De Bruijn Sequences

Last year I wrote an article about 4-digit PIN codes. It became quite popular, and I even got asked to give a TED[X] talk about it which you can watch here (shameless plug).

The basis of the talk was that people are very predictable in the selection of their codes. Mathematically, whilst there are 10,000 ways that the digits 0-9 can be arranged into a four digit PIN, people's selections are not random. Only 426 distinct codes are needed to guess over half the PINs in use!

Perfect World

Imagine, instead of being predictable, that people selected their codes entirely at random. If you wanted to guess the PIN of a four digit combination lock, you might have to walk through all 10,000 four digit combinations. Since there are four digits in each number, the worst case is that you'd have to type 40,000 key presses, and on average it would take you half this number.

That's a lot of key presses.

Can you do better? Can you do it with less key presses?

Well, if the lock is smart, no, you can't do better. But some systems that accept PINs are less sophisticated. These less secure mechanisms don't quantize the inputs in batches of four (or however long the code is), but instead simply look at just the *last four keys pressed.*

An example of this in use is shown below. Imagine you entered the six digits **123479** into one of these systems. As the system examines just the last four digits entered, this string would give you three distinct (overlapping) unlock attempts.

This simpler system has the advantage of not having to worry about what state you are in when you start, or having to code in time-outs into the entry system. Without this simplification, if your code was **1234**, but just before you tried to enter it, unbeknownst to you, someone else had entered **99** on the keypad. When you entered your PIN, the lock would interpret this as **9912** *(fail)*, and then **34** *(still a fail and waiting for more digits before telling you it failed)*. Examining just the last four digits solves this problem.

If our system supports overlapping numbers we can be more efficient in guessing. We can create an input stream that goes through all permutations, but requires less key presses. The question is, how much more efficient? What is the shortest sequence of numbers that we can go through in order to ensure that all possible combinations of the digits are seen? Is it possible to create a sequence that does not repeat any sub-sequence of codes?

Nicolaas Govert de Bruijn

The mathematics, number theory, combinatronics and logic of these types of problems were studied extensively by a Dutch Professor called *Nicolaas Govert de Bruijn* (9 July 1918 – 17 February 2012).

Sequences of these numbers are named after him as *De Bruijn Sequences*.

The quick answer is that, yes, it is possible to make a non-repeating sequence of numbers that covers ever sub-sequence internally, just once. However, before we look at the PIN number solution, let's look at some simpler versions of the problem …

Image Credit: Konrad Jacobs

Simpler Sequences

De Bruijn sequences can be described by two parameters:

- k the number of entities in the alphabet *e.g.* {0,1,2,3,4,5,6,7,8,9} for k=10
- n the order (length of sub-sequence required) *e.g.* n=4 for a four digit long PIN.

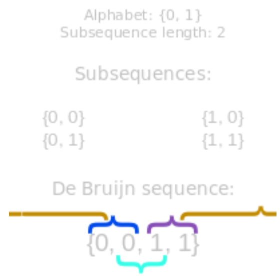
These are typically describe by the representation $B(k,n)$.

For our PIN example, the notation would be $B(10,4)$

B(2,2)

Let's start with a really simple example: $B(2,2)$

We'll use the dictionary {0,1} for the possible values. We want to generate a string that contains substrings each possible combination of two digits. Here is a solution:



0011

The first two digits give us **00**, the next two **01**, then **11**. To get to **10**, we need to *'Wrap Around'* taking the last digit from the string, and the first digit. (If this is not appropriate to do, like the key-press example, then we can simply append the first character from the string to the end to make **00110**).

NOTE: There can be multiple De Bruijn solutions to any problem. You can easily see this using even simple rotations of the string. Since every adjacent pair of digits in this string is unique, it does not matter what the starting position is. As *k* and *n* increase, the number of possible solutions grows rapidly. The hairy looking equation on the left shows the number of distinct solutions.

B(2,3)

Here's a solution for $B(2,3)$:

00010111

Starting from the front, we have **000**, **001**, **010**, **101**, **011**, **111**, then staring to wrap around, we have **110** and finally **100**. All eight possible combinations are present in this string.

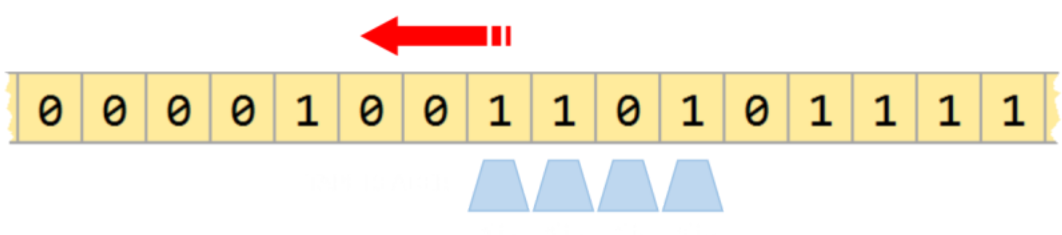
B(2,4)

Here's a solution for $B(2,4)$:

0000100110101111

You can see we have **0000**, **0001**, **0010**, **0100**, **1001** …

And here it is as a pretty ribbon. We can now hint at some of the awesome potential uses for these sequences. Imagine this *De Bruijn* sequence is written onto a looped tape, which passes past a reader head.



Each increment of the tape one position along gives a unique output. We have found an efficient mechanism of encoding position as there is a distinct code for any contiguous set of four digits (in this case, as *n*=4). How cool is that?

This sequence also walks through all permutations of combinations from **0000**-**1111**. If you are a software engineer or tester, I'm sure this is giving you interesting ideas for how you can implement this as test cases to walk through all binary inputs to your functions. Each shift of one bit gives a unique binary word (of length of the substring size *n*), which does not repeat, goes through each possible number, then returns to the start again.

B(2,6)

Skipping ahead a couple, here is a solution for $B(2,6)$:

0000001000011000101000111001001011001101001111010101110110111111

B(6,2)

Of course, we can also change the size of the dictionary. In this example rather than simple binary *k*=2, I've increased this to *k*=6 to use possible values {0,1,2,3,4,5}. Here is the output for $B(6,2)$:

001020304051121314152232425334354455

Reading this from left to right we can see that we have: **00**, **01**, **10**, **02**, **20** … **41**, **15** … **33**, **34** … **55**, **50**

B(5,3)

One last example, here's a sequence for $B(5,3)$:

000100200300401101201301402102202302403103203303404104204304411121131141
22123124132133134142143144222322423323424324433343444

Back to PIN

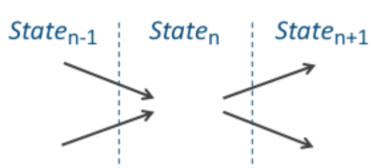
Returning to our PIN cracking, the $B(10,4)$ is 10,000 digits long, and contains every single substring for each combination of digits. As we have to type in this string, the concept of *Wrap-Around* is meaningless, so rather then wrap-around, we simply add the first three numbers again to the end of the strong. So, the maximum number of keypress events is 10,003. Far fewer than the 40,000 that would be required if typing them in completely!

For no useful reason at all, here are all 10,003 digits of this string:

```
0000100020003000400050006000700080009001100120013001400150016001700180019002100220023002400250026002700280029003100320033003400350036003700380039004100420043004400450046004700480049005100520053005400550056005700580059006100620063006400650066006700680069007100720073007400750076007700780079008100820083008400850086008700880089009100920093009400950096009700980099
0101020103010401050106010701080109011101120113011401150116011701180119012101220123012401250126012701280129013101320133013401350136013701380139014101420143014401450146014701480149015101520153015401550156015701580159016101620163016401650166016701680169017101720173017401750176017701780179018101820183018401850186018701880189019101920193019401950196019701980199
020203020402050206020702080209021102120213021402150216021702180219022102220223022402250226022702280229023102320233023402350236023702380239024102420243024402450246024702480249025102520253025402550256025702580259026102620263026402650266026702680269027102720273027402750276027702780279028102820283028402850286028702880289029102920293029402950296029702980299
03030403050306030703080309031103120313031403150316031703180319032103220323032403250326032703280329033103320333033403350336033703380339034103420343034403450346034703480349035103520353035403550356035703580359036103620363036403650366036703680369037103720373037403750376037703780379038103820383038403850386038703880389039103920393039403950396039703980399
0404050406040704080409041104120413041404150416041704180419042104220423042404250426042704280429043104320433043404350436043704380439044104420443044404450446044704480449045104520453045404550456045704580459046104620463046404650466046704680469047104720473047404750476047704780479048104820483048404850486048704880489049104920493049404950496049704980499
050506050705080509051105120513051405150516051705180519052105220523052405250526052705280529053105320533053405350536053705380539054105420543054405450546054705480549055105520553055405550556055705580559056105620563056405650566056705680569057105720573057405750576057705780579058105820583058405850586058705880589059105920593059405950596059705980599
06060706080609061106120613061406150616061706180619062106220623062406250626062706280629063106320633063406350636063706380639064106420643064406450646064706480649065106520653065406550656065706580659066106620663066406650666066706680669067106720673067406750676067706780679068106820683068406850686068706880689069106920693069406950696069706980699
070708070907110712071307140715071607170718071907210722072307240725072607270728072907310732073307340735073607370738073907410742074307440745074607470748074907510752075307540755075607570758075907610762076307640765076607670768076907710772077307740775077607770778077907810782078307840785078607870788078907910792079307940795079607970798 0799
0808090811081208130814081508160817081808190821082208230824082508260827082808290831083208330834083508360837083808390841084208430844084508460847084808490851085208530854085508560857085808590861086208630864086508660867086808690871087208730874087508760877087808790881088208830884088508860887088808890891089208930894089508960897089808 99
09091109120913091409150916091709180919092109220923092409250926092709280929093109320933093409350936093709380939094109420943094409450946094709480949095109520953095409550956095709580959096109620963096409650966096709680969097109720973097409750976097709780979098109820983098409850986098709880989099109920993099409950996099709980999
1111211131114111511161117111811191122112311241125112611271128112911321133113411351136113711381139114211431144114511461147114811491152115311541155115611571158115911621163116411651166116711681169117211731174117511761177117811791182118311841185118611871188118911921193119411951196119711981199
1212131214121512161217121812191222122312241225122612271228122912321233123412351236123712381239124212431244124512461247124812491252125312541255125612571258125912621263126412651266126712681269127212731274127512761277127812791282128312841285128612871288128912921293129412951296129712981299
13131413151316131713181319132213231324132513261327132813291332133313341335133613371338133913421343134413451346134713481349135213531354135513561357135813591362136313641365136613671368136913721373137413751376137713781379138213831384138513861387138813891392139313941395139613971398 1399
141415141614171418141914221423142414251426142714281429143214331434143514361437143814391442144314441445144614471448144914521453145414551456145714581459146214631464146514661467146814691472147314741475147614771478147914821483148414851486148714881489149214931494149514961497149814 99
15151615171518151915221523152415251526152715281529153215331534153515361537153815391542154315441545154615471548154915521553155415551556155715581559156215631564156515661567156815691572157315741575157615771578157915821583158415851586158715881589159215931594159515961597159815 99
161617161816191622162316241625162616271628162916321633163416351636163716381639164216431644164516461647164816491652165316541655165616571658165916621663166416651666166716681669167216731674167516761677167816791682168316841685168616871688168916921693169416951696169716981699
1717181719172217231724172517261727172817291732173317341735173617371738173917421743174417451746174717481749175217531754175517561757175817591762176317641765176617671768176917721773177417751776177717781779178217831784178517861787178817891792179317941795179617971798 1799
18181918221823182418251826182718281829183218331834183518361837183818391842184318441845184618471848184918521853185418551856185718581859186218631864186518661867186818691872187318741875187618771878187918821883188418851886188718881889189218931894189518961897189818 99
1919221923192419251926192719281929193219331934193519361937193819391942194319441945194619471948194919521953195419551956195719581959196219631964196519661967196819691972197319741975197619771978197919821983198419851986198719881989199219931994199519961997199819 99
22223222422252226222722282229223322342235223622372238223922432244224522462247224822492253225422552256225722582259226322642265226622672268226922732274227522762277227822792283228422852286228722882289229322942295229622972298 2299
2323242325232623272328232923332334233523362337233823392343234423452346234723482349235323542355235623572358235923632364236523662367236823692373237423752376237723782379238323842385238623872388238923932394239523962397239823 99
24242524262427242824292433243424352436243724382439244324442445244624472448244924532454245524562457245824592463246424652466246724682469247324742475247624772478247924832484248524862487248824892493249424952496249724982499
252526252725282529253325342535253625372538253925432544254525462547254825492553255425552556255725582559256325642565256625672568256925732574257525762577257825792583258425852586258725882589259325942595259625972598 2599
2626272628262926332634263526362637263826392643264426452646264726482649265326542655265626572658265926632664266526662667266826692673267426752676267726782679268326842685268626872688268926932694269526962697269826 99
27272827292733273427352736273727382739274327442745274627472748274927532754275527562757275827592763276427652766276727682769277327742775277627772778277927832784278527862787278827892793279427952796279727982799
282829283328342835283628372838283928432844284528462847284828492853285428552856285728582859286328642865286628672868286928732874287528762877287828792883288428852886288728882889289328942895289628972898 2899
2929332934293529362937293829392943294429452946294729482949295329542955295629572958295929632964296529662967296829692973297429752976297729782979298329842985298629872988298929932994299529962997299829 99
33334333533363337333833393344334533463347334833493354335533563357335833593364336533663367336833693374337533763377337833793384338533863387338833893394339533963397339833 99
3434353436343734383439344434453446344734483449345434553456345734583459346434653466346734683469347434753476347734783479348434853486348734883489349434953496349734983499
35353635373538353935443545354635473548354935543555355635573558355935643565356635673568356935743575357635773578357935843585358635873588358935943595359635973598 3599
363637363836393644364536463647364836493654365536563657365836593664366536663667366836693674367536763677367836793684368536863687368836893694369536963697369836 99
3737383739374437453746374737483749375437553756375737583759376437653766376737683769377437753776377737783779378437853786378737883789379437953796379737983799
38383938443845384638473848384938543855385638573858385938643865386638673868386938743875387638773878387938843885388638873888388938943895389638973898 3899
393944394539463947394839493954395539563957395839593964396539663967396839693974397539763977397839793984398539863987398839893994399539963997399839 99
444454446444744484449445544564457445844594465446644674468446944754476447744784479448544864487448844894495449644974498 4499
45454645474548454945554556455745584559456545664567456845694575457645774578457945854586458745884589459545964597459845 99
4646474648464946554656465746584659466546664667466846694675467646774678467946854686468746884689469546964697469846 99
47474847494755475647574758475947654766476747684769477547764777477847794785478647874788478947954796479747984799
484849485548564857485848594865486648674868486948754876487748784879488548864887488848894895489648974898 4899
4949554956495749584959496549664967496849694975497649774978497949854986498749884989499549964997499849 99
55556555755585559556655675568556955765577557855795586558755885589559655975598 5599
565657565856595666566756685669567656775678567956865687568856895696569756985699
5757585759576657675768576957765777577857795786578757885789579657975798 5799
58585958665867586858695876587758785879588658875888588958965897589858 99
595966596759685969597659775978597959865987598859895996599759985999
666676668666967776778677967876788678967976798 6799
6767686769677767786779678767886789679767986799
68686968776878687968876888688968976898 6899
6969776978697969876988698969976998 6999
777787779778877897798 7799
78787978887889789878 99
797988798979987999
88889889989899999000
```

How do these things work?

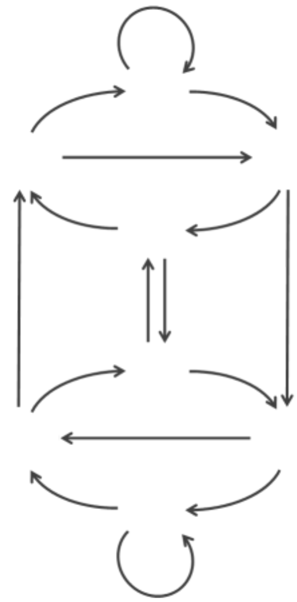
I glossed over how to calculate a De Bruijn sequence earlier. I'll make up for it now by explaining the principles for how these things work. The concept is not complicated, just tedious; things that computers can do easily. We're going to look at this problem using *Directed Graphs*.



For this example, I'm going to use $B(2,3)$. This sequence is simple enough to get the concept without being too complex and busy.

Every sub-string in the sequence can be described as a *node* on a directed graph. From each node, it is possible to add either a **0** or a **1** to make the next number in the sequence.

Similarly, each state/node on the graph could have been formed from adding a digit from one of two earlier states.



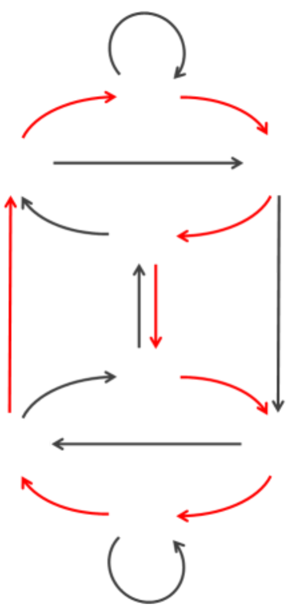
First we write down all the nodes that are possible using our data dictionary.

In this case these are **000** - **1111**.

Then we connect the nodes with lines showing which possible next states are achievable from each location. Each node has two outbound links corresponding to the addition of a **1** or **0** from that state.

These edges are directed. They have a direction (depicted by the arrows)

Note - For States **000** and **111**, the addition of a **0** or **1**, respectively, takes you back to the same node.



In order to make sure every substring is present in the solution, we need to make sure each node is passed through once (and only once). We need to trace a path through the graph (following the arrows) to connect the nodes.

A path that traverses a graph and visits each node exactly once is called a Hamiltonian Path.

One such Hamiltonian Path on the $B(2,3)$ graph is shown in red on the left.

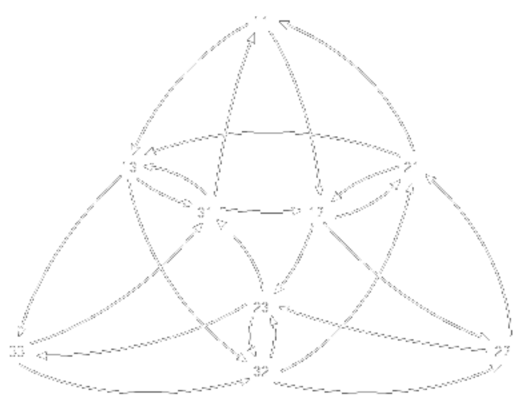
This highlights why there are many different solutions (not just rotations). There is more than one way to walk through this graph.

The way to create a De Bruijn sequence is to find a Hamiltonian Path through a graph their nodes.

As the values of *k* and *n* increase, so does the complexity of the graph, but the principle is the same.

Here's a slightly more complex one using a dictionary {1,2,3) for a sequence $B(3,2)$

Find a path through the graph the passes through each node just once, and you have your solution.



Other uses

We already hinted that De Bruijn sequences can be used for encode/decode positions, and that they can be used by savvy computer programmers. What are some other applications?

I've seen magicians use a similar principle in various card tricks. By pre-arranging (loading) a deck with a known sequence of red and blacks cards in a De Bruijn sequence, it allows him/her to know what the position is, and thus, what the next card will be. Using a binary encoding of the red/black cards to generate a unique number, which can then, be used to encode the value of what the next card will be.

Because of the wrap-around nature of De Bruijn sequences, a loaded deck can be cut as many times as desired without upsetting or disturbing the encoding. I don't want to link directly examples, as I don't want to ruin the tricks for others that might be performing them, but I'm sure if you know how to use a web search tool, you can find some strategies.

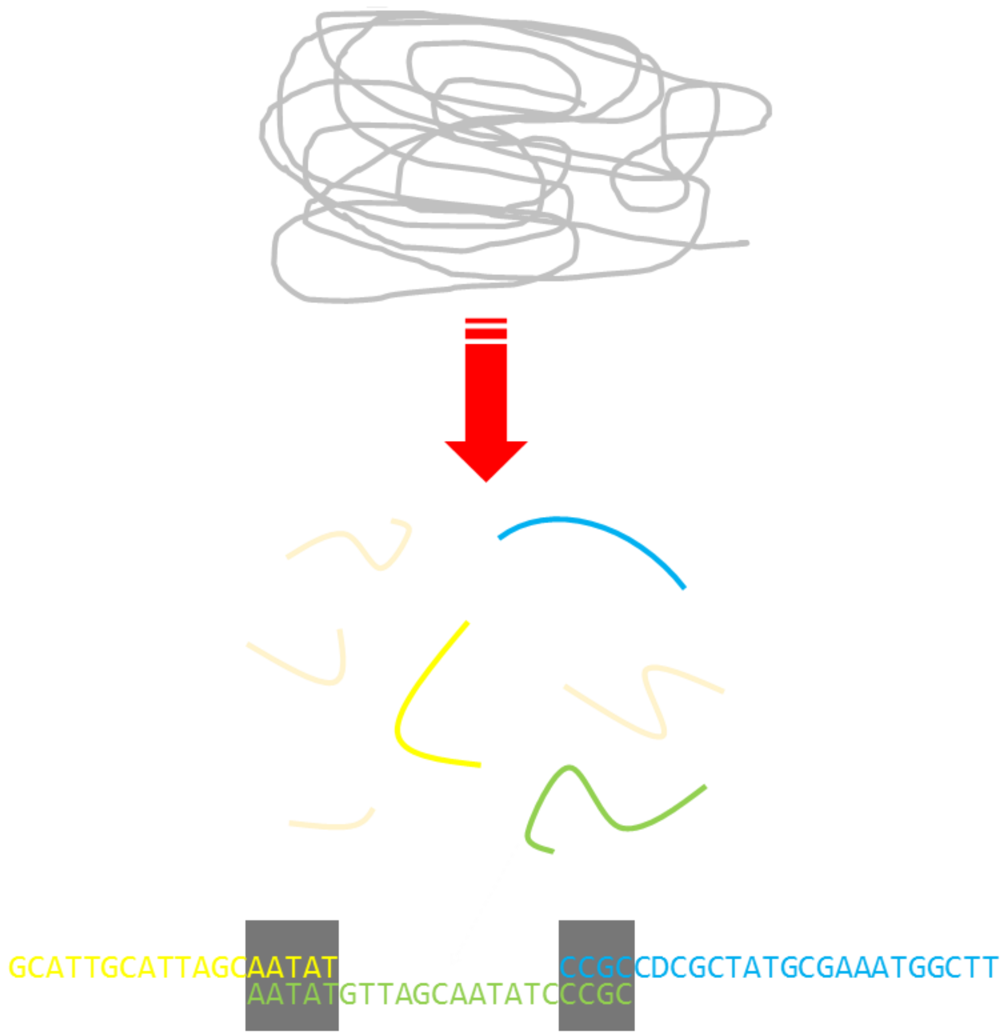
DNA

The concept of Hamilton cycles and De Bruijn are used extensively in modern DNA sequencing techniques.

A large DNA chain can be broken up into smaller pieces (The smaller pieces being easier to process and sequence). Then, the results of these smaller pieces can be 'glued' back together like some kind of giant jigsaw because the individual sub-pieces contain overlaps with other partial strings.

This technique is called **Shotgun Sequencing**.

Below you can see a representation of a long chain of DNA. This is smashed into smaller sub-pieces (of different sizes), which are easier to classify. The classified pieces can then be joined back together to form the complete sequence by looking at the overlaps between the substrands.



In some cases, the shotgun sequencing technique generates multiple canonical sequences. Of course, only one of these can be the real sequence. There are a variety of tests that can narrow down which of these sequences is the original one. Shotgun sequencing, although it does not *necessarily* define the exact sequence, greatly speeds up the process by reducing the number of possible candidates.

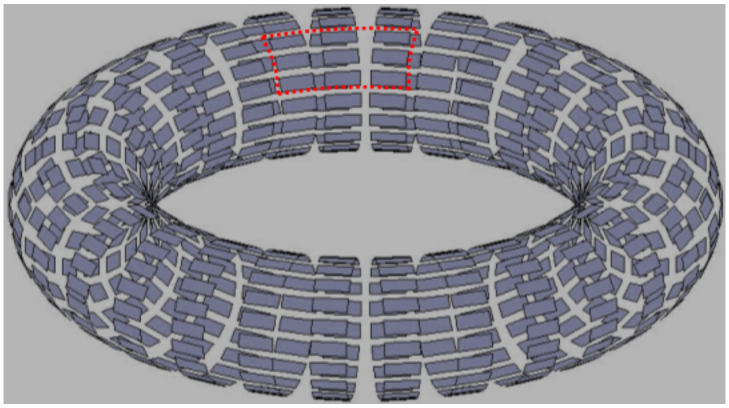
The advent of shotgun sequencing techniques advanced the initial mapping of the human genome by years, and it has provided biologists, geneticists, and doctors with a powerful new tool. The ability to sequence genetic data rapidly has potential benefits not only for life and health science professionals, but also for the public at large. It's pretty cool.

Chess

Computer programs that play chess make use of De Bruijn sequences. A chess board is conveniently *8 x 8* squares, and these can be represented by the numbers 0-63, which is a nice fit for a six bit long De Bruijn sequence.

De Bruijn Toroids

OK, prepare your mind for something cooler still. We can expand a De Bruijn one dimensional sequence into two dimensions! We call these Toroids because every row and column wraps around onto itself.



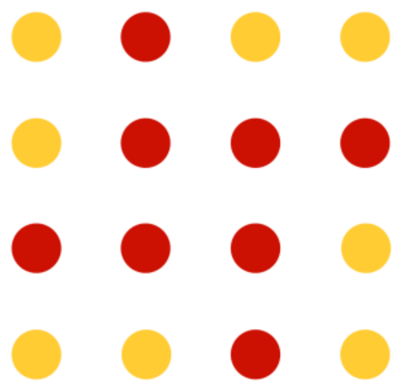
Sliding a window over this surface creates a unique matrix for a well-formed De Bruin array.

Things get much more complicated because not only do we need to specify the number of items in the dictionary, but also the two dimensions of the window.

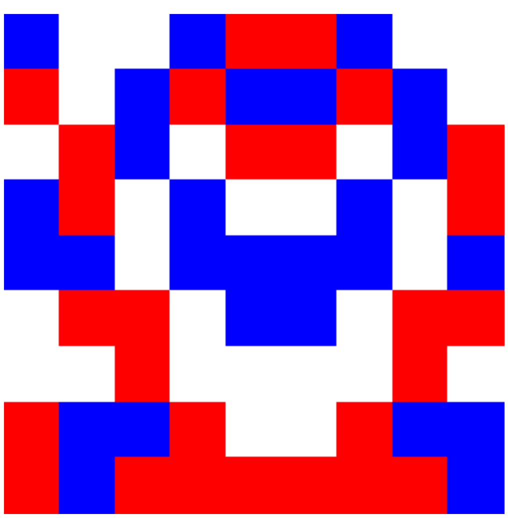
Below is a simple example of an array that has a dictionary of two {red, yellow}, and a (2 x 2) window:

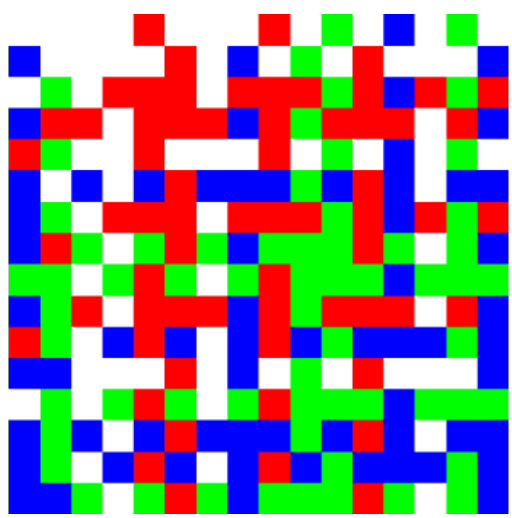
If you look carefully, you will see that every combination of red/yellow dots appears all possible combinations of (2x2) sub-matrices. Remember, you need to wrap-around (for instance to get a matrix containing four yellow dots you need the matrix that is the four corners wrapped around the outside!)

I'm sure you can see instant applications for something like this in determining (x,y) position. Each window is unique and determines coordinate position.



Of course, we're not restricted to using just a dictionary of two items. Below are two larger De Bruijn toroids. The one on the left has a dictionary size of three, and matrix window of (2 x 2). The one on the right has the same matrix window and this time has a dictionary of size four.





The grid on the left is (9 x 9) and that on the right is (16 x 16), allowing representation of all 256 possible combinations that a dictionary of order four into a (2 x 2) matrix. De Bruijn Toroids also don't have to be square; for instance there is a solution the four dictionary solution that fits in a matrix (8 x 32) instead of (16 x 16).

Of course there are multiple equivalent translations of these solutions as any combination of row or column shift is a valid solution. It's also possible to tesselate these tiles perfectly edge to make a map the repeats with the same order frequency.

Digital Paper

Anoto, a Swedish company, has invented, and has numerous patents, concerning the use of digital pens and paper. The Anoto concept is based around a distinct pattern of dots that can be printed onto paper stock. Using a special pen, which contains a small camera, the dot pattern can be read, and exact position on the paper determined.

Whilst not quite the same concept, the Anoto system works by small changes in the displacement in the position of dots from a nominal mean position.

<< Previous Article Next Article >>

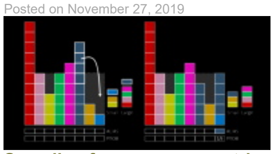
Want more? Here's four more random articles I wrote (Click here to refresh)

Posted on March 10, 2021

Painted Cube Puzzle
Yet another painted cube puzzle.

SHOW

Posted on November 27, 2019



Sampling from non-symmetric discrete outcomes
A super clever algorithm.

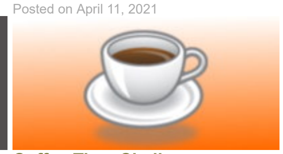
SHOW

Posted on December 30, 2018

Sudoku Solver
Online Sudoku Solver.

SHOW

Posted on April 11, 2021



Coffee Time Challenge
Two unknowns in arithmetic and geometric progressions.

SHOW

You can find a complete list of all the articles here. Click here to receive email alerts on new articles.

Buy me a coffee