

[〆切] 2020/10/01 17:00

[I. 有限桁の呪縛]

I-A. 10進3桁で表された数値 $a = 0.537$, $b = 0.612$, $c = 0.126$ に対して $(a+b)+c$ と $a+(b+c)$ を計算せよ。ただし加算の際に得られた数値は10進4桁以降を切り捨てて計算せよ。

実数 x に対して 10進4桁以降を切り捨てる関数を $r(x)$ とする。

$a = r(a)$, $b = r(b)$, $c = r(c)$ であるため 計算すべき2式はそれぞれ

$$(a+b)+c \longrightarrow \dots \textcircled{1}$$

$$a+(b+c) \longrightarrow \dots \textcircled{2}$$

と書き換える。

・ ①を順番に計算す。

$$r(a+b) =$$

$$r(r(a+b)+c) =$$

$$\text{従って } (a+b)+c = \underline{\hspace{2cm}}$$

・ ②を順番に計算す。

$$r(b+c) =$$

$$r(a+r(b+c)) =$$

$$\text{従って } a+(b+c) = \underline{\hspace{2cm}}$$

① 舛れ算の工夫

カハンの加算 PILLOW ALG

補完項
埋め合われて

Kahan summation algorithm / compensated summation

Input: a_1, a_2, \dots, a_n, n

Output: sum ($= \sum_{i=1}^n a_i$)

\rightarrow たとえは

sum $\leftarrow 0.0$

cp $\leftarrow 0.0$

for $i = 1, 2, \dots, n$ do

$y \leftarrow a_i - cp$

$t \leftarrow sum + y$

$cp \leftarrow (t - sum) - y$

sum $\leftarrow t$

endfor

return sum

1541

I-A の a, b, c をそれぞれ a_1, a_2, a_3 とし、

$n=3$ の場合のカハンの加算 PILLOW ALG を表す。

$$\begin{cases} a_1 \leftarrow a = 0.537 \\ a_2 \leftarrow b = 0.612 \\ a_3 \leftarrow c = 0.126 \end{cases}$$

for $i=1$ の中身

$i=1$

$$\begin{cases} y \leftarrow r(a_1 - cp) \\ t \leftarrow r(sum + y) \end{cases}$$

$$cp \leftarrow r(r(t - sum) - y)$$

$$sum \leftarrow t$$

(summation の途中: $sum = a_1$)

i=2

$$\left\{ \begin{array}{l} y \leftarrow r(a_2 - cp) \\ t \leftarrow r(sum + y) \\ cp \leftarrow r(r(t - sum) - y) \\ sum \leftarrow t \end{array} \right. \quad (\text{summation の途中: } sum = a_1 + a_2)$$

i=3

$$\left\{ \begin{array}{l} y \leftarrow r(a_3 - cp) \\ t \leftarrow r(sum + y) \\ cp \leftarrow r(r(t - sum) - y) \\ sum \leftarrow t \end{array} \right. \quad (\text{summation の完了: } sum = a_1 + a_2 + a_3)$$

return sum

例1.2

加算時10進4桁以下で切り捨ての場合

$$\left\{ \begin{array}{l} a_1 = 100 \\ a_2 = a_3 = \dots = a_{11} = 0.1 \end{array} \right.$$

で、計算された a_1, \dots, a_{11} の総和を考える。普通に a_1 から順番に足すと a_2, \dots, a_{11} がすべて誤差で消えるため総和が a_1 と同じ100になってしまふ。

しかしカバンの加算アルゴリズムを使うと総和は正しく101となる。

この通りに a_1, \dots, a_{11} をソートしてから値が大きいものから足していくと解が正しくなる。

例1.3

加算時10進4桁以下で切り捨ての場合

$$a_1 = \dots = a_{2000} = 0.1$$

で、計算された a_1, \dots, a_{2000} の総和を考える。普通に a_1 から順番に足すと $a_{1001}, \dots, a_{2000}$ がすべて誤差で消えるため総和が100になってしまふ。

しかしカバンの加算アルゴリズムを使うと総和は正しく200となる。

この通りではソートして他の小さいやめる足す方法は使えない。しかし再帰的に2項ずつ足せば解が正しくなる。

[〆切] 2020/10/01 17:00

I. 有限桁の呪縛

I-A. 10進3桁で表された数値 $a = 0.537$, $b = 0.612$, $c = 0.126$ に対して $(a+b)+c$ と $a+(b+c)$ を計算せよ。ただし加算の際に得られた数値は10進4桁以降を切り捨てて計算せよ。

実数 x に対して 10進4桁以降を切り捨てる関数を $r(x)$ とする。

$a = r(a)$, $b = r(b)$, $c = r(c)$ であるため 計算すべき 2式 は それぞれ

$$\begin{aligned} (a+b)+c &\longrightarrow r(r(a+b)+c) \dots \textcircled{1} \\ &\quad r(r(r(a)+r(b))+r(c)) \\ a+(b+c) &\longrightarrow r(a+r(b+c)) \dots \textcircled{2} \end{aligned}$$

と書き換える。

- ①を順番に計算す。

$$r(a+b) = r(0.537+0.612) = \underline{r(1.149)} = 1.14 \quad \text{丸め誤差} -0.009$$

$$r(r(a+b)+c) = r(1.14 + 0.126) = \underline{r(1.266)} = 1.26 \quad \text{丸め誤差} -0.006$$

$$\text{従って } (a+b)+c = \underline{\underline{1.26}} \quad \text{丸め誤差} -0.006$$

- ②を順番に計算す。

$$r(b+c) = r(0.612+0.126) = r(0.738) = 0.738$$

$$r(a+r(b+c)) = r(0.537+0.738) = \underline{r(1.275)} = 1.27 \quad \text{丸め誤差} -0.005$$

$$\text{従って } a+(b+c) = \underline{\underline{1.27}} \quad \text{丸め誤差} -0.005$$

誤差合計 -0.015

キーワード

丸め誤差

$x - r(x)$

絶対誤差 $|x - r(x)|$

相対誤差 $\frac{|x - r(x)|}{x}$

① 小数点位数の工夫

カハンの加算 PILLOW ALG

{補完枝
埋め合あせた}

Kahan summation algorithm / compensated summation

Input: a_1, a_2, \dots, a_n, n

Output: sum ($= \sum_{i=1}^n a_i$)

\rightarrow たとえは

sum $\leftarrow 0.0$

cp $\leftarrow 0.0$

for $i = 1, 2, \dots, n$ do

$y \leftarrow a_i - cp$

$x \leftarrow sum + y$

$cp \leftarrow (x - sum) - y$

sum $\leftarrow x$

endfor

return sum

1541

I-A の a, b, c をそれぞれ a_1, a_2, a_3 とし、

$n=3$ の場合のカハンの加算 PILLOW ALG を表す。

$$\begin{cases} a_1 \leftarrow a = 0.537 \\ a_2 \leftarrow b = 0.612 \\ a_3 \leftarrow c = 0.126 \end{cases}$$

for $i=1$ の中身

$i=1$

$$\begin{cases} y \leftarrow r(a_1 - cp) = r(0.537 - 0.0) = 0.537 \\ x \leftarrow r(sum + y) = r(0.0 + 0.537) = 0.537 \\ cp \leftarrow r(r(x - sum) - y) = r(r(0.537 - 0.0) - 0.537) = 0.0 \\ sum \leftarrow x = 0.537 \quad (\text{summationの途中: } sum = a_1) \end{cases}$$

i=2

$$\left\{ \begin{array}{l} y \leftarrow r(a_2 - cp) = r(0.612 - 0.0) = 0.612 \\ t \leftarrow r(sum + y) = r(0.537 + 0.612) = \underline{r(1.149)} = 1.14 \\ cp \leftarrow r(r(t - sum) - y) = r(r(1.14 - 0.537) - 0.612) = r(0.603 - 0.612) = \underline{-0.009} \\ sum \leftarrow t = 1.14 \quad (\text{summation の途中: } sum = a_1 + a_2) \end{array} \right.$$

↓
れの誤差 -0.009
↑
CP で誤差分の
埋め合せを保持しておく

i=3

$$\left\{ \begin{array}{l} y \leftarrow r(a_3 - cp) = r(0.126 - \underline{(-0.009)}) = r(0.135) = 0.135 \\ t \leftarrow r(sum + y) = r(1.14 + 0.135) = \underline{r(1.275)} = 1.27 \\ cp \leftarrow r(r(t - sum) - y) = r(r(1.27 - 1.14) - 0.135) = r(0.13 - 0.135) = \underline{-0.005} \\ sum \leftarrow t = 1.27 \quad (\text{summation の完了: } sum = a_1 + a_2 + a_3) \end{array} \right.$$

↓
れの誤差分を a_3 に足す
↑
埋め合せ

↓
誤差 -0.005
↑
CP で誤差分の
埋め合せを保持しておく

return sum = 1.27
//

例1.2

加算時10進4桁以下で切り捨ての場合

$$\left\{ \begin{array}{l} a_1 = 100 \\ a_2 = a_3 = \dots = a_{11} = 0.1 \end{array} \right.$$

で、与えられた a_1, \dots, a_n の総和を考える。普通に a_1 から順番に足すと a_2, \dots, a_n がすべて誤差で消えるため総和が a_1 と同じ100になってしまふ。

しかしカバンの加算アルゴリズムを使うと総和は正確に101となる。

この通り a_1, \dots, a_n をソートしてから値が大きいものから足していくと解が正しくなる。

例1.3

加算時10進4桁以下で切り捨ての場合

$$a_1 = \dots = a_{2000} = 0.1$$

で、与えられた a_1, \dots, a_{2000} の総和を考える。普通に a_1 から順番に足すと $a_{1001}, \dots, a_{2000}$ がすべて誤差で消えるため総和が100になってしまふ。

しかしカバンの加算アルゴリズムを使うと総和は正確に200となる。

この通りではソートして他の小さいやめる足す方法は使えない。しかし再帰的に2項ずつ足せば解が正しくなる。