xjb: Fast Float to String Algorithm

ANONYMOUS* and ANONYMOUS*, Anonymous, Anonymous

Efficiently and accurately converting floating-point numbers to decimal strings is a critical challenge in numerical computation and data exchange. While existing algorithms like Ryū, Dragonbox, and Schubfach satisfy the Steele-White (SW) principle for accuracy, they often suffer from performance bottlenecks due to branch prediction failures and high-precision multiplication overhead. This paper presents a novel floating-point to string conversion algorithm called "xjb", an optimized variant of the Schubfach algorithm designed to deliver superior performance for IEEE754 single-precision (binary32) and double-precision (binary64) floating-point numbers. By minimizing instruction dependencies, reducing multiplication operations, mitigating branch prediction penalties and by utilizing the simd instruction set, xjb achieves significant performance gains. The algorithm features concise core implementation, parallel computing support, and excellent portability and scalability. Extensive benchmarking across diverse platforms, including AMD R7-7840H and Apple M1, demonstrates that xjb outperforms state-of-the-art algorithms in most scenarios while maintaining full compliance with the SW principle.

CCS Concepts: • **Computing methodologies → Representation of mathematical objects**.

Additional Key Words and Phrases: floating-point, printing, algorithm, performance

ACM Reference Format:
Anonymous and Anonymous. 2026. xjb: Fast Float to String Algorithm. *ACM Trans. Program. Lang. Syst.* 37, 4, Article 111 (August 2026), 29 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In 1990, Steele and White [1] published the paper *how to print floating-point numbers Accurately* and proposed the optimal principle of floating-point number printing algorithms (hereinafter referred to as the SW principle) :

- **Information preservation**: The print result can be parsed back to the original floating-point number.
- **Minimum length**: The print result should be as short as possible.
- **Correct rounding**: On the basis of satisfying 1 and 2, if there are two candidate values, they should be correctly rounded (i.e., the even value should be selected).
- **Generate from left to right**: The print result is generated from the left.

Floating-point number printing algorithms that satisfy the SW principle convert floating-point numbers into real values with unique and definite results. Over the past few years, a variety of different algorithms have been proposed, such as Grisu3[2], Errol[3], Ryū[4][5], Schubfach[6], Grisu-Exact[7], Dragonbox[8], and yy_double[9].

*Both authors contributed equally to this research.

Authors' address: Anonymous, Anonymous.com; Anonymous, Anonymous.cn, Anonymous, Anonymous, Anonymous, Anonymous.

ACM 0164-0925/2026/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

The algorithm in this paper is based on the Schubfach algorithm, and is inspired by algorithms such as yy_double and Dragonbox. This article only introduces two floating-point number types, IEEE754-binary32 and IEEE754-binary64. To simplify the content, in this article, float represents IEEE754-binary32 and double represents IEEE754-binary64. The article altogether contains nine python code files, and in this paper, the algorithm implementation code you can be found at https://github.com/xjb714/xjb.

2 IEEE754 FLOATING POINT NUMBER REPRESENTATION

Since the print result of a negative floating-point number only has one more negative sign than the print result of its absolute value, this article only discusses positive floating-point numbers and does not include special values such as 0, NaN, and Inf.

The IEEE754 double-precision floating-point number consists of 64 bits, including 1 sign bit ($sign$), 11 exponent bits (exp), and 52 fraction bits ($frac$). $sign$'s range is 0 or 1, exp's range is $[0, 2047]$, and $frac$'s range is $\left[0, 2^{52}-1\right]$.

The IEEE754 single-precision floating-point number consists of 32 bits, including 1 sign bit ($sign$), 8 exponent bits (exp), and 23 fraction bits ($frac$). $sign$'s range is 0 or 1, exp's range is $[0, 255]$, and $frac$'s range is $\left[0, 2^{23}-1\right]$.

When $frac = 0$, it is an irregular floating-point number.

The real value of the positive floating-point number v can be expressed as the following expression:

$$\begin{aligned} double{:}v &= \left(frac + \left(exp \neq 0?2^{52} : 0\right)\right) \cdot 2^{\max(exp,1)-1075} = c \cdot 2^q \\ float{:}v &= \left(frac + \left(exp \neq 0?2^{23} : 0\right)\right) \cdot 2^{\max(exp,1)-150} = c \cdot 2^q \end{aligned} \tag{1}$$

There are two cases in total. When exp equals 0 (referred to as subnormal floating-point numbers), there are:

$$\begin{aligned} double{:}v &= frac \cdot 2^{-1074} \\ float{:}v &= frac \cdot 2^{-149} \end{aligned} \tag{2}$$

When exp is not equal to 0 (referred to as a normal floating-point number), there is:

$$\begin{aligned} double{:}v &= \left(frac + 2^{52}\right) \cdot 2^{exp-1075} \\ float{:}v &= \left(frac + 2^{23}\right) \cdot 2^{exp-150} \end{aligned} \tag{3}$$

In the rounding interval R_v of floating-point numbers, all real numbers will be rounded to this floating-point number when parsed. R_v is:

$$\begin{aligned} v_l &= \begin{cases} \left(c - \dfrac{1}{2}\right) \cdot 2^q, \text{if } frac \neq 0 \text{ or } exp \leqslant 1 \\ \left(c - \dfrac{1}{4}\right) \cdot 2^q, \text{if } frac = 0 \end{cases} \\ v_r &= \left(c + \frac{1}{2}\right) \cdot 2^q \\ R_v &= \begin{cases} [v_l, v_r] \text{ ,if } frac\%2 = 0 \\ (v_l, v_r) \text{ ,if } frac\%2 = 1 \end{cases} \end{aligned} \tag{4}$$

When the floating-point number is a regular floating-point number, 2^{q-1} is the rounded radius.

3 PRINCIPLE OF ALGORITHM

At present, other algorithms use a large number of branches, which can easily lead to branch prediction failure penalties and excessive high multiplication overhead. The algorithm in this paper will minimize the overhead of branch prediction failures and reduce the number of multiplication operations to improve performance. Moreover, the core code for the algorithm implementation in this paper is very concise and it also supports parallel computing. The process of printing floating-point numbers is usually divided into two parts: the first part is to convert the floating-point number to a decimal number, and the second part is to convert the decimal number to a string. And this article will only introduce the first part. All double-precision floating-point numbers are classified into two types: irregular values and regular values. An irregular value is one where all the lower 52 bits are 0, meaning the $frac$ value is 0. There are a total of 2046 valid irregular values (i.e., exp values range from 1 to 2046). Dividing by the irregular values yields the regular value. Similarly, there are a total of 254 irregular values in a single-precision floating-point number. When exp is 0, it is called a subnormal floating-point number.

The valid range for c and q in regular floating-point numbers is:

$$\begin{aligned} float &: \begin{cases} 1 \leqslant c \leqslant 2^{24} - 1, c \neq 2^{23}; q = -149 \\ 2^{23} + 1 \leqslant c \leqslant 2^{24} - 1; -148 \leqslant q \leqslant 104 \end{cases} \\ double &: \begin{cases} 1 \leqslant c \leqslant 2^{53} - 1, c \neq 2^{52}; q = -1074 \\ 2^{52} + 1 \leqslant c \leqslant 2^{53} - 1; -1073 \leqslant q \leqslant 971 \end{cases} \end{aligned} \tag{5}$$

The valid range for c and q in irregular floating-point numbers is:

$$\begin{aligned} float &: \{c = 2^{23}; -149 \leqslant q \leqslant 104 \\ double &: \{c = 2^{52}; -1074 \leqslant q \leqslant 971 \end{aligned} \tag{6}$$

The valid range for c and q in subnormal floating-point numbers is:

$$\begin{aligned} float &: \{c \leqslant 2^{23} - 1; q = -149 \\ double &: \{c \leqslant 2^{52} - 1; q = -1074 \end{aligned} \tag{7}$$

Floating-point numbers that do not fall within the subnormal range are called normal floating-point numbers.

regular floating-point numbers account for the vast majority of all possible values of floating-point numbers and are the most worthy of discussion part. Therefore, unless otherwise specified, only regular floating-point numbers will be discussed below. Suppose the floating-point number v is converted to the optimal solution that satisfies the SW principle as opt, d is a positive integer and k is an integer,which is expressed as:

$$\begin{aligned} v = c \cdot 2^q &\longrightarrow opt = d \cdot 10^k \\ opt \in R_v; d &\in N^+; k \in Z \end{aligned} \tag{8}$$

For example: IEEE754-binary64 floating-point number "1.3", the real value of the floating-point number is 1.3000000000000000444089209850062616169452667236328125, hexadecimal representation of floating-point Numbers is 3ff4cccccccccccd, Then the opt value that meets the SW principle is 1.3. The IEEE754-binary32 floating-point number "1.3" has an actual value of 1.2999999523162841796875, and its hexadecimal representation is 3FA66666. Therefore, the opt value that satisfies the SW principle is 1.3.

3.1 Review the Schubfach algorithm and the derivation of the algorithm in this paper

According to the Schubfach algorithm, the possible values of d can be one of the following four situations:

$$10 \cdot \lfloor v \cdot 10^{-k-1} \rfloor, \lfloor 10 \cdot \left(v \cdot 10^{-k-1}\right) \rfloor, \lfloor 10 \cdot \left(v \cdot 10^{-k-1}\right) \rfloor + 1, 10 \cdot \lfloor v \cdot 10^{-k-1} \rfloor + 10 \tag{9}$$

The calculation method of k in equation (9) is as follows:

$$k = \lfloor q \cdot \lg(2) \rfloor \text{ if } v \in regular \text{ else } \lfloor q \cdot \lg(2) - \lg(\frac{4}{3}) \rfloor \tag{10}$$

In the range of float and double, equation (10) can be equivalent to:

$$\begin{aligned} double: k &= (q \cdot 315653 - (v \in regular?0 : 131072)) \gg 20 \\ float: k &= (q \cdot 1233 - (v \in regular?0 : 512)) \gg 12 \end{aligned} \tag{11}$$

Suppose the integer part of $v \cdot 10^{-k-1}$ is m and the decimal part is n, then we have:

$$\begin{aligned} &\lfloor v \cdot 10^{-k-1} \rfloor = m \\ &v \cdot 10^{-k-1} = m + n \\ &0 \leqslant n = v \cdot 10^{-k-1} - \lfloor v \cdot 10^{-k-1} \rfloor < 1 \end{aligned} \tag{12}$$

Then the decimal part of $v \cdot 10^{-k}$ is expressed as:

$$v \cdot 10^{-k} - \lfloor v \cdot 10^{-k} \rfloor = 10m + 10n - \lfloor 10m + 10n \rfloor = 10n - \lfloor 10n \rfloor \tag{13}$$

The possible values of d obtained from equation (9) are:

$$10m, \lfloor 10\,(m+n) \rfloor, \lfloor 10\,(m+n) \rfloor + 1, 10m + 10 \tag{14}$$

The possible values of d in equation (14) can be simplified to:

$$10m, 10m + \lfloor 10n \rfloor, 10m + \lfloor 10n \rfloor + 1, 10m + 10 \tag{15}$$

Among them, $10m$ represents the minimum possible value and $10m + 10$ represents the maximum possible value. Suppose ten is used to represent $10m$. There are four possible values for one, with $d = ten + one$, denoted as:

$$\begin{aligned} &ten = 10m \\ &one \in \{0, \lfloor 10n \rfloor, \lfloor 10n \rfloor + 1, 10\} \\ &d = ten + one \end{aligned} \tag{16}$$

Calculating d will be converted to calculating ten and one.

The final possible values of d are as follows:

- $10m$
  When the following conditions are met, the result is $10m$ (or equivalent to $one = 0$). That is, the floating-point number v minus the minimum possible value of $10m$ is less than the rounded radius 2^{q-1}.

$$\begin{aligned} c \cdot 2^q - 10m \cdot 10^k &< 2^{q-1} \\ c \cdot 2^q - \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor \cdot 10^{k+1} &< 2^{q-1} \\ c \cdot 2^q \cdot 10^{-k-1} - \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor &< 2^{-1} \cdot 2^q \cdot 10^{-k-1} \\ n &< 2^{-1} \cdot 2^q \cdot 10^{-k-1} \\ 2^{-1} \cdot 2^q \cdot 10^{-k-1} &> n \end{aligned} \tag{17}$$

  Or when $2^{-1} \cdot 2^q \cdot 10^{-k-1} = n$, $c\%2 = 0$ must also be satisfied. Therefore, the following conditions are valid:

$$\text{if } 2^{-1} \cdot 2^q \cdot 10^{-k-1} > n \text{ or } \left(2^{-1} \cdot 2^q \cdot 10^{-k-1} = n \text{ \&\& } c\%2 = 0\right) : one = 0 \tag{18}$$

- $10m + 10$
  When the following conditions are met, the result is $10m + 10$ (or equivalent to $one = 10$). The maximum possible value of $10m + 10$ minus the floating-point number v is less than the rounded radius 2^{q-1}.

$$
\begin{aligned}
(10m + 10) \cdot 10^k - c \cdot 2^q &< 2^{q-1} \\
\lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor \cdot 10^{k+1} + 10^{k+1} - c \cdot 2^q &< 2^{q-1} \\
\lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor - c \cdot 2^q \cdot 10^{-k-1} + 1 &< 2^{-1} \cdot 2^q \cdot 10^{-k-1} \\
1 - n &< 2^{-1} \cdot 2^q \cdot 10^{-k-1} \\
2^{-1} \cdot 2^q \cdot 10^{-k-1} &> 1 - n
\end{aligned} \tag{19}
$$

  Or when $2^{-1} \cdot 2^q \cdot 10^{-k-1} = 1 - n$, $c\%2 = 0$ must also be satisfied. Therefore, the following conditions are valid:

$$
\text{if } 2^{-1} \cdot 2^q \cdot 10^{-k-1} > 1 - n \text{ or } \left(2^{-1} \cdot 2^q \cdot 10^{-k-1} = 1 - n \;\&\&\; c\%2 = 0\right) : one = 10 \tag{20}
$$

- $10m + \lfloor 10n \rfloor$ **or** $10m + \lfloor 10n \rfloor + 1$
  When none of the conditions are met as $d = 10m$ or $d = 10m + 10$, d is either $10m + \lfloor 10n \rfloor$ or $10m + \lfloor 10n \rfloor + 1$. The final value is determined based on the decimal part of $10n$. If the decimal part is 0.5, it is rounded to the nearest even value; if it is not 0.5, it is rounded to the nearest value. For irregular floating-point numbers, it is also necessary to determine whether $10m + \lfloor 10n \rfloor$ is within the rounding interval R_v. If it is not, then $10m + \lfloor 10n \rfloor + 1$.

In summary, the steps of the Schubfach algorithm variants are as follows process (21), that is, the algorithms proposed in this paper. This algorithm process (21) is applicable to float and double floating-point numbers(xjb32 for float, xjb64 for double). Taking a floating-point number v as input, c and q are extracted, and the calculation results d (line 15) and k (line 2) are returned. The real value represented by the returned results is $d \cdot 10^k$, which conforms to the SW principle. The calculation process of k is relatively simple and can be obtained from (11). Therefore, the following only focuses on introducing the rapid calculation process of d. The following will be divided into five parts to introduce the algorithm process (21) :

(1) Introduce the pre-computation process of the algorithm's lookup table.
(2) Quickly calculate m.
(3) Quickly determine whether $one = 0$ or $one = 10$.
(4) Quickly calculate $\lfloor 10n \rfloor$ and determine whether $one = \lfloor 10n \rfloor$ or $one = \lfloor 10n \rfloor + 1$ based on the decimal part of $10n$.
(5) Processing of irregular floating-point numbers.

The following will discuss the above content in detail from Section 3.2 to Section 3.6.

$$
\begin{aligned}
&\text{input} : c, q\\
&\text{output} : d, k\\
&(\ 1)\ v = c \cdot 2^q\\
&(\ 2)\ k = \lfloor q \cdot \lg(2) \rfloor \text{ if } v \in \mathit{regular} \text{ else } \lfloor q \cdot \lg(2) - \lg(\frac{4}{3}) \rfloor\\
&(\ 3)\ m = \lfloor v \cdot 10^{-k-1} \rfloor, n = v \cdot 10^{-k-1} - m\\
&(\ 4)\ \mathit{ten} = 10m\\
&(\ 5)\ \text{if } 10n - \lfloor 10n \rfloor = 0.5 : \mathit{one} = \lfloor 10n \rfloor \text{ if } (\lfloor 10n \rfloor \% 2 = 0) \text{ else } \lfloor 10n \rfloor + 1\\
&(\ 6)\ \text{if } 10n - \lfloor 10n \rfloor < 0.5 : \mathit{one} = \lfloor 10n \rfloor\\
&(\ 7)\ \text{if } 10n - \lfloor 10n \rfloor > 0.5 : \mathit{one} = \lfloor 10n \rfloor + 1\\
&(\ 8)\ \text{if } v \in \mathit{irregular} :\\
&(\ 9)\quad \text{if } 10n - \lfloor 10n \rfloor > 2^{q-2} \cdot 10^{-k} : \mathit{one} = \lfloor 10n \rfloor + 1\\
&(10)\quad \text{if } 2^{q-2} \cdot 10^{-k-1} \geqslant n : \mathit{one} = 0\\
&(11)\ \text{else} :\\
&(12)\quad \text{if } 2^{q-1} \cdot 10^{-k-1} > n \text{ or } \left(2^{q-1} \cdot 10^{-k-1} = n \ \&\&\ c\%2 = 0\right) : \mathit{one} = 0\\
&(13)\ \text{endif}\\
&(14)\ \text{if } 2^{q-1} \cdot 10^{-k-1} > 1 - n \text{ or } \left(2^{q-1} \cdot 10^{-k-1} = 1 - n \ \&\&\ c\%2 = 0\right) : \mathit{one} = 10\\
&(15)\ d = \mathit{ten} + \mathit{one}
\end{aligned}
\tag{21}
$$

3.2 Pre-computation of Lookup Table

The algorithm in this paper uses a lookup table to store the values of 10^{-k-1} for q in the range of $[-149, 104]$ for float and $[-1074, 971]$ for double. In the algorithm of this paper, float uses 64-bit precision and double uses 128-bit precision lookup tables. The code implementation in this section is gen.py. Suppose the bit length of a single value data in the lookup table is B. For float, it has $B = 64$, and for double, it has $B = 128$. Suppose there are integers e_{10} and real numbers e_2, where $1 \leqslant f < 2$. There are:

$$f \cdot 2^{\lfloor e_2 \rfloor} = 2^{e_2} = 10^{e_{10}} \tag{22}$$

Then:

$$\lfloor e_2 \rfloor = \lfloor e_{10} \cdot \log_2(10) \rfloor \tag{23}$$

The calculation leads to f, and the following conclusions are drawn:

$$f = \frac{10^{e_{10}}}{2^{\lfloor e_{10} \cdot \log_2(10) \rfloor}} \tag{24}$$

The way to calculate the lookup table is as follows (using the upward rounding method) :

$$\mathit{lookup}[e_{10}] = \lceil f \cdot 2^{B-1} \rceil = \lceil \frac{10^{e_{10}}}{2^{\lfloor e_{10} \cdot \log_2(10) \rfloor}} \cdot 2^{B-1} \rceil = \lceil 10^{e_{10}} \cdot 2^{B-1-\lfloor e_{10} \cdot \log_2(10) \rfloor} \rceil \tag{25}$$

For float, when $0 \leqslant e_{10} \leqslant 27$, $f \cdot 2^{B-1}$ is an integer in equation (25). For double, when $0 \leqslant e_{10} \leqslant 55$, $f \cdot 2^{B-1}$ is an integer in equation (25). The detailed calculation process is as follows:

- Float

The range of $-k-1$ is calculated to be [-32, 44] through the q value range in equation (5), so the lookup table contains representation values from 10 to the power of -32 to 10 to the power of 44. The calculation process is as follows:

$$
\begin{aligned}
&-32 \leqslant e_{10} \leqslant 44 \\
&e_2 = \left|\lfloor e_{10} \cdot \log_2(10) \rfloor - 63\right| \\
&pow10t = \begin{cases} 2^{e_2} // 10^{|e10|}; \text{if } e_{10} < 0 \\ 10^{|e10|} // 2^{e_2}; \text{if } e_{10} \geqslant 20 \\ 10^{|e10|} \cdot 2^{e_2}; \text{if } 1 \leqslant e_{10} \leqslant 19 \end{cases} \\
&f_{1,e_{10}} = pow10 = pow10t + (e_{10} \geqslant 0 \&\& e_{10} \leqslant 27 ? 0 : 1)
\end{aligned}
\tag{26}
$$

When $0 \leqslant e_{10} \leqslant 27$, the lookup table variable indicates that the values $f_{1,e_{10}} \cdot 2^{\lfloor e_{10} \cdot \log_2(10) \rfloor - 63}$ and $10^{e_{10}}$ are equal. In other cases, the relative error is less than 2^{-63}. Expressed as:

$$
\begin{aligned}
r_{1,e_{10}} &= \frac{f_{1,e_{10}} \cdot 2^{\lfloor e_{10} \cdot \log_2(10) \rfloor - 63}}{10^{e_{10}}} \\
&\in \begin{cases} 1; \text{if } 0 \leqslant e_{10} \leqslant 27 \\ \left(1, 1 + 2^{-63}\right); \text{if } e_{10} < 0 \text{ or } e_{10} > 27 \end{cases}
\end{aligned}
\tag{27}
$$

- Double

  The range of $-k-1$ is calculated to be [-293, 323] through the q value range in equation (5), so the lookup table contains representation values from 10 to the power of -293 to 10 to the power of 323. The calculation process is as follows:

$$
\begin{aligned}
&-293 \leqslant e_{10} \leqslant 323 \\
&e_2 = \left|\lfloor e_{10} \cdot \log_2(10) \rfloor - 127\right| \\
&pow10t = \begin{cases} 2^{e_2} // 10^{|e10|}; \text{if } e_{10} < 0 \\ 10^{|e10|} // 2^{e_2}; \text{if } e_{10} \geqslant 39 \\ 10^{|e10|} \cdot 2^{e_2}; \text{if } 1 \leqslant e_{10} \leqslant 38 \end{cases} \\
&f_{1,e_{10}} = pow10 = pow10t + (e_{10} \geqslant 0 \&\& e_{10} \leqslant 55 ? 0 : 1)
\end{aligned}
\tag{28}
$$

When $0 \leqslant e_{10} \leqslant 55$, the lookup table variable indicates that the values $f_{1,e_{10}} \cdot 2^{\lfloor e_{10} \cdot \log_2(10) \rfloor - 127}$ and $10^{e_{10}}$ are equal. In other cases, the relative error is less than 2^{-127}. Expressed as:

$$
\begin{aligned}
r_{1,e_{10}} &= \frac{f_{1,e_{10}} \cdot 2^{\lfloor e_{10} \cdot \log_2(10) \rfloor - 127}}{10^{e_{10}}} \\
&\in \begin{cases} 1; \text{if } 0 \leqslant e_{10} \leqslant 55 \\ \left(1, 1 + 2^{-127}\right); \text{if } e_{10} < 0 \text{ or } e_{10} > 55 \end{cases}
\end{aligned}
\tag{29}
$$

The following uses r_1 to represent all possible errors of the lookup table values within the float range, r_2 to represent all possible errors of the lookup table values within the double range, and r to represent all possible errors of the lookup table values within either the float or double range. In algorithm process (21), an approximate representation value of 10 to the power of $-k-1$ needs to be obtained through a lookup table. From equation (27) and equation (29), the lookup table representation value is error-free when q is within the following range:

$$
\begin{aligned}
float &: 0 \leqslant -k-1 \leqslant 27 \Rightarrow -93 \leqslant q \leqslant -1 \\
double &: 0 \leqslant -k-1 \leqslant 55 \Rightarrow -186 \leqslant q \leqslant -1
\end{aligned}
\tag{30}
$$

When q is not within the range of equation (30), the error range of the value represented by the lookup table can be concluded as follows:

$$\begin{aligned} float &: 0 < r_1 - 1 < 2^{-63} \\ double &: 0 < r_2 - 1 < 2^{-127} \end{aligned} \tag{31}$$

The introduction of the lookup table calculation process is complete. The storage space required for a float range lookup table is 616 bytes, and that for a double range lookup table is 9872 bytes.

3.3 Quickly Calculate m

Relevant theorems (partially from the Dragonbox algorithm paper) : Suppose there are positive integers n,P,and Q, where P and Q are coprime, $P < Q$, $1 \leqslant n \leqslant n_{max}$,$Q > n_{max}$,$P^*/Q^*$ is the best rational approximation result greater than or equal to P/Q, P_*/Q_* is the best rational approximation result less than or equal to P/Q, and it satisfies $Q^* \leqslant n_{max}$, $Q_* \leqslant n_{max}$. And if $n \cdot P$ does not divide Q evenly, it is expressed as:

$$\lfloor n \cdot \frac{P}{Q} \rfloor + 1 = \lceil n \cdot \frac{P}{Q} \rceil \tag{32}$$

Suppose the following holds true:

$$\lfloor n \cdot \frac{P}{Q} \rfloor = \lfloor n \cdot \xi \rfloor \tag{33}$$

Then there are:

$$\frac{P_*}{Q_*} = \max_{1 \leqslant n \leqslant n_{\max}} \frac{\lfloor n \cdot \frac{P}{Q} \rfloor}{n} \leqslant \xi < \min_{1 \leqslant n \leqslant n_{\max}} \frac{\lfloor n \cdot \frac{P}{Q} \rfloor + 1}{n} = \min_{1 \leqslant n \leqslant n_{\max}} \frac{\lceil n \cdot \frac{P}{Q} \rceil}{n} = \frac{P^*}{Q^*} \tag{34}$$

Therefore, the range of values for ξ is:

$$\frac{P_*}{Q_*} \leqslant \xi < \frac{P^*}{Q^*} \tag{35}$$

And the range of the decimal part with $n \cdot \frac{P}{Q}$ is:

$$\left[\frac{(Q_* P) \,\%Q}{Q}, \frac{(Q^* P) \,\%Q}{Q} \right] \tag{36}$$

That is, when $n = Q_*$, the decimal part is the smallest; when $n = Q^*$, the decimal part is the largest.

The definition of the best rational approximation function is as follows (this function is implemented on line 15 of the test1.py file):

$$(DN, UP) = f\,(C, P, Q) \tag{37}$$

The function (37) calculate the best rational approximation result with a denominator not exceeding C based on the mean term theorem of the Farey sequence. DN and UP are two adjacent terms in the C-order Farey sequence F_C.

In algorithm process (21), m is calculated as $\lfloor v \cdot 10^{-k-1} \rfloor$ (line 3). Just prove that the following equation holds:

$$m = \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor c \cdot 2^q \cdot r \cdot 10^{-k-1} \rfloor \tag{38}$$

Where r is the error of the lookup table value, as defined in equation (27) and equation (29). When the condition (30) is met, r is 1, and the equation (38) clearly holds. When r is not 1, there is:

$$\begin{aligned} float &: 1 < r < 1 + 2^{-63} \\ double &: 1 < r < 1 + 2^{-127} \end{aligned} \tag{39}$$

Calculate the range of $2^q \cdot 10^{-k-1}$ and we get:

$$2^q \cdot 10^{-k-1} = 10^{-1} \cdot \left(10^{q \cdot \lg(2) - \lfloor q \cdot \lg(2) \rfloor}\right) \tag{40}$$

When q is not 0, equation (40) exists:

$$\begin{aligned} q \cdot \lg(2) &\neq \lfloor q \cdot \lg(2) \rfloor \\ 0 < q \cdot \lg(2) &- \lfloor q \cdot \lg(2) \rfloor < 1 \end{aligned} \tag{41}$$

When q is 0, $q \cdot \lg(2) - \lfloor q \cdot \lg(2) \rfloor = 0$, so the final conclusion is:

$$10^{-1} \leqslant 2^q \cdot 10^{-k-1} < 1 \tag{42}$$

Because there is:

$$c \cdot 2^q \cdot 10^{-k-1} = c \cdot \frac{2^{q-k-1}}{5^{k+1}} \in [0.1c\,, c) \tag{43}$$

Therefore:

$$c \cdot 2^q \cdot 10^{-k-1} = \begin{cases} \frac{c \cdot 2^{q-k-1}}{5^{k+1}}; q \geqslant 1 \\ \frac{c}{2^{1+k-q} \cdot 5^{k+1}} = \frac{c}{10}; q = 0 \\ \frac{c \cdot 5^{-k-1}}{2^{1+k-q}}; q < 0 \end{cases} \tag{44}$$

Suppose:

$$c \cdot 2^q \cdot 10^{-k-1} = c \cdot \frac{x}{y} < c \tag{45}$$

Then there are:

$$(x, y) = \begin{cases} \left(2^{q-k-1}, 5^{k+1}\right); q \geqslant 1 \\ (1, 10); q = 0 \\ \left(5^{-k-1}, 2^{1+k-q}\right); q < 0 \end{cases} \tag{46}$$

Suppose:

$$\begin{aligned} float : c \leqslant c_{\max} = C_1 = 2^{24} - 1 \\ double : c \leqslant c_{\max} = C_2 = 2^{53} - 1 \end{aligned} \tag{47}$$

The following is represented by C as C_1 or C_2. C within the float range is C_1, and C within the double range is C_2.

When $y > C$, calculate the P^* and Q^* corresponding to each q by calling $f(C, x, y)$ according to function (37). And calculate the minimum BIT value when the following conditions are met:

$$\frac{x}{y}\left(1 + 2^{-BIT}\right) < \frac{P^*}{Q^*} \tag{48}$$

When $y \leqslant C$, there is:

$$c \cdot \frac{x}{y}\left(1 + \frac{1}{Cy}\right) = \frac{cx + \frac{c}{C} \cdot \frac{x}{y}}{y} < \frac{cx + 1}{y} \tag{49}$$

Therefore:

$$\lfloor c \cdot \frac{x}{y} \rfloor = \lfloor c \cdot \frac{x}{y}\left(1 + \frac{1}{Cy}\right) \rfloor \tag{50}$$

Similarly, calculate the minimum BIT value:

$$\frac{x}{y}\left(1 + 2^{-BIT}\right) < \frac{x}{y}\left(1 + \frac{1}{Cy}\right) \tag{51}$$

In summary, the calculation results of the maximum value among the minimum BIT values corresponding to different q are as follows (the running result is in the test1.py file, and the running time of this code is only about 1 to 2 seconds) :

$$\begin{aligned} float &: BIT_{\max} = 52 \\ double &: BIT_{\max} = 113 \end{aligned} \tag{52}$$

Therefore, the following conclusions exist:

$$\begin{aligned} float &: \lfloor c \cdot \frac{x}{y} \rfloor = \lfloor c \cdot \frac{x}{y} \cdot \left(1 + 2^{-52}\right) \rfloor = \lfloor c \cdot \frac{x}{y} \cdot r_1 \rfloor \\ double &: \lfloor c \cdot \frac{x}{y} \rfloor = \lfloor c \cdot \frac{x}{y} \cdot \left(1 + 2^{-113}\right) \rfloor = \lfloor c \cdot \frac{x}{y} \cdot r_2 \rfloor \end{aligned} \tag{53}$$

This section has been verified.After quickly calculating m, the value of $ten = 10m$ can be obtained very quickly.

3.4 Quickly Determine Whether $one = 0$ or $one = 10$

In algorithm process (21), the conditions for determining $one = 0$ and $one = 10$ are on lines 12, and 14. This section will introduce how to quickly determine whether $one = 0$ or $one = 10$ holds by using equivalent conditions.

When discussing the case of $2^{-1} \cdot 2^q \cdot 10^{-k-1} = n$ (line 12, one might be 0), it is equivalent to:

$$\begin{aligned} c \cdot 2^q \cdot 10^{-k-1} - \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor &= 2^{-1} \cdot 2^q \cdot 10^{-k-1} \\ (2c-1) \cdot 2^{q-1} \cdot 10^{-k-1} &= \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor \end{aligned} \tag{54}$$

When discussing the case of $2^{-1} \cdot 2^q \cdot 10^{-k-1} = 1 - n$ (line 14, one might be 10), it is equivalent to:

$$\begin{aligned} \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor - c \cdot 2^q \cdot 10^{-k-1} + 1 &= 2^{-1} \cdot 2^q \cdot 10^{-k-1} \\ (2c+1) \cdot 2^{q-1} \cdot 10^{-k-1} &= \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor + 1 \end{aligned} \tag{55}$$

Since equation (42), we have:

$$2^{q-1} \cdot 10^{-k-1} \in [0.05\ , 0.5) \tag{56}$$

Therefore, there is:

$$\begin{aligned} \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor - 1 &< c \cdot 2^q \cdot 10^{-k-1} - 0.5 \\ &< (2c-1) \cdot 2^{q-1} \cdot 10^{-k-1} \\ &\leqslant c \cdot 2^q \cdot 10^{-k-1} - 0.05 < \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor + 1 \end{aligned} \tag{57}$$

Therefore, for equation (54), when $(2c-1) \cdot 2^{q-1} \cdot 10^{-k-1}$ is an integer, it must be equal to $\lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor$. Similarly, for equation (55), there is:

$$\begin{aligned} \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor &< c \cdot 2^q \cdot 10^{-k-1} + 0.05 \\ &\leqslant (2c+1) \cdot 2^{q-1} \cdot 10^{-k-1} \\ &< c \cdot 2^q \cdot 10^{-k-1} + 0.5 < \lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor + 2 \end{aligned} \tag{58}$$

Therefore, for equation (55), when $(2c+1) \cdot 2^{q-1} \cdot 10^{-k-1}$ is an integer, it must be equal to $\lfloor c \cdot 2^q \cdot 10^{-k-1} \rfloor + 1$.

In conclusion, it is equivalent to discussing whether $(2c \pm 1) \cdot 2^{q-1} \cdot 10^{-k-1}$ is an integer. There are:

$$(2c \pm 1) \cdot 2^{q-1} \cdot 10^{-k-1} = (2c \pm 1) \cdot 2^{q-k-2} \cdot 5^{-k-1} \tag{59}$$

According to the range of q, there are:

$$\begin{cases} q-k-2 \geqslant 0, -k-1 < 0; q \geqslant 2 \\ q-k-2 < 0, -k-1 < 0; 1 \geqslant q \geqslant 0 \\ q-k-2 < 0, -k-1 \geqslant 0; q < 0 \end{cases} \tag{60}$$

Therefore, equation (59) is equivalent to:

$$(2c \pm 1) \cdot 2^{q-1} \cdot 10^{-k-1} = \begin{cases} \frac{(2c\pm1)\cdot 2^{q-k-2}}{5^{k+1}}; q \geqslant 2 \\ \frac{(2c\pm1)}{2^{2+k-q}\cdot 5^{k+1}}; 1 \geqslant q \geqslant 0 \\ \frac{(2c\pm1)\cdot 5^{-k-1}}{2^{2+k-q}}; q < 0 \end{cases} \tag{61}$$

According to the different ranges of q, the following situations are discussed:

- $q \geqslant 2$
  From $q \geqslant 2$, we get $k \geqslant 0$.When $q \geqslant 2$, it is equivalent to discussing whether $(2c \pm 1)\cdot 2^{q-k-2}$ is divisible by 5^{k+1}. Since 2 and 5 are coprime, it is equivalent to discussing whether $(2c \pm 1)$ is divisible by 5^{k+1}.

$$(2c \pm 1) \ \ \%5^{k+1} = 0 \tag{62}$$

  Suppose t is a positive integer:

$$2c \pm 1 = t \cdot 5^{k+1}; t \geqslant 1 \tag{63}$$

  Since $2c \pm 1$ is odd, t is also odd. Because the following conditions exist:

$$\begin{aligned} float &: 2c-1 \in \left[2^{24}+1, 2^{25}-3\right]; 2c+1 \in \left[2^{24}+3, 2^{25}-1\right]; \\ double &: 2c-1 \in \left[2^{53}+1, 2^{54}-3\right]; 2c+1 \in \left[2^{53}+3, 2^{54}-1\right]; \end{aligned} \tag{64}$$

  Therefore, the following satisfies:

$$\begin{aligned} float &: 2^{24}+1 \leqslant t \cdot 5^{k+1} \leqslant 2^{25}-1 \\ double &: 2^{53}+1 \leqslant t \cdot 5^{k+1} \leqslant 2^{54}-1 \end{aligned} \tag{65}$$

  Therefore, the following conclusions are drawn:

$$\begin{aligned} float &: \frac{2^{24}+1}{5^{k+1}} \leqslant t \leqslant \frac{2^{25}-1}{5^{k+1}}; \\ double &: \frac{2^{53}+1}{5^{k+1}} \leqslant t \leqslant \frac{2^{54}-1}{5^{k+1}}; \end{aligned} \tag{66}$$

  For the above equation (66), the maximum value of k when t can obtain at least one odd number is:

$$\begin{aligned} float &: k_{\max} = 9 \Rightarrow q_{\max} = 33, t = 3 \\ double &: k_{\max} = 22 \Rightarrow q_{\max} = 76, t = 1 \end{aligned} \tag{67}$$

  Therefore, the maximum value of k is 9 within the float range and 22 within the double range. Therefore, when k exceeds the above range, $(2c \pm 1)$ is not divisible by 5^{k+1}.
- $1 \geqslant q \geqslant 0$
  Because the denominator $2^{2+k-q} \cdot 5^{k+1}$ is even and the numerator $(2c \pm 1)$ is odd, the condition is not met.
- $q < 0$
  Because the denominator 2^{2+k-q} is even and the numerator $(2c \pm 1) \cdot 5^{-k-1}$ is odd, the condition is not met.

In summary, the situations when $(2c \pm 1) \cdot 2^{q-1} \cdot 10^{-k-1}$ is an integer are as follows:

$$\begin{aligned} float &: 2 \leqslant q \leqslant 33 \ \&\& \ (2c \pm 1) \,\%5^{k+1} = 0; \\ double &: 2 \leqslant q \leqslant 76 \ \&\& \ (2c \pm 1) \,\%5^{k+1} = 0; \end{aligned} \tag{68}$$

And, the range of $-k-1$ is:

$$\begin{aligned} float &: -10 \leqslant -k-1 \leqslant -1 \\ double &: -23 \leqslant -k-1 \leqslant -1 \end{aligned} \tag{69}$$

When $2^{-1} \cdot 2^q \cdot 10^{-k-1} = n$, the following conclusions can be drawn:

$$\begin{aligned} float &: \left\{ 2^{35} \cdot 2^q \cdot 10^{-k-1} = 2^{36} \cdot n \Rightarrow \lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} \cdot n \rfloor \right. \\ double &: \left\{ 2^{63} \cdot 2^q \cdot 10^{-k-1} = 2^{64} \cdot n \Rightarrow \lfloor 2^{63} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{64} \cdot n \rfloor \right. \end{aligned} \tag{70}$$

When $2^{-1} \cdot 2^q \cdot 10^{-k-1} = 1 - n$, the following conclusions can be drawn:

$$\begin{aligned} float &: \left\{ \begin{array}{c} 2^{35} \cdot 2^q \cdot 10^{-k-1} = 2^{36} - 2^{36} \cdot n \Rightarrow \\ \lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} - 2^{36} \cdot n \rfloor = 2^{36} - 1 - \lfloor 2^{36} \cdot n \rfloor \end{array} \right. \\ double &: \left\{ \begin{array}{c} 2^{63} \cdot 2^q \cdot 10^{-k-1} = 2^{64} - 2^{64} \cdot n \Rightarrow \\ \lfloor 2^{63} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{64} - 2^{64} \cdot n \rfloor = 2^{64} - 1 - \lfloor 2^{64} \cdot n \rfloor \end{array} \right. \end{aligned} \tag{71}$$

The discussion on whether $\lfloor 2^{36} - 2^{36} \cdot n \rfloor = 2^{36} - 1 - \lfloor 2^{36} \cdot n \rfloor$ in equation (71) holds true, that is, whether $2^{36} \cdot n$ in equation (71) is an integer, or equivalent to discussing whether the following values are integers when equation (68) holds true (the same applies to double) :

$$\begin{aligned} float &: 2^{36} \cdot (m+n) = c \cdot 2^{q+36} \cdot 10^{-k-1} = c \cdot 2^{q-k+35} \cdot 5^{-k-1} = c \cdot \frac{2^{q-k+35}}{5^{k+1}} \\ double &: 2^{64} \cdot (m+n) = c \cdot 2^{q+64} \cdot 10^{-k-1} = c \cdot 2^{q-k+63} \cdot 5^{-k-1} = c \cdot \frac{2^{q-k+63}}{5^{k+1}} \end{aligned} \tag{72}$$

Suppose c can divide 5^{k+1} evenly (where t is a temporary integer variable):

$$c = t \cdot 5^{k+1}; t \geqslant 1 \tag{73}$$

Therefore, when equation (73) was established, there were:

$$2c \pm 1 = 2 \cdot t \cdot 5^{k+1} \pm 1 \tag{74}$$

Expression (74) cannot divide 5^{k+1} evenly, which contradicts equation (68), so c cannot divide 5^{k+1} evenly. Therefore, for float, $c \cdot 2^{q+36} \cdot 10^{-k-1}$ and $2^{36} \cdot n$ are not integers; For double, $c \cdot 2^{64+q} \cdot 10^{-k-1}$ and $2^{64} \cdot n$ are not integers, that is:

$$\begin{aligned} float &: \lfloor 2^{36} - 2^{36} \cdot n \rfloor = 2^{36} + \lfloor -2^{36} \cdot n \rfloor = 2^{36} - 1 - \lfloor 2^{36} \cdot n \rfloor \\ double &: \lfloor 2^{64} - 2^{64} \cdot n \rfloor = 2^{64} + \lfloor -2^{64} \cdot n \rfloor = 2^{64} - 1 - \lfloor 2^{64} \cdot n \rfloor \end{aligned} \tag{75}$$

Therefore, the conclusion (71) is correct. Discuss the necessary and sufficient conditions for whether $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} \cdot n \rfloor$ is $2^{-1} \cdot 2^q \cdot 10^{-k-1} = n$. The same applies to double, expressed as:

$$\begin{aligned} float &: 2^{-1} \cdot 2^q \cdot 10^{-k-1} = n \Leftrightarrow \lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} \cdot n \rfloor \\ double &: 2^{-1} \cdot 2^q \cdot 10^{-k-1} = n \Leftrightarrow \lfloor 2^{63} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{64} \cdot n \rfloor \end{aligned} \tag{76}$$

Similarly, the necessary and sufficient conditions for whether $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} - 2^{36} \cdot n \rfloor$ is $2^{-1} \cdot 2^q \cdot 10^{-k-1} = 1 - n$. The same applies to double, expressed as:

$$\begin{aligned} float &: 2^{-1} \cdot 2^q \cdot 10^{-k-1} = 1 - n \Leftrightarrow \lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} - 2^{36} \cdot n \rfloor \\ double &: 2^{-1} \cdot 2^q \cdot 10^{-k-1} = 1 - n \Leftrightarrow \lfloor 2^{63} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{64} - 2^{64} \cdot n \rfloor \end{aligned} \tag{77}$$

The sufficient conditions of equations (76) and (77) are obviously established. Introduce the proof that equation (76) holds. For float, only the necessary conditions need to be discussed, that is, whether $2^{-1} \cdot 2^q \cdot 10^{-k-1} = n$ must hold true when $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} \cdot n \rfloor$ holds, or equivalent to $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor \neq \lfloor 2^{36} \cdot n \rfloor$ must hold true when $2^{-1} \cdot 2^q \cdot 10^{-k-1} \neq n$. The following is proved by proof by contradiction.

Assume that $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} \cdot n \rfloor$ holds when $2^{-1} \cdot 2^q \cdot 10^{-k-1} \neq n$. Then there is:

$$\begin{aligned} &\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} \cdot n \rfloor \\ &\Rightarrow 0 < \left| 2^{35} \cdot 2^q \cdot 10^{-k-1} - 2^{36} \cdot n \right| < 1 \\ &\Rightarrow 0 < \left| (2c-1) \cdot 2^{q-1} \cdot 10^{-k-1} - m \right| < 2^{-36} \end{aligned} \tag{78}$$

As is known from equation (57), there is:

$$m - 1 < (2c-1) \cdot 2^{q-1} \cdot 10^{-k-1} < m + 1 \tag{79}$$

Suppose the decimal part of $(2c-1) \cdot 2^{q-1} \cdot 10^{-k-1}$ is represented as n^-, thus we have:

$$\left| (2c-1) \cdot 2^{q-1} \cdot 10^{-k-1} - m \right| = \begin{cases} n^-; \text{if } (2c-1) \cdot 2^{q-1} \cdot 10^{-k-1} > m \\ 1 - n^-; \text{if } (2c-1) \cdot 2^{q-1} \cdot 10^{-k-1} < m \end{cases} \tag{80}$$

Substitute expression (80) into expression (78), and we get:

$$\begin{aligned} &0 < \left| (2c-1) \cdot 2^{q-1} \cdot 10^{-k-1} - m \right| < 2^{-36} \\ &\Rightarrow 0 < n^- < 2^{-36} \text{ or } 0 < 1 - n^- < 2^{-36} \end{aligned} \tag{81}$$

Similarly, it can be known that the double range is the range of n^-. Therefore, there is:

$$\begin{aligned} float &: n^- \in \left(0, 2^{-36}\right) \cup \left(1 - 2^{-36}, 1\right) \\ double &: n^- \in \left(0, 2^{-64}\right) \cup \left(1 - 2^{-64}, 1\right) \end{aligned} \tag{82}$$

When $2^{-1} \cdot 2^q \cdot 10^{-k-1} \neq n$, it is known from equation (54) that $(2c-1) \cdot 2^{q-1} \cdot 10^{-k-1}$ is not an integer. Therefore, there is:

$$0 < n^- < 1 \tag{83}$$

It is only necessary to prove that equation (82) does not hold. Discuss the range of the decimal part n^- when $(2c-1) \cdot 2^{q-1} \cdot 10^{-k-1}$ is not an integer. According to equation (61), there are:

$$(2c-1) \cdot 2^{q-1} \cdot 10^{-k-1} = (2c-1) \cdot \frac{x}{y} = \begin{cases} \frac{(2c-1) \cdot 2^{q-k-2}}{5^{k+1}}; q \geqslant 2 \\ \frac{(2c-1)}{2^{2+k-q} \cdot 5^{k+1}}; 1 \geqslant q \geqslant 0 \\ \frac{(2c-1) \cdot 5^{-k-1}}{2^{2+k-q}}; q < 0 \end{cases} \tag{84}$$

The maximum value of $2c - 1$ is:

$$\begin{aligned} float &: (2c-1)_{\max} = 2^{25} - 3 \\ double &: (2c-1)_{\max} = 2^{54} - 3 \end{aligned} \tag{85}$$

Discuss based on the denominator range in equation (84).

- $y \leqslant (2c-1)_{\max}$
  When $y \leqslant (2c-1)_{\max}$, $y_{\max}$ is the expression (85), the following holds true:

$$\begin{aligned} \frac{1}{y_{\max}} \leqslant n^{-} \leqslant 1 - \frac{1}{y_{\max}} \\ \frac{1}{y_{\max}} \leqslant 1 - n^{-} \leqslant 1 - \frac{1}{y_{\max}} \end{aligned} \tag{86}$$

  Therefore, when $y \leqslant (2c-1)_{\max}$, equation (82) does not hold true.
- $y > (2c-1)_{\max}$
  Call function (37) to calculate the approximation results $P_* \big/ Q_*$ and $P^* \big/ Q^*$ of all possible upper and lower limit rational numbers:

$$\left(\frac{P_*}{Q_*}, \frac{P^*}{Q^*}\right) = f\left((2c-1)_{\max}, x, y\right) \tag{87}$$

  Therefore, for n^{-}, the following conclusion can be drawn from formula (36).

$$n^{-} \in \left[\frac{(Q_* x)\,\%y}{y}, \frac{(Q^* x)\,\%y}{y}\right] \tag{88}$$

  By exhausting all possibilities, we thus have (the test code file is test3.py) :

$$\begin{aligned} float: 2^{-33} < n^{-} < 1 - 2^{-29} \\ double: 2^{-62} < n^{-} < 1 - 2^{-63} \end{aligned} \tag{89}$$

$$\begin{aligned} float: &\left[\frac{(Q_* x)\,\%y}{y}, \frac{(Q^* x)\,\%y}{y}\right] \cap \left(0, 2^{-36}\right) = \varnothing \\ &\left[\frac{(Q_* x)\,\%y}{y}, \frac{(Q^* x)\,\%y}{y}\right] \cap \left(1 - 2^{-36}, 1\right) = \varnothing \\ double: &\left[\frac{(Q_* x)\,\%y}{y}, \frac{(Q^* x)\,\%y}{y}\right] \cap \left(0, 2^{-64}\right) = \varnothing \\ &\left[\frac{(Q_* x)\,\%y}{y}, \frac{(Q^* x)\,\%y}{y}\right] \cap \left(1 - 2^{-64}, 1\right) = \varnothing \end{aligned} \tag{90}$$

  Therefore, when $y > (2c-1)_{\max}$, equation (82) does not hold true.

In summary, when $2^{-1} \cdot 2^q \cdot 10^{-k-1} \neq n$, equation (82) does not hold true, that is, $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor \neq \lfloor 2^{36} \cdot n \rfloor$ must hold true. Therefore, when $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} \cdot n \rfloor$ holds, $2^{-1} \cdot 2^q \cdot 10^{-k-1} = n$ must hold true. Therefore, equation (76) holds.

Similarly, it can be proved that when $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} - 2^{36} \cdot n \rfloor$ holds, $2^{-1} \cdot 2^q \cdot 10^{-k-1} = 1 - n$ must hold true. The same applies to double. Similarly, by proof of contradiction, for float, it is assumed that when $2^{-1} \cdot 2^q \cdot 10^{-k-1} \neq 1 - n$ holds, $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} - 2^{36} \cdot n \rfloor$ holds. That is:

$$\begin{aligned} &\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} - 2^{36} \cdot n \rfloor \\ &\Rightarrow 0 < \left| 2^{35} \cdot 2^q \cdot 10^{-k-1} - 2^{36} + 2^{36} \cdot n \right| < 1 \\ &\Rightarrow 0 < \left| 2^{q-1} \cdot 10^{-k-1} - 1 + n \right| < 2^{-36} \\ &\Rightarrow -2^{-36} < (2c+1) \cdot 2^{q-1} \cdot 10^{-k-1} - m - 1 < 2^{-36} \end{aligned} \tag{91}$$

As is known from equation (58), there is:

$$m < (2c+1) \cdot 2^{q-1} \cdot 10^{-k-1} < m+2 \tag{92}$$

Suppose the decimal part of $(2c+1) \cdot 2^{q-1} \cdot 10^{-k-1}$ is represented as n^+, thus we have:

$$(2c+1) \cdot 2^{q-1} \cdot 10^{-k-1} - m - 1 = \begin{cases} n^+; \text{if } \ (2c+1) \cdot 2^{q-1} \cdot 10^{-k-1} > m+1 \\ 1-n^+; \text{if } \ (2c+1) \cdot 2^{q-1} \cdot 10^{-k-1} < m+1 \end{cases} \tag{93}$$

Substitute expression (93) into expression (91), and we get:

$$\begin{gathered} 0 < \left|(2c+1) \cdot 2^{q-1} \cdot 10^{-k-1} - m - 1\right| < 2^{-36} \\ \Rightarrow 0 < 1-n^+ < 2^{-36} \text{ or } 0 < n^+ < 2^{-36} \end{gathered} \tag{94}$$

Similarly, it can be known that the double range is the range of n^+. Therefore, there is:

$$\begin{gathered} float : n^+ \in \left(0, 2^{-36}\right) \cup \left(1-2^{-36}, 1\right) \\ double : n^+ \in \left(0, 2^{-64}\right) \cup \left(1-2^{-64}, 1\right) \end{gathered} \tag{95}$$

When $2^{-1} \cdot 2^q \cdot 10^{-k-1} \neq 1-n$, it is known from equation (55) that $(2c+1) \cdot 2^{q-1} \cdot 10^{-k-1}$ is not an integer. Therefore, there is:

$$0 < n^+ < 1 \tag{96}$$

It is only necessary to prove that equation (95) does not hold. Discuss the range of the decimal part n^+ when $(2c+1) \cdot 2^{q-1} \cdot 10^{-k-1}$ is not an integer. According to equation (61), there are:

$$(2c+1) \cdot 2^{q-1} \cdot 10^{-k-1} = (2c+1) \cdot \frac{x}{y} = \begin{cases} \frac{(2c+1) \cdot 2^{q-k-2}}{5^{k+1}}; q \geqslant 2 \\ \frac{(2c+1)}{2^{2+k-q} \cdot 5^{k+1}}; 1 \geqslant q \geqslant 0 \\ \frac{(2c+1) \cdot 5^{-k-1}}{2^{2+k-q}}; q < 0 \end{cases} \tag{97}$$

The maximum value of $2c+1$ is:

$$\begin{gathered} float : (2c+1)_{\max} = 2^{25} - 1 \\ double : (2c+1)_{\max} = 2^{54} - 1 \end{gathered} \tag{98}$$

Discuss based on the denominator range in equation (97).

- $y \leqslant (2c+1)_{\max}$
  When $y \leqslant (2c+1)_{\max}$, $y_{\max}$ is the expression (98), the following holds true:

$$\begin{gathered} \frac{1}{y_{\max}} \leqslant n^+ \leqslant 1 - \frac{1}{y_{\max}} \\ \frac{1}{y_{\max}} \leqslant 1 - n^+ \leqslant 1 - \frac{1}{y_{\max}} \end{gathered} \tag{99}$$

  Therefore, when $y \leqslant (2c+1)_{\max}$, equation (95) does not hold true.
- $y > (2c+1)_{\max}$
  Call function (37) to calculate the approximation results $P_* \big/ Q_*$ and $P^* \big/ Q^*$ of all possible upper and lower limit rational numbers:

$$\left(\frac{P_*}{Q_*}, \frac{P^*}{Q^*}\right) = f\left((2c+1)_{\max}, x, y\right) \tag{100}$$

Therefore, for n^+, the following conclusion can be drawn from formula (36).

$$n^+ \in \left[\frac{(Q_* x)\ \%y}{y}, \frac{(Q^* x)\ \%y}{y}\right] \tag{101}$$

By exhausting all possibilities, we thus have (the test code file is test7.py) :

$$\begin{aligned} float &: 2^{-33} < n^+ < 1 - 2^{-29} \\ double &: 2^{-62} < n^+ < 1 - 2^{-63} \end{aligned} \tag{102}$$

$$\begin{aligned} float &: \left[\frac{(Q_* x)\ \%y}{y}, \frac{(Q^* x)\ \%y}{y}\right] \cap \left(0, 2^{-36}\right) = \varnothing \\ &\left[\frac{(Q_* x)\ \%y}{y}, \frac{(Q^* x)\ \%y}{y}\right] \cap \left(1 - 2^{-36}, 1\right) = \varnothing \\ double &: \left[\frac{(Q_* x)\ \%y}{y}, \frac{(Q^* x)\ \%y}{y}\right] \cap \left(0, 2^{-64}\right) = \varnothing \\ &\left[\frac{(Q_* x)\ \%y}{y}, \frac{(Q^* x)\ \%y}{y}\right] \cap \left(1 - 2^{-64}, 1\right) = \varnothing \end{aligned} \tag{103}$$

Therefore, when $y > (2c+1)_{\max}$, equation (95) does not hold true.

In summary, when $2^{-1} \cdot 2^q \cdot 10^{-k-1} \neq 1 - n$, equation (95) does not hold true, that is, $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor \neq \lfloor 2^{36} - 2^{36} \cdot n \rfloor$ must hold true. Therefore, when $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor = \lfloor 2^{36} - 2^{36} \cdot n \rfloor$ holds, $2^{-1} \cdot 2^q \cdot 10^{-k-1} = 1 - n$ must hold true. The same is true for double.Therefore, equation (77) holds.

The following conclusions hold:

$$\begin{aligned} float : \lfloor 2^{36} - 2^{36} \cdot n \rfloor &= \begin{cases} 2^{36} - 1 - \lfloor 2^{36} \cdot n \rfloor; \text{if } c \cdot 2^{36+q} \cdot 10^{-k-1} \notin Z \\ 2^{36} - \lfloor 2^{36} \cdot n \rfloor; \text{if } c \cdot 2^{36+q} \cdot 10^{-k-1} \in Z \end{cases} \\ double : \lfloor 2^{64} - 2^{64} \cdot n \rfloor &= \begin{cases} 2^{64} - 1 - \lfloor 2^{64} \cdot n \rfloor; \text{if } c \cdot 2^{64+q} \cdot 10^{-k-1} \notin Z \\ 2^{64} - \lfloor 2^{64} \cdot n \rfloor; \text{if } c \cdot 2^{64+q} \cdot 10^{-k-1} \in Z \end{cases} \end{aligned} \tag{104}$$

Discuss whether the following equation (105) holds when conditions (68) and (69) are met:

$$\begin{aligned} float : \lfloor c \cdot \frac{2^{q+35-k}}{5^{k+1}} \rfloor &= \lfloor c \cdot \frac{2^{q+35-k}}{5^{k+1}} \cdot r \rfloor \\ &= \lfloor c \cdot \frac{2^{q+35-k}}{5^{k+1}} \cdot \frac{\left(2^{63 - \lfloor (-k-1) \cdot \log_2(10) \rfloor} // 10^{k+1}\right) + 1}{10^{-k-1}} \cdot 2^{\lfloor (-k-1) \cdot \log_2(10) \rfloor - 63} \rfloor \\ double : \lfloor c \cdot \frac{2^{q+63-k}}{5^{k+1}} \rfloor &= \lfloor c \cdot \frac{2^{q+63-k}}{5^{k+1}} \cdot r \rfloor \\ &= \lfloor c \cdot \frac{2^{q+63-k}}{5^{k+1}} \cdot \frac{\left(2^{127 - \lfloor (-k-1) \cdot \log_2(10) \rfloor} // 10^{k+1}\right) + 1}{10^{-k-1}} \cdot 2^{\lfloor (-k-1) \cdot \log_2(10) \rfloor - 127} \rfloor \end{aligned} \tag{105}$$

There are:

$$\begin{aligned} float : \lfloor c \cdot \frac{2^{q+35-k}}{5^{k+1}} \rfloor = \lfloor 2^{36} \cdot (m+n) \rfloor = 2^{36} \cdot m + \lfloor 2^{36} \cdot n \rfloor \\ double : \lfloor c \cdot \frac{2^{q+63-k}}{5^{k+1}} \rfloor = \lfloor 2^{64} \cdot (m+n) \rfloor = 2^{64} \cdot m + \lfloor 2^{64} \cdot n \rfloor \end{aligned} \tag{106}$$

It has been proven earlier that m can be accurately calculated. Then, when (105) holds true, the values $\lfloor 2^{36} \cdot n \rfloor$ and $\lfloor 2^{64} \cdot n \rfloor$ on the right side of equations (70) and (71) can be accurately calculated.

From equation (63), we have:

$$c = \frac{t \cdot 5^{k+1} - 1}{2} \tag{107}$$

Substituting equation (107) into equation (105), we have:

$$\begin{aligned} float: c \cdot \frac{2^{q+35-k}}{5^{k+1}} &= t \cdot 2^{q+34-k} - \frac{2^{q+34-k}}{5^{k+1}} \\ double: c \cdot \frac{2^{q+63-k}}{5^{k+1}} &= t \cdot 2^{q+62-k} - \frac{2^{q+62-k}}{5^{k+1}} \end{aligned} \tag{108}$$

When conditions (68) and (69) are met, $t \cdot 2^{q+34-k}$ and $t \cdot 2^{q+62-k}$ are integers. Under the condition of meeting condition (68), the decimal part of expression (108) is represented as:

$$\begin{aligned} float: \frac{2^{q+34-k} \% 5^{k+1}}{5^{k+1}}; 2 \leqslant q \leqslant 33 \\ double: \frac{2^{q+62-k} \% 5^{k+1}}{5^{k+1}}; 2 \leqslant q \leqslant 76 \end{aligned} \tag{109}$$

It is only necessary to prove that the increase in the value $c \cdot \frac{2^{q+35-k}}{5^{k+1}} \cdot r$ on the right side of the expression compared to the value $c \cdot \frac{2^{q+35-k}}{5^{k+1}}$ on the left side plus the decimal part of the value on the left side is less than 1 for equation (105) to hold true. That is:

$$\begin{aligned} float: \frac{2^{q+34-k} \% 5^{k+1}}{5^{k+1}} + \left(c \cdot \frac{2^{q+35-k}}{5^{k+1}} \cdot r - c \cdot \frac{2^{q+35-k}}{5^{k+1}}\right) < 1 \\ double: \frac{2^{q+62-k} \% 5^{k+1}}{5^{k+1}} + \left(c \cdot \frac{2^{q+63-k}}{5^{k+1}} \cdot r - c \cdot \frac{2^{q+63-k}}{5^{k+1}}\right) < 1 \end{aligned} \tag{110}$$

By exhaustionally calculating the maximum possible c value under each q and substituting it into equation (110), it holds. The calculation result is in test2.py. The calculation results show that for the float range and the double range, equation (110) always holds true. Therefore, equation (105) holds true, and thus the values of $\lfloor 2^{36} \cdot n \rfloor$ and $\lfloor 2^{64} \cdot n \rfloor$ on the right side of equations (70) and (71) can be accurately calculated. The values of $\lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor$ and $\lfloor 2^{63} \cdot 2^q \cdot 10^{-k-1} \rfloor$ on the left side of equations (70) and (71) can be calculated through lookup tables.

$$\begin{aligned} float: \lfloor 2^{35} \cdot 2^q \cdot 10^{-k-1} \rfloor &= pow10 \gg \left(28 - q - \lfloor (-k-1) \cdot \log_2(10) \rfloor\right) \\ double: \lfloor 2^{64} \cdot 2^q \cdot 10^{-k-1} \rfloor &= pow10 \gg \left(64 - q - \lfloor (-k-1) \cdot \log_2(10) \rfloor\right) \end{aligned} \tag{111}$$

The code file for verifying the validity of equation (111) is test4.py. Therefore, when conditions (68) and (69) are met, the values of both sides of equations (70) and (71) can be accurately calculated.

Discuss the relationship between the following two values within all ranges of floating-point numbers:

$$\begin{aligned} float: \lfloor c \cdot 2^{q+36} \cdot 10^{-k-1} \rfloor; \lfloor c \cdot 2^{q+36} \cdot r \cdot 10^{-k-1} \rfloor; \\ double: \lfloor c \cdot 2^{q+64} \cdot 10^{-k-1} \rfloor; \lfloor c \cdot 2^{q+64} \cdot r \cdot 10^{-k-1} \rfloor; \end{aligned} \tag{112}$$

When $r = 1$, it is obvious that the two values in expression (112) are equal. When $r \neq 1$, or equivalent to $r > 1$, has:

$$\begin{aligned}
&float: \\
&c \cdot 2^{q+36} \cdot r \cdot 10^{-k-1} = c \cdot 2^{q+36} \cdot 10^{-k-1} + c \cdot 2^{q+36} \cdot (r-1) \cdot 10^{-k-1} \\
&\qquad < c \cdot 2^{q+36} \cdot 10^{-k-1} + 2^{24} \cdot 2^{36} \cdot 2^{q} \cdot 10^{-k-1} \cdot (r-1) \\
&\qquad < c \cdot 2^{q+36} \cdot 10^{-k-1} + 2^{-3} \\
&\lfloor c \cdot 2^{q+36} \cdot r \cdot 10^{-k-1} \rfloor \leqslant \lfloor c \cdot 2^{q+36} \cdot 10^{-k-1} \rfloor + 1 \\
&double: \\
&c \cdot 2^{q+64} \cdot r \cdot 10^{-k-1} = c \cdot 2^{q+64} \cdot 10^{-k-1} + c \cdot 2^{q+64} \cdot (r-1) \cdot 10^{-k-1} \\
&\qquad < c \cdot 2^{q+64} \cdot 10^{-k-1} + 2^{53} \cdot 2^{64} \cdot 2^{q} \cdot 10^{-k-1} \cdot (r-1) \\
&\qquad < c \cdot 2^{q+64} \cdot 10^{-k-1} + 2^{-10} \\
&\lfloor c \cdot 2^{q+64} \cdot r \cdot 10^{-k-1} \rfloor \leqslant \lfloor c \cdot 2^{q+64} \cdot 10^{-k-1} \rfloor + 1
\end{aligned} \tag{113}$$

Therefore, there is:

$$\begin{aligned}
float: 0 \leqslant \lfloor c \cdot 2^{q+36} \cdot r \cdot 10^{-k-1} \rfloor - \lfloor c \cdot 2^{q+36} \cdot 10^{-k-1} \rfloor \leqslant 1 \\
double: 0 \leqslant \lfloor c \cdot 2^{q+64} \cdot r \cdot 10^{-k-1} \rfloor - \lfloor c \cdot 2^{q+64} \cdot 10^{-k-1} \rfloor \leqslant 1
\end{aligned} \tag{114}$$

Because there is:

$$\lfloor c \cdot 2^{q} \cdot 10^{-k-1} \rfloor = \lfloor c \cdot 2^{q} \cdot r \cdot 10^{-k-1} \rfloor = m \tag{115}$$

$$\begin{aligned}
float: \lfloor c \cdot 2^{q+36} \cdot 10^{-k-1} \rfloor = 2^{36} \cdot m + \lfloor 2^{36} \cdot n \rfloor \\
double: \lfloor c \cdot 2^{q+64} \cdot 10^{-k-1} \rfloor = 2^{64} \cdot m + \lfloor 2^{64} \cdot n \rfloor
\end{aligned} \tag{116}$$

Suppose:

$$n_r = c \cdot 2^{q} \cdot r \cdot 10^{-k-1} - m \tag{117}$$

Therefore, the following conclusion can be drawn: when condition (68) is met, from equation (105), we have:

$$\begin{aligned}
float: 2 \leqslant q \leqslant 33 \,\&\&\, (2c \pm 1)\,\%5^{k+1} = 0 \Longrightarrow \lfloor 2^{36} \cdot n \rfloor = \lfloor 2^{36} \cdot n_r \rfloor \\
double: 2 \leqslant q \leqslant 76 \,\&\&\, (2c \pm 1)\,\%5^{k+1} = 0 \Longrightarrow \lfloor 2^{64} \cdot n \rfloor = \lfloor 2^{64} \cdot n_r \rfloor
\end{aligned} \tag{118}$$

Within the range of floating-point numbers, there exists:

$$\begin{aligned}
float: \lfloor 2^{36} \cdot n \rfloor \leqslant \lfloor 2^{36} \cdot n_r \rfloor \leqslant \lfloor 2^{36} \cdot n \rfloor + 1 \\
double: \lfloor 2^{64} \cdot n \rfloor \leqslant \lfloor 2^{64} \cdot n_r \rfloor \leqslant \lfloor 2^{64} \cdot n \rfloor + 1
\end{aligned} \tag{119}$$

To simplify the expression, *even* is used to indicate whether c is an even number:

$$even = (c+1)\%2 \in \{0, 1\} \tag{120}$$

When $2^{-1} \cdot 2^{q} \cdot 10^{-k-1} = n$ or $2^{-1} \cdot 2^{q} \cdot 10^{-k-1} = 1 - n$, $2^{-1} \cdot 2^{q} \cdot 10^{-k-1} = n$ is the boundary condition for $one = 0$, and $2^{-1} \cdot 2^{q} \cdot 10^{-k-1} = 1 - n$ is the boundary condition for $one = 10$. Whether one is 0 or 10 is determined based on whether c is an even number.Therefore, the following exists:

$$\begin{aligned}
float: \begin{cases} one = 0: \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + even > \lfloor 2^{36} \cdot n_r \rfloor \\ one = 10: \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + even > 2^{36} - 1 - \lfloor 2^{36} \cdot n_r \rfloor \end{cases} \\
double: \begin{cases} one = 0: \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + even > \lfloor 2^{64} \cdot n_r \rfloor \\ one = 10: \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + even > 2^{64} - 1 - \lfloor 2^{64} \cdot n_r \rfloor \end{cases}
\end{aligned} \tag{121}$$

Therefore, when $2^{-1} \cdot 2^q \cdot 10^{-k-1} = n$ or $2^{-1} \cdot 2^q \cdot 10^{-k-1} = 1 - n$, we can use the condition (122) to determine whether $one = 0$ or $one = 10$.

$$
\begin{aligned}
float &: \begin{cases} \text{if } \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + even > \lfloor 2^{36} \cdot n_r \rfloor : one = 0 \\ \text{if } \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + even > 2^{36} - 1 - \lfloor 2^{36} \cdot n_r \rfloor : one = 10 \end{cases} \\
double &: \begin{cases} \text{if } \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + even > \lfloor 2^{64} \cdot n_r \rfloor : one = 0 \\ \text{if } \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + even > 2^{64} - 1 - \lfloor 2^{64} \cdot n_r \rfloor : one = 10 \end{cases}
\end{aligned} \tag{122}
$$

When $2^{-1} \cdot 2^q \cdot 10^{-k-1} > n$ or $2^{-1} \cdot 2^q \cdot 10^{-k-1} > 1 - n$, We can also use the above condition (122) to determine whether $one = 0$ or $one = 10$. When $2^{-1} \cdot 2^q \cdot 10^{-k-1} < n$ or $2^{-1} \cdot 2^q \cdot 10^{-k-1} < 1 - n$, we can also use the above condition (122) to determine whether $one \neq 0$ or $one \neq 10$. There are a total of four situations.The proof is as follows:

(1)When $2^{-1} \cdot 2^q \cdot 10^{-k-1} < n$, there must exist $one \neq 0$, and there is:

$$
\begin{aligned}
float &: 2^{-1} \cdot 2^q \cdot 10^{-k-1} - n = n^- - 1 \in \left(2^{-33} - 1, -2^{-29}\right) \\
double &: 2^{-1} \cdot 2^q \cdot 10^{-k-1} - n = n^- - 1 \in \left(2^{-62} - 1, -2^{-63}\right)
\end{aligned} \tag{123}
$$

Therefore, the following exists:

$$
\begin{aligned}
float &: 2^{q+35} \cdot 10^{-k-1} - 2^{36} \cdot n \in \left(2^3 - 2^{36}, -2^7\right) \\
double &: 2^{q+63} \cdot 10^{-k-1} - 2^{64} \cdot n \in \left(4 - 2^{64}, -2\right)
\end{aligned} \tag{124}
$$

Suppose there are two real numbers a and b, and the following relationship must exist:

$$
\begin{aligned}
0 \leqslant & b - \lfloor b \rfloor < 1 \\
a - \lfloor a \rfloor - 1 < & b - \lfloor b \rfloor < 1 + a - \lfloor a \rfloor \\
a - b - 1 < & \lfloor a \rfloor - \lfloor b \rfloor < a - b + 1
\end{aligned} \tag{125}
$$

When $a = 2^{q+35} \cdot 10^{-k-1}$ and $b = 2^{36} \cdot n$ or $a = 2^{q+63} \cdot 10^{-k-1}$ and $b = 2^{64} \cdot n$, the following exists:

$$
\begin{aligned}
float &: \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor - \lfloor 2^{36} \cdot n \rfloor < 2^{q+35} \cdot 10^{-k-1} - 2^{36} \cdot n + 1 \\
double &: \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor - \lfloor 2^{64} \cdot n \rfloor < 2^{q+63} \cdot 10^{-k-1} - 2^{64} \cdot n + 1
\end{aligned} \tag{126}
$$

From equation (124), we have:

$$
\begin{aligned}
float &: \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor - \lfloor 2^{36} \cdot n \rfloor < 1 - 2^7 < 0 \\
double &: \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor - \lfloor 2^{64} \cdot n \rfloor < 1 - 2 < 0
\end{aligned} \tag{127}
$$

Therefore, there is:

$$
\begin{aligned}
float : \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + even &\leqslant \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + 1 \\
&< \lfloor 2^{36} \cdot n \rfloor \leqslant \lfloor 2^{36} \cdot n_r \rfloor \\
\Rightarrow \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + even &< \lfloor 2^{36} \cdot n_r \rfloor \\
double : \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + even &\leqslant \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + 1 \\
&< \lfloor 2^{64} \cdot n \rfloor \leqslant \lfloor 2^{64} \cdot n_r \rfloor \\
\Rightarrow \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + even &< \lfloor 2^{64} \cdot n_r \rfloor
\end{aligned} \tag{128}
$$

Therefore, when $2^{-1} \cdot 2^q \cdot 10^{-k-1} < n$, the condition (122) can be used to determine that $one \neq 0$.

(2)When $2^{-1} \cdot 2^q \cdot 10^{-k-1} > n$, there must exist $one = 0$, and there is:

$$
\begin{aligned}
float &: 2^{-1} \cdot 2^q \cdot 10^{-k-1} - n = n^- \in \left(2^{-33}, 1 - 2^{-29}\right) \\
double &: 2^{-1} \cdot 2^q \cdot 10^{-k-1} - n = n^- \in \left(2^{-62}, 1 - 2^{-63}\right)
\end{aligned} \tag{129}
$$

Therefore, the following exists:

$$\begin{aligned} float &: 2^{q+35} \cdot 10^{-k-1} - 2^{36} \cdot n \in \left(2^3, 2^{36} - 2^7\right) \\ double &: 2^{q+63} \cdot 10^{-k-1} - 2^{64} \cdot n \in \left(4, 2^{64} - 2\right) \end{aligned} \tag{130}$$

When $a = 2^{q+35} \cdot 10^{-k-1}$ and $b = 2^{36} \cdot n$ or $a = 2^{q+63} \cdot 10^{-k-1}$ and $b = 2^{64} \cdot n$, from equation (125), the following exists:

$$\begin{aligned} float &: \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor - \lfloor 2^{36} \cdot n \rfloor > 2^{q+35} \cdot 10^{-k-1} - 2^{36} \cdot n - 1 \\ double &: \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor - \lfloor 2^{64} \cdot n \rfloor > 2^{q+63} \cdot 10^{-k-1} - 2^{64} \cdot n - 1 \end{aligned} \tag{131}$$

From equation (130), we have:

$$\begin{aligned} float &: \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor - \lfloor 2^{36} \cdot n \rfloor > 2^3 - 1 \geqslant 0 \\ double &: \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor - \lfloor 2^{64} \cdot n \rfloor > 4 - 1 \geqslant 0 \end{aligned} \tag{132}$$

Therefore, there is:

$$\begin{aligned} float : \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + even &\geqslant \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor \\ &> \lfloor 2^{36} \cdot n \rfloor + 1 \geqslant \lfloor 2^{36} \cdot n_r \rfloor \\ \Rightarrow \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + even &> \lfloor 2^{36} \cdot n_r \rfloor \\ double : \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + even &\geqslant \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor \\ &> \lfloor 2^{64} \cdot n \rfloor + 1 \geqslant \lfloor 2^{64} \cdot n_r \rfloor \\ \Rightarrow \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + even &> \lfloor 2^{64} \cdot n_r \rfloor \end{aligned} \tag{133}$$

Therefore, when $2^{-1} \cdot 2^q \cdot 10^{-k-1} > n$, the condition (122) can be used to determine that $one = 0$.
(3)When $2^{-1} \cdot 2^q \cdot 10^{-k-1} < 1 - n$, there must exist $one \neq 10$, and there is:

$$\begin{aligned} float &: 2^{-1} \cdot 2^q \cdot 10^{-k-1} + n = n^+ \in \left(2^{-33}, 1 - 2^{-29}\right) \\ double &: 2^{-1} \cdot 2^q \cdot 10^{-k-1} + n = n^+ \in \left(2^{-62}, 1 - 2^{-63}\right) \end{aligned} \tag{134}$$

Therefore, the following exists:

$$\begin{aligned} float &: 2^{q+35} \cdot 10^{-k-1} + 2^{36} \cdot n \in \left(2^3, 2^{36} - 2^7\right) \\ double &: 2^{q+63} \cdot 10^{-k-1} + 2^{64} \cdot n \in \left(4, 2^{64} - 2\right) \end{aligned} \tag{135}$$

Suppose there are two real numbers a and b, and the following relationship must exist:

$$\begin{gathered} a - 1 < \lfloor a \rfloor \leqslant a \\ b - 1 < \lfloor b \rfloor \leqslant b \\ a + b - 2 < \lfloor a \rfloor + \lfloor b \rfloor \leqslant a + b \end{gathered} \tag{136}$$

When $a = 2^{q+35} \cdot 10^{-k-1}$ and $b = 2^{36} \cdot n$ or $a = 2^{q+63} \cdot 10^{-k-1}$ and $b = 2^{64} \cdot n$, the following exists:

$$\begin{aligned} float &: \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + \lfloor 2^{36} \cdot n \rfloor \leqslant 2^{q+35} \cdot 10^{-k-1} + 2^{36} \cdot n \\ double &: \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + \lfloor 2^{64} \cdot n \rfloor \leqslant 2^{q+63} \cdot 10^{-k-1} + 2^{64} \cdot n \end{aligned} \tag{137}$$

From equation (135), we have:

$$\begin{aligned} float &: \lfloor 2^{q+35} \cdot 10^{-k-1} \rfloor + \lfloor 2^{36} \cdot n \rfloor < 2^{36} - 2^7 \\ double &: \lfloor 2^{q+63} \cdot 10^{-k-1} \rfloor + \lfloor 2^{64} \cdot n \rfloor < 2^{64} - 2 \end{aligned} \tag{138}$$

Therefore, there is:

$$\begin{aligned}
float: \lfloor 2^{q+35}\cdot 10^{-k-1}\rfloor + even &\leqslant \lfloor 2^{q+35}\cdot 10^{-k-1}\rfloor + 1\\
&< 2^{36} - 2 - \lfloor 2^{36}\cdot n\rfloor\\
&< 2^{36} - 1 - \lfloor 2^{36}\cdot n_r\rfloor\\
\Rightarrow \lfloor 2^{q+35}\cdot 10^{-k-1}\rfloor + even &< 2^{36} - 1 - \lfloor 2^{36}\cdot n_r\rfloor\\
double: \lfloor 2^{q+63}\cdot 10^{-k-1}\rfloor + even &\leqslant \lfloor 2^{q+63}\cdot 10^{-k-1}\rfloor + 1\\
&< 2^{64} - 2 - \lfloor 2^{64}\cdot n\rfloor\\
&< 2^{64} - 1 - \lfloor 2^{64}\cdot n_r\rfloor\\
\Rightarrow \lfloor 2^{q+63}\cdot 10^{-k-1}\rfloor + even &< 2^{64} - 1 - \lfloor 2^{64}\cdot n_r\rfloor
\end{aligned} \tag{139}$$

Therefore, when $2^{-1}\cdot 2^q\cdot 10^{-k-1} < 1-n$, the condition (122) can be used to determine that $one \neq 10$.
(4)When $2^{-1}\cdot 2^q\cdot 10^{-k-1} > 1-n$, there must exist $one = 10$, and there is:

$$\begin{aligned}
float: 2^{-1}\cdot 2^q\cdot 10^{-k-1} + n = n^+ + 1 \in \left(1+2^{-33}, 2-2^{-29}\right)\\
double: 2^{-1}\cdot 2^q\cdot 10^{-k-1} + n = n^+ + 1 \in \left(1+2^{-62}, 2-2^{-63}\right)
\end{aligned} \tag{140}$$

Therefore, the following exists:

$$\begin{aligned}
float: 2^{q+35}\cdot 10^{-k-1} + 2^{36}\cdot n \in \left(2^3+2^{36}, 2^{37}-2^7\right)\\
double: 2^{q+63}\cdot 10^{-k-1} + 2^{64}\cdot n \in \left(4+2^{64}, 2^{65}-2\right)
\end{aligned} \tag{141}$$

When $a = 2^{q+35}\cdot 10^{-k-1}$ and $b = 2^{36}\cdot n$ or $a = 2^{q+63}\cdot 10^{-k-1}$ and $b = 2^{64}\cdot n$, from equation (136), the following exists:

$$\begin{aligned}
float: \lfloor 2^{q+35}\cdot 10^{-k-1}\rfloor + \lfloor 2^{36}\cdot n\rfloor > 2^{q+35}\cdot 10^{-k-1} + 2^{36}\cdot n - 2\\
double: \lfloor 2^{q+63}\cdot 10^{-k-1}\rfloor + \lfloor 2^{64}\cdot n\rfloor > 2^{q+63}\cdot 10^{-k-1} + 2^{64}\cdot n - 2
\end{aligned} \tag{142}$$

From equation (141), we have:

$$\begin{aligned}
float: \lfloor 2^{q+35}\cdot 10^{-k-1}\rfloor + \lfloor 2^{36}\cdot n\rfloor > 2^{36} + 2^3 - 2 > 2^{36}\\
double: \lfloor 2^{q+63}\cdot 10^{-k-1}\rfloor + \lfloor 2^{64}\cdot n\rfloor > 2^{64} + 2 - 2 > 2^{64}
\end{aligned} \tag{143}$$

Therefore, there is:

$$\begin{aligned}
float: \lfloor 2^{q+35}\cdot 10^{-k-1}\rfloor + even &\geqslant \lfloor 2^{q+35}\cdot 10^{-k-1}\rfloor\\
&> 2^{36} - \lfloor 2^{36}\cdot n\rfloor\\
&> 2^{36} - 1 - \lfloor 2^{36}\cdot n_r\rfloor\\
\Rightarrow \lfloor 2^{q+35}\cdot 10^{-k-1}\rfloor + even &> 2^{36} - 1 - \lfloor 2^{36}\cdot n_r\rfloor\\
double: \lfloor 2^{q+63}\cdot 10^{-k-1}\rfloor + even &\geqslant \lfloor 2^{q+63}\cdot 10^{-k-1}\rfloor\\
&> 2^{64} - \lfloor 2^{64}\cdot n\rfloor\\
&> 2^{64} - 1 - \lfloor 2^{64}\cdot n_r\rfloor\\
\Rightarrow \lfloor 2^{q+63}\cdot 10^{-k-1}\rfloor + even &> 2^{64} - 1 - \lfloor 2^{64}\cdot n_r\rfloor
\end{aligned} \tag{144}$$

Therefore, when $2^{-1}\cdot 2^q\cdot 10^{-k-1} > 1-n$, the condition (122) can be used to determine that $one = 10$.

From the above proof, it can be seen that when condition (68) is met, the condition (122) can be used to determine whether $one = 0$ or $one = 10$ when $2^{-1}\cdot 2^q\cdot 10^{-k-1} = n$ or $2^{-1}\cdot 2^q\cdot 10^{-k-1} = 1-n$.

When $2^{-1} \cdot 2^q \cdot 10^{-k-1} > n$ or $2^{-1} \cdot 2^q \cdot 10^{-k-1} > 1 - n$, the condition (122) can be used to determine whether $one = 0$ or $one = 10$. When $2^{-1} \cdot 2^q \cdot 10^{-k-1} < n$ or $2^{-1} \cdot 2^q \cdot 10^{-k-1} < 1 - n$, the condition (122) can be used to determine whether $one \neq 0$ or $one \neq 10$.

The proof process of this section is completed. In the code implementation, the two judgment conditions can be quickly calculated using addition and subtraction shift operations, and can be compiled by the compiler into cmov instructions, thereby reducing the impact of branch prediction failure on performance.

3.5 Determine whether $one = \lfloor 10n \rfloor$ or $one = \lfloor 10n \rfloor + 1$

Determine whether one is $\lfloor 10n \rfloor$ or $\lfloor 10n \rfloor + 1$ based on the decimal part of $10n$. There are two cases: the decimal part of $10n$ is 0.5 and it is not 0.5.

3.5.1 $10n - \lfloor 10n \rfloor = 0.5$ *.* When the decimal part of $10n$ is 0.5, there must be:

$$\begin{aligned} 10n - \lfloor 10n \rfloor &= 0.5 \\ \Rightarrow 10 \cdot c \cdot 2^q \cdot 10^{-k-1} - \lfloor 10 \cdot c \cdot 2^q \cdot 10^{-k-1} \rfloor &= 0.5 \\ \Rightarrow c \cdot 2^q \cdot 10^{-k} - \lfloor c \cdot 2^q \cdot 10^{-k} \rfloor &= 0.5 \\ \Rightarrow c \cdot 2^q \cdot 10^{-k} &= \lfloor c \cdot 2^q \cdot 10^{-k} \rfloor + 0.5 \\ \Rightarrow 2c \cdot 2^q \cdot 10^{-k} &= 2\lfloor c \cdot 2^q \cdot 10^{-k} \rfloor + 1 \end{aligned} \tag{145}$$

So $2c \cdot 2^q \cdot 10^{-k}$ is an odd number. Then the following expression is odd:

$$c \cdot 2^{q+1} \cdot 10^{-k} = c \cdot 2^{q-k+1} \cdot 5^{-k} \tag{146}$$

According to the range of q, there are:

$$c \cdot 2^{q+1} \cdot 10^{-k} = \begin{cases} \frac{c \cdot 2^{q-k+1}}{5^k}; q \geqslant 0 \\ c \cdot 2 \cdot 5^{-k}; q = -1 \\ \frac{c \cdot 5^{-k}}{2^{k-q-1}}; q \leqslant -2 \end{cases} \tag{147}$$

According to the range of q, the following situations are discussed:

- $q \geqslant 0$
  When $q \geqslant 0$, it can be concluded that $q - k + 1 \geqslant 1$, the numerator $c \cdot 2^{q-k+1}$ is even and the denominator 5^k is odd, which does not meet the condition.
- $q = -1$
  When $q = -1$, it can be concluded that $c \cdot 2 \cdot 5^{-k}$ is even, which does not meet the condition.
- $q \leqslant -2$
  5^{-k} is an odd number. c is an odd multiple of 2^{k-q-1}. So:

$$\begin{aligned} float: c \geqslant 2^{k-q-1} \Rightarrow k - q - 1 \leqslant 22 \Rightarrow q \geqslant -34 \\ double: c \geqslant 2^{k-q-1} \Rightarrow k - q - 1 \leqslant 51 \Rightarrow q \geqslant -75 \end{aligned} \tag{148}$$

  Therefore, when q meets the above conditions, c must be an odd multiple of 2^{k-q-1} to meet the condition. Therefore, when the following conditions are met, expression (146) is an odd number:

$$\begin{aligned} float: -34 \leqslant q \leqslant -2 \ \&\& \ c\%2^{k-q} = 2^{k-q-1} \\ double: -75 \leqslant q \leqslant -2 \ \&\& \ c\%2^{k-q} = 2^{k-q-1} \end{aligned} \tag{149}$$

  When q is within the above range (149), $r = 1$ is derived from equation (30).Therefore, there is:

$$n_r = n \tag{150}$$

The following equation holds:

$$20m + 20n = c \cdot 2^q \cdot 10^{-k+1} = c \cdot 2^{q-k+1} \cdot 5^{-k} = \frac{c}{2^{k-q-1}} \cdot 5^{-k} \tag{151}$$

Since $-k \geqslant 1$, 5^{-k} is multiple of 5 and is an odd number. Since $\frac{c}{2^{k-q-1}}$ and 5^{-k} are both odd numbers, $20m$ is an even number, $20n$ is multiple of 5 and is an odd number. Therefore, there is:

$$\begin{aligned} 20n &\in \{5, 15\} \\ \Rightarrow n &\in \{0.25, 0.75\} \\ \Rightarrow n_r &\in \{0.25, 0.75\} \end{aligned} \tag{152}$$

The result of one is an even number between $\lfloor 10n \rfloor$ and $\lfloor 10n \rfloor + 1$. Therefore, when the following conditions are met:

$$one = \begin{cases} \lfloor 10n \rfloor = 2, \text{if } n = 0.25 \\ \lfloor 10n \rfloor + 1 = 8, \text{if } n = 0.75 \end{cases} \Rightarrow one = \lfloor 20n + 1 \rfloor // 2 - (n = 0.25\,?1 : 0) \tag{153}$$

3.5.2 $10n - \lfloor 10n \rfloor \neq 0.5$. When the decimal part of $10n$ is not 0.5, round to the nearest integer value based on the decimal part of $10n$. Therefore, there is:

$$one = \begin{cases} \lfloor 10n \rfloor, \text{if } 10n - \lfloor 10n \rfloor < 0.5 \\ \lfloor 10n \rfloor + 1, \text{if } 10n - \lfloor 10n \rfloor > 0.5 \end{cases} \Rightarrow one = \lfloor 10n + 0.5 \rfloor = \lfloor 20n + 1 \rfloor // 2 \tag{154}$$

Since $\lfloor 20n + 1 \rfloor = \lfloor 20n \rfloor + 1$, it is only necessary to accurately calculate the value of $\lfloor 20n \rfloor$. And, there is:

$$\begin{aligned} d &= ten + one \\ &= 10m + \lfloor 20n + 1 \rfloor // 2 \\ &= (\lfloor 20m + 20n \rfloor + 1) // 2 \end{aligned} \tag{155}$$

Suppose there are:

$$20m + 20n = c \cdot 2^{q+1} \cdot 10^{-k} = c \cdot 2^{q-k+1} \cdot 5^{-k} = c \cdot \frac{x}{y} \tag{156}$$

Suppose the decimal part of $20n$ is n_{20}.

When $y \leqslant c_{\max} = C$, the range of the decimal part must include:

$$\begin{aligned} float &: \frac{1}{2^{24}-1} = \frac{1}{C} \leqslant n_{20} \leqslant 1 - \frac{1}{C} = \frac{2^{24}-2}{2^{24}-1} \\ double &: \frac{1}{2^{53}-1} = \frac{1}{C} \leqslant n_{20} \leqslant 1 - \frac{1}{C} = \frac{2^{53}-2}{2^{53}-1} \end{aligned} \tag{157}$$

When $y > c_{\max} = C$, the range of the decimal part must include(the test file is test5.py):

$$\begin{aligned} float &: 2^{-32} < n_{20} < 1 - 2^{-30} \\ double &: 2^{-64} < n_{20} < 1 - 2^{-62} \end{aligned} \tag{158}$$

Therefore, the range of n_{20} satisfies equation (158). In the code implementation, for float, only the high 36 bits of n_r are retained, and for double, only the high 70 bits of n_r are retained. Suppose the discarded part of a float is represented as n_{36}, and similarly, the discarded part of a double is represented as n_{70}. Therefore, there is:

$$\begin{aligned} float &: n_{36} \in \left[0,\ 2^{-36}\right) \\ double &: n_{70} \in \left[0,\ 2^{-70}\right) \end{aligned} \tag{159}$$

Calculate the boundary conditions of the following expression:

$$\begin{aligned} float : F &= 20 \cdot \left(c \cdot 2^q \cdot r \cdot 10^{-k-1} - n_{36}\right) \\ double : F &= 20 \cdot \left(c \cdot 2^q \cdot r \cdot 10^{-k-1} - n_{70}\right) \end{aligned} \tag{160}$$

Therefore, there is:

$$\begin{aligned} float : F_{\min} &> 20 \cdot \left(c \cdot 2^q \cdot 10^{-k-1} - 2^{-36}\right) \\ &= 20m + 20n - 20 \cdot 2^{-36} \\ F_{\max} &< 20 \cdot \left(c \cdot 2^q \cdot \left(1 + 2^{-63}\right) \cdot 10^{-k-1} - 0\right) \\ &< 20m + 20n + 20 \cdot 2^{-63} \cdot c \\ &< 20m + \lfloor 20n \rfloor + 1 \\ double : F_{\min} &> 20 \cdot \left(c \cdot 2^q \cdot 10^{-k-1} - 2^{-70}\right) \\ &= 20m + 20n - 20 \cdot 2^{-70} \\ &> 20m + \lfloor 20n \rfloor \\ F_{\max} &< 20 \cdot \left(c \cdot 2^q \cdot \left(1 + 2^{-127}\right) \cdot 10^{-k-1} - 0\right) \\ &< 20m + 20n + 20 \cdot 2^{-127} \cdot c \\ &< 20m + \lfloor 20n \rfloor + 1 \end{aligned} \tag{161}$$

Therefore, there is:

$$\begin{aligned} float : \lfloor F \rfloor &= 20m + \lfloor 20n \rfloor \\ double : \lfloor F \rfloor &= 20m + \lfloor 20n \rfloor \end{aligned} \tag{162}$$

In fact, in the above proof process, for float, $\lfloor F_{min} \rfloor \neq 20m + \lfloor 20n \rfloor$ may exist, but the code implementation has passed the exhaustive test, so this not-so-perfect proof process can be ignored. Therefore, the calculation of d can be simplified as follows:

$$\begin{aligned} d &= ten + one \\ &= (\lfloor F \rfloor + 1)//2 \\ &= (\lfloor 20 \cdot (c \cdot 2^q \cdot r \cdot 10^{-k-1} - n_x) \rfloor + 1)//2 \end{aligned} \tag{163}$$

For the float range, $n_x = n_{36}$; for the double range, $n_x = n_{70}$.

For double, quickly determine that $n == 0.25$ in equation (153).
When $n = 0.25$,$\lfloor 2^{64} \cdot n_r \rfloor = \lfloor 2^{64} \cdot n \rfloor = 2^{62}$. Therefore, the following condition can be used to quickly determine whether $n = 0.25$:

$$double : n = 0.25 \text{ if } \lfloor 2^{64} \cdot n_r \rfloor = 2^{62} \tag{164}$$

When $n \neq 0.25$,Calculate the range of the decimal part of the following expression:

$$4m + 4n = c \cdot 2^{q+2} \cdot 10^{-k-1} \tag{165}$$

Therefore, when equation (165) is not an integer, we have(the test file is test6.py):

$$2^{-62} < 4n - \lfloor 4n \rfloor < 1 - 2^{-62} \tag{166}$$

Calculate the two boundary cases of $4n$ that are closest to 1:

$$\begin{aligned} \lfloor 4n \rfloor = 0 \Rightarrow 4n - 0 < 1 - 2^{-62} \Rightarrow \lfloor 2^{64} \cdot n \rfloor \leqslant 2^{62} - 2 \\ \lfloor 4n \rfloor = 1 \Rightarrow 4n - 1 > 2^{-62} \Rightarrow \lfloor 2^{64} \cdot n \rfloor \geqslant 2^{62} + 1 \end{aligned} \tag{167}$$

Then there are:

$$\begin{aligned} \lfloor 2^{64} \cdot n \rfloor \neq 2^{62} \ \&\& \ \lfloor 2^{64} \cdot n \rfloor + 1 \neq 2^{62} \\ \Rightarrow \lfloor 2^{64} \cdot n_r \rfloor \neq 2^{62} \end{aligned} \tag{168}$$

Therefore, the following condition can be used to quickly determine whether $n \neq 0.25$:

$$double : n \neq 0.25 \text{ if } \lfloor 2^{64} \cdot n_r \rfloor \neq 2^{62} \tag{169}$$

In summary, for double, the following condition can be used to quickly determine whether $n = 0.25$:

$$\begin{aligned} double : n = 0.25 \text{ if } \lfloor 2^{64} \cdot n_r \rfloor = 2^{62} \\ double : n \neq 0.25 \text{ if } \lfloor 2^{64} \cdot n_r \rfloor \neq 2^{62} \end{aligned} \tag{170}$$

In the double range, introduce annother faster way to calculate *one*:

$$double : one = \lfloor \frac{\lfloor 2^{64} \cdot n_r \rfloor}{2^{64}} \cdot 10 + (n = 0.25)?0 : \left(2^{-1} + \frac{6}{2^{64}}\right) \rfloor \tag{171}$$

The proof of equation (171) is as follows:
when $n = 0.25$, $\lfloor \frac{\lfloor 2^{64} \cdot n_r \rfloor}{2^{64}} \cdot 10 \rfloor = \lfloor 10n \rfloor = 2$;
when $n \neq 0.25$, equation (171) can be equivalent to the following:

$$double : one = \lfloor \frac{\lfloor 2^{64} \cdot n_r \rfloor}{2^{64}} \cdot 10 + 2^{-1} + \frac{6}{2^{64}} \rfloor \tag{172}$$

According to the $10n - \lfloor 10n \rfloor$ range, *one* is represented as:

$$double : one = \begin{cases} \lfloor 10n \rfloor, \text{if } 10n - \lfloor 10n \rfloor < 0.5 \\ 8, \text{if } 10n - \lfloor 10n \rfloor = 0.5 \\ \lfloor 10n \rfloor + 1, \text{if } 10n - \lfloor 10n \rfloor > 0.5 \end{cases} = \lfloor 20n + 1 \rfloor // 2 \tag{173}$$

Therefore,when $n \neq 0.25$, we need to prove that the following equation holds:

$$\lfloor \frac{\lfloor 2^{64} \cdot n_r \rfloor}{2^{64}} \cdot 10 + 2^{-1} + \frac{6}{2^{64}} \rfloor = \begin{cases} \lfloor 10n \rfloor, \text{if } 10n - \lfloor 10n \rfloor < 0.5 \\ 8, \text{if } 10n - \lfloor 10n \rfloor = 0.5 \\ \lfloor 10n \rfloor + 1, \text{if } 10n - \lfloor 10n \rfloor > 0.5 \end{cases} = \lfloor 20n + 1 \rfloor // 2 \tag{174}$$

From the range of n, there is:

$$\frac{\lfloor 2^{64} \cdot n_r \rfloor}{2^{64}} \in \left(n_r - 2^{-64}, n_r\right] \tag{175}$$

Because the following conditions exist:

$$\begin{aligned} c \cdot 2^q \cdot 10^{-k-1} = m + n \\ c \cdot 2^q \cdot r \cdot 10^{-k-1} = m + n_r \end{aligned} \tag{176}$$

Therefore, the following relationship can be concluded:

$$\begin{aligned} n_r - n &= (r-1) \cdot c \cdot 2^q \cdot 10^{-k-1} \\ n_r &= (r-1) \cdot (m+n) + n \\ \Rightarrow n \leqslant n_r &< 2^{-127} \cdot c + n \\ n \leqslant n_r &< 2^{-127} \cdot 2^{53} + n \\ n \leqslant n_r &< 2^{-74} + n \end{aligned} \tag{177}$$

From equation (175) and (177), it can be concluded that:

$$\begin{aligned} \frac{\lfloor 2^{64} \cdot n_r \rfloor}{2^{64}} &\in \left(n - 2^{-64}, n + 2^{-74}\right) \\ \Rightarrow \frac{\lfloor 2^{64} \cdot n_r \rfloor}{2^{64}} \cdot 10 &\in \left(10n - 10 \cdot 2^{-64}, 10n + 10 \cdot 2^{-74}\right) \\ \Rightarrow \frac{\lfloor 2^{64} \cdot n_r \rfloor}{2^{64}} \cdot 20 &\in \left(20n - 20 \cdot 2^{-64}, 20n + 20 \cdot 2^{-74}\right) \\ \Rightarrow \frac{\lfloor 2^{64} \cdot n_r \rfloor}{2^{64}} \cdot 20 &\in \left(\lfloor 20n \rfloor + n_{20} - 20 \cdot 2^{-64}, \lfloor 20n \rfloor + n_{20} + 20 \cdot 2^{-74}\right) \end{aligned} \tag{178}$$

Discuss the range of values of x when the following conditions are met.

$$\lfloor \frac{\lfloor 2^{64} \cdot n_r \rfloor}{2^{64}} \cdot 20 + 1 + x \rfloor // 2 = \lfloor 20n + 1 \rfloor // 2 = one \tag{179}$$

Therefore, the following conclusions can be drawn:

$$\begin{aligned} \lfloor 20n \rfloor + n_{20} - 20 \cdot 2^{-64} + 1 + x \geqslant \lfloor 20n + 1 \rfloor &\Rightarrow x \geqslant 20 \cdot 2^{-64} - n_{20} \\ \lfloor 20n \rfloor + n_{20} + 20 \cdot 2^{-74} + 1 + x < \lfloor 20n + 2 \rfloor &\Rightarrow x < 1 - 20 \cdot 2^{-74} - n_{20} \end{aligned} \tag{180}$$

Suppose $x = 12 \cdot 2^{-64}$. Through the exhaustive method, all floating-point numbers that do not meet the following conditions can be obtained.

$$x = 12 \cdot 2^{-64} \geqslant 20 \cdot 2^{-64} - n_{20} \tag{181}$$

All floating-point numbers that do not meet condition (181) are as follows (in hexadecimal) :

$$\begin{aligned} &0xd17c0747bd76fa1, \\ &0xd27c0747bd76fa1, \\ &0x4d73de005bd620df, \\ &0x4d83de005bd620df, \\ &0x4d93de005bd620df, \end{aligned} \tag{182}$$

Through the exhaustive method, all floating-point numbers that do not meet the following conditions can be obtained.

$$x = 12 \cdot 2^{-64} < 1 - 20 \cdot 2^{-74} - n_{20} \tag{183}$$

All floating-point numbers that do not meet condition (183) are as follows (in hexadecimal) :

$$\begin{aligned} &0x612491daad0ba280, \\ &0x6159b651584e8b20, \\ &0x619011f2d73116f4, \\ &0x61c4166f8cfd5cb1, \\ &0x61d4166f8cfd5cb1, \end{aligned} \tag{184}$$

There are:

$$2\left(\frac{\lfloor 2^{64}\cdot n_r\rfloor}{2^{64}}\cdot 10+2^{-1}+\frac{6}{2^{64}}\right)=\frac{\lfloor 2^{64}\cdot n_r\rfloor}{2^{64}}\cdot 20+1+x \tag{185}$$

When the floating-point number is not within the above range (182) and (184), the condition (180) is satisfied. We have tested all floating-point numbers within the above-mentioned range (182) and (184), and the algorithm implementation code has output the correct result, that is, it satisfies the SW principle. The test process file is test8.py.

In summary, equation (174) and equation (171) holds. Therefore, equation (171) can be used to quickly calculate *one*.

3.6 Irregular Number

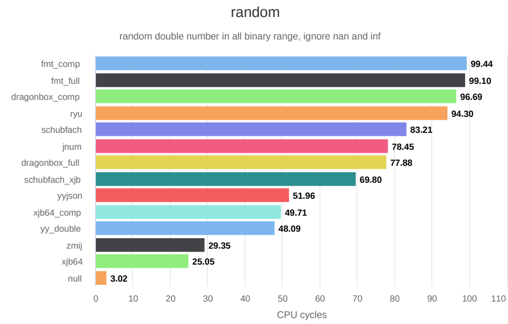
Due to the limited and small number of irregular floating-point numbers, there are a total of 2046 double floating-point numbers and 254 float floating-point numbers. The correctness of the algorithm code in this paper can be proved by the exhaustive method. Therefore, it is not introduced in this article. For the specific implementation process, please refer to the source code.

Table 1. All algorithms in the benchmark test.

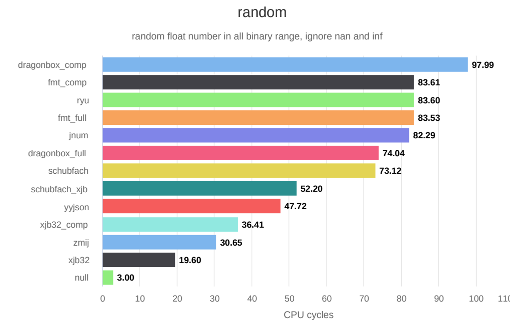
algorithm	float	double	description
Schubfach	Schubfach32	Schubfach64	author:Raffaello Giulietti,https://github.com/c4f7fcce9cb06515/Schubfach.
Schubfach_xjb	Schubfach32_xjb	Schubfach64_xjb	It is improved by Schubfach and has the same output result.
Ryu	Ryu32	Ryu64	author:Ulf Adams,https://github.com/ulfjack/ryu.
Dragonbox	Dragonbox32	Dragonbox64	author:Junekey Jeon,https://github.com/jk-jeon/Dragonbox.
fmt[10]	fmt32	fmt64	author:Victor Zverovich,https://github.com/fmtlib/fmt version:12.1.0
yy_double	-	yy_double	author:GuoYaoYuan,link:yy_double.
yy_json[11]	yy_json32	yy_json64	author:Guo YaoYuan,https://github.com/ibireme/yyjson version:0.12.0
teju_jagua[12]	teju32	teju64	author:Cassio Neri,https://github.com/cassioneri/teju_jagua.
xjb	xjb32	xjb64	this paper,https://github.com/xjb714/xjb.
zmij[13]	zmij32	zmij64	author:Victor Zverovich,https://github.com/vitaut/zmij.
jnum[14]	jnum32	jnum64	author:Jing Leng,https://github.com/lengjingzju/json/jnum.c.

4 BENCHMARK RESULT

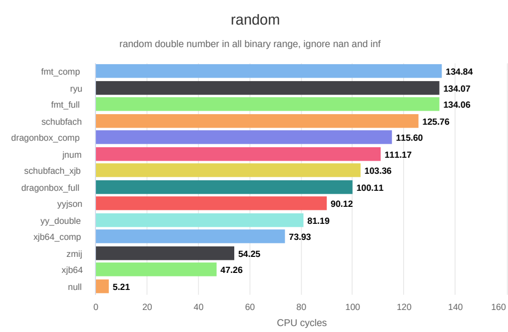
In fact, this article only discusses the binary to decimal part and does not discuss the decimal to string part. In the decimal to string section, the neon instruction set is adopted for the arm64 architecture, and SSE2/SSE4.1/AVX512IFMA instruction set is used for the x86-64 architecture to accelerate the conversion process. Please refer to the source code design. In the performance test comparison, we compared the time spent by the following several different algorithms converting floating-point numbers to decimal results and string, as shown in Table (1). Test process: Generate 2^{24} random numbers without 0, NaN, and Inf, measure the total time spent converting all floating-point numbers to decimal results, and obtain the average time for converting a single floating-point number to decimal and string. The compiler used for AMD R7-7840H is icpx 2025.0.4, and the compiler used for Apple M1 is apple clang 17.0.0. The compilation option for all compilers is ”-O3 -march=native”. We conducted benchmark tests on two processors, and the test results are shown in Fig (1a) to Fig (1d).



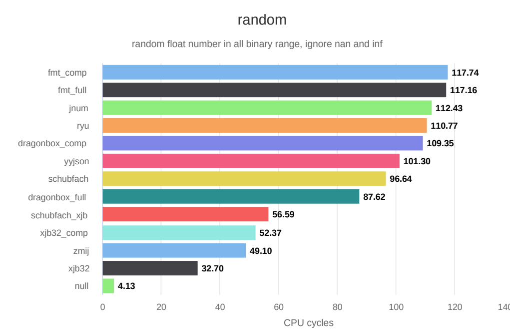
(a) Apple M1 double benchmark result



(b) Apple M1 float benchmark result



(c) AMD R7-7840H double benchmark result



(d) AMD R7-7840H float benchmark result

Fig. 1. Benchmark results

Special note: The algorithm of teju_jagua only supports float/double to decimal, because its author did not implement the source code of decimal to string. yy_double only supports double. Dragonbox_comp,fmt_comp and xjb_comp represent the versions of the compressed constant lookup table. Dragonbox_full and fmt_full represent uncompressed constant lookup table.

From the benchmark results, it can be seen that the performance of the algorithm in this paper is better than other algorithms in most cases.

5 CONCLUSIONS AND FUTURE WORK

This paper proposes a new floating-point number to string conversion algorithm. The algorithm improves the calculation process of Schubfach algorithms, reduces the number of multiplication operations, and optimizes some calculation steps. The algorithm has been implemented in C/C++ language and passed exhaustive tests. The benchmark results show that the performance of the algorithm is better than most existing algorithms in most cases. Future work includes further optimization of the algorithm to improve performance, especially for parallel computing on x86-64 and arm64 architecture,and compatibility with the msvc compiler.

REFERENCES

[1] Guy L. Steele and Jon L. White. 1990. How to print floating-point numbers accurately. *SIGPLAN Not.* 25, 6 (June 1990), 112–126. doi:10.1145/93548.93559

[2] Florian Loitsch. 2010. Printing floating-point numbers quickly and accurately with integers. *SIGPLAN Not.* 45, 6 (June 2010), 233–243. doi:10.1145/1809028.1806623

[3] Marc Andrysco, Ranjit Jhala, and Sorin Lerner. 2016. Printing floating-point numbers: a faster, always correct method. *SIGPLAN Not.* 51, 1 (Jan. 2016), 555–567. doi:10.1145/2914770.2837654

[4] Ulf Adams. 2018. Ryū: fast float-to-string conversion. *SIGPLAN Not.* 53, 4 (June 2018), 270–282. doi:10.1145/3296979.3192369

[5] Ulf Adams. 2019. Ryū revisited: printf floating point conversion. *Proc. ACM Program. Lang.* 3, OOPSLA, Article 169 (Oct. 2019), 23 pages. doi:10.1145/3360595

[6] R. Giulietti. 2020. The Schubfach Way to Render Doubles. (Sept. 2020). https://drive.google.com/file/d/1KLtG_LaIbK9ETXI290zqCxvBW94dj058/view

[7] Junekey Jeon. 2020. Grisu-Exact: A Fast and Exact Floating-Point Printing Algorithm. (Sept. 2020). https://github.com/jk-jeon/Grisu-Exact/blob/master/other_files/Grisu-Exact.pdf.

[8] Junekey Jeon. 2024. Dragonbox: A New Floating-Point Binary-to-Decimal Conversion Algorithm. (July 2024). https://github.com/jk-jeon/dragonbox/blob/master/other_files/Dragonbox.pdf

[9] Guo YaoYuan. 2024. (Nov. 2024). https://github.com/ibireme/c_numconv_benchmark/blob/master/vendor/yy_double/yy_double.c

[10] Victor Zverovich. 2025. (Oct. 2025). https://github.com/fmtlib/fmt

[11] Guo YaoYuan. 2025. (Aug. 2025). https://github.com/ibireme/yyjson

[12] Cassio Neri. 2025. (Nov. 2025). https://github.com/cassioneri/teju_jagua

[13] Victor Zverovich. 2026. (Jan. 2026). https://github.com/vitaut/zmij

[14] Jing Leng. 2025. (Nov. 2025). https://github.com/lengjingzju/json/jnum.c

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009