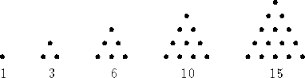
Triangular numbers are indeed a most mysterious sets. They are the collection of all that can be arranged as a triangle of dots with each level one less in size than the level below. In effect, the following are triangular.



We will continue to explore triangular numbers and other related sets below, discovering their properties and proving where they come from.

Definition 1.1, Triangular Number: , or the triangular number, can be written as

. Or in other words:

Lemma 1.1.1) The Symmetry of Triangles:

Q.E.D

This fact was first realized by Carl Freidrich Gauss supposedly as a child when an instructor asked him to add all the numbers from 100 and below. He quickly shouted out “5050,” realizing the geometric symmetry of this problem.

Theorem 1.1) The Triangular Number Theorem:

Q.E.D

Generating such can be shown as follows in Java with relative ease with a time controlled while loop:

**public** **class** PrintTriangularNumbers {

**public** **static** **void** main(String[] args) {

**long** i = 0;

**long** startTime = System.*currentTimeMillis*();

**while**(**false**||(System.*currentTimeMillis*()-startTime)<1000) {

System.***out***.print((i\*i + i)/2 + ", ");

i++;

}

}

}

A sample output of this program is as follows:

1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136, 153, 171, 190, 210, 231, 253, 276, 300, 325, 351, 378, 406, 435, 465, 496, 528, 561, 595, 630, 666, 703, 741, 780, 820, 861, 903, 946, 990, 1035, 1081, 1128, 1176, 1225, 1275, 1326, 1378, 1431, 1485, 1540, 1596, 1653, 1711, 1770, 1830, 1891, 1953, 2016, 2080, 2145, 2211, 2278, 2346, 2415, 2485, 2556, 2628, 2701, 2775, 2850, 2926, 3003, 3081, 3160, 3240, 3321, 3403, 3486, 3570, 3655, 3741, 3828, 3916, 4005, 4095, 4186, 4278, 4371, 4465, 4560, 4656, 4753, 4851, 4950, 5050, 5151, 5253, 5356, 5460, 5565, 5671, 5778, 5886, 5995, 6105, 6216, 6328, 6441, 6555, 6670, 6786, 6903, 7021, 7140, 7260, 7381, 7503, 7626, 7750, 7875, 8001, 8128, 8256, 8385, 8515, 8646, 8778, 8911, 9045, 9180, 9316, 9453, 9591, 9730, 9870, 10011, 10153, 10296, 10440, 10585, 10731, 10878, 11026, 11175, 11325, 11476, 11628, 11781, 11935, 12090, 12246, 12403, 12561, 12720, 12880, 13041, 13203, 13366, 13530, 13695, 13861, 14028, 14196, 14365, 14535, 14706, 14878, 15051, 15225, 15400, 15576, 15753, 15931, 16110, 16290, 16471, 16653, 16836, 17020, 17205, 17391, 17578, 17766, 17955, 18145, 18336, 18528, 18721, 18915, 19110, 19306, 19503, 19701, 19900, 20100, 20301, 20503, 20706, 20910, 21115, 21321, 21528, 21736, 21945, 22155, 22366, 22578, 22791, 23005, 23220, 23436, 23653, 23871, 24090, 24310, 24531, 24753, 24976, 25200, 25425, 25651, 25878, 26106, 26335, 26565, 26796, 27028, 27261, 27495, 27730, 27966, 28203, 28441, 28680, 28920, 29161, 29403, 29646, 29890, 30135, 30381, 30628, 30876, 31125, 31375, 31626, 31878, 32131, 32385, 32640, 32896, 33153, 33411, 33670, 33930, 34191, 34453, 34716, 34980, 35245, 35511, 35778, 36046,

These are the Triangular Numbers. Similar principles extend for other sets such as the Hexagonal numbers. Triangular numbers are indeed housed within other famous sets such as every even perfect number is indeed triangular. However, another skim set brings us to the focus of this paper: The Square Triangular Numbers. These are numbers that are both triangular and square. For example, . Are there any patters of these numbers? How can one generate these numbers? Are there infinitely many numbers of this form?

Let us begin answering these questions.

Definition 1.2, Square Triangular Number: , is a number that is both a perfect square, and triangular in nature as in.

One can start to gain insight by trying to find some of these numbers through computation. For example with the following Java Program.

**public** **class** PrintTriangularSquares {

**public** **static** **void** main(String[] args) {

**long** i = 0;

**while**(**true**) {

**if**(*isPerfectSquare*((i\*i + i)/2))

System.***out***.print((i\*i + i)/2 + ", ");

i++;

}

}

**public** **static** **boolean** isPerfectSquare(**long** i) {

**long** closestRoot = (**long**) Math.*sqrt*(i);

**return** i == closestRoot \* closestRoot;

}

}

In short time, we will come up with some of the following before the Java long overflows with its maximum value of :

0, 1, 36, 1225, 41616, 1413721, 48024900, 1631432881, 55420693056, 1882672131025, 63955431761796, 2172602007770041, 73804512832419600, 2507180834294496361.

You can generate even more using the “BigInteger” class. However, let us revisit this problem from a mathematical perspective:

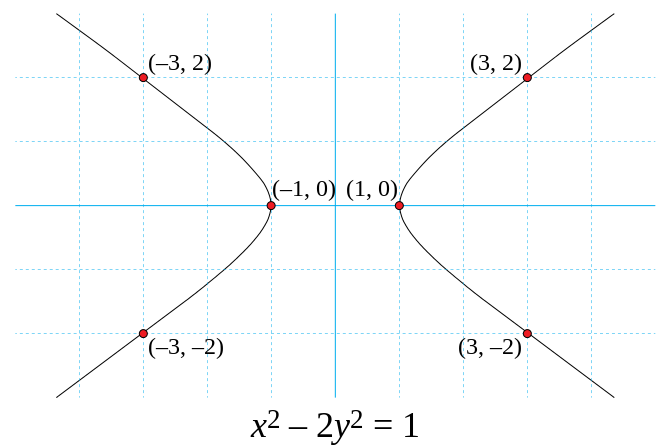
What is a Triangular Number? This problem is indeed a Diophantine equation:

Corollary 1.1: Diophantine equation for Square Triangular numbers. They must be both triangular and square, so the following falls out trivially:

Let us try to solve this equation to see if we can possibly generate Square Triangular Numbers More intelligently. We will use to keep track of the triangular aspect of this problem and to keep track of the square aspect of this problem. Hence,

Taking this equation and completing the square yields the following:

This is indeed an instance of the famous Pell Equation by letting , which is the following Diophantine equation. The Pell equation has many notable properties, relating to the ratio of its solutions, among other things. Some of these properties will be useful in expressing certain patterns of square triangular numbers. The following is a graph of the Pell equation, a graph that on the surface is a hyperbola with some vertical stretch, but is a very fascinating function indeed:



One way of solving Diophantine equations is to look for a way to express the points that lie on the equation by relating them to something else. For example, take the line that intersects the Pell equation at point , . One could attempt to find the other intersection, and thus any coordinate on the Pell equation, based on the rational slope of the line. This is generally done by dividing off by substituting the value into the part of the Diophantine equation, simplifying, and dividing off by the one intersection point of the line, and then solving for This is the strategy employed to solve the Pell equation in our instance as well.

Lemma 1.2:

It is noteworthy to say that all rational coordinates on this hyperbola, like the trivial hyperbola, and the unit circle, are very close to ratios of integer solutions to

According to the Pythagorean Triples Theorem: All Primitive Pythagorean Triples, that is all triples that share no common factors with other members of their triple, are of the form,

Some rearranging of this equation can show that our coordinate based on proves this fact.

We have now expressed the coordinates on the hyperbola in terms of some . However, it is obvious not all of these coordinates are integers. Thus, we have not solved our Diophantine equation. It, therefore, must be true that for only some values of are their integer solutions, and thus square triangles are lurking.

Conjecture 1.1:

This is indeed not intuitive, however, let use use this conjecture nonetheless to show how it solves our Diophantine equation.

Lemma 1.3:

This new solution can be used as and feed into a new rational slope that will generate some new and so on. Because these coordinates grow by an additive square and a multiplicative nature respectively, this will lend itself to the generation of infinitely many points of this form. This does indeed also mean, that the set of natural coordinates becomes more and more rare the larger and larger we go.

Theorem 1.3: Infinitely many triangular square numbers can be generated recursively as follows:

We can now implement this recursive generating function on a timed while loop in Java, using the BigInteger and BigDecimal classes to manage numbers of such enormous size:

**public** **class** SquareTriangularNumbers {

**public** **static** **void** main(String[] args) {

BigInteger uInitial = BigInteger.*valueOf*(3);

BigInteger vInitial = BigInteger.*valueOf*(2);

**long** startTime = System.*currentTimeMillis*();

**while**(**false**||(System.*currentTimeMillis*()-startTime)<10000) {

BigInteger sValue = uInitial.multiply(vInitial);

BigInteger tValue = *bigSqrt*(**new** BigDecimal(BigInteger.*valueOf*(1).add(BigInteger.*valueOf*(8).multiply(sValue).multiply(sValue)))).subtract(BigDecimal.*valueOf*(1)).divide(BigDecimal.*valueOf*(2)).toBigInteger();

BigDecimal tDecimal = **new** BigDecimal(tValue);

BigDecimal sDecimal = **new** BigDecimal(sValue);

System.***out***.println("Square Triangular : " + sValue.multiply(sValue).toString() + " SquareRoot: " + sValue.toString() + " Triangle: " + tValue.toString() + " t/s Ratio: " + tDecimal.divide(sDecimal, MathContext.***DECIMAL128***) + " verified?: " + *verifyTriangularSquare*(sValue,tValue) + " uGenerator: " + uInitial + " vGenerator: " + vInitial);

vInitial = BigInteger.*valueOf*(2).multiply(sValue);

uInitial = *bigSqrt*(**new** BigDecimal(BigInteger.*valueOf*(8).multiply(sValue.multiply(sValue)).add(BigInteger.*valueOf*(1)))).toBigInteger();

}

}

**private** **static** **boolean** verifyTriangularSquare(BigInteger s, BigInteger t)

{

BigInteger a = BigInteger.*valueOf*(2).multiply(s).multiply(s);

BigInteger b = t.multiply(t).add(t);

**return** a.equals(b);

}

**private** **static** **final** BigDecimal ***SQRT\_DIG*** = **new** BigDecimal(150);

**private** **static** **final** BigDecimal ***SQRT\_PRE*** = **new** BigDecimal(10).pow(***SQRT\_DIG***.intValue());

/\* Private utility method used to compute the square root of a BigDecimal.\*/

**private** **static** BigDecimal sqrtNewtonRaphson (BigDecimal c, BigDecimal xn, BigDecimal precision){

BigDecimal fx = xn.pow(2).add(c.negate());

BigDecimal fpx = xn.multiply(**new** BigDecimal(2));

BigDecimal xn1 = fx.divide(fpx,2\****SQRT\_DIG***.intValue(),RoundingMode.***HALF\_DOWN***);

xn1 = xn.add(xn1.negate());

BigDecimal currentSquare = xn1.pow(2);

BigDecimal currentPrecision = currentSquare.subtract(c);

currentPrecision = currentPrecision.abs();

**if** (currentPrecision.compareTo(precision) <= -1){

**return** xn1;

}

**return** *sqrtNewtonRaphson*(c, xn1, precision);

}

/\* Uses Newton Raphson to compute the square root of a BigDecimal.\*/

**public** **static** BigDecimal bigSqrt(BigDecimal c){

**return** *sqrtNewtonRaphson*(c,**new** BigDecimal(1),**new** BigDecimal(1).divide(***SQRT\_PRE***));

}

}

Square Triangular : 36 SquareRoot: 6 Triangle: 8 t/s Ratio: 1.333333333333333333333333333333333 verified?: true uGenerator: 3 vGenerator: 2

Square Triangular : 41616 SquareRoot: 204 Triangle: 288 t/s Ratio: 1.411764705882352941176470588235294 verified?: true uGenerator: 17 vGenerator: 12

Square Triangular : 55420693056 SquareRoot: 235416 Triangle: 332928 t/s Ratio: 1.414211438474870017331022530329289 verified?: true uGenerator: 577 vGenerator: 408

Square Triangular : 98286503002057414584576 SquareRoot: 313506783024 Triangle: 443365544448 t/s Ratio: 1.414213562371500186977083668114926 verified?: true uGenerator: 665857 vGenerator: 470832

Square Triangular : 309127573515950117423442905473334441338685531136 SquareRoot: 555992422174934068969056 Triangle: 786292024016459316676608 t/s Ratio: 1.414213562373095048801687824916866 verified?: true uGenerator: 886731088897 vGenerator: 627013566048

Square Triangular : 3057915414651492622161675552491043409905761391166783571596731306159201431935531704257336356540416 SquareRoot: 1748689627878970586010425926035281459718982106304 Triangle: 2473020588127600939387543243786675530709484249088 t/s Ratio: 1.414213562373095048801688724209698 verified?: true uGenerator: 1572584048032918633353217 vGenerator: 1111984844349868137938112

Square Triangular : 299227093861222721896171967385272039149613186574525328415889329765745335174235644147024020336537274912145702502039856893701098604266445640430763862425589203448396777849818977173988377193152659456 SquareRoot: 17298181807959549882659272694507430758694930035994171324093552383123282847262734884162646088415616 Triangle: 24463323317211940977293404419928347279246091129334268572773850449273611455484253634058690852323328 t/s Ratio: 1.414213562373095048801688724209698 verified?: true uGenerator: 4946041176255201878775086487573351061418968498177 vGenerator: 3497379255757941172020851852070562919437964212608

Square Triangular : 2865179318420255767641041534970326043260818197657906063678551075465860445443459628276129539991106305543195897602468577735720511044798929778609299809080265599679387523995050332634187486120236457469232955240651426956869725486113422934259955622709719893429103742743544508531366000288344882034405659440355046365358348985325359287081297639079856640185669988838417503762716717471495234388809547776 SquareRoot: 1692684057472113065580244544206382206592298598800215212040212244608727820167760432723045923521196141697556515246587635378663538602471805309458536620540130226387527560540974127384423795772481991424 Triangle: 2393816750889781775169375739082176313196905492596202627327114638125962681393885153176192162692298199297165620016318855149608788834131565123446110899404713627587174222798551817391907017545221275648 t/s Ratio: 1.414213562373095048801688724209698 verified?: true uGenerator: 48926646634423881954586808839856694558492182258668537145547700898547222910968507268117381704646657 vGenerator: 34596363615919099765318545389014861517389860071988342648187104766246565694525469768325292176831232

Square Triangular : 262696080854501164531432531781998878049980697473067180753004034091101299937641530750924534011756792933411975683214569080015684222528473399732822269667897664519396704802676547300703288132490340831795342018397595284579757222053171834757244717486887055307160496378690862875511703492735277074051477318244678602776019408991885564583430201030962518879605873759103789263573096291159044883542769124371334305157333976405953568225235223342546743093129400286165664787009826287095498693495314336058881516188520320366477082449553758763912875361667692730191129920908779014287751380954586324902027142580214293254217025087681688200995125437010093617036665842485392811933918843479012957668667904689694390537640718584577684065592525105684935197227737457838407459054225000305150240492816450439287668736 SquareRoot: 16207901802963305358953104995107962944580288226360273862214792773825746520390666916181487945905521185480050502718396729968222061511921853590431963883231862854549365739571307965640599553021472637177206503147912510401725225600535967037936143318401223620375536048011185340256710462785907579977304036581162417504828560754609799801583561441289258661703863156236214119175696133641980180747668153856 Triangle: 22921434547362046141128332279762608346086545581263248509428408603726883563547677026209036319928850444345567180819748621885764088358391438228874398472642124797435100191960402661073499888961891659753863641925211415654957803888907383474079644981677759147432829941948356068250928002306759056275245275522840370922866791882602874296650381112638853121485359910707340030101733739771961875110476382208 t/s Ratio: 1.414213562373095048801688724209698 verified?: true uGenerator: 4787633501779563550338751478164352626393810985192405254654229276251925362787770306352384325384596398594331240032637710299217577668263130246892221798809427255174348445597103634783814035090442551297 vGenerator: 3385368114944226131160489088412764413184597197600430424080424489217455640335520865446091847042392283395113030493175270757327077204943610618917073241080260452775055121081948254768847591544963982848

Square Triangular : 2208295388682067625147422910489195837729489712662773371888389233825965155296821934704568136182038030074931115835542340583499742567952783667393137926901135173397483339705457737327333059673526919139632089590740313188588754437724101875046338614958517783855312279405452783671742928207565313340568563810748739317500491466212387818892163846203275378714819605878829783338552402672002279836080354098751940317206604228798755111651567008270077717785613272692493610115661667067362227237242716570477686159316265558966896329184084465268142575334037882090570122208983252239042227955971611175302491317790514214251700358668203459806768472326900824302107597941286610488045801726075304338264199889830214336238328116637450088538492305757025720276675713741640624418560359073232363588913910167652397693291530352677059245465196975095459949322713606828347563521468023145564349114457986024645576011259013433181827330227343288029379941079492015559531345054643024176451976324500804673889210411130546179306976958616336101438626189852027458863374681356320139745300175439602696146794736859536957353715284807269943030055906312699755981651580050568156261346298328213860980955907525048858277054781408408927836622575512988891708459864402847293678960980942168590349371782881554960496893144122601507960534423061078177201775755729721504679754907105805197279804421539924562078795687378234527862289603448699274027390728070159794303954132920056628458525316892721367868047457880446416649735828114355070945206666998469786773309228667855634505346292141164334053160600085200225160922115953224715514066649481216 SquareRoot: 1486033441306778810926270962584569161654893913314383171171780071875425936026576415095570267169145668695769959434426327408764598604387004080826207364946447825722608444217902053292035518479174723617645682864960623726773337793470268649277425074196386663873709664623368981396501970933468744030283498926010992706031733306114311271873052540329252073646027690073189119594818276863521830397436600095566618935397906874532681492819439519331758432057294644059200167739206662603650183205905354433088423438061376729616147080974895771068867591060476159846259964143617553541819014836648119063351536848673877521446043962096175802966762472293797772473063553835938528554091193675355966450436814968613773462511392769350928301883609593401182527687056602335178566342062906280376220246110161042076156283904 Triangle: 2101568646836009316251460254255991024399845579784537446024032272728810399501132246007396272094054343467295805465716552640125473780227787197862578157343181316155173638421412378405626305059922726654362736147180762276638057776425374678057957739895096442457283971029526903004093627941882216592411818545957428822208155271935084516667441608247700151036846990072830314108584770329272359068342152994970674441258671811247628545801881786740373944745035202289325318296078610296763989547962514688471052129508162562931816659596430070111303002893341541841529039367270232114302011047636690599216217140641714346033736200701453505607961003496080748936293326739883142495471350747832103661349343237517555124301125748676621472524740200845479481577821899662707259672433800002441201923942531603514301349888 t/s Ratio: 1.414213562373095048801688724209698 verified?: true uGenerator: 45842869094724092282256664559525216692173091162526497018856817207453767127095354052418072639857700888691134361639497243771528176716782876457748796945284249594870200383920805322146999777923783319507727283850422831309915607777814766948159289963355518294865659883896712136501856004613518112550490551045680741845733583765205748593300762225277706242970719821414680060203467479543923750220952764417 vGenerator: 32415803605926610717906209990215925889160576452720547724429585547651493040781333832362975891811042370960101005436793459936444123023843707180863927766463725709098731479142615931281199106042945274354413006295825020803450451201071934075872286636802447240751072096022370680513420925571815159954608073162324835009657121509219599603167122882578517323407726312472428238351392267283960361495336307712

Square Triangular : 156050192757583492041585597181180844750441515170255283282290848356200087767760156235350358508153967677787690654993627199341183776880959054235384148586516886077087531087108399352571767489674606015969876815534398860576370250208544971041621323601469388051199794474165833488808763257276765666011656767052252423501548901975290686988842189099747477289147322060615581650073875196229428338619667292668156609335338143058621316603159107569738393474055535197807156432496732016937872131724678600210215195952035772460718976768558103751764801499806196340683375018398075975071616553263655278494859887095618469247653902683704446401991896806762772970420820640601145130228301526821470516668283737073059866513567191370081875338939074791143497568199346699962346027102809718193198608900353629184874786575527973428686271651164899685332058038086772724935944626160169624071671531426737066122036890449354550009650839314146289143553723397379416287574109244175831794846493708174068660627982282140379325814003817501675045434479216378017774053739107412910640514300303485544459060521211485218329749592229845467336593986926787113003219389956044292327817026688398682152639761752054413594225891515867998781195303857186484346679285154278940409080741220572260283055873465778087395428159653541400836281284605663672037395133457201765965958661645012529790207282998661246039549619547103487628975593161821096368473165318727007047609191628781322867207526479202107209098007463047117726002637084800295532312807997295038558876551444019249285471016016493762981899836561821364898065284604357219547011428189177249621448733300193779843258000566361737402724524426105498337010920729189836435282266551480227094761152351194196203992839720895769896187748983062810894919486878066719506835820088584824882871751972370944033528211845706888960930666466215730118105905770524072368723570129876702161869854735457312151810248443776163235027305189306875702487407619581857163590939072328260366199036001753959423539158224317865640151908250107186152569523521757597222471122005859454352569737562691075999285097845504717911101316383212077497515587691476256129449262998876375957087363644931874207284153310878545055019212457840590217144167222324969155923644460819066543225372204050919144121556102323427133655363374457825019637725742888226793000642064296000222170696901564591192451646893650814565583154676649804501695009934160340409866449490903607540079995191875737854986169865151352769355749881321084577848675336685963804251720362378656540682940104748975207944788069193198234680857759851901911893047694143144226327126281979063404665483490407655204370557466193390218221411044139578732308824190234033174699038240068111212798412777739798280180389258925166829532358897367746972645495541216433155966513281946654811623937797164630241922599758581729204515819523572190067422494750550038829747530855959326988247182678543362808104708674791483299766985703690314710093599312346335107663704797348932989652930234382374916387454776087649554682919056058625772333039029425448338513342782868894395412472606623389764851441145734454980934447751905741253054902530460892039489859715125658722025735957857701402010675848576761856 SquareRoot: 12492005153600581669973991678150570808253223832243720091999405312529376084211662438172848742824395170804706001322327814706452566139189608934242278331643887865525422630365039101621299616236372613960339556531897024876218730490708933166065664197877670674876531279696486034040555959172078238806324674297605745533727744618728955077825502053504524090313459645473040770720074400830979282096799010122032752595393201359622692037126352280547640940228448408937201668484791397621629340170449109912756807328724197492711233610505967491631477680248094809736310700142057857644240191135361052270397390261535223346941635173995447734550589188845963131515710740724413747399277370670835131327040086587508141836639967767967426405481003192878600422243232041814645010957935467483400077240083141137500635491344844577291573754340762747421596157711042506083257962145053244868727579309046784202097839984192659342358917041486307110999561088508134795204909007289882339985618470262240376685989415655802794109532466185175657676886395868558167518428061699786119539696222089111353821966106341894033955476555612399572559917748878954393732444700699577611812028789213343199183097788070924182263139699127818972845634123095977998063279204521997955953528699980814295010858098412729909815334009273428919238541677706417818344175128158034057018675637158716535927620830881770912683659676422603156483415134302480959714122535800117040544418152477947704614763539288723995781666998627207681092452209284440344355518778169282291152279701996859185321974890705578813619391390282519083858004033171817804966173276178978816 Triangle: 17666363109456541001179383283913566701835917701302186975107113870607721242374575477636545089456304240599448926684338724667997940543622269339145103415209081387179866717643661898618664477388215353117056716725922505508710035501792815000370708919668142270842498235243622269373943425660522506724548510485989914540003931729699102551137310769626203029718556847030638266708419221376018238688642832790015522537652833830390040893212536066160621742284906181539948880925293336538897817897941732563821489274530124471735170633472675722145140602672303056724560977671866017912337823647772889402419930542324113714013602869345627678454147778615206594416860783530292883904366413808602434706113599118641714689906624933099600708307938446056205762213405709933124995348482872585858908711311281341219181546332242821416473963721575800763679594581708854626780508171744185164514792915663888197164608090072107465454618641818746304235039528635936124476250760437144193411615810596006437391113683289044369434455815668930688811509009518816219670906997450850561117962401403516821569174357894876295658829722278458159544240447250501598047853212640404545250090770386625710887847647260200390866216438251267271422692980604103911133667678915222778349431687847537348722794974263052439683975145152980812063684275384488625417614206045837772037438039256846441578238435372319396496630365499025876222898316827589594192219125824561278354431633063360453027668202535141770942944379663043571333197886624914840567561653335987758294186473829342845076042770337129314672425284800681601801287376927625797724112533195849728 t/s Ratio: 1.414213562373095048801688724209698 verified?: true uGenerator: 4203137293672018632502920508511982048799691159569074892048064545457620799002264492014792544188108686934591610931433105280250947560455574395725156314686362632310347276842824756811252610119845453308725472294361524553276115552850749356115915479790192884914567942059053806008187255883764433184823637091914857644416310543870169033334883216495400302073693980145660628217169540658544718136684305989941348882517343622495257091603763573480747889490070404578650636592157220593527979095925029376942104259016325125863633319192860140222606005786683083683058078734540464228604022095273381198432434281283428692067472401402907011215922006992161497872586653479766284990942701495664207322698686475035110248602251497353242945049480401690958963155643799325414519344867600004882403847885063207028602699777 vGenerator: 2972066882613557621852541925169138323309787826628766342343560143750851872053152830191140534338291337391539918868852654817529197208774008161652414729892895651445216888435804106584071036958349447235291365729921247453546675586940537298554850148392773327747419329246737962793003941866937488060566997852021985412063466612228622543746105080658504147292055380146378239189636553727043660794873200191133237870795813749065362985638879038663516864114589288118400335478413325207300366411810708866176846876122753459232294161949791542137735182120952319692519928287235107083638029673296238126703073697347755042892087924192351605933524944587595544946127107671877057108182387350711932900873629937227546925022785538701856603767219186802365055374113204670357132684125812560752440492220322084152312567808

(10 second run time)

Java 1.6, Eclipse

2015 RMBP, Intel Core i7, 8GB RAM