

Re: Regularly populating the line – revised version

Source: <http://sci.tech-archive.net/Archive/sci.math.num-analysis/2005-01/0494.html>

From: Lord Snooty (*bonzo_at_dog.com*)

Date: 01/30/05

Date: Sun, 30 Jan 2005 21:37:07 GMT

I presume that you meant that a P is emitted at i when

$$((i - 1) * p) / n \neq (i * p) / n$$

where integer divide (rounding down) is used.

This is true, but I still can't prove that this condition is true a total of p times in n trials, for $(n \bmod p) \neq 0$.

LS

"Gordon Sande" <g.sande@worldnet.att.net> wrote in message news:PR6Ld.78622\$Qb.34951@edtnps89...

>

>

> *Lord Snooty wrote:*

>> *Due to a typo, one more time with feeling, and less C-stylistic too:*

>>

>> *for (i=a=0; i < n; i++) {*

>> *a = a + p;*

>> *if (a >= n) {*

>> *//fill at position i*

>> *a = a - n;*

>> *}*

>> *// else leave position i unfilled*

>> *}*

>>

>> *It's trivial to analyse this for $(n \bmod p) = 0$ of course.*

>> *I'm only interested in nonzero $(n \bmod p)$.*

>>

>

>

> *The first observation is that a is just $i*p \bmod n$.*

> *Then one wants to know what make the if condition true.*

> *It is true whenever the value of $i*p/n$ (integer divide)*

> *changes. Otherwise stated as $(i-i)*p/n \neq i*p/n$. This*

> *starts to look like the algorithm for drawing a diagonal*

> *line on an integer lattice, which is Bresenham's algorithm*

> *if you are doing computer graphics. A full citation*

> is left as an exercise for the reader. Some of the dates will
> be in the prehistoric period BG (Before Google).
>
> The curious handle does not suggest a strong academic interest
> so one might even guess that the original problem is computer
> graphics. The ancients did have considerable wisdom and the even
> more curious habit of recording much of it in books. So the
> obvious thing is for Lord Snooty to try reading some books.
>
>
>>
>> "Lord Snooty" <bonzo@dog.com> wrote in message news:...
>>
>>> It's not too uncommon to want to populate n bins with a given number p of
>>> objects, such that they are as regularly spaced as possible – we seek to
>>> avoid
>>> bunching, in other words. For example, if $n=8$ and $p=2$, then trivially we'd
>>> seek one of the cyclic permutations of $PxxxPxxx$, where P represents one of
>>> the
>>> p objects and x represents an unfilled bin.
>>>
>>> I came across the following very efficient algorithm recently, which works
>>> on
>>> all tested integer pairs (n,p) , $n \geq p$. Trouble is, I have difficulty
>>> proving
>>> that it will always emit exactly p objects. I'd appreciate any attempts at
>>> proving this. I'd also like to know if it has a name, because I've never
>>> come
>>> across it before. I can't find it in Knuth, for example, because I can't
>>> find
>>> a good name for it with which to do an index search. Here's the algorithm
>>> (C-code style):
>>>
>>> for ($i=a=0$; $i < n$; $i++$) {
>>> $a += p$;
>>> if ($a \geq n$) {
>>> // fill at position i
>>> $a -= n$;
>>> }
>>> // else leave bin unfilled
>>> }
>>>
>>>
>>
>>