8^{th} *Joint Conference on Mathematics and Computer Science, July 14–17, 2010, Komárno, Slovakia*

TESTING OF SEQUENCES BY SIMULATION

Antal Iványi, Balázs Novák

Eötvös Loránd University,
Dept. of Computer Algebra

1. Introduction

Let ξ be a random integer vector, having uniform distribution

$$P\{\xi = (i_1, i_2, \dots, i_n) = 1/n^n\} \text{ for } 1 \le i_1, i_2, \dots, i_n \le n.$$

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

2. Algorithms

The first algorithm is Backward. It is a naive comparison-based one, which 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.

Theorem 1 *The expected number of the comparisons of the algorithm* Backward *is*

$$C_{exp}(n, \text{Back}) = n + \sqrt{\frac{\pi n}{8}} + \frac{2}{3} - \alpha(n),$$

where $\alpha(n) = \frac{\kappa(n)}{2} + \frac{n!}{n^n}\frac{n+1}{2}$ *monotonically decreasing tends to zero when* n *tends to* ∞*.*

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 whether $v_k > 1$ signalizing a repetition.

n	C	$n-\sqrt{\frac{n}{8}}+\frac{2}{3}$	$\frac{n!}{n^n}\frac{n+1}{2}$	$\kappa(n)$	$\alpha(n)=\frac{(n)}{2}+\frac{n!}{n^n}\frac{n+1}{2}$
1	0.000000	1.040010	1.000000	0.080019	1.040010
2	1.000000	1.780440	0.750000	0.060879	0.780440
3	2.111111	2.581265	0.444444	0.051418	0.470154
4	3.156250	3.413353	0.234375	0.045455	0.257103
5	4.129600	4.265419	0.115200	0.041238	0.135819
6	5.058642	5.131677	0.054012	0.038045	0, 073035
7	5.966451	6.008688	0.024480	0.035515	0.042237
8	6.866676	6.894213	0.010815	0.033444	0.027536
9	7.766159	7.786695	0.004683	0.031707	0.020537
10	8.667896	8.685003	0.001996	0.030222	0.017107

Table 1: Values of C $n-\sqrt{\frac{n}{8}}+\frac{2}{3}$, $\frac{n!}{n^n}\frac{n+1}{2}$, $\kappa(n)$, and $\alpha(n)=\frac{(n)}{2}+\frac{n!}{n^n}\frac{n+1}{2}$ for $n=1,\ 2,\ \ldots,\ 10$ (Number of comparisons of BACKWARD)

Theorem 2 *The expected number of comparisons of* LINEAR *is*

$$C_{exp}(n, \mathrm{LI}) = \sqrt{\frac{\pi n}{2}} + \frac{2}{3} - \frac{n!}{n^n} + \kappa(n),$$

where $\kappa(n)$ tends monotonically decreasing to zero when n tends to infinity.

TREE builds a random search tree from the elements of the realization

n	C	$\sqrt{\frac{n}{2}} + \frac{2}{3}$	$\frac{n!}{n^n}$	$\kappa(n)$	$\delta(n) = \kappa(n) - \frac{n!}{n^n}$	$\sigma(n)$
1	1.000000	1.919981	1.000000	0.080019	−0.919981	0.025808
2	2.000000	2.439121	0.500000	0.060879	−0.439121	0.013931
3	2.666667	2.837470	0.222222	0.051418	−0.170804	0.009504
4	3.125000	3.173295	0.093750	0.045455	−0.048295	0.007205
5	3.472000	3.469162	0.038400	0.041238	+0.002838	0.005799
6	3.759259	3.736647	0.015432	0.038045	+0.022612	0.004852
7	4.012019	3.982624	0.006120	0.035515	+0.029395	0.004170
8	4.242615	4.211574	0.002403	0.033444	+0.031040	0.003656
9	4.457379	4.426609	0.000937	0.031707	+0.030770	0.003255
10	4.659853	4.629994	0.000363	0.030222	+0.029859	0.002933

Table 2: Values of C, $\sqrt{\frac{n}{2}} + \frac{2}{3}$, $\frac{n!}{n^n}$, $\kappa(n)$, $\delta(n)$, and $\sigma(n)$ for $n = 1, 2, \ldots, 10$ (Number of comparisons of LINEAR)

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 without a repetition (then the realization is good).

Theorem 3 *The expected number of comparisons of* TREE *is*

$$C_{exp}(n, \text{TREE}) = \Theta(\sqrt{n} \lg n),$$

and the expected number of assignments is

$$A_{exp}(n, \text{Tree}) = \Theta(\sqrt{n}),$$

The next table shows some results of the simulation experiments (the number of random input sequences is 100 000 in all cases).

Using the method of the smallest squares to find the parameters of the formula $a\sqrt{n}\log_2 n + b$, resp. $c\sqrt{n}\log_2 n + d$ we received the following approximation formula for the expected number of comparisons, resp. for the expected number of assignments:

$$C_{exp}(n, \text{Tree}) =$$

$$= 0.585205\sqrt{n}\log_2 n + 0.447973,$$

n	number of good inputs	number of comparisons	number of assignments
1	100 000.000000	0.000000	1.000000
2	49 946.000000	1.000000	1.499460
3	22 243.000000	2.038960	1.889900
4	9 396.000000	2.921710	2.219390
5	3 723.000000	3.682710	2.511409
6	1 569.000000	4.352690	2.773160
7	620.000000	4.985280	3.021820
8	251.000000	5.590900	3.252989
9	104	6.148550	3.459510
10	33	6.704350	3.663749
11	17	7.271570	3.860450
12	3	7.779950	4.039530
13	3	8.314370	4.214370
14	0	8.824660	4.384480
15	2	9.302720	4.537880
16	0	9.840690	4.716760
17	0	10.287560	4.853530
18	0	10.719770	4.989370
19	0	11.242740	5.147560
20	0	11.689660	5.279180

Table 3: Values of n, number of good inputs, number of comparisons, number of assignments for $n = 1,\ 2,\ \ldots,\ 10$ (Number of comparisons of LINEAR)

and

$$A_{exp}(n, \textsc{Tree}) =$$

$$= 1.245754\sqrt{n}\log_2 n - 0.273588.$$

Finally BUCKET handles $\rceil\sqrt(n)\rceil$ buckets. It tests whether i_j appeared earlier in the $\lceil i_j/\sqrt{n} = r$-th bucket (using linear search in the corresponding bucket). If yes, then the input is bad, otherwise BUCKET puts the investigated element into the r-th bucket.

Lemma 4 *Let* b_j $(j = 1,\ 2,\ \ldots,\ m)$ *be a random variable characterising the number of elements in bucket* B_j *at the moment of the first repetition. Then*

$$E\{b_j\} = \sqrt{\frac{\pi}{2}} - \mu(n) \textit{ for } j = 1, 2, \ldots, m,$$

where

$$\mu(n) = \frac{1}{3\sqrt{n}} - \frac{\kappa(n)}{\sqrt{n}},$$

and $\mu(n)$ *tends monotonically decreasing to zero when* n *tends to infinity.*

n	$E\{b_1\}$	$\sqrt{\frac{}{2}}$	$\frac{1}{3\sqrt{n}}$	$\frac{(n)}{\sqrt{n}}$	$\mu(n) = \frac{1}{3\sqrt{n}} - \frac{(n)}{\sqrt{n}}$
1	1.000000	1.253314	0.333333	0.080019	0.253314
2	1.060660	1.253314	0.235702	0.043048	0.192654
3	1.090055	1.253314	0.192450	0.029686	0.162764
4	1.109375	1.253314	0.166667	0.022727	0.143940
5	1.122685	1.253314	0.149071	0.018442	0.130629
6	1.132763	1.253314	0.136083	0.015532	0.120551
7	1.147287	1.253314	0.125988	0.013423	0.112565
8	1.147287	1.253314	0.117851	0.011824	0.106027
9	1.152772	1.253314	0.111111	0.010569	0.100542
10	1.157462	1.253314	0.105409	0.009557	0.095852

Table 4: Values of $E\{b_1\}$, $\sqrt{\frac{}{2}}$, $\frac{1}{3\sqrt{n}}$, $\frac{(n)}{\sqrt{n}}$, and $\mu(n) = \frac{1}{3\sqrt{n}} - \frac{(n)}{\sqrt{n}}$ for $n = 1, 2, \ldots, 10$ (Number of elements in a bucket)

Theorem 5 *The expected number of comparisons in a bucket is*

$$E\{C_1(n)\} = 1 - \phi(n).$$

where

$$\phi(n) = \frac{1}{n+\sqrt{n}}\left(\sqrt{n} - \sqrt{\frac{\pi n}{2}} + \frac{1}{3} - \kappa(n)\right),$$

and $\phi(n)$ *tends to zero when* n *tends to infinity.*

n	$E\{C_1(n)\}$	$a(n)\left(\sqrt{n}-\sqrt{\frac{n}{2}}+\frac{1}{3}-\kappa(n)\right)$	$\frac{a(n)}{n+\sqrt{n}}$	$1-\frac{a(n)}{n+\sqrt{n}}$
1	0.000000	1.000000	0.000000	0.000000
2	0.146447	2.914214	0.853553	0.146447
3	0.234805	3.620940	0.765195	0.234805
4	0.296875	4.218750	0.703125	0.296875
5	0.344054	4.746468	0.655946	0.344054
6	0.381716	5.224181	0.618284	0.381716
7	0.412810	5.663890	0.587190	0.412810
8	0.439120	6.073445	0.560880	0.439120
9	0.461807	6.458316	0.538193	0.461807
10	0.481663	6.822493	0.518337	0.481663

Table 5: Values of $E\{C_1(n)\}$, $\sqrt{\frac{n}{2}}-\frac{1}{3}+\kappa(n)$, $a(n)=2n-2(\sqrt{\frac{n}{2}}-\frac{1}{3}+\kappa(n))$, and $\frac{a(n)}{2n+2\sqrt{n}}$ for $n=1,\ 2,\ \ldots,\ 10$

3. Summary (efficiency of the algorithms)

Index and Algorithm	$T_{best}(n)$	$T_{worst}(n)$	$T_{exp}(n)$
1. LINEAR	$\Theta(n)$	$\Theta(n)$	$n + \Theta(\sqrt{n})$
2. BACKWARD	$\Theta(1)$	$\Theta(n^2)$	$\Theta(n)$
3. TREE	$\Theta(1)$	$\Theta(n^2)$	$\Theta(\sqrt{n} \lg n)$
4. BUCKET	$\Theta(\sqrt{n})$	$\Theta(n\sqrt{n})$	$\Theta(\sqrt{n})$

Table 6: The running times of the investigated algorithms in best, worst and expected cases

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

[3] 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

[4] Iványi, A., Kátai, I. Testing of random matrices, *Annales Univ. Sci. Budapest., Sectio Computatorica*, Submitted. ⇒ 2

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

[6] Novák, B. Analysis of Sudoku algorithms (in Hungarian). Master thesis, Eötvös Loránd University, Budapest, 2010. http://compalg.inf.elte.hu/~tony/Oktatas/Rozsa/Sudoku-thesis/ ⇒ 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

E-mail: tony@compalg.inf.elte.hu
E-mail: pszihopata85@gmail.com