8th *International Conference on Applied Informatics*
Eger, Hungary, January 30, 2010.

QUICK TESTING OF RANDOM VARIABLES

Antal Iványi,[a] Imre Kátai[a]

[a] Eötvös Loránd University,
Dept. of Computer Algebra

E-mail: tony@compalg.inf.elte.hu
E-mail: katai@compalg.inf.elte.hu

This presentation can be downloaded from
http://compalg.inf.elte.hu/~tony/ICAI

1. Introduction

Let ξ be a random integer vector, having uniform distribution

$$\mathbf{P}\{\xi = (i_1, i_2, \ldots, i_n) = 1/n^n\} \text{ for}$$

$$1 \le i_1, i_2, \ldots, i_n \le n.$$

A realization $(i_1, i_2, \ldots, i_n)$ of ξ is called *good*, if its elements are different. We present five comparison-based algorithms which decide whether a given realization is good [1, 2, 3, 4, 5, 6, 7, 8].

2. Algorithms

The first algorithms are Forward and Backward. These algorithms are naive comparison-based ones.

FORWARD compares the first (i_1), second (i_2), ..., last but one (i_{n-1}) element of the realization with the following elements until the first collision or until the last pair of elements.

BACKWARD compares the second (i_2), third (i_3), ..., last (i_n) element of the realization with the previous elements until the first collision or until the last pair of elements.

LINEAR writes zero into the elements of an n length vector $\mathbf{v} = (v_1, v_2, \ldots, v_n$, then investigates the elements of the realization and if $i_j = k$, then adds 1 to v_k and tests wheatherer $v_k > 1$ signalizing a repetition.

TREE builds a random search tree from the elements of the realization and finishes the construction of the tree if it finds the following element of the realization in the tree (then the realization is not good) or it tested the last element too without a collision (then the realization is good).

Finally MODULAR handles $m = \lceil\sqrt(n)\rceil$ queues and puts the element i_j into the i-th queue if i_j gives a residue i mod m. MODULAR tests wheather i_j appeared earlier using linear search only in the corresponding queue.

3. Efficiency of the algorithms

FORWARD and BACKWARD work in best case ($\Theta(1)$ time, in worst case in$\Theta(n^2)$ time and in expected case in $\Theta(n)$ time.

LINEAR works in $\Theta(n)$ time in all cases.

In expected time TREE works in $\Theta(\log n \sqrt{}(n)$ time while MODULAR is asypptotically optimal since it works $\Theta(\sqrt{n})$ time.

References

[1] Bailey, R. A., Cameron, P. J., Connelly, R., Sudoku, gerechte designs, resolutions, affine space, spreads, reguli, and Hamming codes. *Amer. Math. Monthly,* Vol. 115 (2008), no. 5, 383-404. ⇒2

[2] Behrens, W. U. Feldversuchsanordnungen mit verbessertem Ausgleich der Bodenunterschiede, *Zeitschrift für Landwirtschaftliches Versuchs- und Untersuchungswesen,* Vol. 2, (1956), no. 2, 176-193. ⇒2

[3] Crook, J. F. A pencil-and-paper algorithm for solving Sudoku puzzles. *Notices Amer. Math. Soc.*, Vol. 56 (2009), no. 4, 460-468. ⇒2

[4] Iványi, A., Kátai, I. Estimates for speed of computers with interleaved memory systems, *Annales Univ. Sci. Budapest., Sectio Mathematica*, Vol. 19 (1976), 159-164. ⇒2

[5] Knuth, D. E., *The Art of Computer Programming. Vol. 1. Fundamental Algorithms* (third edition). Addison–Wesley, 1997. ⇒2

[6] Thom, D., SUDOKU ist NP-vollstndig. PhD Dissertation. Stuttgart, 2007. ⇒2

[7] Vaughan, E. R. The complexity of constructing gerechte designs. *The Electronic Journal of Combinatorics.* ⇒2

[8] Xu, C., Xu, W., The model and algorithm to estimate the difficulty levels of Sudoku puzzles. *J. Math. Res.*, Vol. 11 (2009), no. 2, 43-46. ⇒2