

Forum: [Hack Forums](#)

Thread: [Generating Large Relative Primes](#)

Goodies - September 30, 2015, 12:03 am

This is more or less a code review. It's just a way to generate very large relatively prime numbers. Please let me know if (read "when") you find any issues or something that can be improved. I know I'm not a great C programmer, so I'm here for tips. Please be constructive. Thanks.

<https://github.com/GoodiesHQ/RPrimeGen/blob/master/rprimegen.c>

NTTITONV228 - September 30, 2015, 12:25 am

I could definitely optimize this in c without the usage of `mpz_*` to make it more clear for those that have never used `mpz_*` datatypes & functions such as myself (primitive types).

Goodies - September 30, 2015, 12:46 am

(Sep 30, 2015 12:25 AM)NTTITONV228 Wrote: [\[CITE\]](#) I could definitely optimize this in c without the usage of `mpz_*` to make it more clear for those that have never used `mpz_*` datatypes & functions such as myself (primitive types).

`mpz_t` is the data type used in GMP. In order to work with numbers potentially hundreds if not thousands of characters in length, I need to use `mpz_t`.

Voxfames - September 30, 2015, 12:52 am

(Sep 30, 2015 12:46 AM)Goodies Wrote: [\[CITE\]](#) `mpz_t` is the data type used in GMP. In order to work with numbers potentially hundreds if not thousands of characters in length, I need to use `mpz_t`.

Just ignore him he has over 228 accounts here. He is only here to troll

NTTITONV228 - September 30, 2015, 12:58 am

(Sep 30, 2015 12:46 AM)Goodies Wrote: [\[CITE\]](#) `mpz_t` is the data type used in GMP. In order to work with numbers potentially hundreds if not thousands of characters in length, I need to use `mpz_t`.

And how do you think this works in memory? Hence primitive types.

There's something called the big integer class in java; most similarly can be constructed in C using an integer array (Think of integer vectors in c++). You don't need to use `mpz_*`, although there is nothing wrong with it. But it's better to use primitive types and take your time to show your true understanding instead of using unnecessary libraries... otherwise this project is useless because anyone that wants to waste 10 minutes downloading a library in order to write 50 loc for finding prime numbers is just ridiculous, at least show some understanding barriers of memory and how to manage such large numbers... like seriously man ??? It's like releasing an ssl bot with boost library, anyone can fucking do it.

Quote:According to the website the code can be used to calculate 1000000000 digits of pi in 1957 seconds on an AMD Phenom II @ 3.2 GHz.

The code has been developed since 1991.

Jeez. Better algorithms have been developed since then, at least use something viable. 1 billion digits in 1957 seconds.... numberworld.org/y-cruncher/

do the math (Yes i notice it's different processors but compare the rates, download the program y cruncher and use it yourself for '1957' seconds for constant pi) it's not only performance that's the issue here..

(Sep 30, 2015 12:52 AM)Voxfames Wrote: [\[CITE\]](#) Just ignore him he has over 228 accounts here. He is only here to troll

Yeah, sorry if you didn't understand what i said. I only troll when it's not relevant to code you god damn shit box.

Goodies - September 30, 2015, 1:17 am

(Sep 30, 2015 12:58 AM)NTTITONV228 Wrote: [\[CITE\]](#) And how do you think this works in memory? Hence primitive types.Relevance? Why should I re-invent the wheel starting from the bottom instead of using tools I have available to me?

Quote:There's something called the big integer class in java; most similarly can be constructed in C using an integer array (Think of integer vectors in c++).

I don't do C++ and I hate Java.

Quote:You don't need to use mpz_*, although there is nothing wrong with it.

Agreed.

Quote:But it's better to use primitive types and take your time to show your true understanding instead of using unnecessary libraries.

Agreed.

Quote:otherwise this project is useless because anyone that wants to waste 10 minutes downloading a library in order to write 50 loc for finding prime numbers is just ridiculous,

Agreed.

Quote:at least show some understanding barriers of memory and how to manage such large numbers... like seriously man ???

Every time I researched managing large numbers manually, I continually was referred back to the GMP library. It's fairly recent, too. Yes, it was first developed decades ago, but was most recently updated last year.

However, you are right that I should learn about managing such large numbers manually.

Quote:It's like releasing an ssl bot with boost library, anyone can fucking do it.

Jeez. Better algorithms have been developed since then, at least use something viable. 1 billion digits in 1957 seconds....

k.

NTTITONV228 - September 30, 2015, 1:19 am

(Sep 30, 2015 01:17 AM)Goodies Wrote: [\[CITE\]](#) Relevance? Why should I re-invent the wheel starting from the bottom instead of using tools I have available to me?

I don't do C++ and I hate Java.

Agreed.

Agreed.

Agreed.

Every time I researched managing large numbers manually, I continually was referred back to the GMP library. It's fairly recent, too. Yes, it was first developed decades ago, but was most recently updated last year.

However, you are right that I should learn about managing such large numbers manually.

k.Ok you hate java, the big integer class was an example. You can rip it right from the jdk, it's not difficult to convert and understand.

Relevance? Why should I re-invent the wheel starting from the bottom instead of using tools I have available to me?

Re-invent what wheel exactly? a third party library capable of computing with shitty performance? Yeah.. you might want to fix that flat tire mate.

Goodies - September 30, 2015, 1:26 am

(Sep 30, 2015 01:19 AM)NTTITONV228 Wrote: [\[CITE\]](#) Ok you hate java, the big integer class was an example. You can rip it right from the jdk, it's not difficult to convert and understand.

Relevance? Why should I re-invent the wheel starting from the bottom instead of using tools I have available to me?

Re-invent what wheel exactly? a third party library capable of computing with shitty performance? Yeah.. you might want to fix that flat tire mate.

I'm not doing that. That seems like a huge waste of time. I'm sure there's plenty of documentation in C to guide me.

By the way, this all stemmed from a completely different project that I'm doing. For that project, I needed a simple LCG, and for that, I needed relative primes. I wrote this code in maybe 3 hours with Netflix in the background, but the point is that I'm not a programmer, so I'd like to know some rookie mistakes that I've undoubtedly made.

NTTITONV228 - September 30, 2015, 1:33 am

(Sep 30, 2015 01:26 AM)Goodies Wrote: [\[CITE\]](#) I'm not doing that. That seems like a huge waste of time. I'm sure there's plenty of documentation in C to guide me.

By the way, this all stemmed from a completely different project that I'm doing. For that project, I needed a simple LCG, and for that, I needed relative primes. I wrote this code in maybe 3 hours with Netflix in the background, but the point is that I'm not a programmer, so I'd like to know some rookie mistakes that I've undoubtedly made.

If this is for engineering/math major, why aren't you using fortran? regardless, it doesn't matter considering you're not a programmer.

Oh well, LCG isn't all that, once you get more into number theory you'll develop your own algorithms. Hopefully it all goes well..

Goodies - September 30, 2015, 9:25 am

(Sep 30, 2015 01:33 AM)NTTITONV228 Wrote: [\[CITE\]](#) If this is for engineering/math major, why aren't you using fortran? regardless, it doesn't matter considering you're not a programmer. My major is in Computer Networking (dual majoring in Philosophy of Science btw).

Quote:Oh well, LCG isn't all that, once you get more into number theory you'll develop your own algorithms. Hopefully it all goes well..

I'm aware, but chances are I won't be developing my own algorithms.

0xDEAD10CC - October 4, 2015, 3:12 pm

Code #include <stdbool.h>

^ You already include this (C99)

So what's the point of this:

Code #define FALSE 0

#define TRUE 1

Additionally, <Windows.h> has it's own BOOL, TRUE, and FALSE.

Also, why?

Code int16_t main(uint16_t argc, char **argv) {

And you define FAILURE, but only use it 50% of the time:

Code if(mpz_set_str(pair, *(argv+1), 10) == -1)

'-1'

And these should be a bit more generic than argument-less functions:

Code setlimits();

setseed();

genpair();

Goodies - October 4, 2015, 3:56 pm

(Oct 4, 2015 03:12 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Code #include <stdbool.h>

^ You already include this (C99)

So what's the point of this:

Code #define FALSE 0

#define TRUE 1

Additionally, <Windows.h> has it's own BOOL, TRUE, and FALSE.

Also, why?

Code int16_t main(uint16_t argc, char **argv) {

And you define FAILURE, but only use it 50% of the time:

Code if(mpz_set_str(pair, *(argv+1), 10) == -1)

'-1'

And these should be a bit more generic than argument-less functions:

Code setlimits();

setseed();

genpair();

Thanks! A couple things.

1) I'm doing this on Linux. I honestly have no idea if it will compile/run or not Windows. I added Windows.h as an afterthought, and I didn't know that it had TRUE or FALSE in it at all.

2) I changed the in16_t for main() back to int. I'm not sure why, but I prefer uniform sizes regardless of OS, even if it's completely (admittedly) unnecessary.

3) The reason I wasn't passing mpz_t variables into the setlimit() setseed() and genpair() functions is that I didn't intend to make them at all. it was all in one main() function at first until it got to be too cluttered and I had already made them global. Out of curiosity, does this bring any downsides? Or is it just convention?

