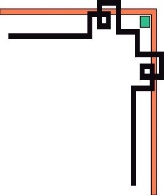
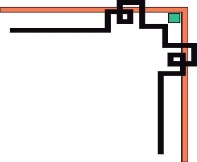
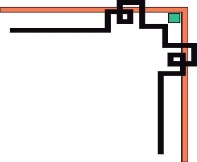
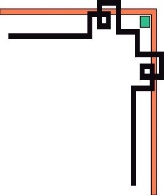
**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**



**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 lớp: DASA230179\_09**

**GVHD: Huỳnh Xuân Phụng**

**Thực hiện: Nhóm 10**

**Sinh viên thực hiện**

|  |  |  |
| --- | --- | --- |
| STT | Họ và tên | MSSV |
| 1 | Trần Tuấn Anh | 21110369 |
| 2 | Hồ Khánh Đăng | 21110876 |
| 3 | Lê Hữu Huy | 21110471 |

***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

[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)

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

[1. Chuyển đổi phân số thập phân thành số nhị phân dấu chấm động 2](#_Toc123134594)

[1.1 Nội dung chi tiết 2](#_Toc123134595)

[1.2 Phân tích thuật toán P1 4](#_Toc123134596)

[2. Chuyển đổi số nhị phân dấu chấm động thành phân số thập phân 5](#_Toc123134597)

[2.1 Nội dung chi tiết 5](#_Toc123134598)

[2.2 Phân tích thuật toán P2 6](#_Toc123134599)

[3.Chứng minh thuật toán P2 7](#_Toc123134604)

[3.1 Phân tích 7](#_Toc123134606)

[4. Mở rộng thuật toán 10](#_Toc123134600)

[4.1 Nội dung chi tiết 10](#_Toc123134601)

[4.2 Phân tích thuật toán P3 11](#_Toc123134602)

[PHẦN 3: PHÂN TÍCH ỨNG DỤNG VÀ HƯỚNG PHÁT TRIỂN CỦA THUẬT TOÁN 13](#_Toc123134607)

[1. Ứng dụng của thuật toán 13](#_Toc123134608)

[2. Hướng phát triển 14](#_Toc123134612)

# PHẦN 1: TÌM HIỂU VẤN ĐỀ

## 1. Nội dung chính của bài báo

- Cách chuyển qua lại giữa phân số thập phân và số chấm động nhị phân.

- Cách chứng minh tính đúng đắng của 1 chương trình đơn giản khi chuyển qua lại giữa phân số thập phân và số chấm nhị phân

## 2. Cấu trúc của bài báo

- Cấu trúc của bài báo gồm 5 phần.

Chương 1: Trình bày cách chuyển đổi từ phân số thập phân sang số nhị phân dấu chấm động

Chương 2: Trình bày cách chuyển đổi từ số nhị phân dấu chấm động sang phân số thập phân

Chương 3: Chứng minh tính đúng đắng của chương 2

Chương 4: Chứng minh tính đúng đắng cho chương 2

Chương 5: Mở rộng thuật toán

## 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**, viết không định dạng là **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.

-Nó phổ biến trong môi trường hàn lâm, đặc biệt là trong cộng đồng toán học, vật lý, khoa học máy tính, kinh tế, thống kê, kỹ thuật. Trong hầu hết các bản cài đặt Unix, nó gần như thế chỗ của troff, cũng là một chương trình định dạng văn bản được ưa thích khác.

-TeX là phần mềm miễn phí. Nói chung nó được xem là cách tốt nhất để gõ công thức toán học phức tạp, đặc biệt là ở dạng LaTeX, ConTeXt hoặc các gói khuôn mẫu (template package) khác, nó cũng được dùng cho các tác vụ sắp chữ khác.

# PHẦN 2: PHÂN TÍCH CHI TIẾT

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

-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.

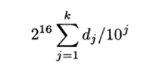
-Mục đích của tác giả trong phần ghi chú này là để trình bày rằng chương trình con với một phác thảo về bằng chứng tốt nhất mà ông biết, hi vọng rằng những độc giả với nhiều kinh nghiệm hơn trong phương pháp chính thức sẽ đồng ý rằng thuật toán này thú vị và sẽ giúp ông tìm ra tôi nên làm điều gì.

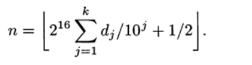
### 1.1 Nội dung chi tiết

-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.

 -Mục tiêu*:* 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

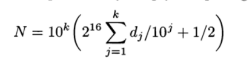
*Ví dụ:* Với phân số thập phân đầu vào là: 0.214 thì bội số nguyên nhỏ nhất của 2-16 cần tìm đó là :

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

được làm tròn lên thành số nguyên

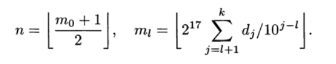
n = 14024.704 + 1/2 = 14025.204 14025

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:

**1.2 Phân tích thuật toán**

**Các bước trong thuật toán**

Từ công thức (1.2)

Bước 1: Gắn biến l theo k và tối là của l có thể nhận là 17 số nên l=min(k,17) ; gắn m=0;

Bước 2: Thực hiện vòng lặp tính m theo công thức ml

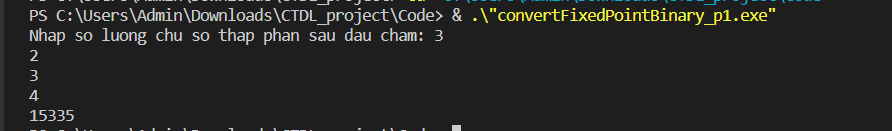
Bước 3: Tính n sau khi tính được m theo công thức (m+1)/2.

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

* #include <bits/stdc++.h>
* using namespace std;
* long long fixedPointBinary(int D[],int k);
* void nhapMang(int D[],int &k);
* int main()
* {
* int D[18],k;
* nhapMang(D,k);
* long long kq=fixedPointBinary(D,k);
* cout<<kq;
* }
* long long fixedPointBinary(int D[],int k)
* {
* int l=min(k,17);
* double m=0;
* do{
* m = (131072\*D[l-1]+m)/10;
* l--;
* }while(l!=0);
* long long n=(m+1)/2;
* return n;
* }
* void nhapMang(int D[],int &k)
* {
* cout<<"Nhap so luong chu so thap phan sau dau cham: ";
* cin>>k;
* for(int i=0;i<k;i++)
* cin>>D[i];
* }

Kết quả sau khi chạy chương trình:

Với n = 125



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

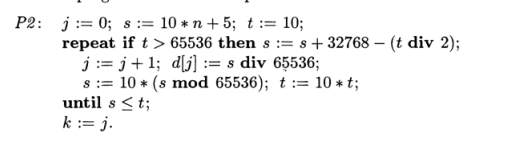
**2.1. Nội dung chi tiết**

-Tác giả đưa ra thuật toán giải quyết bài toán sau :

Cho số n thuộc (0,2^16), tìm các chữ số phần thập phân .d1d2…dk của n/2^16

-Thuật toán sẽ tìm 5 chữ số của số thập phân n/2^16 (tức là k=5). Vì thuật toán sẽ tìm ra .d1d2…dk là số ít chữ số thập phân nhất biểu diễn gần đúng nhất với n/2^16. Vì sao chọn k = 5?, câu hỏi sẽ được giải quyết ở phần sau.

-Tác giả trình bày thuật toán như sau:



**2.2 Phân tích thuật toán**

* **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 P3 bằng ngôn ngữ C++**

#include <bits/stdc++.h>

#define int long long

using namespace std;

int gcd(int a,int b)

{

    if (b==0) return a;

    return gcd(b,a%b);

}

void decimalFraction()

{

  int n; cin>>n;

  int j=0;

  int s = 10\*n;

  int t = 10;

  int d[100];

  while (s>=t)

  {

     if (t>65536) s = s + (65536/2);

     j++; d[j] = s/65536;

     s = 10\*(s%65536); t\*=10;

  }

  int k = j;

  cout<<k<<endl;

  for (int i=1;i<=k;i++) cout<<d[i]<<" ";

}

int32\_t main()

{

   int t=1; //cin>>t;

   while (t--)

   {

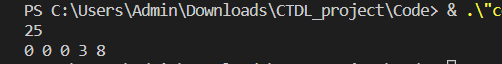
      decimalFraction();

   }

}

Kết quả sau khi chạy chương trình:

Với n = 25 thì ta có kết quả như sau:



**3. Chứng minh thuật toán p2**

***\*Germs of a Proof***

-Tại sao thuật toán chỉ tìm nhiều nhất 5 chữ số của số n/2^16 ?

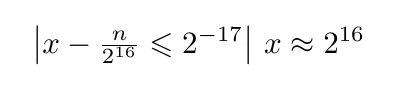
+Tác giả chứng minh được rằng số thập phân mà tác giả tìm được là tối ưu nhất

+Gọi số thập phân của tác giả là .d1d2…dk (k<=5)

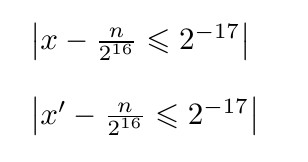
Chứng minh phản chứng:

-Giả sử có số .d1’d2’…dk’ với k’ <k và .d1’d2’…dk’ < .d1d2…dk

-Đặt x =.d1d2…dk; x’ =.d1’d2’…dk’

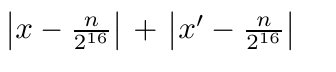
-Số thỏa mãn gần đúng nhất với n/2^16 là số: 

-Nên ta có :



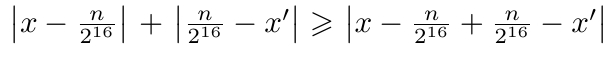
-Ta có : x – x’ >= 10^-k’ (vì k’ <k)

-Xét :

Ta có BĐT đã học :



=> 

Suy ra: 10^-k’ <= 2^-17 + 2^-17 = 2^-16

⬄k’ >= 5 >= k

Mâu thuẫn với giả sử ban đầu là : k’ <k

Suy ra giả sử sai

Vậy số thập phân của tác giả là số thập phân tối ưu nhất biểu diễn gần đúng n/2^16. Hay nói cách khác là không có số thập phân nào ngắn hơn .d1…dk thỏa tính chất

|.d1d2…dn – n/2^16| <=2^-17

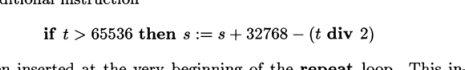
***\*Completion of the proof***

-Thuật toán của tác giả lấy nhiều nhất 5 chữ số thập phân nên điều là s<=t

Khi t = 10^6 tức là thuật toán chạy được 5 vòng lặp, ta thấy: t lúc này sẽ lớn hơn s. Bởi vì :mã(s) = 10(2^16-1) = 655350 < 10^6 = t

Nên khi xong vòng lặp thứ 5 thì thuật toán chắc chắn dừng lại

-Ta giả thích ý nghĩa câu lệnh



Đặt câu lệnh này là (\*)

Ta xuất phát từ dòng lệnh đầu tiên

+Tác giả đặt s= 10n+5

+Ta chú ý đến số 5, ta thấy sau mỗi vòng lặp thì 5 sẽ nhân 10

Khi ở vòng lặp thứ 6 thì 5 sẽ tăng thành 5\*10^5 > 65536

-Như vậy từ vòng lặp thứ 5 trở về trước thì yếu tố 5 không làm ảnh hưởng tới kết quả

-Vì để ngăn chặn yếu tố 5 làm sai lệch kết quả thì tác giả mới cho -t/2 =50000 ở vòng lặp thứ 5

-Điều kiện : t>65536, 10^I 65536 -> t>=5

Vậy tại vòng lặp thứ 5 tác giả đã triệt tiêu 50000 để tránh nó ảnh hưởng tới kết quả ở vòng lặp thứ 6

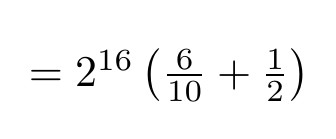
-Tại sao lại cộng 5 với 32768?

Mục đích để làm tròn lên kết quả vì 32768 = 2^15 = 2^16/2

-Tại vòng lặp thứ 5 (tìm chữ số cuối cùng) biết s >=2^16 thì chữ số cuối sẽ tăng 1 đơn vị và ngược lại giữ nguyên

VD: tại vòng lặp 5

Ta có

S 

Thì ta thao tác S = 6/10 \* 10^16 + 2^16/2

= 2^16(6/10 +1/2)

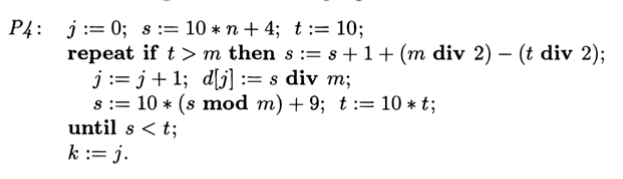
6/10+1/2 > 1 suy ra chữ số cuối sẽ tăng 1 đơn vị

## 4. Mở rộng thuật toán

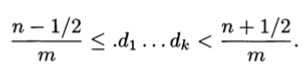
### 4.1 Nội dung chi tiết

Thuật toán P3 tương tự như P2 nhưng có vài sự thay đổi

Chi tiết công thức như sau:



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



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’

### 4.2 Phân tích thuật toán P3

- Input: số nguyên n,m

- 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

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

#include <bits/stdc++.h>

using namespace std;

void Extension(){

    long long n,m;

    cin >> n >>m;

    int j=0;

    long long s= 10\*n+4;

    long long t =10;

    long long d[100];

    while (s>= t){

        if (t>m){

            s = s+1 +(m/2)-(t/2);

        }

        j++;

        d[j] = s/m;

        s = 10\*(s%m)+9;

        t\*=10;

    }

    long long k=j;

    for (int i=1;i<=k;i++){

        cout <<d[i] <<" ";

    }

}

int main(){

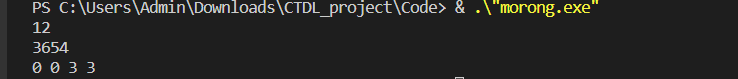
    Extension();

    return 0;

}

Kết quả sau khi chạy chương trình:

Với n = 12, m= 3654



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

## Ứng dụng của thuật toán

*Ứng dụng cho ngành toán học*

-Chuyển số thập phân có dấu sang số nhị phân đơn giản nhất

*Chương trình tex*

-Là tiền thân của chương trình LaTex sau này

*Các ngôn ngữ như c++,c#...hay tin học văn phòng*

-Sử dụng cho các hàm về làm tròn

*Lập trình máy tính casio*

-Ứng dụng rộng rãi cho lập trình máy tính casio

*Các app khác*

-Ứng dụng rộng rãi trong các app để hiển thị ra kết quả vừa ngắn và xấp xỉ gần đúng nhất

## 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 sử dụng hứng thú hơn khi cố gắng xây dựng nên một thuật toán đọc phân số đầu vào thập phân và xác định giá trị tuyệt đối của n đáp ứng các bất đẳng thức trên khi tính trước m.