Penentuan Bilangan Terkecil yang Memenuhi Sistem Kongruensi Chinese Remainder Theorem dengan Memanfaatkan Brute Force

Damiann Muhammad Mangan/13510071[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*damiann.mm@students.itb.ac.id*

***Abstrak*—Penulis memperkenalkan algoritma yang mampu mengembalikan bilangan terkecil yang memenuhi kongruensi-kongruensi yang diberikan. Dengan mereduksi jumlah kongruensi, memanfaatkan lemma kongruensi yang akan diuraikan, brute force dapat menyelesaikan dengan O(*n*). Penulis akan menyajikan contoh implementasi algoritma ini dalam bentuk yang sederhana, tanpa *heuristic*.**

***Index Terms*—FPB, KPK, relasi kongruensi, Chinese Remainder Theorem.**

I. PENDAHULUAN

Chinese Remainder Theorem, secara umum merupakan hasil dari beberapa kongruensi di teori bilangan dan generalisasinya di aljabar abstrak. CRT, yang dikenal juga dalam bahasa Indonesia sebagai Teorema Sisa Tiongkok, dipicu kemunculannya oleh Sun Zi yang hidup di abad ke-3 hingga ke-5 sebelum akhirnya dirumuskan secara utuh oleh Qin Jiushao pada tahun 1247 dengan nama Da yan shu pada buku berjudul Shushu Jiuzhang.

Sebelum CRT dibahas lebih dalam, untuk kejelasan pembacaan, akan dibahas hal-hal yang perlu diketahui, seperti FPB (Faktor Persekutuan Terbesar, GCD), KPK (Kelipatan Persekutuan Terkecil), dan relasi kongruensi, yang menggunakan konsep modulo.

II. TEORI BILANGAN

FPB dikenal dunia sebagai GCD (Greatest Common Divisor). FPB dari suatu himpunan bilangan adalah bilangan terbesar yang mampu membagi setiap seluruh anggota himpunan bilangan tersebut dan menghasilkan bilangan bulat. Cara termudah untuk menentukan FPB dari himpunan bilangan adalah dengan menuliskan setiap faktor prima beserta pangkatnya dari setiap himpunan tersebut. Kasus khusus saat FPB dari dua bilangan adalah 1, disebut saling prima.

Contoh FPB dari 500 dan 300 adalah 100, berdasarkan perhitungan dibawah ini.

$500 = 2^2 \times 5^3$
$300 = 2^2 \times 3 \times 5^2$

Terlihat bahwa FPB-nya merupakan $2^2 \times 5^2$yang senilai dengan 100. Selain dengan memfaktorkan, cara lain adalah dengan algoritma Euclid, yang secara formal dapat ditulis sebagai berikut.

$\gcd(a, a) = a$,
$\gcd(a, b) = \gcd(b, a - b)$, jika $a > b$, dan
$\gcd(a, b) = \gcd(b, a)$, jika $b > a$

KPK atau LCM (Least Common Multiple) dari suatu himpunan bilangan adalah bilangan terkecil yang merupakan kelipatan dari setiap anggota himpunan itu. Contohnya, KPK dari 4 dan 6 dengan perhitungan manual dapat ditulis seperti dibawah ini.

$4, 8, 16, 20, 24, 28, 32, 26, 40, 44, 48, 52, \ldots$
$6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72, \ldots$

Kelipatan persekutuannya adalah bilangan-bilangan yang muncul pada kedua baris geometri tersebut, yakni $12, 24, 36, 48, \ldots$. Jadi, KPK dari 4 dan 6 adalah 12. Cara lain, dengan sebelumnya mencari nilai FPB, adalah dengan memanfaatkan persamaan berikut.

$$\text{lcm}(a, b) = \frac{|a \cdot b|}{\gcd(a, b)}$$

II. TEORI BILANGAN

Dua bilangan *a* dan *b* akan kongruen modulo *c* apabila selisih *a* dan *b* habis dibagi *m*. Sifat-sifat pada relasi kongruensi ini diantaranya *equivalence*, *determination*, *reflexivity*, *symmetry*, dan *transitivity*, yang masing-masing secara berurutan adalah sebagai berikut:

$a \equiv b \pmod 0$, maka $a = b$,
$a \equiv b \pmod m$ atau $a \not\equiv b \pmod m$ pasti benar,
$a \equiv a \pmod m$,
$a \equiv b \pmod m$, maka $b \equiv a \pmod m$, dan
apabila $a \equiv b \pmod m$ dan $b \equiv c \pmod m$, maka $a \equiv c \pmod m$.

Untuk menggabungkan dua atau lebih kongruensi, penulis menggunakan *lemma* bahwa apabila b_1 dan b_2 adalah bilangan bulat positif yang saling prima dan diberikan bilangan bulat r_1 dan r_2, maka dengan V yang memenuhi sistem kongruensi ini,

$$V \equiv r_1 \pmod{b_1}$$
$$V \equiv r_2 \pmod{b_2}$$

Akan menghasilkan kongruensi berikut, yang didapatkan dengan menjumlahkan kedua kongruensi setelah menyamakan basis dari masing-masing kongruensi, dan masih memenuhi masing-masing kongruensi awal.

$$V(b_1 + b_2) \equiv b_1 r_2 + b_2 r_1 \ (mod\ b_1 b_2)$$

Perhitungan baik oleh manusia dan komputer akan lebih mudah dengan kokngruensi tunggal, karena variabel yang perlu dihitung atau dipertahankan hanya dua, yaitu koefisien dari nilai yang dicari, *V*, dan hasil kongruensinya.

III. Chinese Remainder Theorem

CRT memperlihatkan bahwa pasti akan ada nilai *x* yang akan memenuhi sistem kongruensi berikut.

$$x \equiv a_1 \pmod{n_1}$$
$$x \equiv a_2 \pmod{n_2}$$
$$\vdots$$
$$x \equiv a_k \pmod{n_k}$$

Apabila n_1, n_2, …, n_k adalah bilangan bulat positif yang setiap kombinasi keduanya saling prima dan a_1,a_2, …, a_k, adalah bilangan bulat bebas.

Untuk mencari nilai *x*, dapat dilakukan metode substitusi berulang-ulang, memanfaatkan bentuk modulo dalam persamaan. Yaitu:

$$a \equiv b \pmod{m} \leftrightarrow a - b = mk$$

Dengan *k* adalah bilangan bulat positif. Contohnya, nilai *x* terkecil yang apabila dibagi 2 akan bersisa 1 dan akan bersisa 2 apabila dibagi 3, dalam bentuk sistem kongruensi akan seperti dibawah ini.

$$x \equiv 1 \pmod{2}$$
$$x \equiv 2 \pmod{3}$$

Ubah kongruensi pertama menjadi persamaan, dan substitusi, akan menghasilkan,

$$x - 1 = 2k,$$
$$x \equiv 2 \pmod{3} \rightarrow 2k + 1 \equiv 2 \pmod{3},$$
$$2k \equiv 1 \pmod{3} \leftrightarrow k \equiv 2 \pmod{3},$$

Dengan memilih *k* terkecil yang memenuhi, yaitu 2, akan didapatkan nilai *x* akan sebesar $2 \times 2 + 1$ atau 5.

Cara lain adalah dengan menggunakan *lemma* metode reduksi kongruensi pada bab sebelumnya. Dengan sistem kongruensi seperti dibawah, contoh penggunaannya adalah seperti berikut.

$$x \equiv 1 \pmod{2}$$
$$x \equiv 2 \pmod{3}$$
$$x \equiv 4 \pmod{5}$$

Dengan sifat relasi kongruensi, sistem ini akan diubah menjadi lebih sederhana, dan selanjutnya akan disamakan basisnya.

$$x \equiv -1 \pmod{2} \rightarrow 15x \equiv -15 \pmod{30}$$
$$x \equiv -1 \pmod{3} \rightarrow 10x \equiv -10 \pmod{30}$$
$$x \equiv -1 \pmod{5} \rightarrow 6x \equiv -6 \pmod{30}$$

Hasil penjumlahan seluruhnya adalah,

$$31x \equiv -31 \pmod{30} \leftrightarrow x \equiv -1 \equiv 29 \pmod{30}$$

Didapat 29, 59, 89, ... memenuhi nilai *x*, sehingga cukup dipilih *x* = 29 untuk mendapatkan nilai terkecil yang memenuhi.

IV. Algoritma

Tanpa *heuristic* yang mampu mengubah hasil sisa pembagian pada relasi kongruensi menjadi negatif agar lebih mudah dihitung, setelah menerima seluruh kongruensi, kode cukup mendapatkan KPK dari seluruh basis seiringan dengan mengubah koefisien pada nilai yang dicari dan memperbarui hasil sisa bagi masing-masing kongruensi sebelum menjumlahkan seluruhnya. Kongruensi terakhir yang merupakan hasil dari jumlah seluruh sistem akan dicari solusinya dengan metode *brute force*. Bagian iterasi terakhir ini juga dijalankan tanpa *heuristic*.

Berikut ini adalah implementasi sederhananya dengan menggunakan bahasa *C++*, dengan masukan pertama adalah jumlah kongruensinya dan masukan selanjutnya merupakan sistem yang diketahui. Program akan menampilkan nilai terkecil yang memenuhi beserta waktu yang ditempuh seluruhnya.

```
//
//  main.cpp
//  bruteCRT
//
//  Created by Damiann Muhammad Mangan on 12/15/12.
//  Copyright (c) 2012 Damiann Muhammad Mangan. All
rights reserved.
//

#include <iostream>
#include <ctime>

// t : waktu dalam mikro detik
// s : jumlah kongruensi
```

```
// c[s] : konstanta dari nilai yang ingin dicari pada kongruensi ke-s
// r[s] : sisa pembagian pada kongruensi ke-s
// b[s] : basis pada kongruensi ke-s
// C : konstanta untuk kongruensi terakhir, berubah sesuai dengan B
// R : sisa pembagian untuk kongruensi terakhir, berubah sesuai dengan B
// B : basis untuk kongruensi terakhir, hasil perkalian seluruh basis
// temp : nilai untuk memperkecil perlunya perhitungan (perkalian)
int main(int argc, const char * argv[])
{

    clock_t t;
    int s;
    scanf("%d", &s);

    // inisialisasi variabel berdasarkan jumlah kongruensi
    unsigned long long c[s];
    unsigned long long r[s];
    unsigned long long b[s];
    unsigned long long B = 1;
    unsigned long long C = 0;
    unsigned long long R = 0;
    unsigned long long temp;

    for (int i = 0; i < s; i++) {
        scanf("%llu %llu", &b[i], &r[i]);
    } // r[s], b[s], dan B sudah siap

    t = clock();

    for (int i = 0; i< s; i++) {
        r[i] %= b[i];
        B *= b[i];
    }

    for (int i = 0; i < s; i++) {
        c[i] = B / b[i];
        C += c[i];
        R += c[i] * r[i];
        R %= B;
    } // c[s] dan R sudah siap
    C %= B; // C sudah siap

    // brute force pencarian nilai yang memenuhi
    temp = 0;
    for (unsigned long long i = 1; i <= B; i++) {
        temp += C;
        temp %= B;
        if (temp == R) {
            t = clock() - t;
            printf("%llu\n", i);
            printf("took %lu clocktime", t);
            return 0; // cukup dikeluarkan nilai terkecil
        }
    }

    return 0;
}
```

Dibawah ini adalah contoh *input* dan *output* dengan menggunakan komputer yang memiliki *processor* 1.7 GHz Intel Core *i*5 dan *memory* 4GB 1333 MHz DDR3.

```
8
2 1
3 2
5 4
7 6
11 10
13 12
17 16
19 18
9699689
took 143043 clocktime
```

```
9
2 1
3 2
5 4
7 6
11 10
13 12
17 16
19 18
23 22
223092869
took 3367896 clocktime
```

Dengan perhitungan manual, didapati bahwa hasil tersebut, 9699689 dan 223092869 memang merupakan nilai yang memenuhi kedua sistem kongruensi tersebut. Masing-masing nilai kedua hasil tersebut juga merupakan nilai terkecil yang memenuhi sistem kongruensi yang bersangkutan, karena lebih kecil dari KPK seluruh basis sistem kongruensinya.

Contoh lain,

```
3
7 4
1998 200
3 2002
9524
took 171 clocktime
```

```
4
3 773
5 8219
7 8
11 10
659
took 15 clocktime
```

```
20
2 1
3 1
5 1
7 1
11 1
13 1
17 1
19 1
23 1
29 1
31 1
37 1
41 1
43 1
47 1
53 1
59 1
61 1
67 1
71 1
1
took 3 clocktime
```

Terlihat waktu yang ditempuh program akan berbanding lurus langsung dengan nilai KPK terakhir. Dapat dilihat pada bagian kode yang disediakan, iterasi yang dilakukan tidak rekursif maupun memiliki rekursif didalanya, sehingga waktu yang diperlukan untuk menentukan hasil akhir adalah O(*n*), dengan n adalah nilai KPK seluruh basis di sistem.

V. KEKURANGAN UMUM

Algoritma yang penulis sediakan masih tanpa *heuristic* baik pada bagian penyederhanaan relasi kongruensi maupun iterasi untuk menetukan nilai akhir. Sedangkan metode substitusi berulang kali akan menggunakan waktu yang serupa, implementasi untuk nilai yang dicari akan lebih sulit, dan kongruensi terakhir akan masih perlu dicari nilai yang memenuhi dengan metode pencarian buta dengan *brute force*.

VI. KESIMPULAN

Algoritma *brute force* mampu dan dapat diandalkan untuk mencari nilai terkecil yang tepat selama sistem kongruensi yang diberikan memenuhi syarat CRT. Tanpa *heuristic*, waktu pengerjaan dibandingkan dengan jumlah relasi kongruensi akan *astronomical*! Karena KPK yang didapat akan sangat besar. Algoritma ini menggunakan O(*n*) dengan *n* sebagai hasil KPK.

VII. PENGHARGAAN

Penulis secara pribadi berterima kasih kepada saudara kandung penulis, Dannis, atas saran dan koreksinya dalam isi makalah. Penulis juga berterima kasih kepada Bapak Rinaldi Munir selaku dosen mata kuliah Strategi Algoritma dan referensi dari beliau.

REFERENSI

[1] Chinese Remainder Theorem – Wikipedia, the free encyclopedia (2012). http://en.wikipedia.org/wiki/Chinese_remainder_theorem. Tanggal akses 15 Desember 2012.

[2] Congruence – from Wolfram MathWorld (2012). http://mathworld.wolfram.com/Congruence.html. Tanggal akses 15 Desember 2012.

[3] Congruence Relation – Wikipedia, the free encyclopedia (2012). http://en.wikipedia.org/wiki/Congruence_relation. Tanggal akses 15 Desember 2012.

[4] Greatest common divisor – Wikipedia, the free encyclopedia (2012). http://en.wikipedia.org/wiki/Greatest_common_divisor. Tanggal akses 15 Desember 2012.

[5] Least common multiple – Wikipedia, the free encyclopedia (2012). http://en.wikipedia.org/wiki/Least_common_multiple. Tanggal akses 15 Desember 2012.

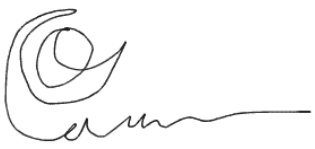
[6] Greatest Common Divisor – from Wolfram MathWorld (2012). http://mathworld.wolfram.com/GreatestCommonDivisor.html. Tanggal akses 15 Desember 2012.

[7] Least Common Multiple – from Wolfram MathWorld (2012). http://mathworld.wolfram.com/LeastCommonMultiple.html. Tanggal akses 15 Desember 2012.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 15 Desember 2012



Damiann M Mangan, 13510071