

That is not the algorithm I proposed

Source: <http://www.derkeiler.com/Newsgroups/sci.crypt/2004-11/1123.html>

From: Paul Tomkins (*tomkinsp_at_iinet.net.au*)

Date: 11/23/04

Date: Tue, 23 Nov 2004 16:26:57 +1100

It may not have been clear in this post or my initial post in the thread, but, yes, my algorithm does require sampling without replacement. You are certainly right to point that out.

I realise that `deckB[i] = deckA[rand[i]]` will not work unless `rand[i]` samples without replacement and I agree that the extra code required to turn a PRG sampling with replacement into one sampling without replacement far outweighs the savings that can be made by avoiding swapping.

If you read my post immediately after "Swapping is inefficient and wasteful" entitled "My fantastic shuffling algorithm" you will see that I didn't propose `deckB[i] = deckA[rand[i]]`. I actually proposed `deckB[i] = deckC[deckA[i]]`.

A PR sequence of 52 cards 52 cards long drawn without repetition is simply a deck of cards drawn sequentially, which is why I can use `deckA[i]` in place of the more general `rand[i]`.

It should be noted that repeating `deckB[i] = deckC[deckA[i]]` by itself in a loop will not work. `deckB` will be changed the first time but remain the same thereafter. As I suggested before, the programmer needs to call the shuffle algorithm twice in the loop, or needs to swap the pointers A and B to get around this problem. In this regard, it is very different from other shuffling algorithms and PRNGs, so people could be confused and need to be clear about this point.

```
{
deckB = deckC[deckA];
play(deckB);
if(playagain()==0)
    return;
deckA = deckC[deckB];
play(deckA);
if(playagain()==0)
    return;
}
```

Alternatively:

sci.crypt: That is not the algorithm I proposed

```
{
deckB = deckC[deckA];
play(deckB);
if(playagain()==0)
    return;
swap(*deckA,*deckB);
}
```

deckB[i] = deckC[deckA[i]] is a very simple way to sample without replacement. It does avoid the need for swapping yet doesn't require any complicated code to turn a sampling with replacement generator into a sampling without replacement generator.

Paul Tomkins.

"Brian Hetrick" <bhetrick@notinnedmeats.iname.com> wrote in message news:auKdndFXzq92Lj_cRVn-qQ@comcast.com...

> "Paul Tomkins" <tomkinsp@iinet.net.au> wrote ...

> [snip]

>

> *The algorithm you propose, deckB [i] = deckA [rand [i]], needs rand [i] to do sampling without replacement. Otherwise, deckB has multiples of some cards and therefore does not have other cards. The overhead of keeping track of which cards have and have not been used essentially duplicates the swapping necessary for shuffling a single deck.*

>

> *I seem to recall from a course long ago that the algorithm (WARNING, air code follows):*

>

> *card d [52];*

> *int i;*

> *for (i = 0; i < 52; i ++)*

> {

> *d [i] = (card) i;*

> }

> *for (i = 0; i < 52; i ++)*

> {

> *card t;*

> *k = (int) (rand () * 52);*

> *t = d [i];*

> *d [i] = d [k];*

> *d [k] = t;*

> }

>

> *does not generate all possible permutations with equal probabilities, but*

> *I remember proving the algorithm:*

>

> *card d [52];*

> *int i;*

> *for (i = 0; i < 52; i ++)*

> {

That is not the algorithm I proposed

sci.crypt: That is not the algorithm I proposed

```
> d [i] = (card) i;
> }
> for (i = 0; i < 51; i ++ )
> {
> card t;
> k = i + (int) (rand () * (52 - i));
> t = d [i];
> d [i] = d [k];
> d [k] = t;
> }
>
> does. The difference is that in the second algorithm, the swap of element
> [i] is always with an element at position i or after. (In both cases, rand
> () is assumed to produce a U[0,1) variate.)
>
> Just about any elementary probability text would have an algorithm for
> random shuffling, but the second algorithm above is, as far as I know,
> "standard."
>
```