192
32	384
64	768

$384 \leq 540 < 768$, donc compter 32

$540 - 384 = 156$, $96 \leq 156 < 192$, compter 8

$156 - 96 = 60$, $48 \leq 60 < 96$, compter 4

$60 - 48 = 12$, donc compter 1

donc $540 \div 12 = 32 + 8 + 4 + 1 = 45$

- Ils pouvaient aussi aller jusqu'aux fractions; par exemple pour $549/12$, le calcul précédent donnera un reste de 9; en ajoutant

1/4	3
1/3	4
1/2	6

donc $549/12 = 45 + 1/2 + 1/4$

Modernité de la division égyptienne

- Cette division repose sur le fait que si $q = a \div b$, alors $a = b \times q + r$ ($0 \leq r < b$), donc $a = b \times (2^n q_n + \dots + 2q_1 + q_0)$, avec $q_n = 1$ (on ne s'intéresse pas aux 0 non significatifs)
 - On cherche donc le n tel que $2^n b \leq a < 2^{n+1} b$
 - Alors, $a = 2^n b + R$, et il suffit de recommencer pour $R \div b$ si $R \geq b$
- D'où l'algorithme

division (a, b) \rightarrow (q, r)
 $p := 1, q := 0, r := a$

tant que $p * b < r$ *faire*
 $p := p * 2$

tant que $r \geq b$ *faire*
si $p \times b \leq r$ *alors*
 $q := q + p$
 $r := r - p \times b$
 $p := p \div 2$

Trace de l'exécution (2^{ème} boucle)
pour a = 540 et b = 12

r	p	p.b	q
540	64	768	0
	32	384	
156	16	192	32
	8	96	
60	4	48	40
12	2	24	44
	1	12	
0	0	0	45

La division en binaire

- Pour en comprendre le principe, revenons à la division (avec des nombres écrits en base 10) comme nous l'avons apprise à l'école

- soit à calculer $5823 \div 27$, on pose

dans 56, il y va $2 \times 27 = 54$,
 $56 - 54 = 2$ et je descends 3 ;
 dans 23, il y va 0×27 ,
 je descends 7 ;
 dans 237 il y va $8 \times 27 = 216$,
 $237 - 216 = 21$, $21 < 27$, c'est fini

$$\begin{array}{r|l}
 5637 & 27 \\
 \hline
 56 & 2 \\
 -54 & \\
 \hline
 23 & 0 \\
 237 & 8 \\
 -216 & \\
 \hline
 21 &
 \end{array}$$

donc $q = 208$, $r = 21$

- Idem en binaire : $29 \div 5$

il suffit d'ajouter 0 à droite du
 quotient si le nombre abaissé
 est $<$ au diviseur, et 1 sinon

donc $q = 5$, $r = 4$

$$\begin{array}{r|l}
 11101 & 101 \\
 \hline
 111 & 1 \\
 -101 & \\
 \hline
 100 & 0 \\
 1001 & 1 \\
 - 101 & \\
 \hline
 100 &
 \end{array}$$

Division égyptienne et division binaire

- Reprenons l'algorithme de la division égyptienne, et regardons comment il s'exécute pour $29 \div 5$ en écrivant les nombres en binaire sur 6 bits

```

division (a, b) → (q, r)
  p := 1, q := 0, r := a

  tant que p * b < r faire
    p := p * 2

  tant que r ≥ b faire
    si p × b ≤ r alors
      q := q + p
      r := r - p × b
    p := p ÷ 2
  
```

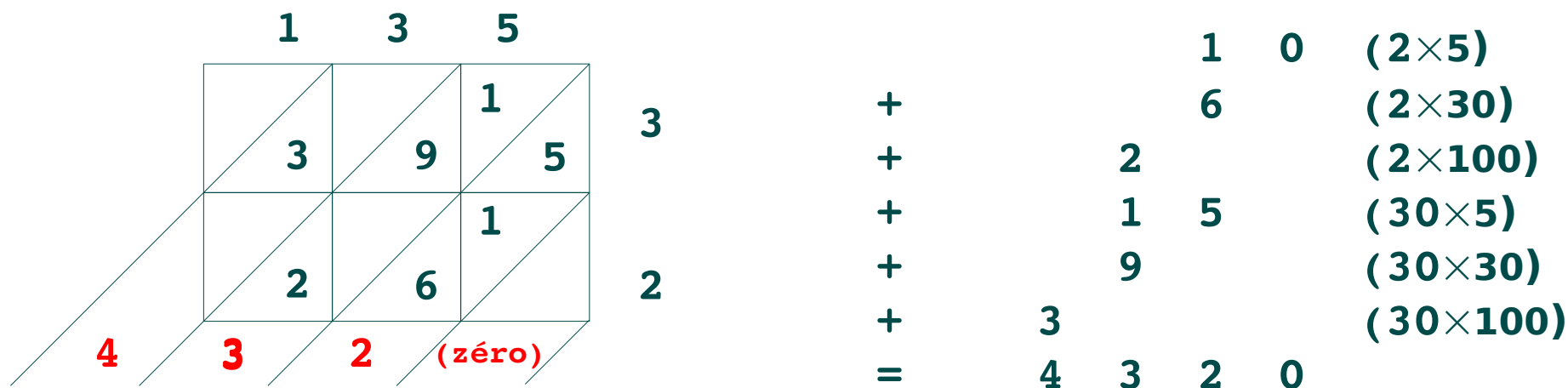
Trace de l'exécution (b=000101)

r	p	p.b	q
	<i>première boucle tant que</i>		
011101	0001	000101	000000
	0010	001010	
	0100	010100	
	1000	101000	
	<i>deuxième boucle tant que</i>		
	0100	010100	
011101-010100=			000000+000100 =
001001	0010	001010	000100
	0001	000101	
001001-000101=			000100+000001 =
000100 (soit 4)			000101(soit 5)

- On voit apparaître les mêmes valeurs que pour la division « à la main »

Multiplication à l'aide d'un tableau

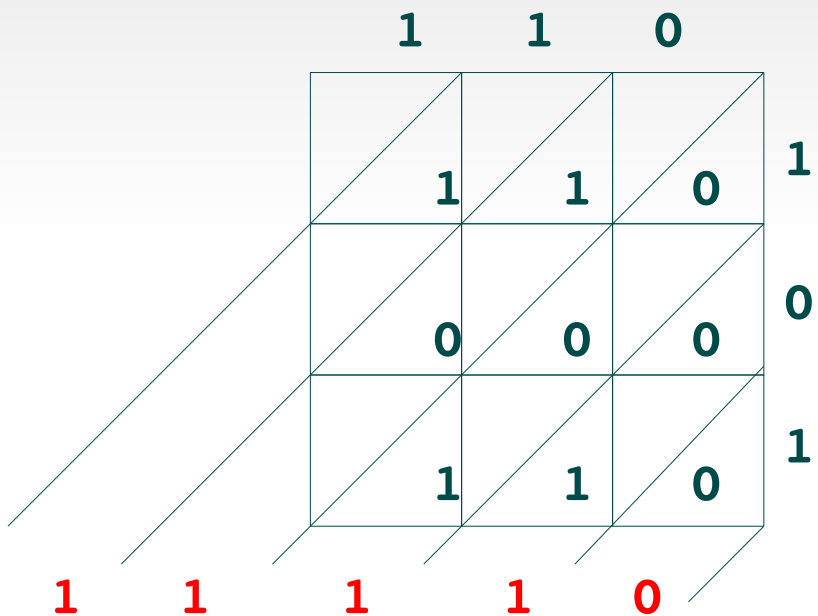
- Technique initialement arabe ou indienne (12e siècle), mais aussi connue en Chine et en Europe
 - Soit à calculer $135 \times 32 = 4320$: chaque case (i,j) contient les résultats de la multiplication du chiffre de la ligne i et de celui de la colonne j, le chiffre des dizaines (s'il y en a un) en haut ; le reste ensuite à additionner en diagonale en partant de la droite



Il fallait évidemment connaître ses tables de multiplication !

Multiplication binaire revue

- On peut évidemment multiplier en binaire avec un tableau : par exemple 6×5 (110×101 en binaire)



On peut imaginer un circuit où chaque case est un circuit très simple (*AND*)

	0	1
0	0	0
1	0	1

il suffit ensuite d'additionner leurs sorties « en diagonale »

Pas besoin de connaître ses tables de multiplication !

Les différences finies

- Toute fonction continue peut être approchée aussi près que l'on veut par une fonction polynomiale (théorème de Taylor, 1712)
 - **d'où l'importance du calcul des valeurs d'un polynôme : c'est ainsi que les ordinateurs calculent les fonctions trigonométriques, exponentielles, etc. par développement limité**
- Pour un polynôme $P(x)$ de degré n , quand on calcule n fois les différences entre $x+1$ et x , on arrive toujours à des différences constantes. Par exemple pour x^3

x	x^3	D_1	D_2	D_3	D_4
0	0				
1	1	1			
2	8	7	6		
3	27	19	12	6	
4	64	37	18	6	0
5	125	61	24	6	0

Car la dérivée n -ième d'un polynôme de degré n est une constante

La machine à différences de Charles Babbage (1822)

- Charles Babbage a utilisé la propriété des différences finies pour faire calculer des valeurs de polynômes de degré n par une machine mécanique avec simplement des additions :
 - On dispose de $n+2$ « accumulateurs » A_0 à A_{n+1} initialisés avec les premières différences $D_0, D_1 \dots D_{n+1}$
 - on aura donc, pour l'exemple précédent, 5 « accumulateurs » A_0 à A_4 initialisés à 0, 1, 6, 6, 0 (les nombres en gras du tableau précédent)
 - A chaque étape, on ajoute à chaque A_i la valeur de A_{i+1}
 - La $k^{\text{ème}}$ étape donne dans A_0 la valeur de $P(x)$ en $x=k$

L'algorithme de la machine à différence

- On aurait donc l'algorithme suivant pour calculer la valeur d'un polynome $P(x)$ de degré n en $x=k$ (k entier ≥ 0)
 - initialiser les $n+2$ accumulateurs avec les différences initiales*
 - répéter k fois*
 - pour i de 1 à n faire $A_{i-1} := A_{i-1} + A_i$*
 - le résultat est dans A_0*
- Exemple de fonctionnement pour calculer la valeur de $2x^3+5x^2+3x+4$ en $x=4$

différences initiales : 4, 10, 22, 12, 0

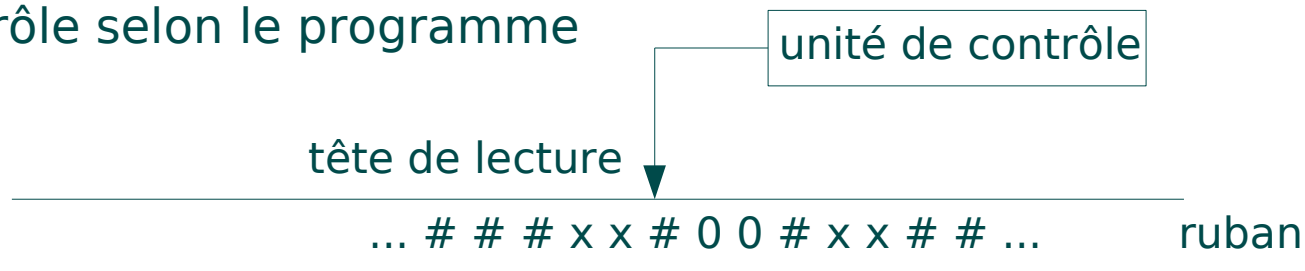
x	A_0	A_1	A_2	A_3	A_4
0	4	10	22	12	0
1	14	32	34		
2	46	66	46		
3	112	112	58		
4	224	170	70		

Décidabilité et complexité

- Un gros problème se pose aux informaticiens : étant donné un certain problème, existe-t-il un algorithme pour en donner une solution (ce problème est-il *décidable* ou non ?)
 - Il est par exemple indécidable (dans le cas général) de savoir si un programme s'arrêtera quel que soient ses données !
- Et si ce problème est décidable, combien faudra-t-il de temps pour le résoudre en fonction de la taille de ses données : c'est la *complexité* de l'algorithme mis en œuvre
 - S'il faut 8h à un programme pour prévoir la météo du lendemain, et que la prévision pour chaque jour supplémentaire double le temps de calcul (donc 16h pour la météo du surlendemain), prévoir la météo à 5 jours (120h) demandera 128h de calcul. Votre programme donnera une « prévision » pour le journal télé de 20h de ce que tout le monde aura pu constater à midi ! Pas très utile pour les téléspectateurs.
- Il est nécessaire d'avoir des outils théoriques pour apprécier la décidabilité d'un problème, et estimer la complexité d'un algorithme

La machine de Turing (1936)

- Pour résoudre ces questions, Alan Turing a défini un modèle abstrait d'ordinateur ; il comprend
 - Un *ruban* (infini) composé de cases dans lesquelles peuvent être inscrits des **symboles quelconques** appartenant à un ensemble fini, par exemple {0, x, #}
 - D'une unité de contrôle qui peut prendre un nombre fini d'états (le processeur et son *programme*)
 - D'une tête de lecture-écriture du ruban, commandée par l'unité de contrôle selon le programme



- Selon la thèse de Church-Turing, les problèmes résolubles par une machine de Turing universelle sont exactement ceux qui sont résolubles par un algorithme

Exemple de machine de Turing

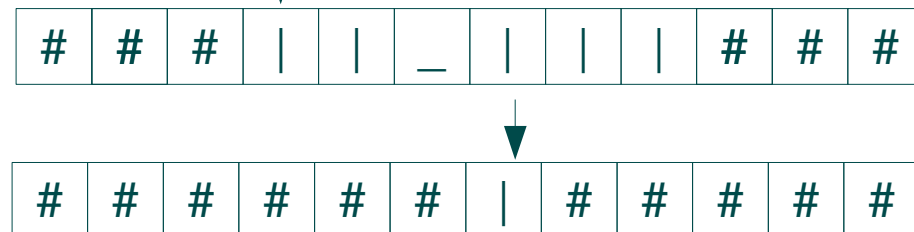
- On associe à chaque **état** $q_1, q_2 \dots$ de la machine et à chaque symbole de l'alphabet un triplet
(état suivant, symbole à écrire, déplacement de la tête)

- Par exemple, avec pour symboles $\{_, |, \#\}$ et le programme :

état	symbole lu		
		-	#
q_1	$(q_2, \#, \blacktriangleright)$	$(q_5, \#, \blacktriangleright)$	
q_2	$(q_2, , \blacktriangleright)$	$(q_2, -, \blacktriangleright)$	$(q_3, \#, \blacktriangleleft)$
q_3	$(q_4, \#, \blacktriangleleft)$		
q_4	$(q_4, , \blacktriangleleft)$	$(q_4, -, \blacktriangleleft)$	$(q_1, \#, \blacktriangleright)$
q_5			

en q_1 , si le symbole lu est | écrire #, avancer la tête d'une case à droite et aller en q_2 , etc.

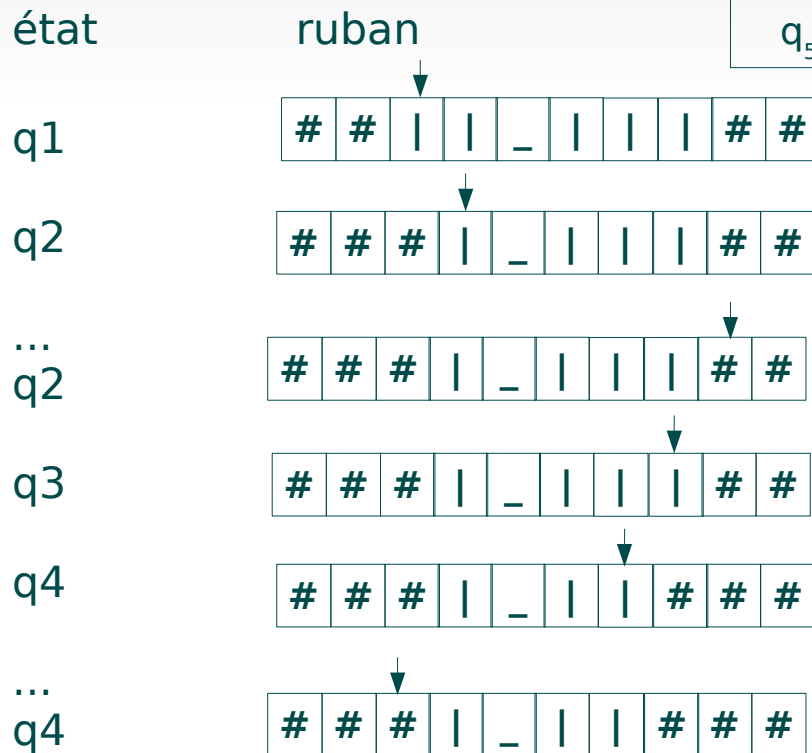
- A partir du ruban initial, représentant les nombres 2 (||) et 3 (|||), et en partant de l'état q_1 , on obtient le ruban suivant : la machine a calculé $3 - 2 = 1$



Fonctionnement de la machine

- Rappel du programme :

état		-	#
q_1	$(q_2, \#, \blacktriangleright)$	$(q_5, \#, \blacktriangleright)$	
q_2	$(q_2, , \blacktriangleright)$	$(q_2, -, \blacktriangleright)$	$(q_3, \#, \blacktriangleleft)$
q_3	$(q_4, \#, \blacktriangleleft)$		
q_4	$(q_4, , \blacktriangleleft)$	$(q_4, -, \blacktriangleleft)$	$(q_1, \#, \blacktriangleright)$
q_5			



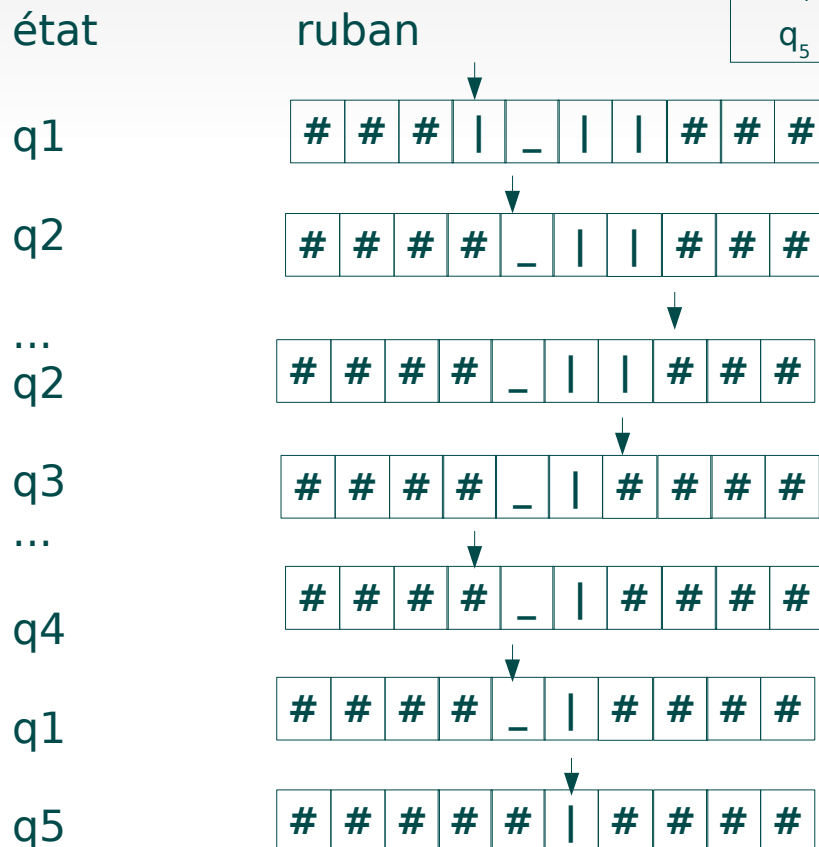
Dans cette séquence, le | le plus à gauche est effacé (q_1), puis on cherche le | le plus à droite pour l'effacer aussi (boucle sur q_2).

On va ensuite chercher le | le plus à gauche (boucle sur q_4), puis on revient en q_1 avec la tête sur le | le plus à gauche (même configuration qu'au début), et on recommence

Fonctionnement de la machine (suite)

- Rappel du programme :

état		-	#
q ₁	(q ₂ , #, ▶)	(q ₅ , #, ▶)	
q ₂	(q ₂ , , ▶)	(q ₂ , -, ▶)	(q ₃ , #, ◀)
q ₃	(q ₄ , #, ◀)		
q ₄	(q ₄ , , ◀)	(q ₄ , -, ◀)	(q ₁ , #, ▶)
q ₅			



Dans cette séquence, la machine efface de nouveau le | le plus à gauche et celui le plus à droite.

En q₅, la machine n'a aucune action à effectuer sur le symbole |, elle s'arrête. Donc, elle a éliminé autant de | à droite qu'il y en a à gauche, c'est bien une soustraction.

Vous pouvez vérifier que pour le ruban initial # _ || | #, elle calcule bien 3 - 0.

Petite leçon de modestie

Nous sommes des nains juchés sur des épaules de géants. Nous voyons ainsi davantage et plus loin qu'eux, non parce que notre vue est plus aigüe ou notre taille plus haute, mais parce qu'ils nous portent en l'air et nous élèvent de toute leur hauteur gigantesque.

Attribué à Bertrand de Chartres, 12^e siècle, reprise plus tard par Newton (entre autres)