<< Back to CodeChef

CODECHEF Discuss
A Directi Educational Initiative

questions tags users badges unanswered ask a question about

CodeChef Discussion

Search Here... questions tags us

A tutorial on the Extended Euclid's Algorithm

26

18

Hello @all,

Following my previous tutorial on Repeated Squaring, I will now focus on the **Extended Euclid's Algorithm**, which as you will be able to see, can be seen as the reciprocal of modular exponentiation.

But, before delving deeper into this algorithm, it might be worthwhile to review the most basic algorithm, the **Euclidean Algorithm.**

- **Foreword about the Euclidean Algorithm**

The Euclidean Algorithm is possibly one of the oldest numerical algorithms still in use (its first appearance goes back to 300 BC, making it over 2000 years old), and it is used to find the GCD of two numbers, i.e., the greatest common divisor of both numbers.

It's easily implemented in C++ as:

```
#include <iostream>
#include <algorithm>
using namespace std;
#define builtin_gcd __gcd

int gcd(int a, int b)
{
    if(b==0)
        return a;
    else
        return gcd(b,a%b);
}

int main()
{
    cout << gcd(252,105) << endl;
    cout << builtin_gcd(252,105) << endl;
    return 0;
}
```

Also, please note that if you include the header `<algorithm>` on your code, you can actually use the built-in gcd function, by renaming the language function __gcd (note the two underscore characters to the left of gcd) to something you would like (on the code above, I renamed it to builtin_gcd, just to distinguish it from my own implemented gcd function).

Note that I suggest a renaming of the built-in function solely for you not to use the full name **gcd, but something more convenient, but, you can also use** gcd and everything will work completely fine as well. :)

Returning to our algorithm discussion, it's easy to see that this algorithm finds the greatest number that divides both numbers passed as arguments to the gcd() function.

The gcd() has some interesting properties related to the arguments it receives as well as its number. Two interesting properties are:

- gcd(a,b) = 1, implies that the integers a and b are coprime (this will have implications further on this text);
- It's possible to find the gcd of several numbers by finding the pairwise gcd of every 2 numbers, i.e., say we have three numbers a,b,c, then gcd(a,b,c) = gcd(gcd(a, b), c);

This sums up the basic properties of the gcd, which will allow us to understand a small extension to its algorithm, which will, in turn, allow us to understand how division works over a given modulo, m (concept commonly known as **modular multiplicative inverse**).

- **An extension of Euclid's Algorithm**

The main motivation to have devised an extension of the original algorithm comes from the fact, that we might want **to actually check that a given integer number, say, d, is indeed the gcd of two other integer numbers, say a and b, i.e., we want to check d = gcd(a,b).**

As you might have noticed, it's not enough to check that d divides both a and b, to safely claim that d is the largest number that does so, as this only shows that d is a common factor and not necessarily the largest one.

To do so, we need to turn ourselves to a mathematical identity called the **Bézout's identity.**

- **The Bézout's identity**

Follow this question
By Email:
Once you sign in you will be able t
subscribe for any updates here
By RSS:
Answers
Answers and Comments

Question tags:
algorithm ×1,608
maths ×1,069
tutorial ×630
euclidean ×25
extended ×14
question asked: 14 Aug '13, 02:17
question was seen: 31,671 times
last updated: 22 May '17, 17:28

Related questions
Minimum Number of Calls to be made ?
Algorithmic problems
Algorithm to find all the divisors of a number
Proving the Fact of balanced edge bina tree
EULER TOTIENT FUNCTION
How to learn to solve codechef problem
The Rise and Fall of Power (A4) WA
Best place to practice and learn basics
Approach for problem GAMCOUNT
Need easy to understand tutorials some mathematical concepts used competitive programming

d = a**x** + b**y**

holds.

This is, in very simple terms, the Bézout's identity. (An outline of a proof might be found online)

What our extended Euclid's algorithm will allows us to do is to simultaneously find the value of d = gcd(a,b) **and** the values of x and y that actually "solve" (verify) the Bézout's identity.

- **A simple implementation of the Extended Euclid's Algorithm in Python**

Below you can find the implementation of the recursive version of this algorithm in the Python language (I must admit I haven't yet implemented it myself before, so I am also learning as I go, although I believe implementing the non-recursive version in C++ shouldn't be too complicated):

```
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)
```

This now solves, as desired, our original issue and allows us to conclude without any doubt that the value **d** on our original equation is indeed the **gcd(a,b)**.

- **An application: Computing the modular multiplicative inverse of a modulo m**

The most used application of this algorithm (at least, as far as I know and in the ambit of programming competitions) is the computation of the modular multiplicative inverse of a given integer a modulo m.

It is given by:

$$a^{-1} \equiv x \pmod{m}.$$

and mathematically speaking (as in, quoting Wikipedia), it is the multiplicative inverse in the ring of integers modulo m.

What the above means is that we can multiply both sides by a and we can obtain the identity:

$$ax \equiv aa^{-1} \equiv 1 \pmod{m}.$$

This means that m is a divisor of ax-1, which means we can have something like:

ax-1 = qm,

where q is an integer multiple that will be discarded.

If we rearrange the above as:

ax-mq = 1

we can now see that the above equation has the exact same form as the equation that the Extended Euclid's Algorithm solves (with a and m given as original parameters, x being the inverse and q being a multiple we can discard), **with a very subtle but important difference: gcd(a,m) NEEDS to be 1.**

What this basically means is that it is mandatory that a is coprime to the modulus, or else the inverse won't exist.

To wrap this text up, I will now leave you the code in Python which finds the modular multiplicative inverse of a modulo m using the Extended Euclid's Algorithm:

```
def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        return None  # modular inverse does not exist
    else:
        return x % m
```

- **Further explorations and a final note**

Number Theory is a beautiful field of Mathematics, but it is at the same time, one of the most vast and in my personal opinion, hardest fields to master.

The need of gcd(a,m) = 1, allows one to exploit this fact and use Euler's Theorem, along with Euler's Totient Function to find the modular inverse as well.

In fact, on the popular and most widely spread case where the modulus, m, happens to be a prime number, we can use the simple formula:

a^{m-2} (mod m)

to find the multiplicative inverse of a.

This result follows from Euler's Theorem directly.

Nontheless, besides my tutorial, my own personal experience is that it can be hard to actually understand all these ideas clearly in order to apply them successfully on a live contest (at least, it is hard for me), but, I hope that with some practice and also a lot more training and studying I can get better at it. So far, my study alongside with wikipedia and other books allowed me to write this tutorial which imho finds the best bits of information and puts them together on a same post.

Best regards,

Bruno

maths euclidean extended algorithm tutorial

edited 07 Mar '14, 01:59
3★ kcahdog
[10.0k]•28•54•129

asked 14 Aug '13, 02:17
2★ kuruma
[17.6k]•72•143•209
accept rate: 8%

In the last parts, where you mention Euler's Theorem, you can also mention Fermat's Little Theorem.

2★ tijoforyou (14 Aug '13, 09:03)

In the 1st code given can anyone tell me why it is necessary to rename builtin_gcd to __gcd, if we dont do so it is giving compilation error..??

4★ coding_addict (14 Aug '13, 10:46)

1 @coding_addict You can simply choose to use __gcd itself. But, it will look nice, if we have some other names than the preceding double underscores. It is not a rule. It is our choice! :D

2★ tijoforyou (14 Aug '13, 11:01)

@coding_addict, Thanks for your question, I will clarify that part better peraphs

2★ kuruma (14 Aug '13, 15:09)

1 @saikrishna173, it's the same as floor division, or, applying floor function to result of divison.

2★ kuruma (14 Aug '13, 19:47)

showing 5 of 7 show all

6 Answers:

oldest answers newest answers popular answers

2

@kuruma In Bezout's Identity you have mentioned that d = gcd(a,b) if and only if there are two positive integers x and y such that the identity : d = ax + by holds.

However the integers x and y need not be positive. Only the value ax + by i.e. d should be positive. eg. a=6 and b=3. then gcd(6,3)=3

Now 3= 6 * 0 + 1 * 3 only and 3 = 6x + 3y does not hold for any positive value of x and y.

link

answered 07 Mar '14, 01:58
3★ kcahdog
[10.0k]•28•54•129
accept rate: 14%

2 Thank you @kcahdog for pointing it out and for fixing it :)

2★ kuruma (09 Mar '14, 16:12)

1 I should thank you for writing such awesome tutorials! Have learnt a lot of stuff from them.

3★ kcahdog (09 Mar '14, 16:45)

2 :) Thanks a lot for your words! I've also learnt a lot by writing them and reading all of your corrections and ideas, so, this has been being a great synergy :D Hope this lasts for quite a while eheh

2★ kuruma (09 Mar '14, 16:48)

0

Can you please provide the c implementation of extended Euclidean algorithm?

link

answered 20 Apr '14, 21:14
5★ ashishtilokani
[30]•1•2•6
accept rate: 0%

0

I just found this question - http://discuss.codechef.com/questions/47223/new-method-for-finding-modular-inverse seems interesting...

link

answered 20 Nov '14, 12:26
3★ betlista ♦♦
[16.9k]•49•115•225
accept rate: 11%

0

Shouldn't the return value in the python implementation of Extended euclidean algo be **(g,x,y-(b//a)x)**, instead of **(g,x-(b//a)y,y)**. I'm following the proof mentioned here : http://e-maxx.ru/algo/extended_euclid_algorithm It would be great if someone could help me out.

link

edited 27 May '15, 21:37

answered 27 May '15, 21:33
5★ rtheman
[69]•3
accept rate: 0%

0

The Bézout's identity states that given two numbers a and b, passed as arguments to the gcd function, we can be sure that d = gcd(a,b) **only if** there are two integers x and y such that the identity:

d = ax + by

holds.

link

answered 18 Sep '16, 02:15
2★ flatballoon
[1]
accept rate: 0%

2 Actually if there exist two integers x and y such that
d = ax + by then d = gcd(a, b) iff d = min(ax+by) > 0. In your example min(4x+3y) = 1 (for x = 1 and y = -1) so 1 is the gcd of 4 and 3.

Yes he should have mention that d should be least positive linear combination of a and b.

3★ ashwanigautam (18 Sep '16, 02:42)

0

i am not able to understand the code of extend euclid algo in python ,,plz explain me how this code works,its working fine in my pc but i am not able to understand it on copy

link

answered 22 May '17, 17:28
arjarjun
[1]
accept rate: 0%

[hide preview] community wiki:

Preview

reCAPTCHA V1 IS SHUTDOWN
Direct site owners to g.co/recaptcha/upgrade

Type the text

Post Your Answer