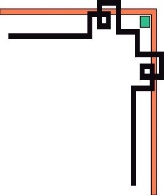
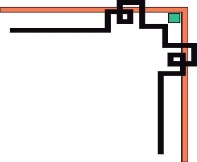
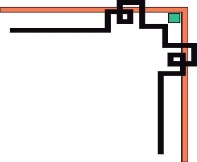
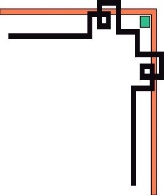
**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**



**KHOA CÔNG NGHỆ THÔNG TIN**

**🕯✡🕮🕮✡🕯**

**DỰ ÁN CUỐI KỲ**

**MÔN HỌC: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**CHAPTER 11**

**A SIMPLE PROGRAM WHOSE PROOF ISN’T**

**Mã LHP: DASA230179\_08**

**GVHD: Huỳnh Xuân Phụng**

**Thực hiện: Nhóm 10. Lớp thứ tư. Tiết 1-4**

**Sinh viên thực hiện**

|  |  |  |
| --- | --- | --- |
| STT | Họ và tên | MSSV |
| 1 | Ngô Thanh Thanh | 21110643 |
| 2 | Võ Đức Toàn | 21110689 |
| 3 | Trần Minh Hoàng | 21110461 |

***Tp. Hồ Chí Minh, tháng 12 năm 2022***

# LỜI CẢM ƠN

Lời nói đầu tiên, em xin được gửi đến thầy Huỳnh Xuân Phụng– giảng viên bộ môn lời cảm ơn chân thành và sâu sắc nhất.

Em xin cảm ơn sự quan tâm và giúp đỡ tận tình của thầy trong suốt quá trình giảng dạy. Cảm ơn thầy đã luôn giải đáp những thắc mắc cũng như đưa ra những nhận xét, góp ý giúp nhóm thực hiện cải thiện chất lượng công việc.

Vì khả năng còn hạn chế nên trong quá trình thực hiện báo cáo không tránh khỏi sai sót, kính mong nhận được những ý kiến đóng góp từ thầy để em có thể cải thiện hơn sau này.

Em xin chân thành cảm ơn.

MỤC LỤC

[LỜI CẢM ƠN 2](#_Toc123134584)

[PHẦN 1: TÌM HIỂU VẤN ĐỀ 1](#_Toc123134585)

[1. Nội dung chính của bài báo 1](#_Toc123134586)

[2. Cấu trúc của bài báo 1](#_Toc123134587)

[3. Ý nghĩa của bài báo 1](#_Toc123134588)

[4. Các kiến thức liên quan để đọc hiểu bài báo 1](#_Toc123134589)

[5. Giải thích ý nghĩa các từ vựng chuyên ngành 3](#_Toc123134590)

[PHẦN 2: PHÂN TÍCH CHI TIẾT 6](#_Toc123134591)

[Chương 11 6](#_Toc123134592)

[MỘT CHƯƠNG TRÌNH ĐƠN GIẢN MÀ SỰ KIỂM CHỨNG NÓ THÌ KHÔNG 6](#_Toc123134593)

[1. Chuyển đổi phân số thập phân thành số nhị phân dấu chấm động 6](#_Toc123134594)

[1.1 Nội dung chi tiết 6](#_Toc123134595)

[1.2 Phân tích thuật toán P1 8](#_Toc123134596)

[2. Chuyển đổi theo cách khác 10](#_Toc123134597)

[2.1 Nội dung chi tiết 10](#_Toc123134598)

[2.2 Phân tích thuật toán P2 13](#_Toc123134599)

[3. Mầm mống của bằng chứng 15](#_Toc123134600)

[3.1 Nội dung chi tiết 15](#_Toc123134601)

[3.2 Phân tích thuật toán P3 23](#_Toc123134602)

[4. Hoàn thành bằng chứng 25](#_Toc123134603)

[5. Kết luận 27](#_Toc123134604)

[5.1 Nội dung chi tiết 27](#_Toc123134605)

[5.2 Phân tích thuật toán P4 29](#_Toc123134606)

[PHẦN 3: PHÂN TÍCH ỨNG DỤNG VÀ HƯỚNG PHÁT TRIỂN CỦA THUẬT TOÁN 32](#_Toc123134607)

[1. Ứng dụng của thuật toán 32](#_Toc123134608)

[1.1 So sánh 2 phân số thập phân 32](#_Toc123134609)

[1.2 Cộng 2 phân số thập phân 33](#_Toc123134610)

[1.3 Trừ 2 phân số thập phân 34](#_Toc123134611)

[2. Hướng phát triển 35](#_Toc123134612)

# PHẦN 1: TÌM HIỂU VẤN ĐỀ

Trước khi đi vào phân tích chi tiết các thuật toán có trong bài báo, bước đầu tiên nhóm chúng em thực hiện là tìm hiểu vấn đề, nội dung chính của bài báo cũng như mục đích và ý nghĩa mà tác gỉả muốn truyền tải trong công trình nghiên cứu của ông.

## 1. Nội dung chính của bài báo

Bài báo trình bày quá trình tìm cách chứng minh tính đúng đắn của một chương trình đơn giản chuyển đổi từ sô nguyên n sang phân số thập phân trong chương trình TeX.

## 2. Cấu trúc của bài báo

Cấu trúc của bài báo gồm 5 phần.

Phần 1: Trình bày cách chuyển đổi từ phân số thập phân sang số nhị phân dấu chấm động với việc chứng minh râtts dễ dàng giúp ngừoi đọc làm quen, nắm bắt được vấn đề trước khi đi vào nội dung chính

Phần 2: Tác giả trình bày thuật toán cho vấn đề ngược mà chính ông nghĩ ra, hóa ra lại rất đơn giản nhưng chỉ có thể chứng minh bằng cách thực thi chương trình

Phần 3: Trình bày ý tưởng, nguồn gốc của việc chứng minh

Phần 4: Hoàn thành bằng chứng cụ thể

Phần 5: Kết luận

## 3. Ý nghĩa của bài báo

Phát hiện của tác giả giúp người đọc có một cái nhìn, lối tư duy hoàn toàn mới với những phân tích, lập luận vô cùng sắc bén của ông.

## 4. Các kiến thức liên quan để đọc hiểu bài báo

**Tìm hiểu về chương trình TeX**

TeX là một ngôn ngữ lập trình máy tính mô tả trang nhằm cải thiện chất lượng ký hiệu toán học trong sách được phát triển trong suốt nhưng năm 1977- 1986 bởi Donald Knuth, một giáo sư nổi tiếng ở trường đại học Stanford[[1]](#footnote-1)

TeX uses scaled point as its minimal unit.[[2]](#footnote-2) (TeX sử dụng điểm tỷ lệ như là đơn vị tối thiểu)

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

Khi nói:

\dimen 0 = 0.00000762939453125 pt

Lệnh \showthe\dimen0 sẽ trả lời

> 0.00002pt.

bởi vì đây là sự thể hiện của 1sp khi được hiển thị theo điểm.

Nghĩa là sao? TeX thực hiện số học nhị phân, sâu bên trong chương trình. Số bạn nhập được nhận dạng lớn hơn 0 (nhìn vào mã chương trình) và do đó, thứ nguyên được đặt thành giá trị dương nhỏ nhất.

Giá trị bạn nhập vào có thể xuất hiện nhỏ hơn 1sp, nhưng giá trị mà TeX sử dụng là 1sp.

*Chuyển đổi từ số thập phân sang số nguyên chia tỷ lệ*

- The integral part is just multiplied by 65536, so it's only interesting for numbers in the form 0.d0d1…dk–1

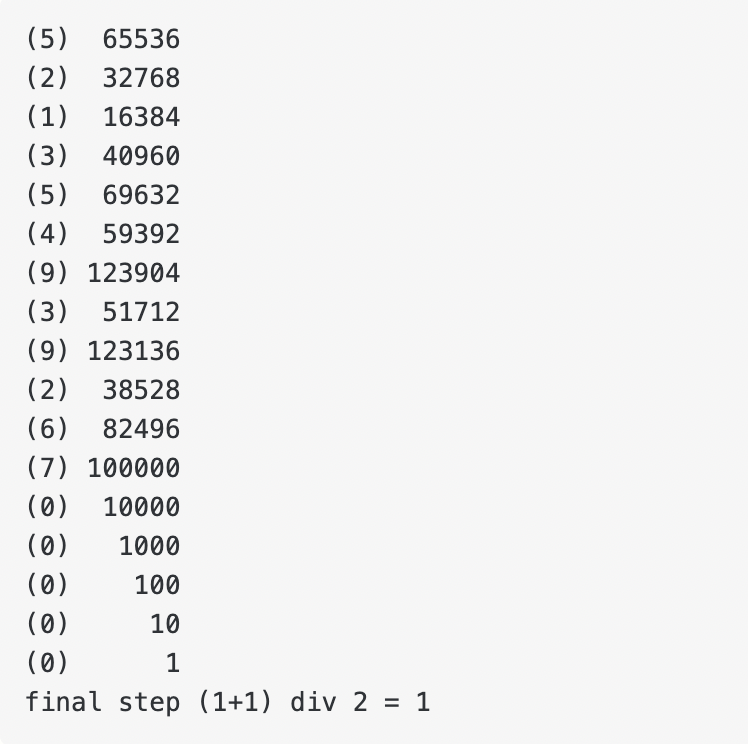
- The final scaled integer is stored in a which is initialized as 0

- Then we start from the far right and at each step a is assigned the result of

(a + di\* 217) div 10

- Finally, a is assigned the value (a + 1) div 2 (with truncation)

*For example, If we carry out this algorithm, we get the following numbers as values of a (in parentheses the digit examined)*



## 5. Giải thích ý nghĩa các từ vựng chuyên ngành

* TEX-program: 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, đã dẫn đến một số luận án của các học trò của Knuth.
* Proof: a fact or a piece of information that shows something exist or true
* Subroutine : in a large computer program, a set of instructions for a particular small task that is available for repeated use (chương trình con) conventional method: phương pháp thông thường
* Decimal fractions: the fractions in which the denominator (mẫu số) must be 10 or a multiple of 10 like 100, 1000, 10000, and so on. The numerator (tử số )can be any integer (between -infinity and +infinity). These decimal fractions are usually expressed in decimal numbers (numbers with a decimal point).
* Fixed-Point binary: the binary point is set in a fixed position, and therefore it does not need to be stored in memory (số nhị phân dấu chấm động)
* Integer multiples: bội số nguyên - Input language : ngôn ngữ đầu vào
* Decimal notation: The representation of a number in the form of a fraction with the base as 10 along with a decimal point. Ex: Rewrite in decimal notation 7.85\*10^6 = 7850000
* Digit: any one of the ten numbers 0 to 9
* Nonnegative integer: số nguyên không âm
* Compute: to calculate an answer or amount by using a machine
* Integer arithmetic: số nguyên số học - Intermediate results: kết quả trung gian
* Arbitrarily large: tùy tiện lớn
* Quotient: the result you get when you divide one number by another (thương số)
* Discard: throw sth away or get rid of it because you no longer want or need it
* Recurrence: sự tái diễn, sự lặp lại
* Conversion: sự chuyển đổi
* Inverse problem: vấn đề đảo ngược
* Conversion algorithm: thuật toán đảo ngược
* Implementation: sự thi hành
* Typeset: sắp chữ
* Yield: năng suất
* Dijkstra's famous dictum (Châm ngôn): “Testing can reveal the presence of faults, but not their absence.”: It's mean that testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use. When you test software, you execute a program using artificial data. Counterexample: an example that opposes or contradicts an idea or theory.
* Invariant relationship: mối quan hệ bất biến
* Intrinsic meaning: ý nghĩa nội hàm
* non-empty: a non-empty set is one which contains at least one member: Lyle Ramshaw's convention (quy ước)
* Syntax: cú pháp
* Interval: a space between 2 point (a,b)
* Close interval: [a,b] - Half-open interval: [a,b)
* Mathematical interpretation: Giải thích toán học
* Permissible value: giá trị cho phép
* Intersection: sự giao nhau
* Manipulation: thao tác
* Odd multiple: bội số lẻ
* Prime number: số nguyên tố
* Auxiliary memory: Auxiliary memory holds programs and data for future use, and, because it is nonvolatile (like ROM), it is used to store inactive programs and to archive data.



# PHẦN 2: PHÂN TÍCH CHI TIẾT

Sau khi nắm được nội dung, mục đích chính của bài báo cũng như có được các kiến thức, nguyên liệu cần thiết để hiểu những gì tác giả đề cập tới, nhóm em sẽ đi vào phân tích chi tiết bài báo gồm: chứng minh các công thức, phân tích thuật toán, chạy thuật toán,… Bản trình bày và phân tích bằng tiếng việt dưới đây là những gì mà chúng em hiểu và tiếp thu được trong quá trình tìm hiểu và tham khảo từ nhiều nguồn tài liệu khác nhau.

# Chương 11

# MỘT CHƯƠNG TRÌNH ĐƠN GIẢN MÀ SỰ KIỂM CHỨNG NÓ THÌ KHÔNG

Khi tác giả đang viết chương trình TEX, ông cần xây dựng các chương trình con để giải quyết các nhiệm vụ nhỏ. Giải pháp cho một trong những vấn đề hóa ra lại đặc biệt thú vị bởi nì nó rất ngắn và là vài dòng code đơn giản nhưng tác giả thấy không có một cách nào dễ dàng để thể hiện tính đúng đắn của nó bằng phương pháp thông thường.

Mục đích của tác giả trong phần ghi chú này là để trình bày rằng chương trình con với một phác thảo về bằng chứng tốt nhất mà ông biết, hi vọng rằng những độc giả với nhiều kinh nghiệm hơn trong phương pháp chính thức sẽ đồng ý rằng thuật toán này thú vị và sẽ giúp ông tìm ra tôi nên làm điều gì.

## 1. Chuyển đổi phân số thập phân thành số nhị phân dấu chấm động

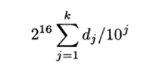
### 1.1 Nội dung chi tiết

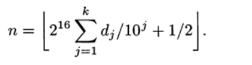
Đầu tiên, tác giả đã giới thiệu một chương trình ngắn và việc kiểm chứng nó rất dễ dàng.

*Vấn đề:* TEX làm việc nội bộ với bội số nguyên của (như , ,…) 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

.d1d2…dk.  (k 1; 0 )

thành phân số nhị phân gần nhất có thể biểu diễn được.

* Việc cần làm:* Tìm bội số nguyên nhỏ nhất của 2-16  hay nói cách khác là làm tròn lượng:

thành số nguyên n gần nhất bằng cách lấy phần nguyên của

*Ví dụ:* Với phân số thập phân đầu vào là: 0.214 thì bội số nguyên nhỏ nhất của 2-16 cần tìm đó là :

216 \* ( 2\*10-1 + 1\*10-2 + 4\*10-3 ) = 216\* 0.214 = 14024.704

được làm tròn lên thành số nguyên

n = 14024.704 + 1/2 = 14025.204 14025

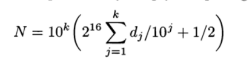
*Chú ý rằng:* - Số nguyên nhỏ nhất của n là 0 tức:

.d1d2…dk < .00000762939453125 = 2-17

- Giá trị lớn nhất có thể của n là 65536 tức:

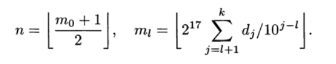
.d1d2…dk > .99999237060546875 = 1-2-17

Bởi vì giá trị đầu ra của n là một số nguyên không âm trong một phạm vi giới hạn và vì các giá trị đầu vào dj là các số nguyên không âm nhỏ nên kỳ vọng tính toán n với số học nguyên theo cách mà giá trị trung gian luôn nhỏ một các hợp lý.

Tổng số chữ số k có thể lớn tùy ý. Do đó, chúng ta không thể giải quyết vấn đề chỉ bằng cách tính số nguyên

Và sau đó để n = N/10k ; giá trị của N và 10k có thể quá lớn đối với phần cứng máy tính của chúng ta.

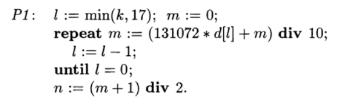
Tuy nhiên, cần chú ý rằng giá trị của dj với j > 17 hoàn toàn không gây ảnh hưởng gì tới kết quả n vì vậy chúng ta có thể bỏ qua từ giá trị từ d18 trở đi.

Khi đó, có thể tính n bằng cách viết:

Trong đó: l = min (k,17)

Giá trị trung gian ml tuân theo sự lặp lại đơn giản:

Dưới đây chính là là một đoạn của chương trình chuyển đổi phân số thập phân thành phân số nhị phân với nhiệm vụ đi tìm số nguyên n là bội của 2-16



### 1.2 Phân tích thuật toán P1

Từ các ý trên, ta có thể khái quát hóa thuật toán chuyển đổi từ số thập phân sang số nguyên chia tỷ lệ (bội của 2-16) như sau:

- Input: Phân số thập phân có dạng 0.d1d2…dk

- Output: Sô nguyên n sao cho 0.d1d2…dk = n\*2-16

* **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

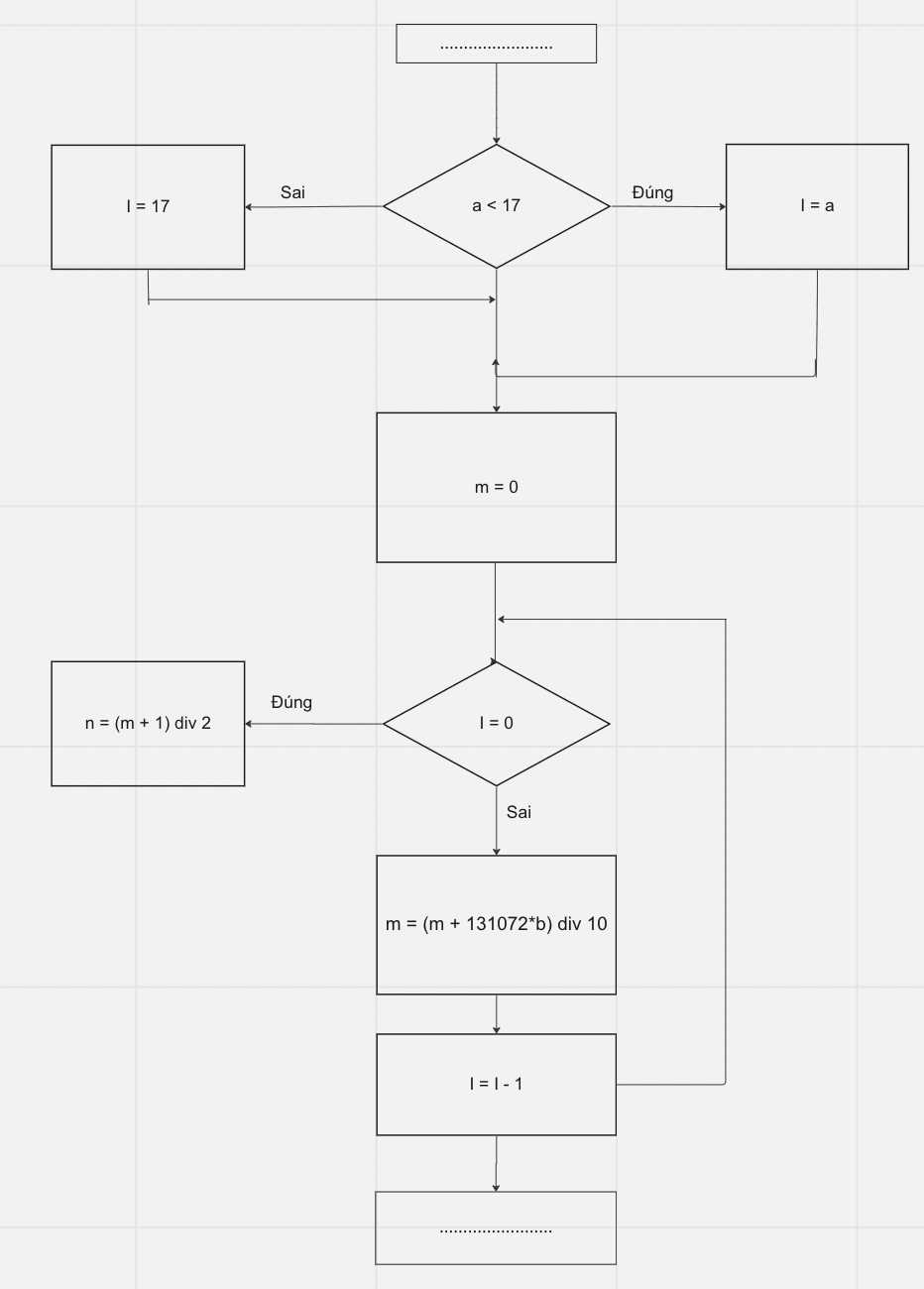
Bước 3: Nếu l = 0 thì chuyển tới bước 7

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

* **Sơ đồ giải thuật**

- Tính hữu hạn (Finiteness): thuật toán được xác định hữu hạn vì kết thúc sau một số bước và trả về được kết quả là số nguyên n thỏa điều kiện.

* **Cài đặt chương trình P1 bằng ngôn ngữ C++**

#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;

}

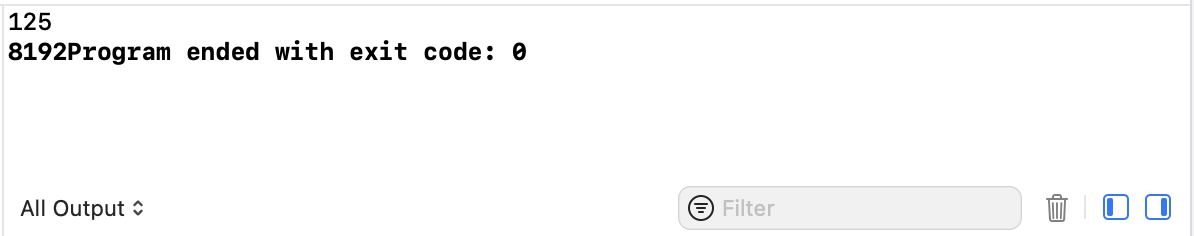
**int** n = (m + 1) / 2;

cout << n;

**return** 0;

}

Kết quả sau khi chạy chương trình:

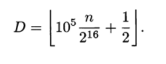


## 2. Chuyển đổi theo cách khác

### 2.1 Nội dung chi tiết

*Vấn đề ngược lại:* Cho số nguyên n giả sử trong phạm vi 0 216. Tìm phân số thập phân .d1d2…dk sao cho .d1d2…dk xấp xỉ 2-16n

Bài toán này giống như những bài toán khác, có một giải pháp đơn giản: Để k = 5 và d1d2d3d4d5 là các chữ số thập phân của số nguyên:



*Ví dụ:* Cho n = 8192 thì D = 105 \* 8192 \* 2-16 + = 12500.5

Sau đó, D/105 phải tái tạo n để thuật toán chuyển đổi tìm thấy một số n’ sao cho



(1)

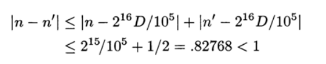
Mặt khác:



(2)

Áp dụng bất đẳng thức của trị tuyệt đối |a+b| |a| +|b|

Từ (1) & (2), ta có:

 + +

Hay

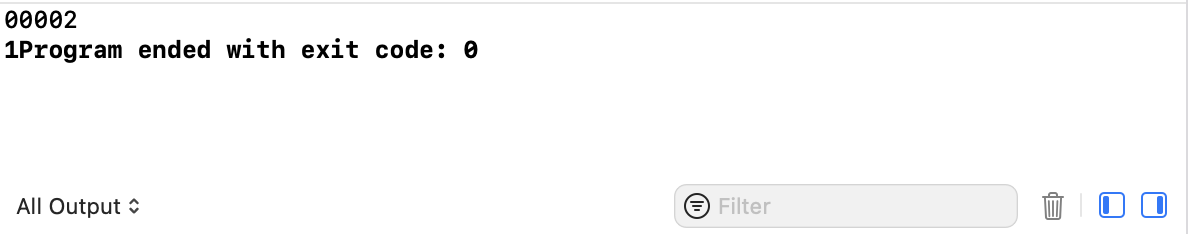
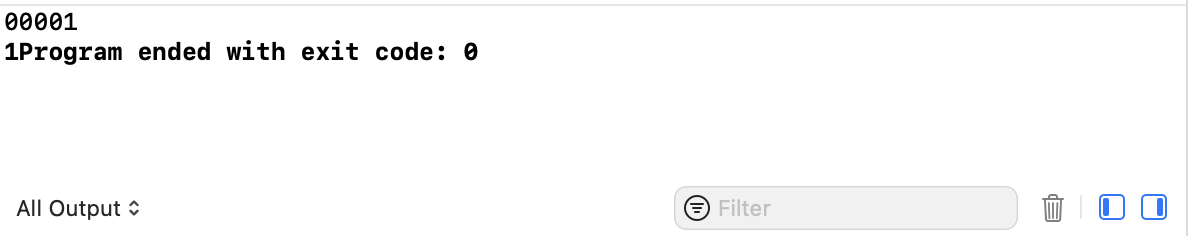
Và n phải bằng n’

Việc triển khai ban đầu của TeX lấy k = 5 như gợi ý ở trên không thỏa mãn bởi vì một người dùng đã yêu cầu quy tắc .4 điểm cho biết rằng TeX đã thực sự thiết lập quy tắc .39999 điểm. Phản hồi của TeX rất trugn thực nhưng chi tiết thừa đã gây mất tập trung và lộc xộn một cách không cần thiết.

Do đó, tốt hơn là tìm một giải pháp cho bài toán 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.

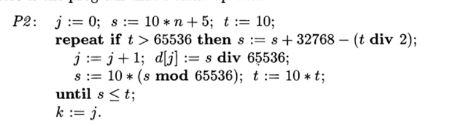
Hơn nữa, thường có hai phân số thập phân có cùng độ dài mà cả hai đều mang lại cùng một giá trị.

*Ví dụ:* Cả 0.00001 và 0.00002 đều trả về n = 1. Dùng chương trình P1 để kiểm tra



Vì cả 0.65536 và 1.31072 đều được làm tròn thành 1,0. Trong những trường hợp như vậy, nên chọn phân số thập phân gần nhất với n/216 (cụ thể là .00002 khi n = 1)

Hóa ra rằng có một chương trình đơn giản để tính toán các phân số thập phân ngắn nhất như vậy. Nhưng như tác giả nói ở phần giới thiệu, ông không biết một cách đơn giản để chứng minh nó đúng. Dưới đây là chương trình mà ông đã nghĩ ra



Để kiểm tra tính đúng đắn của chương trình trên ta có thể giả sử kiểm tra cho tất cả 65535 giá trị của n và giả sử kết quả các phân số .d1 ... dk đều tái tạo lại được giá trị ban đầu khi được chuyển đổi trở lại. Sau đó, chúng ta chỉ cần xác minh rằng không có phân số nào ngắn hơn hoặc phân số lân cận có độ dài bằng nhau tốt hơn thì thử nghiệm này sẽ chứng minh được chương trình đúng.

Nhưng thử nghiệm không phải là một cách tốt để đảm bảo tính đúng đắn, bởi vì nó không cho chúng ta cái nhìn sâu sắc về tổng quát hóa. Do đó, tác giả đã nỗ lực tìm kiếm một bằng chứng có thể hiểu được và mang tính giáo dục.

### 2.2 Phân tích thuật toán P2

Từ các ý tưởng trên, ta có thể khái quát thuật toán chuyển đổi ngược lại, từ số nguyên n thành phân số thập phân

- Input: số nguyên n

- Output: số thập phân dưới dạng .d1d2…dk sao cho n\*2-16 = .d1d2…dk

* **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

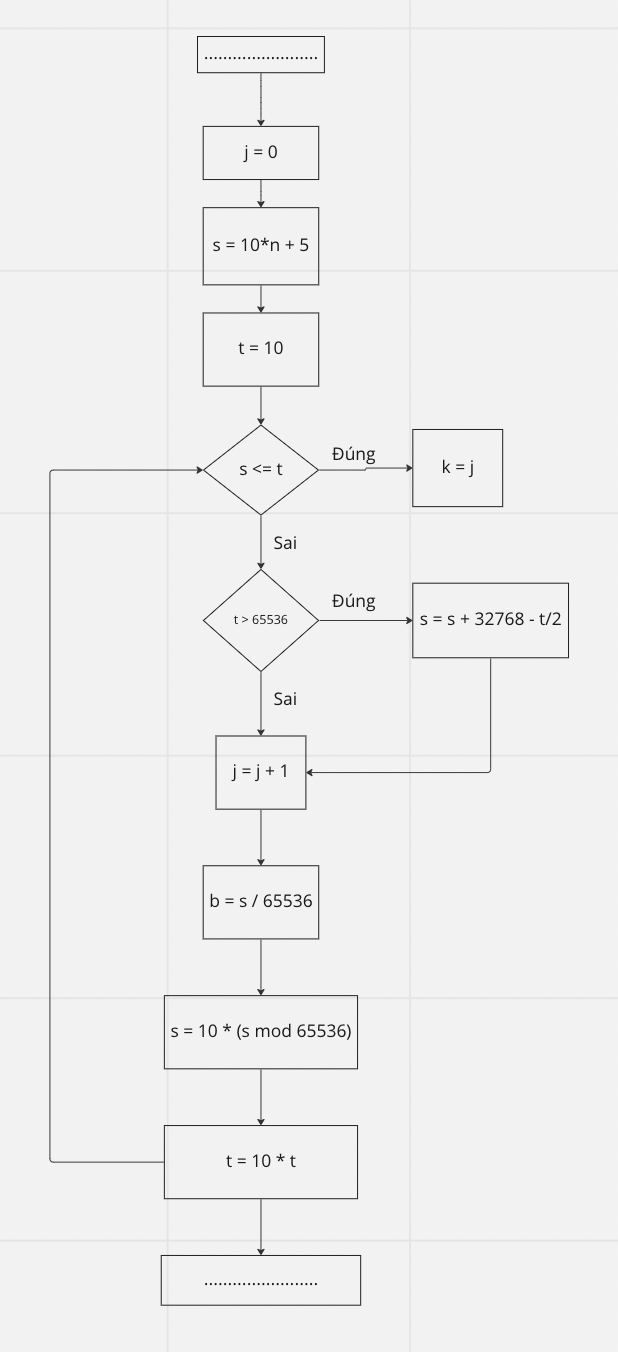
Bước 8: s = 10 \* (s mod 65536)

Bước 9: t =10 \* t

Bước 10: Quay lại bước 4

Bước 11: k = j

* **Sơ đồ giải thuật:**



- Tính hữu hạn (Finiteness): thuật toán được xác định hữu hạn vì kết thúc sau một số bước và trả về được kết quả là số thập phân thỏa điều kiện.

* **Cài đặt chương trình P2 bằng ngôn ngữ C++**

#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 sau khi chạy chương trình:



## 3. Mầm mống của bằng chứng

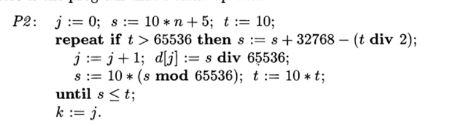
### 3.1 Nội dung chi tiết

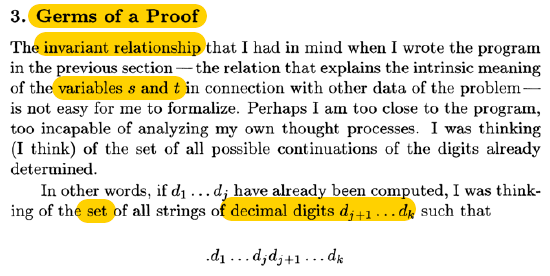
Tóm tắt: Tác giả đưa ra các bằng chứng để chứng minh thuật toán P3 gần thực hiện đúng mong muốn rồi từ đó chúng minh được thuật toán P2 đúng

Proof: a [fact](https://dictionary.cambridge.org/vi/dictionary/english/fact) or [piece](https://dictionary.cambridge.org/vi/dictionary/english/piece) of [information](https://dictionary.cambridge.org/vi/dictionary/english/information) that [shows](https://dictionary.cambridge.org/vi/dictionary/english/show) that something [exists](https://dictionary.cambridge.org/vi/dictionary/english/exist) or is [true](https://dictionary.cambridge.org/vi/dictionary/english/true) – bằng chứng

germs of a proof: Mầm mống của bằng chứng

Thuật toán P2





invariant: not [changing](https://dictionary.cambridge.org/vi/dictionary/english/changing) – bất biến

relationship: the way in which two things are [connected](https://dictionary.cambridge.org/vi/dictionary/english/connected) – mối quan hệ

invariant relationship: mối quan hệ bất biến

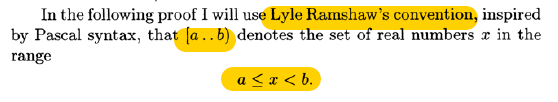
tác giả giải thích mối quan hệ của biến s và t trong chương trình

set: to put something in a [particular](https://dictionary.cambridge.org/vi/dictionary/english/particular) [place](https://dictionary.cambridge.org/vi/dictionary/english/place) or [position](https://dictionary.cambridge.org/vi/dictionary/english/position) – tập hợp



tập hợp tất cả các chuỗi chữ số thập phân d­j+1….dksaocho sẽ tạo ra số n khi được xử lý bởi chương trình đầu tiên

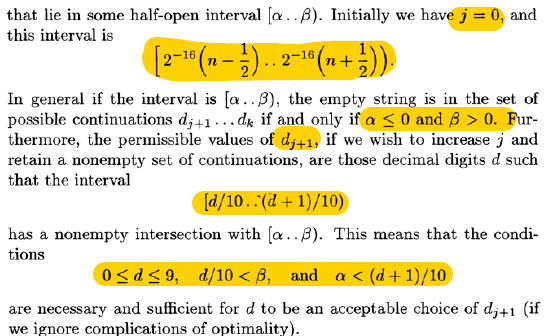
dừng nếu chuỗi trống có trong tập hợp



convention: a [large](https://dictionary.cambridge.org/vi/dictionary/english/large) [formal](https://dictionary.cambridge.org/vi/dictionary/english/formal) [meeting](https://dictionary.cambridge.org/vi/dictionary/english/meeting) of [people](https://dictionary.cambridge.org/vi/dictionary/english/people) who do a [particular](https://dictionary.cambridge.org/vi/dictionary/english/particular) [job](https://dictionary.cambridge.org/vi/dictionary/english/job) or have a [similar](https://dictionary.cambridge.org/vi/dictionary/english/similar) [interest](https://dictionary.cambridge.org/vi/dictionary/english/interest), or a [large](https://dictionary.cambridge.org/vi/dictionary/english/large) [meeting](https://dictionary.cambridge.org/vi/dictionary/english/meeting) for a [political](https://dictionary.cambridge.org/vi/dictionary/english/political) [party](https://dictionary.cambridge.org/vi/dictionary/english/party) – quy ước

quy ước của Lyle Ramshaw

[a ... b) biểuthị tập hợp a



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].

Ban đầu j = 0

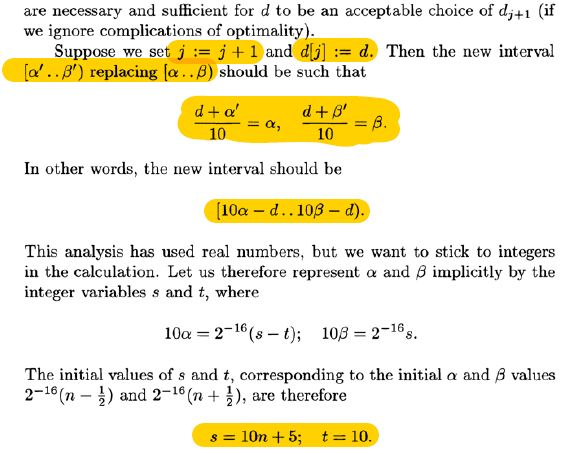


chuỗi rỗng nằm trong tập chỉ khi a <= 0 và B> 0

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

 có một giao điểm khác với [a..B).Có nghĩa là các điều kiện





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

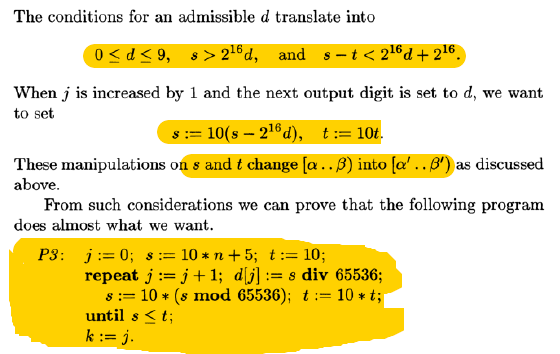


Nói cách khác, khoảng thời gian mới phải saocho



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.



Các điềukiệnđể được chấp nhận d chuyển thành

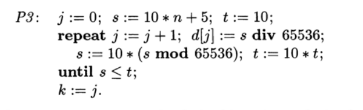


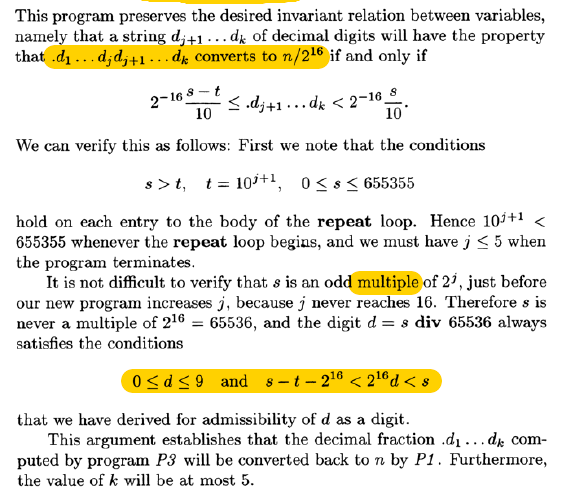
Khi j tăng lên 1 và chữ số đầu ra tiếp theo được đặt thành d



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

chứng minh rằng chương trình sau gần như thực hiệnnhững gì mong muốn





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



xác minh điều này lưu ý rằng các điềukiện



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.

odd multiple: bội số lẻ

Không khó để xác minhrằng s trong bội số lẻ của 2 ^ j, ngay trước khi chương trình mới của chúng ta tăng j, bởi vì j không bao giờ đạt đến 16. Trước đó s không bao giờ là bội số của 2 ^ 16 = 65536 và chữ số d = s div 65535 luôn đáp ứng các điềukiện



* mà chúng ta đã suy ra để có thể chấp nhận d dưới dạng một chữ số.

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 thỏamãn



thì s div 65536 là lớn nhất.Do đó, bấtkỳ phân số nào .d'1... d'k, với, k' <k mà chương trình P1 chuyển đổi thành n có d'j<djđối với j nào đó.Điều này ngụ ý rằng



Nhưng chúng ta có



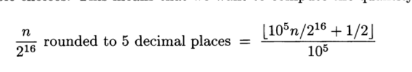
Kể từ đây



Và k’ >= 5 >= k, mâu thuẫn

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. Nếu đầu ra của P3 không phải là giá trị gần đúng nhất thì các giá trị hoàn toàn tốt hơn ít nhất 10-k nhỏ hơn .d1...dk và đối số vừa đưa ra ngụ ý rằng chỉ có thể có hai xấp xỉ khác nhau về độ dài k khi k> = 5. Do đó P3 đưa ra câu trả lời chính xác bất cứ khi nào nó tìm thấy một xấp xỉ có độ dài 4 hoặc nhỏ hơn.

Trong tất cả các trường hợp khác, chương trình P3 cuối cùng bắt đầu vòng lặp của nó với j = 4 và t = 105 . Chúng tôi 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. Điều này có nghĩa là chúng tôi muốn tính toán số lượng.



bất cứ khi nào không có giá trị gần đúng phù hợp với ít hơn 5 chữ số thập phân

### 3.2 Phân tích thuật toán P3

Từ các ý tưởng trên, ta có thể khái quát thuật toán chuyển đổi ngược lại, từ số nguyên n thành phân số thập phân

- Input: số nguyên n

- Output: số thập phân dưới dạng .d1d2…dk sao cho n\*2-16 = .d1d2…dk

* **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 10

Bước 5: j = j+1

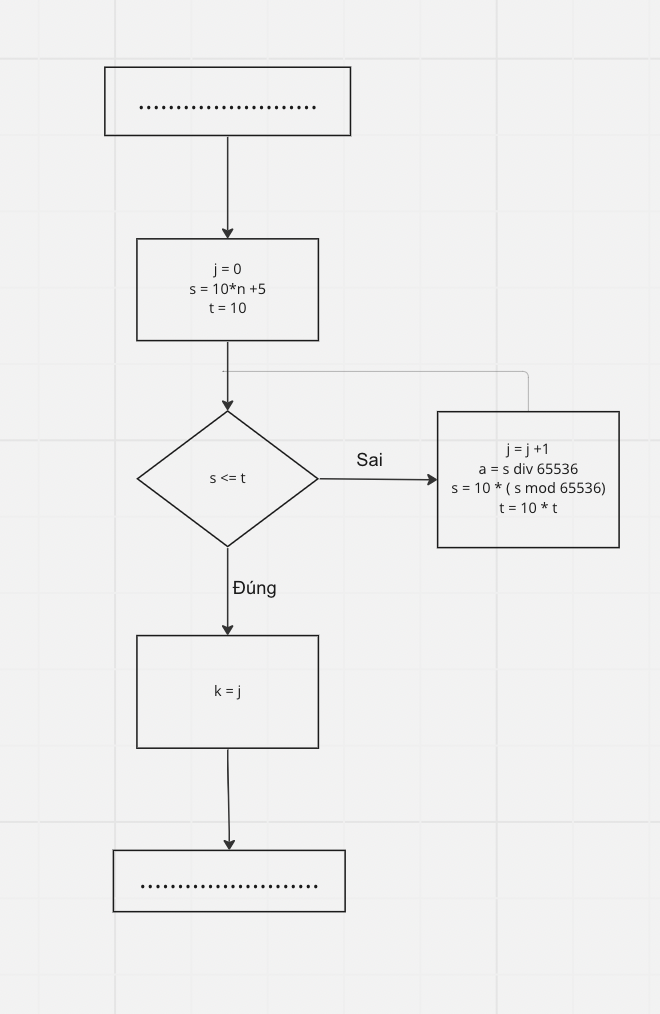
Bước 6: Chữ số phần thập phân thứ j = s div 65536

Bước 7: s = 10 \* (s mod 65536)

Bước 8: t =10 \* t

Bước 9: Quay lại bước 4

Bước 10: k = j

*  **Sơ đồ giải thuật:**

- Tính hữu hạn (Finiteness): thuật toán được xác định hữu hạn vì kết thúc sau một số bước và trả về được kết quả là số thập phân thỏa điều kiện.

* **Cài đặt chương trình P3 bằng ngôn ngữ C++**

#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){

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;

}

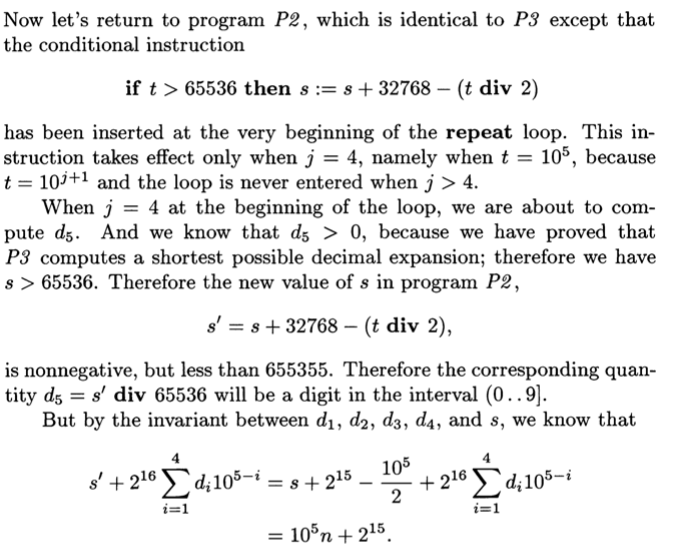
Kết quả sau khi chạy chương trình:



## 4. Hoàn thành bằng chứng

Tóm tắt: Chứng minh P2 đúng

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 chứng ra sẽ quay lại chương trình P2, gần giống với chương trình P3 ngoài trự lện có điều kiện

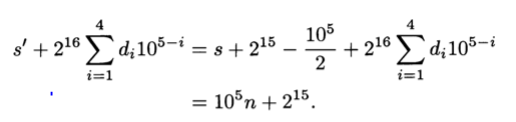
If t > 65536 then s:= s + 32768 – (t **div** 2)

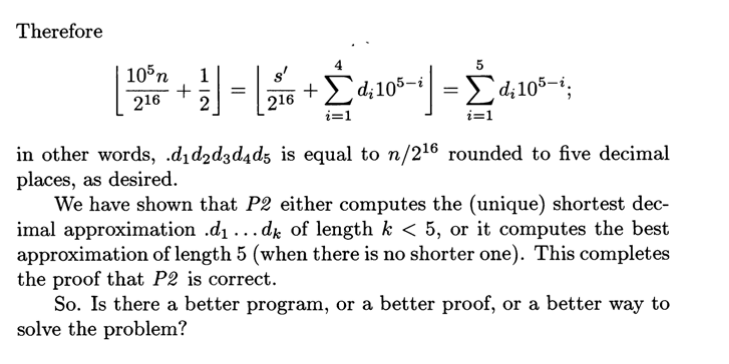
được chèn vào đầu dòng của vòng lặp. Điều này chỉ có hiệu lực khi j=4, cụ thể là khi t = 105, vì t = 10j+1 và vòng lặp không bao giơ được nhập khi j>4

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

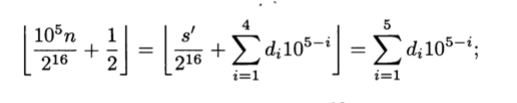
s’ = s + 32768 – (t div 2)

s’ là không âm nhưng s’ < 655355 nên là đại lượng d5 = s’ div 65536 sẽ là một chữ số từ 0 đến 9. Nhưng bằng sự bất biến giữa d1, d2, d3, d4 và s, chúng ta biết rằng





Vì vậy



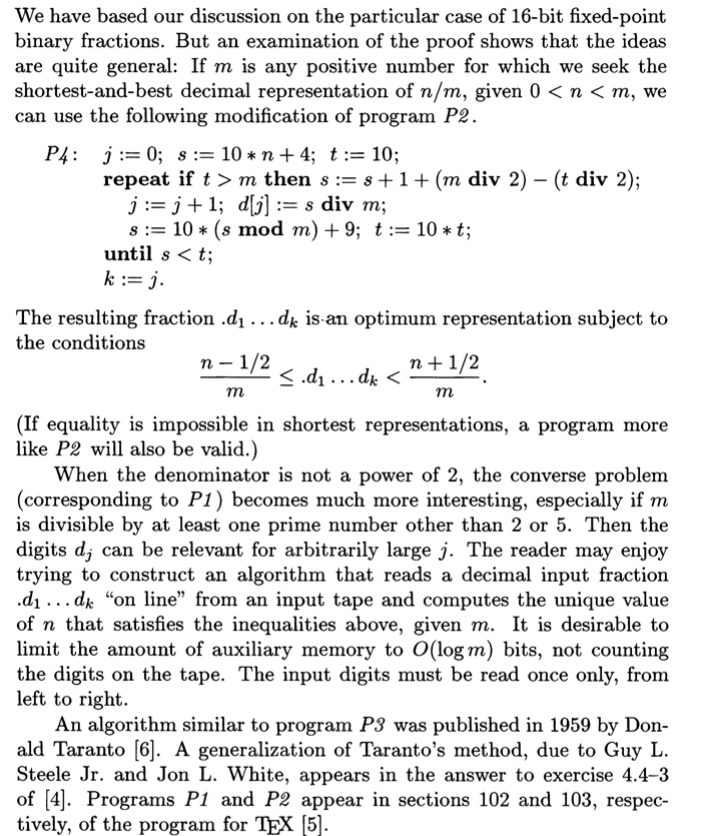
Nói cách khác, thì .d1d2d3d4d5 bằng n/216 được làm tròn thành 5 chữ số thập phân như mong muốn

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.

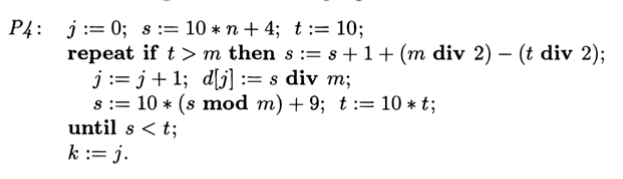
## 5. Kết luận

### 5.1 Nội dung chi tiết

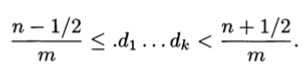
Tóm tắt : Cho thuật toán P4 tương tự như P2, P3 có sửa đổi một chút



Khi kiểm tra bằng chứng thấy các ý tưởng vẫn khá chung chung. Nếu m là bất kỳ số dương nào mà chúng r tìm biểu diễn thập phân ngắn nhất và tốt nhất của n/m, với 0 < n <m, chúng ta có thể sử dụng chương trình này



Kết quả .d1...dk là một biểu diễn tối ưu tùy thuộc vào điều kiện :



(Nếu không thể có đẳng thức trong các biểu diễn ngắn nhất thì một chương trình giống P2 hơn cũng sẽ hợp lệ)

Khi mẫu số không phải là lũy thừa của 2, bài toán ngược lại ( tương ứng với P1) trở nên thú vị hơn nhiều. Nếu m chia hết cho một số nguyên tố khác 2 hoặc 5 thì các chữ số dj có thể liên quan tùy ý hơn j.

Sau khi viết bài báo này, tác giả quyết định tối ưu hóa P2 bằng cách thay đổi ‘ + 32768 – (t div 2) ‘ đến ‘17232’

### 5.2 Phân tích thuật toán P4

- Input: số nguyên n

- Output: số thập phân dưới dạng .d1d2…dk sao cho n\*2-16 = .d1d2…dk

* **Thuật toán:**

Bước 1: j = 0

Bước 2: s = 10\*n+4

Bước 3: t =10

Bước 4: Nếu s < t thì chuyển tới bước 11

Bước 5: Nếu t > m thì s = s + 1 + (m div 2) - (t div 2)

Bước 6: j = j+1

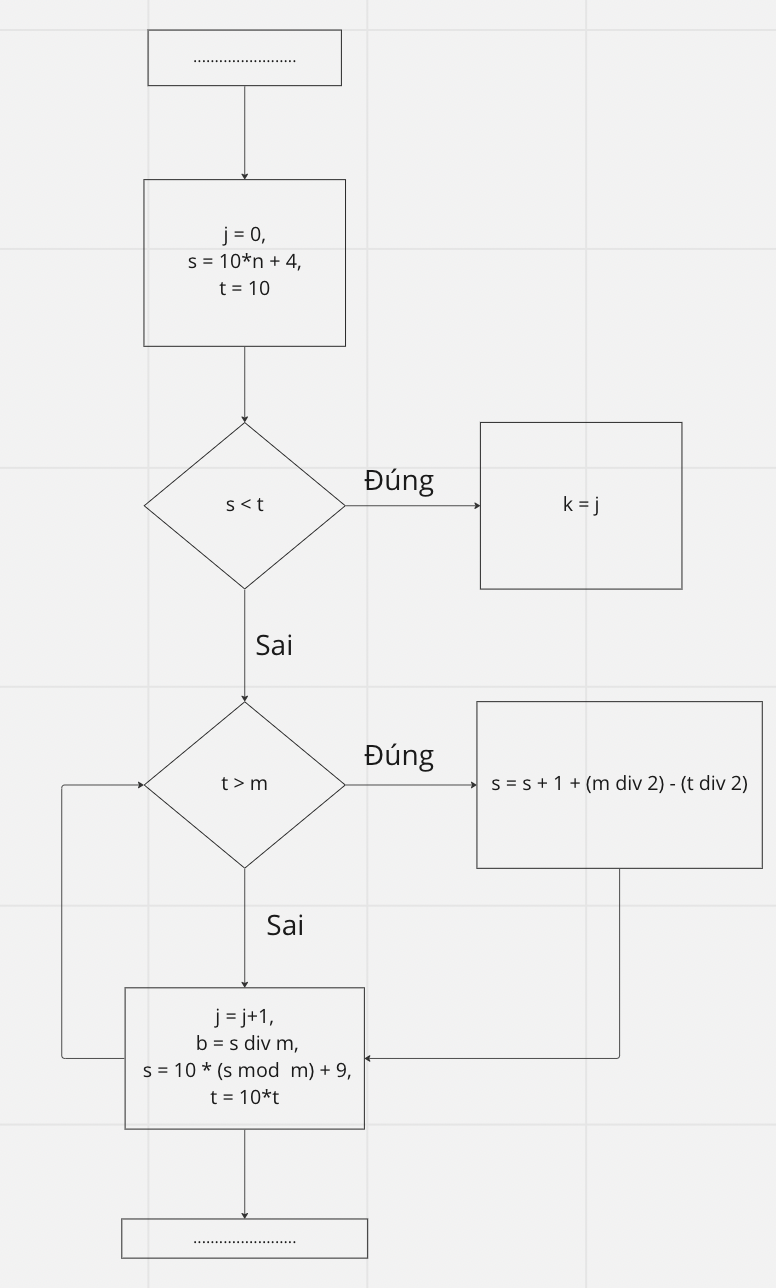
Bước 7: Chữ số phần thập phân thứ j = s div m

Bước 8: s = 10 \* (s mod m) + 9

Bước 9: t =10 \* t

Bước 10: Quay lại bước 4

Bước 11: k = j

*  **Sơ đồ giải thuật:**

- Tính hữu hạn (Finiteness): thuật toán được xác định hữu hạn vì kết thúc sau một số bước và trả về được kết quả là số thập phân thỏa điều kiện.

* **Cài đặt chương trình P3 bằng ngôn ngữ C++**

#include <iostream>

**using** **namespace** std;

**int** main ()

{

**int** d[10];

**int** m,n;

cin >> n;

cin >> m;

**int** j = 0;

**int** s = 10\*n + 4;

**int** t = 10;

**while** (s >= t){

**if**(t>m)

{

s = s + 1 + (m/2) - (t/2);

j = j + 1;

d[j] = s/m;

s = 10\*(s%m) + 9;

t = 10\*t;

}

}

**int** k = j;

**for** (**int** i = 1; i <= k; i++)

cout << d[i];

**return** 0;

}

Kết quả sau khi chạy chương trình:



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

## 1. Ứng dụng của thuật toán

### 1.1 So sánh 2 phân số thập phân

* **Cài đặt chương trình bằng ngôn ngữ C++**

#include <iostream>

**using** **namespace** std;

**int** chusangso (**char** x)

{

**return** (**int**)x - ('0' - 0);

}

**int** multi (string x)

{

**int** m = 0;

**int** l = min((**int**)x.length(), 17);

**for** (**int** i = l; i >= 1; i--){

m = (131072 \* chusangso(x[i-1]) + m) / 10;

}

**int** n = (m + 1) / 2;

**return** n;

}

**int** main ()

{

string a, b;

cout << "Nhap 2 so de so sanh: ";

cin >> a >> b;

**if** (multi(a) > multi(b)) cout << a << " > " << b;

**else** **if** (multi(a) < multi(b)) cout << a << " < " << b;

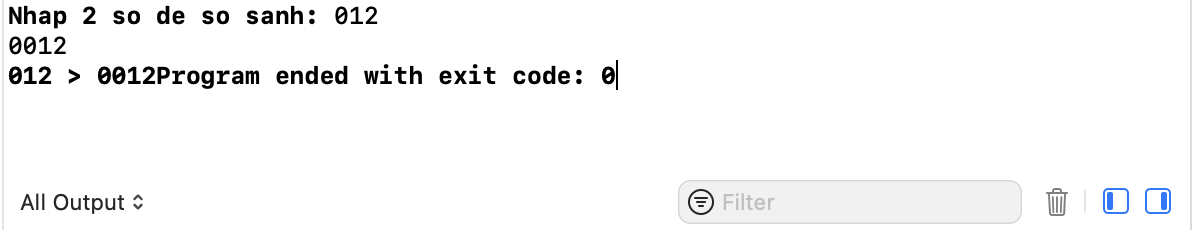
**else** cout << a << '=' << b;

**return** 0;

}

Kết quả sau khi chạy chương trình:

Vì cả 0.00001 và 0.00002 đều trả về n = 1



### 1.2 Cộng 2 phân số thập phân

* **Cài đặt chương trình bằng ngôn ngữ C++**

#include <iostream>

**using** **namespace** std;

**int** ans[10];

**int** chusangso (**char** x)

{

**return** (**int**)x - ('0' - 0);

}

**int** multi (string x)

{

**int** m = 0;

**int** l = min((**int**)x.length(), 17);

**for** (**int** i = l; i >= 1; i--){

m = (131072 \* chusangso(x[i-1]) + m) / 10;

}

**int** n = (m + 1) / 2;

**return** n;

}

**void** decimal (**int** x, **int**& k)

{

**int** j = 0;

**int** s = 10\*x + 5;

**int** t = 10;

**while** (s > t){

**if** (t > 65536) s = s + 32768 - (t/2);

j++;

ans[j] = s / 65536;

s = 10 \* (s % 65536);

t = 10 \* t;

}

k = j;

}

**int** main ()

{

//Dieu kien tong cua 2 so < 1 - 2^-17

string a, b;

cout << "Nhap 2 so can cong: ";

cin >> a >> b;

**int** c = multi(a) + multi(b);

**int** k = 0;

decimal(c,k);

**for** (**int** i = 1; i <= k; i++) cout << ans[i];

**return** 0;

}

Kết quả sau khi chạy chương trình:



### 1.3 Trừ 2 phân số thập phân

* **Cài đặt chương trình bằng ngôn ngữ C++**

#include <iostream>

**using** **namespace** std;

**int** ans[10];

**int** chusangso (**char** x)

{

**return** (**int**)x - ('0' - 0);

}

**int** multi (string x)

{

**int** m = 0;

**int** l = min((**int**)x.length(), 17);

**for** (**int** i = l; i >= 1; i--){

m = (131072 \* chusangso(x[i-1]) + m) / 10;

}

**int** n = (m + 1) / 2;

**return** n;

}

**void** decimal (**int** x, **int**& k)

{

**int** j = 0;

**int** s = 10\*x + 5;

**int** t = 10;

**while** (s > t){

**if** (t > 65536) s = s + 32768 - (t/2);

j++;

ans[j] = s / 65536;

s = 10 \* (s % 65536);

t = 10 \* t;

}

k = j;

}

**int** main ()

{

//Dieu kien hieu 2 so > 0 va < 1 - 2^-17

string a, b;

cout << "Nhap 2 so can tru: ";

cin >> a >> b;

**int** c = multi(a) - multi(b);

**int** k = 0;

decimal(c,k);

**for** (**int** i = 1; i <= k; i++) cout << ans[i];

**return** 0;

}

Kết quả chạy chương trình



## 2. Hướng phát triển

Cho m là số dương bất kì để ta có thể tìm biểu diễn thập phân ngắn nhất .d1 xấp xỉ n/m với d1 là kết quả của chương trình P4.

Có thể giúp người dùng thích thú hơn khi cố gắng xây dưng một thuật toán đọc phân số đầu vào thập phân và tính giá trị duy nhất của n thỏa mãn các bất đẳng thức trên khi cho trước m.

1. TeX Definition & Facts <https://www.britannica.com/topic/Web-script> , 24/11/2022 [↑](#footnote-ref-1)
2. What value TeX uses as its minimal unit? <https://tex.stackexchange.com/questions/231269/what-value-tex-uses-as-its-minimal-unit> , 25/11/2022 [↑](#footnote-ref-2)