Algorithme de calcul d'une racine carrée

Benjamin.Barras@epfl.ch, EPFL - Domaine-IT, Responsable Linux

The algorithm presented here is used to calculate a square root simply, efficiently and very accurately.

L'algorithme présenté ici, permet de calculer une racine carrée de manière simple, efficace et d'une précision redoutable.

Introduction

Après avoir parlé de l'algorithme CORDIC **[1]** dans le numéro précédent, je vous propose de découvrir en détail l'algorithme de calcul d'une racine carrée utilisé dans les calculatrices HP. On trouvera l'article original, qui décrit cet algorithme, dans le HP Journal **[2]**.

Vocabulaire

Avant de décrire cet algorithme, nous avons besoin de définir un peu de vocabulaire:

- 0,1,2,3,4,5,6,7,8,9 sont des chiffres *(digit)*;
- on forme un nombre avec des chiffres;
- exemple: 1267 est un nombre formé avec les chiffres 1, 2, 6, 7;
- on travaillera toujours en base 10;
- rappel: $1267=1\cdot10^3+2\cdot10^2+6\cdot10^1+7$.

Présentation de l'algorithme

Première approche

- On cherche la valeur de $\sqrt{x}$,
- Prenons a une valeur approchée de $\sqrt{x}$, et calculons le reste: $R=x-a^2$,
- Si $R>0$, on augmente a et on recommence,
- Encore faut-il choisir une méthode ?

Premier exemple

Première approche, calcul de $\sqrt{54756}$

a	$R=54756-a^2$	Signe	Notes
200	14756	$R>0$	$a=200$ est une valeur approchée
210	10656	$R>0$	on augmente a
220	6356	$R>0$	on augmente a
230	1856	$R>0$	on augmente a
240	-2844	$R<0$	stop, nouvelle valeur approchée: $a=230$

Simple, pas compliqué, mais on peut faire mieux !

Calculs

Cherchons la prochaine valeur du chiffre b qui se trouve à la position j: $a_j=a+b\cdot10^j$ avec a qui est notre une valeur approchée (nombre). Les restes sont:

$$R_a=x-a^2 \quad \text{et} \quad R_{aj}=x-a_j^2$$

Remarque: $R_a \geq R_{aj}$ puisque a_j est plus proche de x. Définissons R_b comme le solde de la différence lorsque l'on ajoute $b\cdot10^j$ à a:

$$R_b=R_a-R_{aj}=a_j^2-a^2$$

Donc, d'après l'équation ci-dessus:

$$R_a-R_b=R_{aj}\geq0 \quad \text{donc} \quad R_a\geq R_b$$

Premiers développements

Calculons R_b:

$R_b=a_j^2-a^2=(a+b\cdot10^j)^2-a^2=a^2+2ab\cdot10^j+b^2\cdot10^{2j}-a^2=2ab\cdot10^j+b^2\cdot10^{2j}$

Donc b est le plus grand chiffre possible tel que:

$$2ab\cdot10^j+b^2\cdot10^{2j}\leq R_a$$

Une fois que l'on a trouvé b, a est recalculé en y ajoutant la valeur $b\cdot10^j$, soit $a_j=a+b\cdot10^j$. On recalcule R_a, soit $R_{aj}=R_a-R_b$. Enfin, on décrémente j de 1.

Deuxième exemple

Deuxième approche, calcul de $\sqrt{54756}$

$a=200$, $R_a=x-a^2=54756-40000=14756$, $j=1$

b	$R_b=2ab\cdot10^j+b^2\cdot10^{2j}$	R_a-R_b
0	0	14756
1	4100	10656
2	8400	6356
3	12900	1856
4	17600	-2844

Donc $a=200+3\cdot10^1=230$, $R_a=1856$, $j=0$

Simple, pas compliqué, mais on peut faire mieux !

Première subtilité

Il faut savoir qu'on écrit

- un nombre pair: $p=2n$ avec $n\in\mathbb{N}$
- un nombre impair: $p=2n-1$ avec $n\in\mathbb{N}-0$

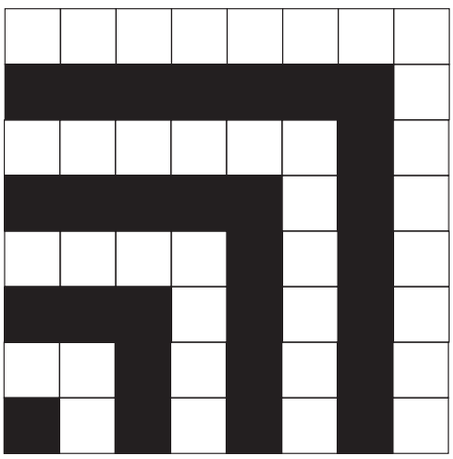
Théorème

$b^2=\sum_{n=1}^{b} 2n-1$ somme des nombres impairs jusqu'à $2b-1$.

Exemple:

$$b=6$$
$$2b-1=11$$
$$b^2=36=1+3+5+7+9+11$$

Preuve



Les lignes blanches et noires représentent les nombres impairs. Il suffit de compter le nombre de carré. Reprenons notre calcul, avec:

$$b^2=\sum_{n=1}^{b} 2n-1 \qquad b=\sum_{n=1}^{b} 1$$

Ce qui nous donne:

$$R_b=2ab\cdot 10^j+b^2\cdot 10^{2j}=\sum_{n=1}^{b}(2a\cdot 10^j+(2n-1)\cdot 10^{2j})$$

Deuxième subtilité

Reprenons notre inégalité: $R_a \geq R_b$.
Multiplions par 5 des deux côtés: $5\cdot R_a \geq 5\cdot R_b$.
Calculons $5\cdot R_b$:

$$5\cdot R_b=5\sum_{n=1}^{b}(2a\cdot 10^j+(2n-1)\cdot 10^{2j})=\sum_{n=1}^{b}(10a\cdot 10^j+(10n-5)\cdot 10^{2j})$$

Rappel: b est le plus grand chiffre possible tel que $5\cdot R_a \geq 5\cdot R_b$ reste vérifiée: $R_{aj}=R_a-R_b \geq 0$, donc
$5\cdot R_{aj}=5\cdot R_a-5\cdot R_b \geq 0$.

Troisième exemple

$\sqrt{x}=\sqrt{54756}$,a=200,

$R_a=x-a^2=54756-40000=14756$, $j=1$

$$5\cdot R_b=\sum_{n=1}^{b}(10a\cdot 10^j+(10n-5)\cdot 10^{2j}),\ 5\cdot R_a=73780$$

n	$10a\cdot 10^j+(10n-5)\cdot 10^{2j}$	$5\cdot R_b$	$5\cdot R_a-5\cdot R_b$
1	20500	20500	53280
2	21500	42000	31780
3	22500	64500	9280
4	23500	88000	-14220

Donc b=3, a=200+3·10[1]=230, $5\cdot R_a$=9280, j=0
Remarque: 23500 sont les premiers chiffres qui apparaissent pour $\sqrt{54756}$.

Suite

$\sqrt{x}=\sqrt{54756}$,a=230, $R_a=x-a^2=54756-52900=1856$, $j=0$

$$5\cdot R_b=\sum_{n=1}^{b}(10a\cdot 10^j+(10n-5)\cdot 10^{2j}),\ 5\cdot R_a=9280$$

n	$10a\cdot 10^j+(10n-5)\cdot 10^{2j}$	$5\cdot R_b$	$5\cdot R_a-5\cdot R_b$
1	2305	2305	6975
2	2315	4620	4660
3	2325	6945	2335
4	2335	9280	0
5	2345	11625	-2345

Donc b=4, a=230+4=234 qui correspond aux chiffres 2345.

Quelques questions

Que prenons-nous pour la valeur approchée de a?
– La plus simple et logique possible, soit: a=0
Que prenons-nous pour la valeur de j?
– La plus simple possible soit: j=0
En effet, un nombre s'écrit de la manière suivante: 0.0000000E00. Le premier chiffre ∈0,1,2,3,4,5,6,7,8,9. On commence donc par calculer la partie entière, et après les décimales. On fait comme à l'école, on efface la virgule et on la recolle à la fin du calcul. De toute façon, les registres ne s'encombrent pas de virgules. Mais on y reviendra à la fin.

De la théorie à la pratique

Exemple pratique $\sqrt{2}$
Rappel: $10a\cdot 10^j+(10n-5)\cdot 10^{2j}$

```
x     :00000002    8 chiffres (digits)
a     :00000000
5Ra   :00000010    Ra = x - a² = 2
```

Premier problème: on doit faire un décalage *(shift)*

```
    5Ra  :10000000   avec shift
    5Rb  :05000000   n = 1
          ===========
5Ra-5Rb  :05000000
    5Rb  :15000000   n = 2
          ===========
5Ra-5Rb  :990000000  overflow => b = 1 & a = 10000000
          j = j - 1 => a = shift droit, (10n-5) double shift droit
    5Ra  :05000000 5Ra = 5Ra-5Rb
    5Rb  :01050000   n = 1
          ===========
5Ra-5Rb  :03950000
    5Rb  :01150000   n = 2
          ===========
```

```
5Ra-5Rb  : 02800000
    5Rb  : 01250000   n = 3
           ==========
5Ra-5Rb  : 01550000
    5Rb  : 01350000   n = 4
           ==========
5Ra-5Rb  : 00200000
    5Rb  : 01450000   n = 5
           ==========
5Ra-5Rb  : 998750000  overflow => b = 4 & a = 01400000 & j = j - 1
    5Ra  : 00200000   5Ra = 5Ra-5Rb
    5Rb  : 00140500   n = 1
           ==========
5Ra-5Rb  : 00059500
    5Rb  : 00141500   n = 2
           ==========
5Ra-5Rb  : 999918000  overflow => b = 1 & a = 00141000 & j = j - 1
    5Ra  : 00059500   5Ra = 5Ra-5Rb
    5Rb  : 00014105   n = 1
           ==========
5Ra-5Rb  : 00045395
    5Rb  : 00014115   n = 2
           ==========
5Ra-5Rb  : 00031280
    5Rb  : 00014125   n = 3
           ==========
5Ra-5Rb  : 00017155
    5Rb  : 00014135   n = 4
           ==========
5Ra-5Rb  : 00003020
    5Rb  : 00014145   n = 5
           ==========
5Ra-5Rb  : 999988875  overflow => b = 4 & a = 00014140 & j = j - 1
    5Ra  : 00003020   5Ra = 5Ra-5Rb
    5Rb  : 0000141405 n = 1
           ==========
```

Deuxième problème: on a épuisé notre quota avec seulement 4 chiffres *(digits)* trouvés.

Solution: on fait un shift gauche à chaque étape.

On retrouve à ce sujet, une phrase dans l'article original qu'il faut apprendre à décoder: *During the process of finding* $\sqrt{x}$ *the remainder* R_a *progressively decreases. To avoid losing accuracy, this remainder is multiplied by* 10^j *after finding each new digit b. This avoids shifting a at all, once the square root extraction process begins.*

On recommence

```
    5Ra  : 10000000
    5Rb  : 05000000   n = 1
           ==========
5Ra-5Rb  : 05000000
    5Rb  : 15000000   n = 2
           ==========
5Ra-5Rb  : 990000000  overflow => b = 1 & a = 10000000 & j = j - 1
    5Ra  : 50000000   5Ra = 5Ra-5Rb avec shift
    5Rb  : 10500000   n = 1
           ==========
5Ra-5Rb  : 39500000
    5Rb  : 11500000   n = 2
           ==========
5Ra-5Rb  : 28000000
    5Rb  : 12500000   n = 3
           ==========
5Ra-5Rb  : 15500000
    5Rb  : 13500000   n = 4
           ==========
5Ra-5Rb  : 02000000
    5Rb  : 14500000   n = 5
           ==========
5Ra-5Rb  : 987500000  overflow => b = 4 & a = 14000000 & j = j - 1
    5Ra  : 20000000   5Ra = 5Ra-5Rb avec shift
    5Rb  : 14050000   n = 1
           ==========
5Ra-5Rb  : 05950000
    5Rb  : 14150000   n = 2
           ==========
5Ra-5Rb  : 991800000  overflow => b = 1 & a = 14100000 & j = j - 1
    5Ra  : 59500000   5Ra = 5Ra-5Rb avec shift
    5Rb  : 14105000   n = 1
           ==========
5Ra-5Rb  : 45395000
    5Rb  : 14115000   n = 2
           ==========
```

```
5Ra-5Rb  : 31280000
    5Rb  : 14125000   n = 3
           ===========
5Ra-5Rb  : 17155000
    5Rb  : 14135000   n = 4
           ===========
5Ra-5Rb  : 03020000
    5Rb  : 14145000   n = 5
           ===========
5Ra-5Rb  : 988875000  overflow => b = 4 & a = 14140000 & j = j - 1
    5Ra  : 30200000   5Ra = 5Ra-5Rb avec shift
    5Rb  : 14140500   n = 1
           ===========
5Ra-5Rb  : 16059500
    5Rb  : 14141500   n = 2
           ===========
5Ra-5Rb  : 01918000
    5Rb  : 14142500   n = 3
           ===========
5Ra-5Rb  : 987775500  overflow => b = 2 & a = 14142000 & j = j - 1
    5Ra  : 30200000   5Ra = 5Ra-5Rb avec shift
    5Rb  : 14140500   n = 1
           ===========
5Ra-5Rb  : 16059500
    5Rb  : 14141500   n = 2
           ===========
5Ra-5Rb  : 01918000
    5Rb  : 14142500   n = 3
           ===========
5Ra-5Rb  : 987775500  overflow => b = 2 & a = 14142000 & j = j - 1

    5Ra  : 19180000 5Ra = 5Ra-5Rb avec shift
    5Rb  : 14142050   n = 1
           ===========
5Ra-5Rb  : 05037950
    5Rb  : 14142150   n = 2
           ===========
5Ra-5Rb  : 990895800  overflow => b = 2 & a = 14142100 & j = j - 1
    5Ra  : 50379500 5Ra = 5Ra-5Rb avec shift
    5Rb  : 14142105   n = 1
           ===========
5Ra-5Rb  : 36237395
    5Rb  : 14142115   n = 2
           ===========
5Ra-5Rb  : 22095280
    5Rb  : 14142125   n = 3
           ===========
5Ra-5Rb  : 07953155
    5Rb  : 14142135   n = 4
           ===========
5Ra-5Rb  : 993811020  overflow => b = 3 & a = 14142130 & j = j - 1
    5Ra  : 79531550 5Ra = 5Ra-5Rb avec shift
    5Rb  : 141421305  n = 1
```

Troisième problème: il nous manque un chiffre à droite

Solution: on ajoute un chiffre à droite

```
    5Ra  : 795315500 5Ra = 5Ra-5Rb avec shift
    5Rb  : 141421305 n = 1
           ===========
5Ra-5Rb  : 653894195
    5Rb  : 141421315 n = 2
           ===========
5Ra-5Rb  : 512472880
    5Rb  : 141421325 n = 3
           ===========
5Ra-5Rb  : 371051555
    5Rb  : 141421335 n = 4
           ===========
5Ra-5Rb  : 229630220
    5Rb  : 141421345 n = 5
           ===========
5Ra-5Rb  : 088208875
    5Rb  : 141421355 n = 6
           ===========
5Ra-5Rb  : 9946787520 overflow => b = 5 & a = 14142135
```

Réponse: $\sqrt{2} = 1.4142135$

Vérification

Il est toujours très important de vérifier ses résultats [3]. On a trouvé: $\sqrt{2} = 1.4142135$ que l'on peut facilement vérifier à l'aide de la commande **bc** (bc - *An arbitrary precision calculator language*) de la fondation **GNU** (gnu.org):

```
echo 'scale=7; sqrt(2)' | bc
1.4142135
```

Bug du *shift* gauche

En faisant un *shift* à gauche pour R_a, on a introduit un bug. Démonstration:

```
    5Ra  :  13441930443550000
    5Rb  :  14142135623730500   n = 1
            ==================
5Ra-5Rb  :  999299794819819500   overflow => b = 0

    5Ra  :  134419304435500000   5Ra = 5Ra-5Rb avec shift
    5Rb  :   14142135623730050   n = 1
            ==================
5Ra-5Rb  :   20277168811769950

5Ra-5Rb  :  120277168811769950   La vraie soustraction
```

Quatrième problème: il nous manque un chiffre à gauche

Solution: on ajoute un chiffre à gauche

```
    5Ra  :   0134419304435500000
    5Rb  :   0141421356237305000   n = 1
             ====================
5Ra-5Rb  :   999299794819819500   overflow => b = 0

    5Ra  :   1344193044355000000   5Ra = 5Ra-5Rb avec shift
    5Rb  :   0141421356237300500   n = 1
             ====================
5Ra-5Rb  :   120277168811769950
```

Calcul de l'exposant

Reprenons notre nombre:
`0.0000000E00`

$$\sqrt{x \cdot 10^e} = \sqrt{x} \cdot \sqrt{10^e} = \sqrt{x} \cdot 10^{e/2}$$

Si *e* est pair: *no problemo*.
Si *e* est impair: $\sqrt{x} \cdot 10^{e/2} = \sqrt{10 \cdot x} \cdot 10^{(e-1)/2}$. Il faut faire un *shift* gauche pour *x* avant de commencer les calculs.

Troisième subtilité

Comment fait-on pour diviser par 2 puisque l'on travaille en base 10 et que l'on a ni division, ni multiplication? Difficile de trouver plus simple: $x \div 2 = (x \cdot 5) \div 10$. Donc, on fait 5 additions et un *shift* droit. Exemple $123 \div 2 = (123 \cdot 5) \div 10$:

```
00123     00246     00492
00123     00246     00123
=====     =====     =====
00246     00492     00615

00615
000615

Réponse: 61 reste 0.5 (carry)
```

C'est également un moyen de savoir si un nombre est pair ou impair *(carry)*.

Quelques commentaires

On ne trouve pas dans l'article de William E. Egbert trois choses:

1. le chiffre ajouté à droite;
2. Le chiffre ajouté à gauche;
3. La subtilité de la division par 2.

Par contre, on trouve ces astuces dans le *listing* **[4]** d'une calculatrice HP35 ainsi que dans la trace d'exécution d'un calcul de racine carrée **[5]**.

Bonus

Je vous propose de tester **[6]** cet algorithme à l'aide d'un programme que j'ai écrit en *javascript* que l'on peut exécuter à l'aide d'une simple page *HTML* ou de télécharger l'archive complète **[7]** si vous souhaitez voir le code source. Avec ce programme, vous pouvez choisir le nombre de virgules que vous voulez et vous pouvez voir toute la trace du calcul tel que présenté dans le présent article. J'ai poussé la plaisanterie assez loin, puisque je n'utilise pas d'addition ni de soustraction, tous les calculs se font à l'aide de tables. Cela juste pour prouver que l'on peut calculer une racine carrée sans avoir besoin d'unité arithmétique.

Références

[1] Algorithme CORDIC: FI6/13, flashinformatique.epfl.ch/spip.php?article2692.
[2] Personal Calculator Algorithms I: Square Roots.
[3] J'ai poussé la vérification de cet algorithme jusqu'à 100'000 décimales pour le calcul de la racine carrée de 2, à l'aide d'un programme écrit en perl et n'utilisant que des tables d'additions et de soustractions (pas d'unité arithmétique) dans le même esprit que le programme en javascript proposé. On peut trouver facilement des valeurs sur le Web qui permettent la comparaison.
[4] HP35 square root algorithm: www.jacques-laporte.org/Square_root.htm.
[5] Trace de l'algorithme d'une calculatrice HP: www.jacques-laporte.org/sqrt_1156.txt.
[6] Programme en javascript: https://documents.epfl.ch/users/b/ba/barras/www/Download/squareRoot/sqrt.html.
[7] L'archive complète: https://documents.epfl.ch/users/b/ba/barras/www/Download/squareRoot/javascript.tar.gz. ■