# Does It Beat the Minimal Standard?

M.E. O'Neill — 2018-03-24 17:52

Suppose that you've written (or just seen someone else announce) a brand new PRNG. Cool! It's nice to have new things. But here's the question you should ask yourself; “Is it better than a reasonable ‘minimal standard’? Does it beat methods devised more than sixty years ago?”

## Wisdom from the Past

Back in 1987, more than thirty years ago now, Steve Park and Keith Miller wrote the paper *Random Number Generators: Good Ones Are Hard to Find* (https://dl.acm.org/citation.cfm?id=63042), which lamented the poor quality of many of the PRNGs in use at the time and proposed a bare mininum for a “good enough” PRNG (the paper appeared in CACM early the following year). In that paper, they wrote

> In retrospect it is evident that a generally satisfactory algorithm for random number generation was proposed by D. H. Lehmer 36 years ago [26]. This parametric multiplicative linear congruential algorithm has withstood the test of time. It can be implemented efficiently [27, 31, 37, 41], numerous empirical tests of the randomness of its output have been published [8, 15, 27, 28, 37], and its important theoretical properties have been analyzed [9, 14, 18, 30]. The conclusion to be drawn from all this research is now clear. Although Lehmer’s algorithm has some statistical defects, if the algorithm parameters are chosen properly and if the software implementation of the algorithm is correct, the resulting generator can produce a virtually infinite sequence of numbers that will satisfy almost any statistical test of randomness. In other words, with properly chosen parameters, Lehmer’s algorithm, correctly implemented, represents a good minimal standard generator against which all other random number generators can—and should—be judged.

Thirty years later, some specifics may have changed (i.e., the necessary parameters for what we might consider a reasonable LCG), but the fundamental idea—that any proponent of a new PRNG should be asked to explain how their technique is better than an appropriate “minimal standard” LCG—remains as relevant today is it was then.

If you can't show that your PRNG technique beats an idea from over sixty years ago, you haven't proven its worth.

## Today's 32-bit Minimal Standard: A 96-bit Truncated LCG or MCG

The code below represents a reasonable minimal-standard MCG:

```
uint96_t state = 1;   // can be seeded to any odd number

uint32_t next()
{
    state *= 0xdc87976860b11728995deb95;
    return state >> 64;
}
```

And this version is a reasonable minimal-standard LCG:

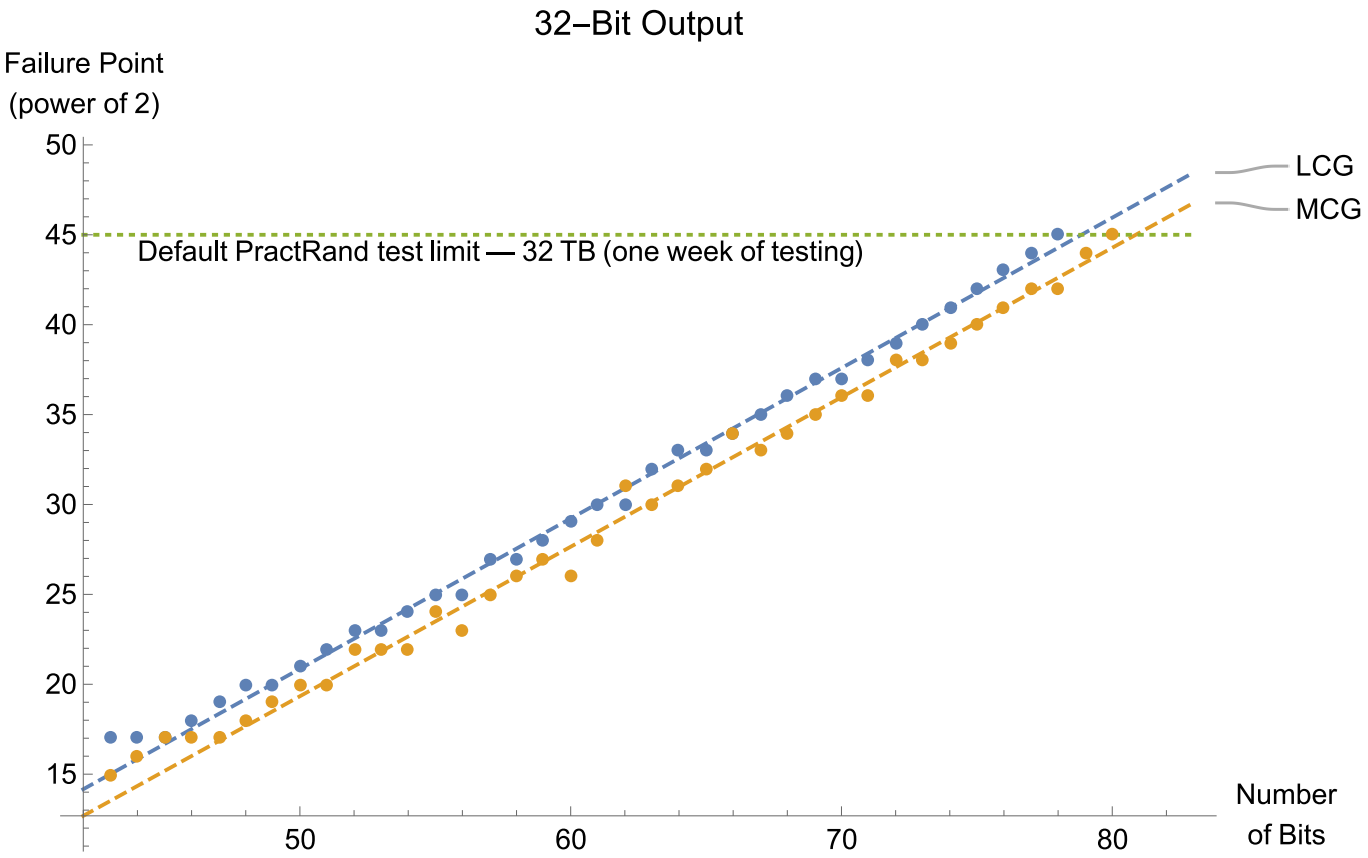
```
uint96_t state;   // can be seeded to any number

uint32_t next()
{
    const uint96_t MULTIPLIER = 0xc580cadd754f7336d2eaa27d;
    state *= MULTIPLIER;
    state += MULTIPLIER;
        // Any odd odd number will do, the multiplier is odd and
        // conveniently to hand
    return state >> 64;
}
```

What we know about these two PRNGs is that

- The code is incredibly simple (although it does presume the existence of a 96-bit (or larger) integer type).
- On 64-bit machines, the necessary multiplication is not difficult to perform. In fact, on modern x86 machines, the entire generation process can take place in less than a nanosecond.
- The generators are known to have long periods and no short cycles. The LCG has period $2^{96}$, which is maximal, and no bad initializations

—any initial seed is okay. The MCG provides two separate cycles of size $2^{62}$ and only requires that the seed be odd.
- The underlying theory ensures that the generators are uniform (each output occurs the same number of times).
- Because the output function is truncation, each output occurs a vast number of times ($2^{62}$ and $2^{64}$, respectively), meaning that producing one output does not significantly reduce the chance of it recurring.
- The underlying theory is very well understood. The multiplicative constants were chosen to have excellent "spectral test" (https://en.wikipedia.org/wiki/Spectral_test) values (for the MCG, normalized LLL-spectral test (http://dx.doi.org/10.1287/ijoc.9.2.206) values are $M_8 = 0.70279$, $M_{16} = 0.65345$, $M_{24} = 0.58118$ and for the LCG, $M_8 = 0.73201$, $M_{16} = 0.64438$, $M_{24} = 0.53151$, as computed by software by Karl Entacher (http://random.mat.sbg.ac.at/results/karl/spectraltest/)).
- The output passes stringent statistical tests, such as PractRand up to 32 TB and TestU01's BigCrush test (http://simul.iro.umontreal.ca/testu01/tu01.html).

In fact, not only does this 96-bit PRNG pass statistical tests, it passes them *well*. Testing with PractRand, we can plot the failure point for different sized 32-bit generators as follows:




This graph shows us that we leave clear-fail territory at 78 bits for LCGs and 80 bits for MCGs. Thus a 96-bit LCG or MCG has considerable *headroom* available to allow it to pass larger statistical tests.

Extrapolating the line from this graph shows that we'll get to the clear failure point for a 96-bit MCG at 128 petabytes of data, and for the LCG at the half exabyte point. If it takes PractRand a week of testing to test 32 TB, half an exabyte will require more than three hundred years of testing to detect statistical flaws.

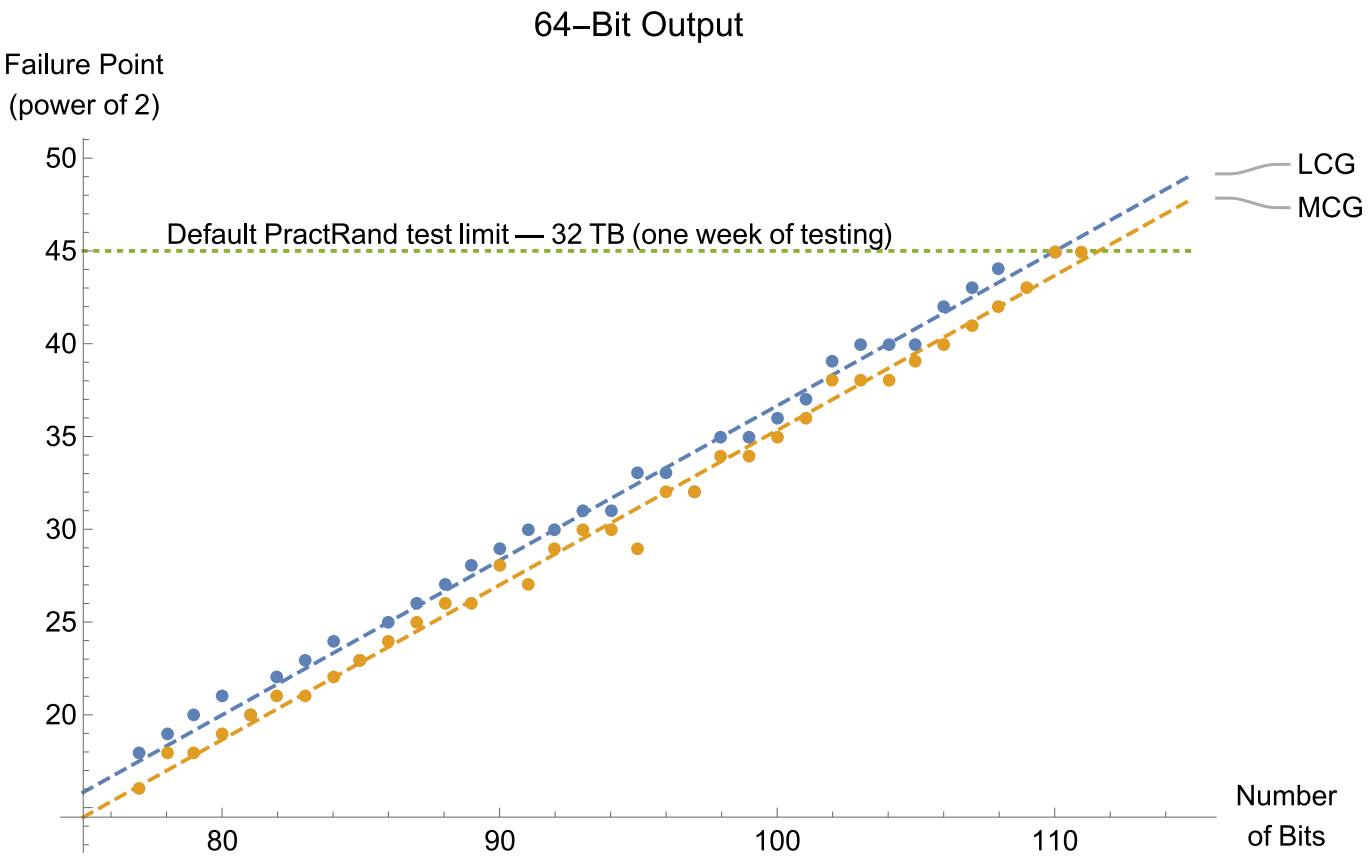
## Today's 64-bit Minimal Standard: A 128-bit Truncated LCG or MCG

Similarly, if you want 64-bit output, a 128-bit truncated LCG or MCG represents a reasonable minimal standard. In fact, on 64-bit hardware, these generators are no more costly to compute that the 32-bit output versions given above.

```
uint128_t state = 1;   // can be seeded to any odd number

uint64_t next()
{
    state *= 0x0fc94e3bf4e9ab32866458cd56f5e605;
            // Spectral test: M8 = 0.71005, M16 = 0.66094, M24 = 0.61455
    return state >> 64;
}
```

```
uint128_t state;   // can be seeded to any number

uint64_t next()
{
    const uint128_t MULTIPLIER = 0x2d99787926d46932a4c1f32680f70c55;
    state *= MULTIPLIER;
        // Spectral test: M8 = 0.70420, M16 = 0.65412, M24 = 0.60209
    state += MULTIPLIER;
        // Any odd odd number will do, the multiplier is odd and
        // conveniently to hand
    return state >> 64;
}
```

Essentially the same observations we made for the 32-bit generators apply here as well. Once again, we can plot the failure point for different sized 64-bit generators as follows:



This graph shows us that we leave clear-fail territory at 109 bits for LCGs and 111 bits for MCGs. Thus a 128-bit LCG or MCG has considerable *headroom* available to allow it to pass larger statistical tests.

As before, we can extrapolate the line from this graph, this time finding that we reach the clear failure point for a 128-bit MCG at 256 petabytes of data, and for the LCG at the 1 exabyte point. If it takes PractRand a week of testing to test 32 TB, an exabyte will require more than six hundred years of testing to detect statistical flaws.

## Conclusion

Linear congruential generators are frequently cited as being a poor choice for random-number generation, and at small sizes that remains true, but as we have seen, at larger sizes these generators do very well in statistical tests. Forty years ago, these larger versions might have been unwieldy, but that has not been the case for a long time. Today modern 64-bit architectures have little difficulty performing 128-bit math quickly, and since 2016 (https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/), even a $35 Raspberry Pi (https://www.raspberrypi.org/blog/raspberry-pi-3-model-bplus-sale-now-35/) has a 64-bit quad-core CPU. And older 32-bit machines can perform 96-bit math acceptably fast—it's worth realizing that the longstanding `drand48` generator present in all Unix systems was designed in the age of 16-bit machines and can be trivially adapted to use 32-bit data, creating `drand96`.

Any new PRNG with 64-bit output needs to answer the question, “How is it better than the minimal standard, a 128-bit LCG?” and likewise any PRNG with 32-bit output needs to answer the question “How is it better than the minimal standard, a 96-bit LCG?”. It is not enough just to be fairly fast, because these minimal standards are fast. It is not enough to pass stringent statistical tests, because these minimal standard generators pass those same tests, and pass them *easily*.

Moreover, the graphs we have seen show that it is possible to take a scientific approach to testing PRNGs. The trend lines we have seen allow us to make good predictions about tests that that might be prohibitively expensive to run, so we see another useful tip for designers of PRNGs. Don't just provide a single data point, provide many and do some science.

## Appendix

To allow others to recreate the above graphs, below are multiplicative constants for MCGs and LCGs of different sizes. The constants were found by using a random search to find constants that had good normalized LLL-spectral test results; each constant is intended to be "very good" rather than truly outstanding. Each table shows the $M_8$ result for the normalized LLL-spectral test. (http://dx.doi.org/10.1287/ijoc.9.2.206) The $M_{16}$ values are all above 0.625.

***MCG Constants***

| Bits | Multiplicative Constant | Spectral Test |
|---|---|---|
| 32 | 2739110765 | 0.716768631223 |
| 33 | 6416791565 | 0.741609517211 |
| 34 | 2110734621 | 0.730357331437 |
| 35 | 9409920061 | 0.727139444967 |
| 36 | 8124055821 | 0.726338729537 |
| 37 | 85397059285 | 0.745062524229 |
| 38 | 268237653253 | 0.742041038480 |
| 39 | 220322046269 | 0.729472249491 |
| 40 | 533248573853 | 0.741393982264 |
| 41 | 1159580999389 | 0.721303065856 |
| 42 | 4207420190797 | 0.727290948336 |
| 43 | 1162000279733 | 0.717554748268 |
| 44 | 14291355528493 | 0.721569546451 |
| 45 | 32316819487421 | 0.722852594721 |

| Bits | Multiplicative Constant | Spectral Test |
|---|---|---|
| 46 | 27424978670797 | 0.723975492235 |
| 47 | 112357400097837 | 0.725660187612 |
| 48 | 203797216008469 | 0.728474675519 |
| 49 | 14838598453045 | 0.736512565883 |
| 50 | 406663326808437 | 0.755377478054 |
| 51 | 942227593642253 | 0.727499556775 |
| 52 | 276168391955373 | 0.738214029774 |
| 53 | 7207664573232381 | 0.747016695380 |
| 54 | 1678565460045845 | 0.727249278946 |
| 55 | 15485172535814365 | 0.710832966834 |
| 56 | 65157942103163701 | 0.729099254104 |
| 57 | 41826564414174485 | 0.737033978607 |
| 58 | 265278118812494821 | 0.720996827858 |
| 59 | 43534115661926286 1 | 0.713519610623 |
| 60 | 2229153327831421 17 | 0.737161497144 |
| 61 | 20420333521707024 85 | 0.730916689500 |
| 62 | 38522130061940321 49 | 0.717693049282 |
| 63 | 26814003317027713 81 | 0.730222321935 |
| 64 | 14647171131086947261 | 0.738375341778 |
| 65 | 23586512187791901165 | 0.749719574465 |
| 66 | 4021396209304885645 | 0.737035361383 |
| 67 | 713388175221274576 61 | 0.749586645884 |
| 68 | 235371599704391449 81 | 0.741383430613 |
| 69 | 5744099463553435900 93 | 0.728499816481 |
| 70 | 847531188765324779717 | 0.735818093856 |
| 71 | 154015064707048177 6589 | 0.736156422148 |
| 72 | 362951267625149354 7269 | 0.733279877031 |
| 73 | 569150550081143949 4453 | 0.731779832097 |
| 74 | 268758921655152549 3141 | 0.717594710954 |
| 75 | 594402455649377636 5117 | 0.731218718355 |
| 76 | 254000411055292709 3773 | 0.741476230780 |
| 77 | 961482343430672208 5205 | 0.716421804008 |

| Bits | Multiplicative Constant | Spectral Test |
|---|---|---|
| 78 | 256445958763511920394925 | 0.733008689483 |
| 79 | 601712692362582751632037 | 0.744911691987 |
| 80 | 396062012093570494875 89 | 0.725258169835 |
| 81 | 129013195733174607937330 1 | 0.734219557372 |
| 82 | 397767538977878279138816 5 | 0.756010134150 |
| 83 | 762152609915050521565922 1 | 0.725543053961 |
| 84 | 987723928296148257293750 1 | 0.725928715061 |
| 85 | 252364371504075552473930 29 | 0.749180805625 |
| 86 | 817103110113322906103303 7 | 0.740169374705 |
| 87 | 493625406713896980041183 17 | 0.724803286548 |
| 88 | 257722390823181249111569 3 | 0.719764670518 |
| 89 | 249522469266726746228387 597 | 0.749826167287 |
| 90 | 839446671699896991292676 421 | 0.727750604637 |
| 91 | 958856983424445520621212 381 | 0.720868271564 |
| 92 | 936151086994606743394269 365 | 0.717081973076 |
| 93 | 439855118747073666576541 0061 | 0.737375655009 |
| 94 | 165757175167195745695189 05037 | 0.733588588969 |
| 95 | 169244510418128200169380 51413 | 0.738072389915 |
| 96 | 636842078722189695046391 12949 | 0.722889364342 |
| 97 | 123612427885422382889677 805949 | 0.720047707563 |
| 98 | 135644778577393329179563 37261 | 0.723224899486 |
| 99 | 567192134585768783122181 505413 | 0.727178102380 |
| 100 | 605302915592620321688005 792373 | 0.745754947466 |
| 101 | 194067547904011311755189 2359605 | 0.739161068408 |
| 102 | 365891647347813458216718 3652629 | 0.753704464419 |
| 103 | 157135932919948579414179 2737933 | 0.730697664731 |
| 104 | 169044450179127387607677 35186589 | 0.742586816074 |
| 105 | 224858555707681665672059 90626413 | 0.718485657687 |
| 106 | 743132798534582399975754 61406493 | 0.720008226330 |
| 107 | 697895680924988112461305 93789565 | 0.731015334466 |
| 108 | 199185148331384386512624 391509277 | 0.753423145342 |
| 109 | 445995822870053631298584 088971085 | 0.737911420945 |

| Bits | Multiplicative Constant | Spectral Test |
|---|---|---|
| 110 | 52203894373882052611337698157076 5 | 0.728651914451 |
| 111 | 17901662234667289288940314456592 21 | 0.745853280704 |
| 112 | 23762325944193467387935164310342 85 | 0.739509229331 |
| 113 | 17393655162080712030479898291643 73 | 0.752781474861 |
| 114 | 13159937790046761463548017391558 589 | 0.735305543510 |
| 115 | 26771634451446385007601761491488 901 | 0.724847025658 |
| 116 | 14592594772439033291729959800277 197 | 0.744686357317 |
| 117 | 60966792259065248069366924264146 173 | 0.741555898957 |
| 118 | 68836090842512112509636367404547 397 | 0.746365192374 |
| 119 | 52418635260008058404993056727452 2237 | 0.739180239317 |
| 120 | 61077515213203575558359089985519 5013 | 0.747582831613 |
| 121 | 90038733420382331122954654054952 9981 | 0.745007443886 |
| 122 | 23067259533481257461900753364185 08061 | 0.727909253640 |
| 123 | 18004139486006149787253846790208 07301 | 0.723551474912 |
| 124 | 18484242130631853726820903993747 978157 | 0.718440060620 |
| 125 | 26907765095583321268647989660234 740333 | 0.722561248710 |
| 126 | 25076288795215231685469951584245 457885 | 0.724645916794 |
| 127 | 34218442865726806866117399643452 229733 | 0.719956739979 |
| 128 | 63788880824840432877499191278319 602189 | 0.731606096275 |

***LCG Constants***

| Bits | Multiplicative Constant | Spectral Test |
|---|---|---|
| 32 | 1019135901 | 0.726561516888 |
| 33 | 6527674237 | 0.718126733013 |
| 34 | 15029734005 | 0.723277276010 |
| 35 | 3660796997 | 0.742645750618 |
| 36 | 29510867981 | 0.719429027076 |
| 37 | 63514268605 | 0.732551422652 |
| 38 | 116624537293 | 0.724558743348 |
| 39 | 194195883829 | 0.722415400434 |
| 40 | 568512975829 | 0.730711283118 |
| 41 | 1322742409629 | 0.738030287624 |

| Bits | Multiplicative Constant | Spectral Test |
|---|---|---|
| 42 | 126882788757 | 0.728710926727 |
| 43 | 1391190336909 | 0.729854488924 |
| 44 | 9272727027237 | 0.722758310108 |
| 45 | 12790945375773 | 0.737588630205 |
| 46 | 15105438078773 | 0.739295363228 |
| 47 | 50521972618661 | 0.723112105229 |
| 48 | 249419361368573 | 0.730241902102 |
| 49 | 404273197363917 | 0.750413967945 |
| 50 | 1785193840416 29 | 0.731476180651 |
| 51 | 435569983484149 | 0.734296368387 |
| 52 | 672969485740877 | 0.748370625927 |
| 53 | 1997786961644461 | 0.721800591403 |
| 54 | 7946058339844821 | 0.741940429724 |
| 55 | 3421042040467 0109 | 0.735305419562 |
| 56 | 4968223042472 1469 | 0.751246366335 |
| 57 | 4155281800741 2365 | 0.732767196482 |
| 58 | 20876812449011 9261 | 0.720192561204 |
| 59 | 53418670303654 5445 | 0.732258727961 |
| 60 | 71819893795646 3965 | 0.717132171609 |
| 61 | 14927610482496 20805 | 0.732951920801 |
| 62 | 2857626640570 93061 | 0.720297424345 |
| 63 | 55624086579458 04781 | 0.740322612872 |
| 64 | 91999403085852 34877 | 0.740159647448 |
| 65 | 11160153871882 754749 | 0.722374537852 |
| 66 | 30609332369975 655877 | 0.750132565315 |
| 67 | 10250457471057 4106733 | 0.727072404636 |
| 68 | 28900880863544 7097333 | 0.737066005448 |
| 69 | 20096507486281 2859469 | 0.727446643227 |
| 70 | 99521660642577 5766885 | 0.742508065267 |
| 71 | 13834462501191 46840917 | 0.735902660153 |
| 72 | 12779747553922 84889725 | 0.745059476563 |
| 73 | 75562100772493 7721413 | 0.723731561436 |

| Bits | Multiplicative Constant | Spectral Test |
|---|---|---|
| 74 | 1403601537669505748437 | 0.735159325664 |
| 75 | 3443051036027360753871 7 | 0.739570300200 |
| 76 | 5602134104600432135767 7 | 0.732060732087 |
| 77 | 5742854999522408107061 3 | 0.737418286180 |
| 78 | 1612799811754133975053 41 | 0.736420906775 |
| 79 | 6010095727327928605130 77 | 0.744752347985 |
| 80 | 8129784566064384657178 61 | 0.725653623399 |
| 81 | 2230578935970333198265 389 | 0.726202750322 |
| 82 | 3110954275176992599611 333 | 0.722803877503 |
| 83 | 2549771676202439997119 837 | 0.728988412935 |
| 84 | 1724111934827110856193 2573 | 0.733198806677 |
| 85 | 2645773155329572386128 4917 | 0.751942767791 |
| 86 | 5909079172946320149419 1589 | 0.726268118857 |
| 87 | 1527188166399838488681 7965 | 0.734997793995 |
| 88 | 7062241603228225017008 2701 | 0.730360551193 |
| 89 | 4265411516667560816290 40421 | 0.732673387845 |
| 90 | 1033056376707427530835 730733 | 0.730196493664 |
| 91 | 3960241768858220285464 35213 | 0.730113412997 |
| 92 | 2388182321528562871077 577165 | 0.737154295148 |
| 93 | 8010225565907072943071 535853 | 0.728882371264 |
| 94 | 1326120662959507290825 2835605 | 0.725329755925 |
| 95 | 1690242280411359261125 1823813 | 0.750861377225 |
| 96 | 6112424744292873273619 0063229 | 0.732014535166 |
| 97 | 2791178274538571056357 4645949 | 0.734864728131 |
| 98 | 2993863475295039028618 6539269 | 0.723675443983 |
| 99 | 2622922570895768685141 1456901 | 0.735866461121 |
| 100 | 8190297234186974164411 04865757 | 0.716956721057 |
| 101 | 2322146411595516589823 065344981 | 0.741946609263 |
| 102 | 1784183314298914792221 437953997 | 0.730507073217 |
| 103 | 3235104345222420208314 737169685 | 0.726129037214 |
| 104 | 6718814192664462072263 8870253 | 0.747754151762 |
| 105 | 2436249846286644479688 629522261 | 0.723418570653 |

| Bits | Multiplicative Constant | Spectral Test |
|---|---|---|
| 106 | 182567073328247796690884427354 9 | 0.760448328874 |
| 107 | 324653970483671718516754476042 9 | 0.732216405411 |
| 108 | 303587630294941123502336685825389 | 0.719120482824 |
| 109 | 113926489974744232355028377198 21 | 0.751921546570 |
| 110 | 964268383221670159557101452712557 | 0.734010450204 |
| 111 | 639338642199625092864608394898789 | 0.735970283015 |
| 112 | 190636817896380433646609610754845 3 | 0.747352401943 |
| 113 | 674973075863102259514945295869340 5 | 0.730041084931 |
| 114 | 167076543753511380708458525724472 29 | 0.723933957671 |
| 115 | 172076203640458720193947065344264 21 | 0.740743821760 |
| 116 | 156234954177642516079846243568185 01 | 0.731135297999 |
| 117 | 131745201739913730912678565034201 349 | 0.737765053219 |
| 118 | 342011107102696511776925457344658 77 | 0.726921739220 |
| 119 | 144754161049240217053622458007212997 | 0.719557456180 |
| 120 | 113902786893147305053150670351992 1533 | 0.723535881344 |
| 121 | 261210642589146262073919652551158 4853 | 0.734994863425 |
| 122 | 200951714646543999995751798245400 9829 | 0.744269096369 |
| 123 | 834257908093988255830252141117197 7885 | 0.745363344384 |
| 124 | 177749641913694747615888462506811 98813 | 0.726741230435 |
| 125 | 184182245510097692968253990459807 69997 | 0.724759980253 |
| 126 | 569679959619315054419146244977783 63077 | 0.735443492446 |
| 127 | 921554234107826724372006033275852 09317 | 0.742100869722 |
| 128 | 199967246047888932297834045878657 099405 | 0.723283410960 |