Sampling without Replacement: Durstenfeld-Fisher-Yates Permutation for Very Large Permutation Sizes

Author: Martin Harriman

Publication Date: March 17, 2017
Summary

We present a modification of the Durstenfeld-Fisher-Yates random-permutation algorithm for use in sampling without replacement from a large population.

Motivation

We would like to sample without replacement from a potentially large population. The Durstenfeld-Fisher-Yates random permutation algorithm is attractive for small populations, as it generates the desired sample in time and memory proportional to the population size. We modify Durstenfeld to generate a partial permutation (and thus, sample without replacement) in time and memory proportional to the sample size.

Description

Given a vector \( p \), the Durstenfeld random permutation algorithm repeatedly swaps two elements, call them \( p[i] \) and \( p[j] \) (in Durstenfeld's algorithm, \( i \) iterates sequentially over each element of \( p \), and \( j \) is a random value, \( i \leq j \leq \text{length of } p \)).

Instead of using a vector \( p \), we will use a hash table \( h \) to record the entries we modify. The hash table records only those elements of \( p \) that we have changed, so it will require memory proportional to the length of the partial permutation we generate.

When swapping \( p[i] \) and \( p[j] \), for \( i \neq j \), look for the index \( j \) in \( h \). If \( j \) is in \( h \), use the value from the map as \( p[j] \), otherwise use \( j \) as the value of \( p[j] \). Similarly, if \( i \) is in the hash map, use the value from the map as \( p[i] \), and if not, use \( i \). Then record the swap, and in particular enter the new value of \( p[j] \) in the map. For Durstenfeld, we will never examine \( p[i] \) again, so we can record the new value of \( p[i] \) however we please.

Though \( h \) is described as a hash table, one could use any associative structure to implement it. One could equally well implement \( h \) as content-addressable memory in hardware, for instance.
References

Following documents were referred in preparing this document:


About the Author

Martin Harriman has been happily generating pseudorandom permutations using Durstenfeld’s algorithm since 1973: many thanks to the Stanford Bookstore for displaying Knuth’s *The Art of Computer Programming* so prominently (and of course, many thanks to Professor Knuth for writing it).
© 2017 Pure Storage, Inc. All rights reserved. Pure Storage and the "P" Logo registered trademarks of Pure Storage, Inc. in the U.S. and other countries. Any other trademarks are the property of their respective owners.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. PURE STORAGE SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION.

Pure Storage, Inc. 650 Castro Street, Mountain View, CA 94041
http://www.purestorage.com