
Subject: Re: Randomize array order

Posted by [David Streutker](#) on Fri, 27 Jul 2007 17:54:07 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Jul 27, 10:05 am, kuyper <kuy...@wizard.net> wrote:

> David Streutker wrote:

>> On Jul 27, 6:03 am, Allan Whiteford

> ...

>>> If you have a million elements then you have 1000000! (i.e. one million
>>> factorial) different ways to re-order the data. However, your seed is a
>>> 4 byte integer which can only take 2^{32} different values.

>

>>> Some messing about suggests that:

>

>>> $1000000! \approx 10^{5568636}$

>

>>> which means there are $\sim 10^{5568636}$ different ways to re-arrange your
>>> elements as opposed to the 4×10^9 values your seed can take.

>

>>> Thus, using any of the algorithms suggested you're only going to sample

>

>>> $10^{-5568625} \%$

>

>>> of the possible values. This is a really small number. It means that no
>>> matter how hard you try and how many times you do things you'll never be
>>> able to access anything but a tiny number of the possibilities without
>>> doing multiple shufflings - I think it's something like 618737
>>> sub-shufflings (i.e. $5568636 / 9$) but that could be wrong. However, that
>>> requires producing 618737 seeds per major-shuffle (and you can't use a
>>> generator based on a 4 byte seed to produce these seeds).

>

>>> But, since you're only going to be running the code 1000-10,000 times
>>> (which is much smaller than $4e9$) I guess everything will be ok. I don't
>>> know if anyone has studied possible correlations of results as a
>>> function of the very small number of seeds (compared to the data),
>>> whatever random number generator is used and the shuffling method.
>>> Presumably they have and presumably everything is ok. Does anyone know?

>

>>> Thanks,

>

>>> Allan

>

>> I'm not sure that I agree. Where in any of our algorithms are we
>> unable to access a (theoretically) possible outcome? As long as we
>> are able to randomly select any element of the array in each step, it
>> should work, right? (i.e., as long as the input array has fewer than
>> 2^{32} elements.) In your analysis, shouldn't we be using $(2^{32})^n$ for
>> the maximum possible number of randomly generated combinations, where

>> n is the number of steps/elements?
>
> No, because the entire sequence of numbers is uniquely determined by
> initial internal state of the generator. If you knew the algorithm
> used, and the internal state, that's all the information you'd need to
> predict, precisely, the entire sequence of numbers generated, no
> matter how long that sequence was. If the internal state is stored in
> a 32 bit integer, that means there's only 2^{32} possible different
> sequences.
>
>> From that fact, it can also be shown that every possible sequence must
>
> start repeating, exactly, with a period that is less than 2^{32} . If one
> of the possible sequences has starts repeating with a period T, then
> at least T-1 of the other possible sequences generate that same repeat
> cycle, with various shifts.
>
> There's a reason why these things are called PSEUDO-random number
> generators.

Interesting. I hadn't really thought it through before.

If there are only 2^{32} possible sequences, then why is the internal state characterized by a 36-element array?

```
IDL> test = randomu(seed)
IDL> help, seed
SEED      LONG      = Array[36]
```

Is it that there are only 2^{32} possible sequences available during any given session? With a new set being available in a different session?
