数値計算法 第1回

数の表現

濱本 雄治 [1]

情報工学部 情報通信工学科

2025 年 4 月 14 日

[1] hamamoto@c.oka-pu.ac.jp

整数の 2 進数表現

- 10 進数の整数 $(x)_{10}$ を $2^0, 2^1, 2^2, \cdots$ で展開

$$\begin{aligned}(x)_{10} &= b_n \cdot 2^n + b_{n-1} \cdot 2^{n-1} + \cdots + b_0 \cdot 2^0 \\ &= (b_n b_{n-1} \cdots b_0)_2\end{aligned}$$

- c.f. 10 進数の整数は $10^0, 10^1, 10^2, \cdots$ の展開係数を並べたもの

$$\begin{aligned}(x)_{10} &= d_n \cdot 10^n + d_{n-1} \cdot 10^{n-1} + \cdots + d_0 \cdot 10^0 \\ &= (d_n d_{n-1} \cdots d_0)_{10}\end{aligned}$$

2 進整数を 10 進数に変換

方針

右から n 桁目の数字 (0 or 1) を 2^{n-1} に掛けてすべて足す

- 例: 2 進整数 00100101 を 10 進数で表す

$$1 \cdot 2^5 + 1 \cdot 2^2 + 1 \cdot 2^0 = 32 + 4 + 1 = 37$$

- 0 or 1 を n 個並べた 2 進数の情報量は n ビット

10 進整数を 2 進数に変換

方針

商を 2 で次々に割って出てきた余りを右から並べる

- 例: 10 進数 37 を 2 進数で表す

$$
\begin{array}{r|rlll}
2 & 37 & & & \\ \hline
2 & 18 & \cdots & \color{red}{1} & \leftarrow \color{red}{2^0\text{の桁}} \\ \hline
2 & 9 & \cdots & \color{red}{0} & \leftarrow \color{red}{2^1\text{の桁}} \\ \hline
2 & 4 & \cdots & \color{red}{1} & \leftarrow \color{red}{2^2\text{の桁}} \\ \hline
2 & 2 & \cdots & \color{red}{0} & \leftarrow \color{red}{2^3\text{の桁}} \\ \hline
2 & 1 & \cdots & \color{red}{0} & \leftarrow \color{red}{2^4\text{の桁}} \\ \hline
 & 0 & \cdots & \color{red}{1} & \leftarrow \color{red}{2^5\text{の桁}}
\end{array}
$$

余りを順に右から並べると $(37)_{10} = (100101)_2$

10進整数を2進数に変換

- なぜ上の手順でうまくいくのか？
- $(x)_{10} = \underbrace{b_n \cdot 2^n + b_{n-1} \cdot 2^{n-1} + \cdots}_{2\text{ の倍数}} + \underbrace{b_0 \cdot 2^0}_{0 \text{ or } 1}$ を 2 で次々に割ると

$$
\begin{aligned}
(x)_{10} \div 2 &= b_n \cdot 2^{n-1} + b_{n-1} \cdot 2^{n-2} + \cdots + b_1 \cdot 2^0 & \cdots \quad & b_0 \\
\text{上式の商} \div 2 &= b_n \cdot 2^{n-2} + b_{n-1} \cdot 2^{n-3} + \cdots + b_2 \cdot 2^0 & \cdots \quad & b_1 \\
&\vdots \\
\text{上式の商} \div 2 &= b_n & \cdots \quad & b_{n-1} \\
\text{上式の商} \div 2 &= 0 & \cdots \quad & b_n
\end{aligned}
$$

低い方の桁の数字から順に余りとして出てきた

小数の 2 進数表現

- 10 進数の小数 $(x)_{10}$ を $2^{-1}, 2^{-2}, 2^{-3}, \cdots$ で展開

$$
\begin{aligned}
(x)_{10} &= b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} + \cdots + b_{-n} \cdot 2^{-n} \\
&= (0.b_{-1}b_{-2}\cdots b_{-n})_2
\end{aligned}
$$

- c.f. 10 進数の小数は $10^{-1}, 10^{-2}, \cdots$ の展開係数を並べたもの

$$
\begin{aligned}
(x)_{10} &= d_{-1} \cdot 10^{-1} + d_{-2} \cdot 10^{-2} + \cdots + d_{-n} \cdot 10^{-n} \\
&= (0.d_{-1}d_{-2}\cdots d_{-n})_{10}
\end{aligned}
$$

2進小数を10進数に変換

方針

小数点以下 n 桁目の数字 (0 or 1) を 2^{-n} に掛けてすべて足す

- 例: 2進小数 0.1101 を 10 進数で表す

$$1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-4} = 0.5 + 0.25 + 0.0625 = 0.8125$$

10 進小数を 2 進数に変換

方針

小数部に 2 を次々に掛けて出てきた整数部を小数点以下に左から並べる

- 例: 10 進小数 0.8125 を 2 進数で表す

$$\begin{aligned}
0.8125 \times 2 &= 0.625 + 1 \quad &\leftarrow 2^{-1}\text{の桁}\\
0.625 \times 2 &= 0.25 + 1 \quad &\leftarrow 2^{-2}\text{の桁}\\
0.25 \times 2 &= 0.5 + 0 \quad &\leftarrow 2^{-3}\text{の桁}\\
0.5 \times 2 &= 0 + 1 \quad &\leftarrow 2^{-4}\text{の桁}
\end{aligned}$$

整数部を小数点以下に左から並べると $(0.8125)_{10} = (0.1101)_2$

10 進小数を 2 進数に変換

- なぜ上の手順でうまくいくのか？
- $(x)_{10} = b_{-1}2^{-1} + b_{-2}2^{-2} + \cdots + b_{-n}2^{-n}$ に 2 を次々に掛けると

$$(x)_{10} \times 2 = \underbrace{{\color{red}b_{-1}} \cdot 2^0}_{\text{整数部 0 or 1}} + \underbrace{b_{-2} \cdot 2^{-1} + \cdots + b_{-n} \cdot 2^{-n+1}}_{\text{小数部}}$$

$$\text{上式の小数部} \times 2 = {\color{red}b_{-2}} \cdot 2^0 + b_{-3} \cdot 2^{-1} + \cdots + b_{-n} \cdot 2^{-n+2}$$

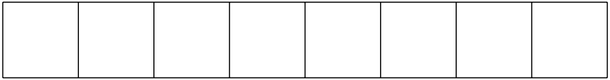
$$\vdots$$

$$\text{上式の小数部} \times 2 = {\color{red}b_{-n}} \cdot 2^0$$

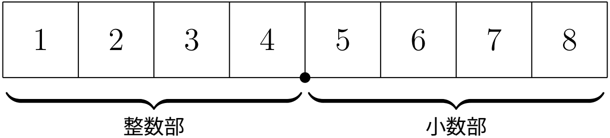
小数点以下の高い方の桁の数字から順に整数部に出てきた

固定小数点数

- 小数を 8 桁の 10 進数で表現する方法を考える (精度 8 桁)



- 仮に 4 桁目と 5 桁目の間に小数点を固定すると 1234.5678 は



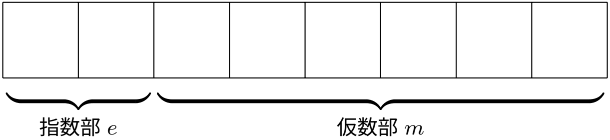
- この方法の問題
  - 9999.9999 より大きな小数が表現できない
  - 0.0001 より小さい小数が表現できない

浮動小数点数

- より広い小数を表現するために、2 桁を小数点の移動に使う
- 残りの 6 桁に有効数字を割り当てる
- 小数を次の形に変形する (正規化)

$$0.m \times 10^{e-50} \qquad (0 \le e \le 99)$$

ただし m は 6 つの数字の並びを表し、一番左の数字は 0 でない



浮動小数点数の例

- 例えば 1234.5678 の場合は

$$1234.5678 = 0.12345678 \times 10^{4} = 0.123456\cancel{78} \times 10^{54-50},$$

$$\therefore m = 123456, \qquad e = 54$$

$$\underbrace{\boxed{5}\boxed{4}}_{\text{指数部 } e}\underbrace{\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{6}}_{\text{仮数部 } m}$$

末尾の 78 は記憶できない

オーバーフローとアンダーフロー

- 指数部 2 桁、仮数部 6 桁で扱える浮動小数点数には限界がある
- 浮動小数点数の最大値

$$0.999999 \times 10^{+49} \simeq 10^{+49}$$

  これより大きな数は扱えない (オーバーフロー)
- 浮動小数点数の最小値

$$0.100000 \times 10^{-50} \simeq 10^{-51}$$

  これより小さい数は扱えない (アンダーフロー)

精度の比較

	固定小数点数	浮動小数点数
例	1234.5678	0.123456×10^4
最大値	9999.9999	$\simeq 10^{+49}$
最小値	0.0001	$\simeq 10^{-51}$
有効桁数	8	6

※ 表現できる範囲が増えた反面、有効桁数が減った点に注意

負の数も含めた浮動小数点数

- $(-1)^0 = +1, (-1)^1 = -1$ に注意
- 符号も含めた小数は

$$(-1)^s \times 0.m \times 10^{e-50} \qquad (s = 0, 1)$$

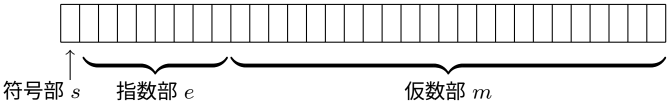
- 左端に符号を表現する桁を追加

- $s = 0, 1$ なので符号部は 1 ビット (2 進数 1 桁) でよい

現実の浮動小数点数

- 基数の種類
  - 2 進数
  - 16 進数
- 規格の種類
  - IBM 形式
  - IEEE 形式
- 精度の種類
  - 単精度 (32 ビット)
  - 倍精度 (64 ビット)

IEEE 単精度 (32 ビット) 浮動小数点数



- 符号部は 1 ビットの 2 進数で $s = 0, 1$
- 指数部は 8 ビットの 2 進数で $(00000000)_2 \leq e \leq (11111111)_2$
- 仮数部は残りの 23 ビット分だけ 0 or 1 を並べて表現
  - $(0.m)_2$ と表すと m の一番左は明らかに 1 なので無駄
  - $(1.m)_2$ と表して左から 2 桁目以降のみを考慮

$0 < e < 255$ のとき (両端 $e = 0, 255$ は含まない)

$$(-1)^s \times (1.m)_2 \times 2^{e-127}$$

(正の) 最大値と "最小値"

- 最大値

$$(-1)^0 \times (\underbrace{1.11\cdots1}_{24\text{ 桁}})_2 \times 2^{254-127} = (2 - 2^{-23}) \times 2^{127}$$
$$= 3.40\cdots \times 10^{38}$$

- "最小値"

$$(-1)^0 \times (\underbrace{1.00\cdots0}_{24\text{ 桁}})_2 \times 2^{1-127} = 2^{-126}$$
$$= 1.17\cdots 10^{-38}$$

C 言語で確認

overflow1.c

```
#include <stdio.h>
#include <float.h> /* FLT_MAX, FLT_MIN */
#include <math.h>  /* nextafterf */

int main(){
  printf("%.8e\n", FLT_MAX);
  printf("%.8e\n\n", nextafterf(FLT_MAX, INFINITY));
  printf("%.8e\n", FLT_MIN);
  printf("%.8e\n", nextafterf(FLT_MIN, 0.0f));
}
```

実行結果

```
$ gcc overflow1.c -lm
$ ./a.out
3.402823e+38
inf

1.17549435e-38
1.17549421e-38
```

- 最大値 (`FLT_MAX`) の 1 つ次の値は `inf`
- “最小値” (`FLT_MIN`) の 1 つ前の値はゼロにならない

"最小値" より小さい値

- 2^{-126} より小さい数は非正則数で表現

$$(-1)^s \times ({\color{red}0}.m)_2 \times 2^{-126}$$

ただし m は 23 個の 0 or 1 の任意の並び (一番左は 0 でもよい)

- 有効桁は 24 桁未満
- 真の (正の) 最小値

$$\begin{aligned}(-1)^0 \times (\underbrace{0.0\cdots0}_{\text{23 桁}}\,1)_2 \times 2^{-126} &= 2^{-149} \\ &= 1.40\cdots \times 10^{-45}\end{aligned}$$

C 言語で確認

overflow2.c

```
#include <stdio.h>
#include <float.h> /* FLT_MIN, FLT_TRUE_MIN */
#include <math.h>  /* nextafterf */

int main(){
  printf("%.8e\n", FLT_MIN);
  printf("%.8e\n", FLT_TRUE_MIN);
  printf("%.8e\n", nextafterf(FLT_TRUE_MIN, 0.0f));
}
```

実行結果

```
$ gcc overflow2.c -lm
$ ./a.out
1.17549435e-38
1.40129846e-45
0.00000000e+00
```

- 最小値 (FLT_TRUE_MIN) の 1 つ前の値はゼロ

IEEE 単精度浮動小数点数のまとめ

- $0 < e < 255$ のとき (再掲)

$$(-1)^s \times (1.m)_2 \times 2^{e-127}$$

- $e = 0$ or 255 のとき

	$m = 0$	$m \neq 0$
$e = 0$	符号付きゼロ $(-1)^s \times 0$	$(-1)^s \times (0.m)_2 \times 2^{-126}$
$e = 255$	符号付き無限大 $(-1)^s \times \infty$	非数 (Not a number, NaN)

IEEE 単精度浮動小数点数の例

1	1	0	0	0	0	0	1	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- 符号 $s = 1$
- 指数部 $e = (10000011)_2 = 1 \cdot 2^7 + 1 \cdot 2^1 + 1 \cdot 2^0 = 131$
- 仮数部 $(1.m)_2 = (1.010101)_2$
- よって上の IEEE 単精度浮動小数点数が表す値は

$$
\begin{aligned}
(-1)^s \times (1.m)_2 \times 2^{e-127} &= (-1)^1 \times (1.010101)_2 \times 2^{131-127} \\
&= -(1.010101)_2 \times 2^4 \\
&= -(10101.01)_2 \\
&= -(1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-2}) \\
&= -21.25
\end{aligned}
$$

IEEE 倍精度 (64 ビット) 浮動小数点数

- 符号部は 1 ビットの 2 進数で $s = 0, 1$
- 指数部は 11 ビットの 2 進数で

$$0 = (00000000000)_2 \leq e \leq (11111111111)_2 = 2047$$

- 仮数部は残りの 52 ビット分だけ 0 or 1 を並べて表現

$0 < e < 2047$ のとき (端の $e = 0, 1024$ は含まない)

$$(-1)^s \times (1.m)_2 \times 2^{e-1023}$$

整数型と実数型の比較

- 符号付き整数型 (32 ビット)

$$\underbrace{100\cdots0}_{32\text{ 桁}} = -2147483648 \sim \underbrace{011\cdots1}_{32\text{ 桁}} = 2147483647$$

- 単精度実数型

$$(-1)^s \times (1.m)_2 \times 2^{e-127}$$

	Fortran	C, Java
整数型	`INTEGER I`	`int i;`
単精度実数型	`REAL F`	`float f;`
倍精度実数型	`DOUBLE PRECISION D`	`double d;`