BÁO CÁO DỰ ÁN CUỐI KỲ

Chapter 11

A Simple Program Whose Proof Isn't

NHÓM 10

GIẢNG VIÊN HƯỚNG DẪN: THS. HUỲNH XUÂN PHỤNG

MÔN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

LỚP: DASA230179_08

THÀNH VIÊN NHÓM 10

NGÔ THANH THANH

21110643

VÕ ĐỨC TOÀN

21110689

TRẦN MINH HOÀNG

21110461

TỔNG QUAN

PHẦN 1: TÌM HIỂU VẤN ĐỀ

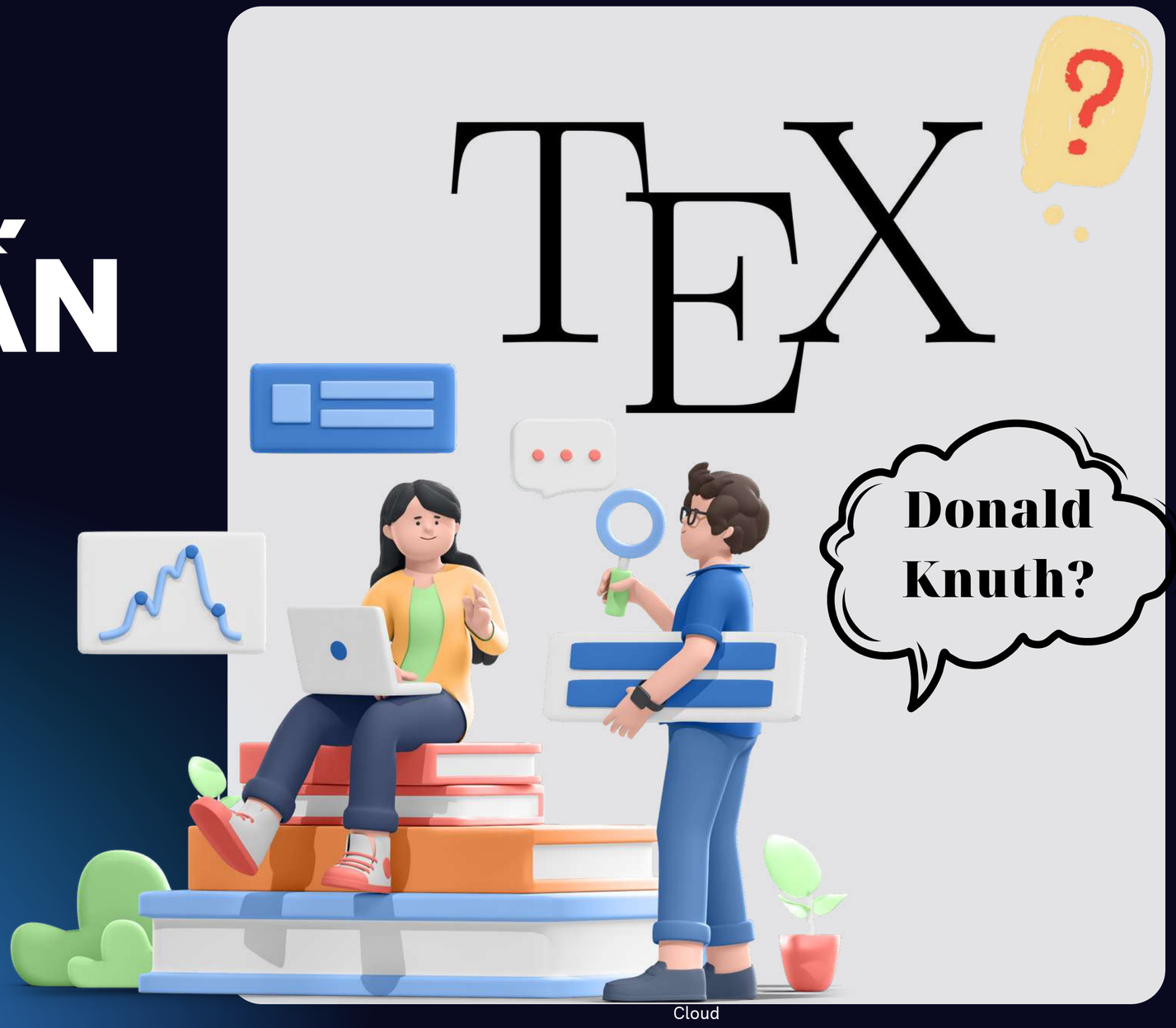
PHẦN 2: PHÂN TÍCH CHI TIẾT

PHẦN 3: PHÂN TÍCH ỨNG DỤNG VÀ HƯỚNG PHÁT TRIỂN CỦA THUẬT TOÁN

PHẦN 1

TÌM HIỂU VẤN ĐỀ

- NỘI DUNG
- CẤU TRÚC
- Ý NGHĨA
- KIẾN THỨC LIÊN QUAN
- THUẬT NGỮ



PHẦN 1: TÌM HIỂU VẤN ĐỀ

1.1 Nội dung chính

Trình bày quá trình tìm cách chứng minh tính đúng đắn của một chương trình đơn giản chuyển đổi từ số nguyên n sang phân số thập phân trong chương trình TeX mà chính tác giả đã tìm ra

PHẦN 1: TÌM HIỂU VẤN ĐỀ

1.2 Cấu trúc của bài báo

GỒM 5 PHẦN

1. Chuyển đổi từ phân số thập phân sang số nhị phân dấu chấm động
2. Chuyển đổi theo cách khác
3. Ý tưởng chứng minh
4. Hoàn thành bằng chứng
5. Kết luận

PHẦN 1: TÌM HIỂU VẤN ĐỀ

1.3 Ý nghĩa của bài báo

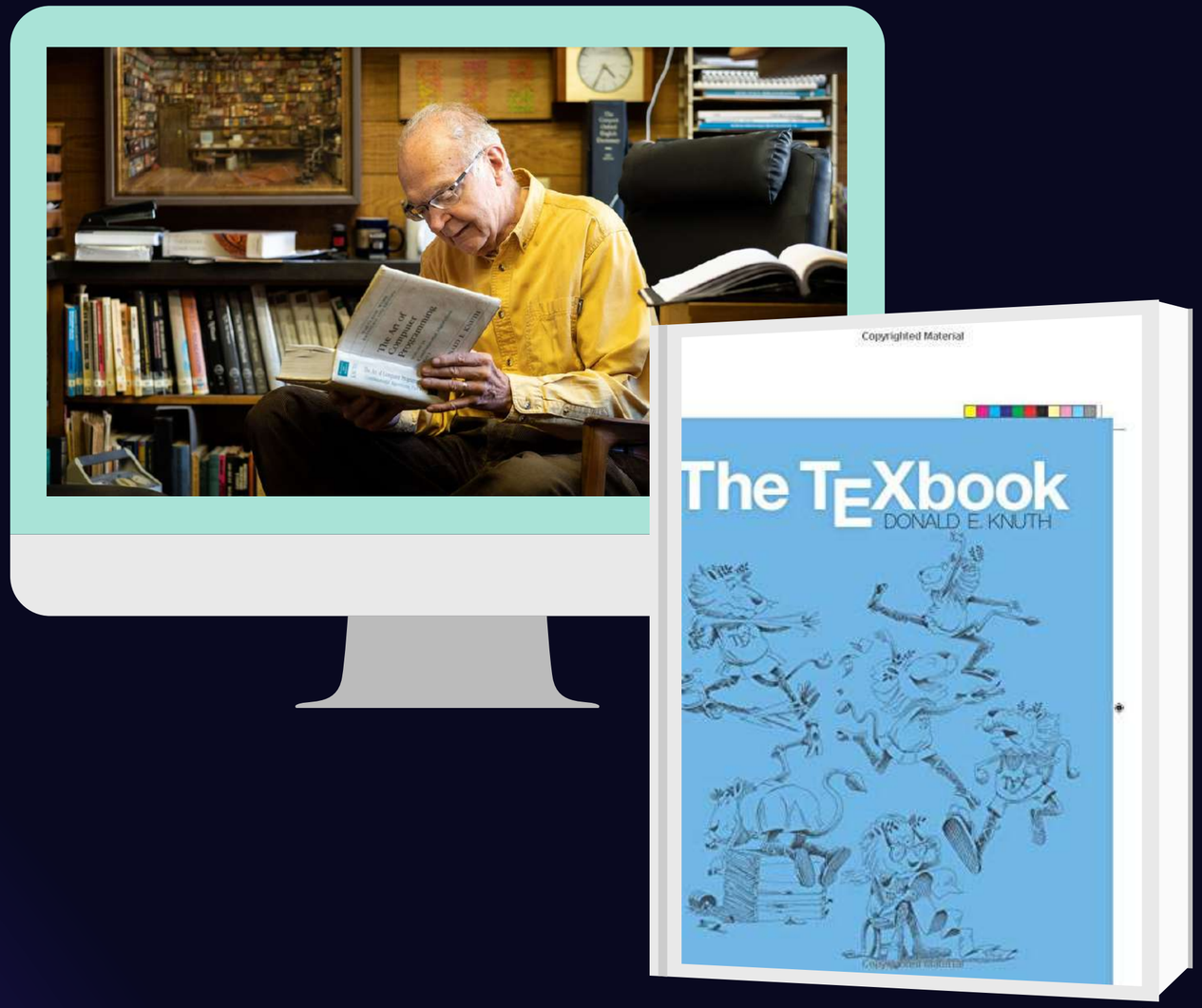
Giúp người đọc có một cái nhìn, lối tư duy hoàn toàn mới

Phân tích, lập luận sắc bén

1.4 Kiến thức liên quan

TÌM HIỂU VỀ CHƯƠNG TRÌNH TEX

- Là một hệ thống sắp chữ được viết bởi Donald Knuth và giới thiệu lần đầu vào năm 1978.
- TeX được thiết kế với hai mục đích chính: cho phép bất kì ai cũng có thể tạo ra những cuốn sách chất lượng cao với ít công sức nhất, và cung cấp một hệ thống sắp chữ cho ra cùng một kết quả trên mọi máy tính, ngay bây giờ và cả trong tương lai.
- Phần mềm TeX hội tụ một số thuật toán thú vị và đã dẫn đến một số luận án của các học trò của Knuth.



PHẦN 1: TÌM HIỂU VẤN ĐỀ

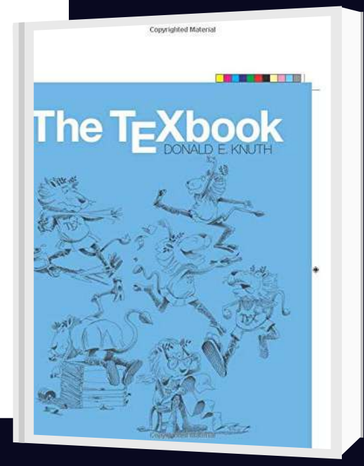
1.4 Kiến thức liên quan

Nghĩa là sao?

TEX USES SCALED POINT AS ITS MINIMAL UNIT.

TeX book said : “TeX represents all dimensions internally as an integer multiple of the tiny units called sp: 65536 spo = 216 sp = 1 pt. TeX actually does its calculations with integer multiples of 2-16 ”..

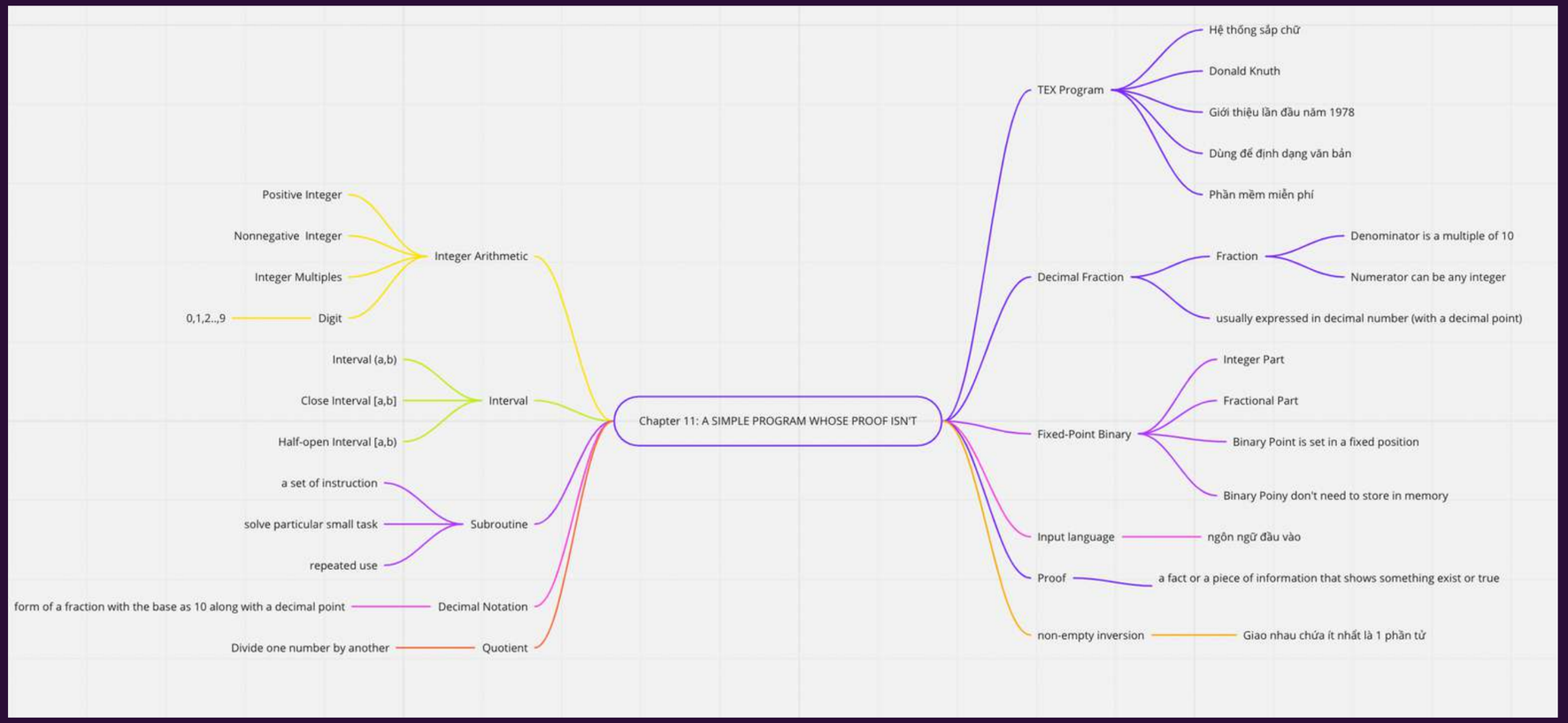
TeX thực hiện số học nhị phân, sâu bên trong chương trình. Số bạn nhập được nhận dạng lớn hơn 0 và do đó, thứ nguyên được đặt thành giá trị dương nhỏ nhất.



Và để thực hiện được điều này TeX cần thuật toán chuyển đổi từ số thập phân sang số nguyên chia tỷ lệ.

PHẦN 1: TÌM HIỂU VẤN ĐỀ

1.5 Giải thích thuật ngữ



PHẦN 2

PHÂN TÍCH CHI TIẾT

- Chuyển đổi phân số thập phân thành số nhị phân dấu chấm động
- Chuyển đổi theo cách khác
- Mầm mống của bằng chứng
- Hoàn thành bằng chứng
- Kết luận

PHẦN 2: PHÂN TÍCH CHI TIẾT

2.1 Chuyển đổi phân số thập phân thành số nhị phân dấu chấm động

Bài toán đặt ra

TEX làm việc nội bộ với bội số nguyên của 2^-16 nhưng ngôn ngữ đầu vào lại dùng ký hiệu thập phân (như 0.1, 0.07,...). Vì vậy, TEX cần một quy trình để chuyển phân số thập phân đã cho

PHẦN 2: PHÂN TÍCH CHI TIẾT

2.1 Chuyển đổi phân số thập phân thành số nhị phân dấu chấm động

INPUT	Phân số thập phân có dạng 0.d1d2...dk
OUTPUT	Số nguyên n sao cho 0.d1d2...dk = n*2^-16

$$n = \left\lfloor \frac{m_0 + 1}{2} \right\rfloor, \quad m_l = \left\lfloor 2^{17} \sum_{j=l+1}^{k} d_j / 10^{j-l} \right\rfloor.$$

THUẬT TOÁN P1

Thuật toán có tính hữu hạn.

- **Thuật toán:**

Bước 1: Gán số nhỏ hơn khi so sánh giữa số chữ số phần thập phân với 17 cho l
Bước 2: m = 0
Bước 3: Nếu l = 0 thì chuyển tới bước 7
Bước 4: m = (m+131072*chữ số phần thập phân thứ l) div 10
Bước 5: l = l - 1
Bước 6: Quay lại bước 3
Bước 7: n = (m+1) div 2

Cài đặt chương trình P1 bằng ngôn ngữ C++

```cpp
#include <iostream>
#include <string.h>
using namespace std;
int chusangso (char x)
{
    return (int)x - ('0' - 0);
}
int main ()
{
    string s;
    cin >> s;
    double m = 0;
    int l = min((int)s.length(), 17);
    for (int i = l; i >= 1; i--){
        m = (131072 * chusangso(s[i-1]) + m) / 10;
    }
    int n = (m + 1) / 2;
    cout << n;
    return 0;
}
```

KẾT QUẢ KHI CHẠY CHƯƠNG TRÌNH

Giả sử ta cần chuyển đổi số thập phân 0.125 thành số nguyên nguyên thì ta sẽ nhập 125 và đây là kết quả sau khi chạy chương trình trả về 8192 là bội của 2^-16

```
125
8192Program ended with exit code: 0
```

All Output ⌃ Filter

PHẦN 2: PHÂN TÍCH CHI TIẾT

2.2 Chuyển đổi theo cách khác

Bài toán ngược lại

Cho số nguyên n giả sử trong phạm vi 0<n< 2^16. Tìm phân số thập phân .d1d2...dk sao cho .d1d2...dk xấp xỉ 2^-16n

PHẦN 2: PHÂN TÍCH CHI TIẾT

2.2 Chuyển đổi theo cách khác

INPUT	Số nguyên n
OUTPUT	Số thập phân dưới dạng .d1d2...dk sao cho n*2-16 = .d1d2...dk

Giải pháp cho bài toán này là chuyển đổi nghịch đảo sao cho k càng nhỏ càng tốt. Tức là tìm kiếm một phân số thập phân ngắn nhất có thể tái tạo giá trị n đã cho. Hơn nữa, thường có hai phân số thập phân có cùng độ dài mà cả hai đều mang lại cùng một giá trị.

THUẬT TOÁN P2

Thuật toán có tính hữu hạn.

- **Thuật toán:**

Bước 1: j = 0
Bước 2: s = 10*n+5
Bước 3: t =10
Bước 4: Nếu s <= t thì chuyển tới bước 11
Bước 5: Nếu t > 65536 thì s = s + 32768 - (t div 2)
Bước 6: j = j+1
Bước 7: Chữ số phần thập phân thứ j = s div 65536
Bước 8: s = 10 * (s mod 65536)
Bước 9: t =10 * t
Bước 10: Quay lại bước 4
Bước 11: k = j

Cài đặt chương trình P2 bằng ngôn ngữ C++

```cpp
#include <iostream>
using namespace std;
int main ()
{
    int d[10];
    int n;
    cin >> n;
    int j = 0;
    int s = 10*n + 5;
    int t = 10;
    while (s > t){
        if (t > 65536) s = s + 32768 - (t/2);
        j++;
        d[j] = s / 65536;
        s = 10 * (s % 65536);
        t = 10 * t;
    }
    int k = j;
    for (int i = 1; i <= k; i++) cout << d[i];
    return 0;
}
```

KẾT QUẢ KHI CHẠY CHƯƠNG TRÌNH

```
1
00002Program ended with exit code: 0
```

All Output

Filter

3. Germs of Proof

-Tóm tắt: Tác giả đưa ra các bằng chứng để chứng minh thuật toán P3 gần thực hiện đúng mong muốn rồi từ đó chúng minh được thuật toán P2 đúng

3. Germs of a Proof

The invariant relationship that I had in mind when I wrote the program in the previous section — the relation that explains the intrinsic meaning of the variables s and t in connection with other data of the problem — is not easy for me to formalize. Perhaps I am too close to the program, too incapable of analyzing my own thought processes. I was thinking (I think) of the set of all possible continuations of the digits already determined.

In other words, if $d_1 \ldots d_j$ have already been computed, I was thinking of the set of all strings of decimal digits $d_{j+1} \ldots d_k$ such that

$$.d_1 \ldots d_j d_{j+1} \ldots d_k$$

-Tác giả giải thích mối quan hệ của biến s và t trong chương trình

-Tập hợp tất cả các chuỗi chữ số thập phân dj+1....dk sao cho sẽ tạo ra số n khi được xử lý bởi chương trình đầu tiên

-Dừng nếu chuỗi trống có trong tập hợp

In the following proof I will use Lyle Ramshaw's convention, inspired by Pascal syntax, that $[a\,..\,b)$ denotes the set of real numbers x in the range

$$a \le x < b.$$

-Quy ước của Lyle Ramshaw

-[a … b) biểuthị tập hợp a <= x < b

-Tập hợp các chuỗi chữ số dj+1...dk tạo ra kết quả cho trước n nằm trong khoảng nửa mở nào đó [a..B].

-Các giá trị cho phép của dj+1, nếu chúng ta muốn tăng j và giữ lại một số liên tục khác, là các chữ số thập phân d sao cho khoảng

$[d/10\,..\,(d+1)/10)$ có một giao điểm khác với [a..B)

Có nghĩa là các điều kiện $0 \le d \le 9, \quad d/10 < \beta, \quad \text{and} \quad \alpha < (d+1)/10$

that lie in some half-open interval $[\alpha\,..\,\beta)$. Initially we have $j = 0$, and this interval is

$$\left[2^{-16}\left(n - \frac{1}{2}\right)\,..\,2^{-16}\left(n + \frac{1}{2}\right)\right).$$

In general if the interval is $[\alpha\,..\,\beta)$, the empty string is in the set of possible continuations $d_{j+1} \ldots d_k$ if and only if $\alpha \le 0$ and $\beta > 0$. Furthermore, the permissible values of d_{j+1}, if we wish to increase j and retain a nonempty set of continuations, are those decimal digits d such that the interval

$$[d/10\,..\,(d+1)/10)$$

has a nonempty intersection with $[\alpha\,..\,\beta)$. This means that the conditions

$$0 \le d \le 9, \quad d/10 < \beta, \quad \text{and} \quad \alpha < (d+1)/10$$

are necessary and sufficient for d to be an acceptable choice of d_{j+1} (if we ignore complications of optimality).

-đặt j := j + 1 và d[j] := d.Sau đó khoảng mới [a '.. B') thay thế [a..B) nên saocho

$$[10\alpha - d\,..\,10\beta - d).$$

-Các giá trị phân tích này của s và t, tương ứng với các giá trị a và B ban đầu là 2^-16(n-1/2) và 2^-16(n+1/2) , do đó s = 10n + 5; t = 10.

are necessary and sufficient for d to be an acceptable choice of d_{j+1} (if we ignore complications of optimality).

Suppose we set $j := j + 1$ and $d[j] := d$. Then the new interval $[\alpha'\,..\,\beta')$ replacing $[\alpha\,..\,\beta)$ should be such that

$$\frac{d+\alpha'}{10} = \alpha, \quad \frac{d+\beta'}{10} = \beta.$$

In other words, the new interval should be

$$[10\alpha - d\,..\,10\beta - d).$$

This analysis has used real numbers, but we want to stick to integers in the calculation. Let us therefore represent α and β implicitly by the integer variables s and t, where

$$10\alpha = 2^{-16}(s - t); \quad 10\beta = 2^{-16}s.$$

The initial values of s and t, corresponding to the initial α and β values $2^{-16}(n - \frac{1}{2})$ and $2^{-16}(n + \frac{1}{2})$, are therefore

$$s = 10n + 5; \quad t = 10.$$

The conditions for an admissible d translate into

$$0 \le d \le 9, \quad s > 2^{16}d, \quad \text{and} \quad s - t < 2^{16}d + 2^{16}.$$

When j is increased by 1 and the next output digit is set to d, we want to set

$$s := 10(s - 2^{16}d), \quad t := 10t.$$

These manipulations on s and t change $[\alpha\,..\,\beta)$ into $[\alpha'\,..\,\beta')$ as discussed above.

From such considerations we can prove that the following program does almost what we want.

```
P3:  j := 0; s := 10 * n + 5; t := 10;
     repeat j := j + 1; d[j] := s div 65536;
       s := 10 * (s mod 65536); t := 10 * t;
     until s ≤ t;
     k := j.
```

-Các điều kiệnđể được chấp nhận d chuyển thành

$$0 \le d \le 9, \quad s > 2^{16}d, \quad \text{and} \quad s - t < 2^{16}d + 2^{16}.$$

-Khi j tăng lên 1 và chữ số đầu ra tiếp theo được đặt thành d

$$s := 10(s - 2^{16}d), \quad t := 10t.$$

-Các thao tác trên s và t thay đổi [a..B) thành [a '.. B')

-Chứng minh rằng chương trình P3 sau gần như thực hiệnnhững gì mong muốn

-.d1....djdj+1....dk chuyển đổi thành n / 2 ^ 16 nếu và chỉ nếu

$$2^{-16}\frac{s-t}{10} \le .d_{j+1}\dots d_k < 2^{-16}\frac{s}{10}.$$

-Xác minh điều này lưu ý rằng các điều kiện

$$s > t, \quad t = 10^{j+1}, \quad 0 \le s \le 655355$$

-Do đó 10j+1 < 655355 bất cứ khi nào bắt đầu lặp lại vòng lặp và chúng ta phải có j <= 5 khi chương trình kết thúc.

-Giá trị của k sẽ nhiều nhất là 5.

-Giá trị của k do P3 tạo ra thực tếlà nhỏ nhất; không có phân số rút gọn nào hơn .d1 ... dk sẽ tái tạo n

-Để chứng minh điều này, chúng ta quansátrằngthuật toán luôn chọn chữ số d lớn nhất có thể;nếu hai hoặc nhiều giá trị của d thỏa mãn

$$s - t - 2^{16} < 2^{16}d < s,$$

thì s div 65536 là lớn nhất

-Do đó P3 gần như là một giải pháp cho vấn đề của chúng ta. Nhiệm vụ còn lại duy nhất là tìm một giá trị gần đúng .d1...dk có độ dài nhỏ nhất gần với n/216 nhất có thể

-chương trình P3 cuối cùng bắt đầu vòng lặp của nó với j = 4 và t = 105 . Tác giả muốn sửa đổi phép tính ở vòng cuối cùng để chữ số cuối cùng d5 là "tốt nhất có thể" trong số các lựa chọn có sẵn.

This program preserves the desired invariant relation between variables, namely that a string $d_{j+1} \dots d_k$ of decimal digits will have the property that $.d_1 \dots d_j d_{j+1} \dots d_k$ converts to $n/2^{16}$ if and only if

$$2^{-16}\frac{s-t}{10} \le .d_{j+1}\dots d_k < 2^{-16}\frac{s}{10}.$$

We can verify this as follows: First we note that the conditions

$$s > t, \quad t = 10^{j+1}, \quad 0 \le s \le 655355$$

hold on each entry to the body of the **repeat** loop. Hence $10^{j+1} < 655355$ whenever the **repeat** loop begins, and we must have $j \le 5$ when the program terminates.

It is not difficult to verify that s is an odd multiple of 2^j, just before our new program increases j, because j never reaches 16. Therefore s is never a multiple of $2^{16} = 65536$, and the digit $d = s$ **div** 65536 always satisfies the conditions

$$0 \le d \le 9 \quad \text{and} \quad s - t - 2^{16} < 2^{16}d < s$$

that we have derived for admissibility of d as a digit.

This argument establishes that the decimal fraction $.d_1 \dots d_k$ computed by program *P3* will be converted back to n by *P1*. Furthermore, the value of k will be at most 5.

$$\frac{n}{2^{16}} \text{ rounded to 5 decimal places } = \frac{\lfloor 10^5 n/2^{16} + 1/2 \rfloor}{10^5}$$

4. Completion of the Proof

Chúng ta có thể chứng minh một các dễ dàng rằng các biến d1,....,dj và s của chương trình P3 tuân theo quan hệ bất biến
Giờ chúng ta sẽ quay lại chương trình P2.

If t > 65536 then s := s + 32768 – (t div 2)

Điều này chỉ có hiệu lực khi j=4, cụ thể là khi t = 105, vì t = 10j+1 và vòng lặp không bao giơ được nhập khi j>4
Khi j=4 ở đầu vòng lặp thì chúng ta tính d5. Và d5 >0 nên là ta có s > 65536. Nên ta có giá trị mới của s trong chương trinh P2

s' = s + 32768 – (t div 2)

s' là không âm nhưng s' < 655355 nên là đại lượng d5 = s' div 65536 sẽ là một chữ số từ 0 đến 9.

Now let's return to program *P2*, which is identical to *P3* except that the conditional instruction

$$\textbf{if } t > 65536 \textbf{ then } s := s + 32768 - (t \textbf{ div } 2)$$

has been inserted at the very beginning of the **repeat** loop. This instruction takes effect only when $j = 4$, namely when $t = 10^5$, because $t = 10^{j+1}$ and the loop is never entered when $j > 4$.

When $j = 4$ at the beginning of the loop, we are about to compute d_5. And we know that $d_5 > 0$, because we have proved that *P3* computes a shortest possible decimal expansion; therefore we have $s > 65536$. Therefore the new value of s in program *P2*,

$$s' = s + 32768 - (t \textbf{ div } 2),$$

is nonnegative, but less than 655355. Therefore the corresponding quantity $d_5 = s' \textbf{ div } 65536$ will be a digit in the interval $(0\,..\,9]$.

But by the invariant between d_1, d_2, d_3, d_4, and s, we know that

$$\begin{aligned} s' + 2^{16} \sum_{i=1}^{4} d_i 10^{5-i} &= s + 2^{15} - \frac{10^5}{2} + 2^{16} \sum_{i=1}^{4} d_i 10^{5-i} \\ &= 10^5 n + 2^{15}. \end{aligned}$$

4. Completion of the Proof

Nhưng bằng sự bất biến giữa d1, d2, d3, d4 và s, chúng ta biết rằng

$$s' + 2^{16} \sum_{i=1}^{4} d_i 10^{5-i} = s + 2^{15} - \frac{10^5}{2} + 2^{16} \sum_{i=1}^{4} d_i 10^{5-i}$$
$$= 10^5 n + 2^{15}.$$

Now let's return to program *P2*, which is identical to *P3* except that the conditional instruction

$$\textbf{if } t > 65536 \textbf{ then } s := s + 32768 - (t \textbf{ div } 2)$$

has been inserted at the very beginning of the **repeat** loop. This instruction takes effect only when $j = 4$, namely when $t = 10^5$, because $t = 10^{j+1}$ and the loop is never entered when $j > 4$.

When $j = 4$ at the beginning of the loop, we are about to compute d_5. And we know that $d_5 > 0$, because we have proved that *P3* computes a shortest possible decimal expansion; therefore we have $s > 65536$. Therefore the new value of s in program *P2*,

$$s' = s + 32768 - (t \textbf{ div } 2),$$

is nonnegative, but less than 655355. Therefore the corresponding quantity $d_5 = s' \textbf{ div } 65536$ will be a digit in the interval $(0\,..\,9]$.

But by the invariant between d_1, d_2, d_3, d_4, and s, we know that

$$s' + 2^{16} \sum_{i=1}^{4} d_i 10^{5-i} = s + 2^{15} - \frac{10^5}{2} + 2^{16} \sum_{i=1}^{4} d_i 10^{5-i}$$
$$= 10^5 n + 2^{15}.$$

4. Completion of the Proof

Vì vậy

$$\left\lfloor \frac{10^5 n}{2^{16}} + \frac{1}{2} \right\rfloor = \left\lfloor \frac{s'}{2^{16}} + \sum_{i=1}^{4} d_i 10^{5-i} \right\rfloor = \sum_{i=1}^{5} d_i 10^{5-i};$$

Nói cách khác, thì .d1d2d3d4d5 bằng n/2^16 được làm tròn thành 5 chữ số thập phân như mong muốn

Ta đã chỉ ra rằng P2 hoặc tính toán xấp xỉ thập phân ngắn nhất (duy nhất) .d1....dk có độ dài bé hơn 5 hoặc xấp xỉ độ dài 5 là lớn nhất(khi không có độ dài nhỏ hơn). Nên đã chứng minh được P2 là đúng.

Therefore

$$\left\lfloor \frac{10^5 n}{2^{16}} + \frac{1}{2} \right\rfloor = \left\lfloor \frac{s'}{2^{16}} + \sum_{i=1}^{4} d_i 10^{5-i} \right\rfloor = \sum_{i=1}^{5} d_i 10^{5-i};$$

in other words, $.d_1 d_2 d_3 d_4 d_5$ is equal to $n/2^{16}$ rounded to five decimal places, as desired.

We have shown that *P2* either computes the (unique) shortest decimal approximation $.d_1 \ldots d_k$ of length $k < 5$, or it computes the best approximation of length 5 (when there is no shorter one). This completes the proof that *P2* is correct.

So. Is there a better program, or a better proof, or a better way to solve the problem?

5. Kết luận

We have based our discussion on the particular case of 16-bit fixed-point binary fractions. But an examination of the proof shows that the ideas are quite general: If m is any positive number for which we seek the shortest-and-best decimal representation of n/m, given $0 < n < m$, we can use the following modification of program *P2*.

P4: $j := 0$; $s := 10 * n + 4$; $t := 10$;
repeat if $t > m$ **then** $s := s + 1 + (m$ **div** $2) - (t$ **div** $2)$;
$j := j + 1$; $d[j] := s$ **div** m;
$s := 10 * (s$ **mod** $m) + 9$; $t := 10 * t$;
until $s < t$;
$k := j$.

The resulting fraction $.d_1 \ldots d_k$ is an optimum representation subject to the conditions

$$\frac{n - 1/2}{m} \le .d_1 \ldots d_k < \frac{n + 1/2}{m}.$$

(If equality is impossible in shortest representations, a program more like *P2* will also be valid.)

When the denominator is not a power of 2, the converse problem (corresponding to *P1*) becomes much more interesting, especially if m is divisible by at least one prime number other than 2 or 5. Then the digits d_j can be relevant for arbitrarily large j. The reader may enjoy trying to construct an algorithm that reads a decimal input fraction $.d_1 \ldots d_k$ "on line" from an input tape and computes the unique value of n that satisfies the inequalities above, given m. It is desirable to limit the amount of auxiliary memory to $O(\log m)$ bits, not counting the digits on the tape. The input digits must be read once only, from left to right.

An algorithm similar to program *P3* was published in 1959 by Donald Taranto [6]. A generalization of Taranto's method, due to Guy L. Steele Jr. and Jon L. White, appears in the answer to exercise 4.4–3 of [4]. Programs *P1* and *P2* appear in sections 102 and 103, respectively, of the program for TEX [5].

THANKS FOR YOUR LISTENING!