Annales Univ. Sci. Budapest., Sectio Comp., **??** (????) ??–??

TESTING OF RANDOM MATRICES

A. Iványi and **I. Kátai**
(Budapest, Hungary) **JULY 14, 2010**

Dedicated to Professor Péter Simon on his 60th birthday

Abstract. Let n be a positive integer and $X = [x_{ij}]_{1 \le i,j \le n}$ be an $n \times n$ sized matrix of independent random variables having joint uniform distribution

$$\Pr\{x_{ij} = k \text{ for } 1 \le k \le n\} = \frac{1}{n} \quad (1 \le i, j \le n).$$

A realization $\mathcal{M} = [m_{ij}]$ of X is called *good*, if its each row and each column contains a permutation of the numbers $1,\ 2,\ \ldots,\ n$. We present and analyse algorithms which decide whether a given realization is good.

1 Introduction

Some subsets of the elements of Latin squares [1, 8, 18, 19, 20, 33, 34, 41, 42], of Sudoku squares [3, 10, 12, 15, 16, 17, 29, 30, 31, 35, 38, 42, 44, 46, 48, 49, 52], of de Bruijn arrays [2, 5, 7, 13, 23, 25, 26, 27, 28, 36, 43, 51, 53] and gerechte designs, connected with agricultural and industrial experiments [3, 4, 22] have to contain different elements. The one dimensional special case is also studied is several papers [9, 21, 24].

The testing of these matrices raises the following problem.

Let n $(1 \le n)$ be integers and $X = [x_{ij}]_{1 \le i \le m, 1 \le j \le n}$ be an $m \times n$ sized matrix of independent random variables having joint uniform distribution

$$\Pr\{x_{ij} = k \text{ for } 1 \le k \le n\} = \frac{1}{n} \quad (1 \le i \le m, 1 \le j \le n). \tag{1}$$

A realization $\mathcal{M} = [m_{ij}]$ of X is called *good*, if its each row and each column contain different elements (in the case $m = n$ a permutation of the numbers $1,\ 2,\ \ldots,\ n$. We present and analyse algorithms which decide whether a given realization is good. If the realization is good then the output of the algorithms is True, otherwise is False.

In Section 2 we analyse the running times of the algorithms testing the first row of $\mathcal{M}$ in worst, best and expected cases. Section 3 contains an algorithm testing some subsets of two and three dimensional matrices. In Section 4 the results are summarised, while in Section 5 the pseudocodes of the investigated algorithms are presented.

Computing Classification System 1998: F.2 [Theory of Computation]: Subtopic – Analysis of algorithms and problem complexity
Mathematics Subject Classification 2010: 68Q25
Key words and phrases: random sequences, efficiency of algorithms

2 Test of random sequences

We start with the first step of the testing of $\mathcal{M}$: describe and analyse several algorithms testing the first row of $\mathcal{M}$. The inputs of these algorithms are n (the length of the first row of $\mathcal{M}$) and the elements of the first row $\mathbf{m} = (m_{11}, m_{12}, \ldots, m_{1n})$. For the simplicity we use the notation $\mathbf{s} = (s_1, s_2, \ldots, s_n)$. The output is always a logical variable g (its value is True, if the input sequence is good, and False otherwise).

We will denote the binomial coefficient $\binom{n}{k}$ by $B(n,k)$ and the function $\log_2 n$ by $\lg n$ [14].

We characterise the running time of the algorithms by the number of necessary assignments and comparisons and denote the running time of algorithm Alg by $T_{worst}(n, \text{Alg})$, $T_{best}(n, \text{Alg})$ and $T_{exp}(n, \text{Alg})$ in the worst, best, resp. expected case. The numbers of the corresponding assignments and comparisons are denoted by A, resp. C.

Before the investigation of the concrete algorithms we formulate several lemmas. The first lemma is the following version of the well-known Stirling's formula.

Lemma 1 ([14]) *If $n \geq 1$ then*

$$n! = \left(\frac{n}{e}\right)^n \sqrt{2\pi n} e^{\tau(n)}, \tag{2}$$

where

$$\frac{1}{12n+1} < \tau(n) < \frac{1}{12n}, \tag{3}$$

and τ_n tends monotonically decreasing to zero when n tends to infinity.

Let $a_k(n) = a_k$ and $S_i(n) = S_i$ defined for any positive integer n as follows:

$$a_k(n) = a_k = \frac{n^k}{k!} \quad (k = 0,\ 1,\ 2,\ \ldots), \tag{4}$$

$$S_i(n) = S_i = \sum_{k=0}^{n-1} a_k k^i \quad (i = 0,\ 1,\ 2,\ \ldots). \tag{5}$$

If in (5) $k = i = 0$, then $k^i = 0$.

Solving a problem posed by S. Ramanujan [45] Gábor Szegő [47] proved the following connection between e^n and S_0.

Lemma 2 ([47]) *The function $\sigma(n)$ defined by*

$$\frac{e^n}{2} = S_0 + \left(\frac{1}{3} + \sigma(n)\right) a_n = \sum_{k=0}^{n-1} \frac{n^k}{k!} + \left(\frac{1}{3} + \sigma(n)\right) a_n \quad (n = 1,\ 2,\ \ldots) \tag{6}$$

and

$$\sigma(0) = \frac{1}{6}, \tag{7}$$

tends monotonically decreasing to zero when n tends to ∞.

The following lemma shows the connection among S_i and $S_0,\ S_1,\ \ldots,\ S_{i-1}$.

Lemma 3 *If i and n are positive integers, then*

$$S_i(n) = n\sum_{k=0}^{i-1} B(i-1,k)S_k(n) - n^{i-1}a_{n-1} \tag{8}$$

and

$$S_i(n) = \Theta(e^n n^i). \tag{9}$$

Proof. Omitting the member belonging to the index $k = 0$ in S_i, then simplifying by k and using the substitution $k-1 = j$ we get

$$S_i = \sum_{k=0}^{n-1} \frac{n^k}{k!}k^i = n\sum_{k=1}^{n-1} \frac{n^{k-1}}{(k-1)!}k^{i-1} = n\sum_{j=0}^{n-2} \frac{n^j}{j!}(j+1)^{i-1}. \tag{10}$$

Completing the sum with the member belonging to index $j = n-1$ results

$$S_i = n\sum_{j=0}^{n-1} \frac{n^j}{j!}(j+1)^{i-1} - n^i a_{n-1}. \tag{11}$$

Now the application of the binomial theorem results (8).

According to (9) $S_0 = \Theta(e^n)$, therefore using induction and (10) we get (9). ■

In this paper we need only the simple form of $S_0,\ S_1,\ S_2$ and S_3 what is presented in the next lemma.

Lemma 4 *If n is a positive integer then*

$$S_0 = \frac{e^n}{2} - \frac{n^n}{n!}\left(\frac{1}{3} + \sigma(n)\right), \tag{12}$$

$$S_1 = nS_0 - na_{n-1}, \quad S_2 = S_0(n^2+n) - 2n^2 a_n, \tag{13}$$

and

$$S_3 = S_0(n^3 + 3n^2 + n) - (3n^3 + 2n^2)a_n. \tag{14}$$

Proof. Expressing S_0 from (6), and using recursively Lemma 3 for $i = 1,\ 2$ and 3 we get the required formula for $S_0,\ S_1,\ S_2,$ and S_3. ■

We remark that (11) was proved by Balázs Novák in [40] directly.

We introduce also another useful function

$$R_i(n) = \sum_{k=1}^{n} p_k(n)k^i \quad (i = 0,\ 1,\ 2,\ \ldots), \tag{15}$$

where $p_k(n)$ is the key probability of this paper, defined in [21] as

$$p_k(n) = p_k = \frac{n}{n}\frac{n-1}{n}\cdots\frac{n-k+1}{n}\frac{k}{n} = \frac{n!k}{(n-k)!n^{k+1}} \quad (k = 1,\ 2,\ \ldots,\ n). \tag{16}$$

The following lemma mirrors the connection between the function $R_i(n)$ and the functions $S_0,\ S_1,\ \ldots,\ S_{i+1}$.

Lemma 5 *If i and n are positive integers, then*

$$R_i(n) = \frac{n!}{n^{n+1}} \sum_{l=0}^{i+1} (-1)^l \binom{i+1}{l} n^{i+1-l} S_l(n). \tag{17}$$

Proof. Using (15) and (16) the substitution $n - k = j$ results

$$R_i(n) = \sum_{k=1}^{n} \frac{n!k^{i+1}}{(n-k)!n^{k+1}} = \frac{n!}{n^{n+1}} \sum_{j=0}^{n-1} \frac{n^j (n-j)^{i+1}}{j!} \ . \tag{18}$$

From here using the binomial theorem we get (2). ■

In this paper we need only the following consequence of Lemma 5.

Lemma 6 *If n is a positive integer, then*

$$R_0(n) = 1, \quad R_1(n) = \frac{n!}{n^n} S_0, \tag{19}$$

and

$$R_2(n) = 2n - \frac{n!}{n^n} S_0. \tag{20}$$

Proof. $R_0(n) = 0$ follows from the definition of the probabilities $p_k(n)$. Substituting $i = 1$ into (2) we get

$$R_1(n) = \frac{n!}{n^{n+1}} \left(n^2 \sum_{j=0}^{n-1} \frac{n^j}{j!} - 2n \sum_{j=0}^{n-1} \frac{n^j}{j!} j + \sum_{j=0}^{n-1} \frac{n^j}{j!} j^2 \right). \tag{21}$$

From here using (5) we get

$$R_1(n) = \frac{n!}{n^{n+1}} (n^2 S_0 - 2nS_1 + S_2), \tag{22}$$

and using (11) the required formula for $R_1(n)$.

Substituting $i = 2$ into () we get

$$R_2(n) = \frac{n!}{n^{n+1}} \left(n^3 \sum_{j=0}^{n-1} \frac{n^j}{j!} - 3n^2 \sum_{j=0}^{n-1} \frac{n^j}{j!} j + 3n \sum_{j=0}^{n-1} \frac{n^j}{j!} j^2 - \sum_{j=0}^{n-1} \frac{n^j}{j!} j^3 \right). \tag{23}$$

From here using (5) we have

$$R_2(n) = \frac{n!}{n^{n+1}} (n^3 S_0 - 3n^2 S_1 + 3nS_2 - S_3), \tag{24}$$

and using (13) and (14) the required formula for $R_2(n)$. ■

The following lemma gives some further properties of $R_1(n)$.

Lemma 7 *If n is a positive integer, then*

$$R_1(n) = \frac{n!}{n^n} S_0 = \sqrt{\frac{\pi n}{2}} - \frac{1}{3} + \kappa(n), \tag{25}$$

where

$$\kappa(n) = \sqrt{\frac{\pi n}{2}} \left(e^{\tau(n)} - 1 - \frac{2\sigma(n) e^{\tau(n)}}{e^n} \right), \tag{26}$$

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

Proof. Substituting S_0 according to (12) in the formula (20) for $R_1(n)$ we get

$$R_1(n) = \frac{n!}{n^n} \left[\frac{e^n}{2} - \frac{n^n}{n!} \left(\frac{1}{3} + \sigma(n) \right) \right] = -\frac{1}{3} + \frac{n!}{n^n} \left(\frac{e^n}{2} - \frac{n^n}{n!} \sigma(n) \right). \tag{27}$$

Substitution of $n!$ according to (2) (Stirling's formula) and writing $1 + (e^{\tau(n)} - 1)$ instead of $e^{\tau(n)}$ results

$$R_1(n) = -\frac{1}{3} + \frac{1}{n^n} \left(\frac{n}{e} \right)^n \sqrt{2\pi n} \left[1 + (e^{\tau(n)} - 1) \right] \left[\frac{e^n}{2} - \sigma(n) \right]. \tag{28}$$

The product P of the expressions in the square brackets is

$$P = \frac{e^n}{2} + \frac{e^n}{2} \left(e^{\tau(n)} - 1 \right) - \sigma(n) e^{\tau(n)}, \tag{29}$$

therefore

$$R_1(n) = \sqrt{\frac{\pi n}{2}} - \frac{1}{3} + \frac{\sqrt{2\pi n}}{e^n} \left[\frac{e^n}{2} \left(e^{\tau(n)} - 1 \right) - \sigma(n) e^{\tau(n)} \right], \tag{30}$$

implying

$$R_1(n) = \sqrt{\frac{\pi n}{2}} - \frac{1}{3} + \sqrt{\frac{\pi n}{2}} \left(e^{\tau(n)} - 1 \right) - \sqrt{\frac{\pi n}{2}} \frac{2\sigma(n) e^{\tau(n)}}{e^n}. \tag{31}$$

Let

$$\kappa_1(n) = \sqrt{\frac{\pi n}{2}} (e^{\tau(n)} - 1), \quad \kappa_2(n) = \sqrt{\frac{\pi n}{2}} \frac{2\sigma(n) e^{\tau(n)}}{e^n}, \quad \text{and } \kappa(n) = \kappa_1 + \kappa_2, \tag{32}$$

and

$$\gamma(n) = \frac{\kappa(n+1)}{\kappa(n)} = \frac{\kappa_1(n+1) - \kappa_2(n+1)}{\kappa_1(n) - \kappa_2(n)} \quad \text{for } n = 1,\ 2,\ \ldots. \tag{33}$$

Since all κ functions are positive for all positive integer n's, therefore $\gamma(n) < 1$ for $n \geq 1$ implies the monotonity of $\kappa(n)$. Numerical results in Table 1 show that $\gamma(n) < 1$ for $n = 1,\ 2,\ \ldots,\ 9$, therefore it remained to show $\gamma(n) < 1$ for $n \geq 10$.

$\kappa_2(n)$ can be omitted from the nominator of (32). Since $\sigma(n)$ and $\tau(n)$ are monotone decreasing functions, and $0 < \sigma(5) < 0.0058$, and $0 < e^{\tau(5)} < 1.02$, and $n^2 < e^n$ for $n \geq 10$, therefore

$$\frac{2\sigma(n)e^{\tau(n)}}{e^n} < \frac{2 \cdot 0.0058 \cdot 1.02}{e^n} < \frac{0.012}{n^2} \text{ for } n \geq 10. \tag{34}$$

Using (33), (34) and the Lagrange remainder of the Taylor series of the function e^x we have

$$\gamma(n) < \frac{\sqrt{n+1}}{\sqrt{n}} \frac{\tau(n+1) + \frac{\tau^2(\xi_{n+1})}{2}}{\tau(n) + \frac{\tau^2(\xi_n)}{2} - \frac{0.012}{n^2}}\ , \tag{35}$$

where $0 < \xi_{n+1} < n+1$ and $0 < \xi_n < n$, therefore using Lemma 1 we get

$$\gamma(n) < \frac{\sqrt{n+1}}{\sqrt{n}} \frac{\frac{1}{12(n+1)+1}}{\frac{1}{12n} + \frac{1}{2}\left(\frac{1}{12n}\right)^2 - \frac{0.012}{n^2}}\ . \tag{36}$$

Now multiplying the nominator and denominator of the right side of (**??**) by $(12n)^2$ results

$$\gamma(n) = \frac{\sqrt{n+1}}{\sqrt{n}} \frac{\frac{12n \cdot 12n}{12n+13}}{12n + 0.5 - 1.728} = \frac{\sqrt{n+1}}{\sqrt{n}} \frac{12n}{(12n - 1.228)\left(1 + \frac{13}{12n}\right)}\ . \tag{37}$$

Since

$$(12n - 1.228)\left(1 + \frac{13}{12n}\right) < 12n + 11, \tag{38}$$

(37) and (38) imply

$$\gamma(n) < \frac{\sqrt{144n^3 + 144n^2}}{\sqrt{144n^3 + 264n^2}} < 1, \tag{39}$$

finishing the proof of the monotonity of $\kappa(n)$. ■

We remark, that the monotonity of κ_n was published in [24] without proof.

In the following seven analyses let $n \geq 1$ and let $\mathbf{x} = (x_1,\ x_2,\ \ldots,\ x_n)$ be independent random variables having uniform distribution on the set $\{1, 2, \ldots, n\}$. The input sequence of the algorithms is $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ (a realization of $\mathbf{x}$).

2.1 Running time of algorithm LINEAR

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 $\mathbf{s}$ and if $v[s_k] > 0$ (signalising a repetition), then returns False, otherwise adds 1 to v_k. If Linear does not find a repetition among the elements of $\mathbf{s}$ then it returns finally True.

Linear needs assignments in lines 01, 03, and 08, and it needs comparisons in line 05. The number of assignments in lines 01 and 03 equals to $n+1$ for arbitrary input and varies between 1 and n in line 08. The number of comparisons in line 08 also varies between 1 and n. Therefore the running time of Linear is $\Theta(n)$ in the best, worst and expected case too. The following theorem gives a more precise characterisation of the expected running time of $T_{exp}(n, \text{Linear})$.

Theorem 8 *The expected running time of* LINEAR *is*

$$T_{exp}(n, \textsc{Linear}) = n + \sqrt{2\pi n} + \frac{7}{3} - 2\frac{n!}{n^n} - 2\kappa(n), \tag{40}$$

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

Proof. Let

$$y(n) = y = \max\{k : 1 \le k \le n \text{ and } s_1,\ s_2,\ \ldots,\ s_k \text{ are different}\} \tag{41}$$

be a random variable characterising the maximal length of the prefix of $\mathbf{s}$ containing different elements. Then

$$\Pr\{y = k\} = p_k(n) = p_k \quad (k = 1,\ 2,\ \ldots,\ n), \tag{42}$$

where $p_k(n)$ is the probability introduced in (16).

At first we compute the expected value of the necessary comparisons $C_{exp}(n, \textsc{Linear})$ which for the simplicity will be denoted by C.

If $y = k$ and $1 \le k \le n-1$, then LINEAR executes $k+1$ comparisons, and only n comparisons, if $y = n$, therefore

$$C = \sum_{k=1}^{n-1} p_k(k+1) + p_n n = \sum_{k=1}^{n} p_k(k+1) - p_n = 1 - \frac{n!}{n^n} + \sum_{k=1}^{n} p_k k, \tag{43}$$

from where using Lemma (7) we receive

$$C = 1 - \frac{n!}{n^n} + R_1(n) = \sqrt{\frac{\pi n}{2}} + \frac{2}{3} - \frac{n!}{n^n} + \kappa(n). \tag{44}$$

LINEAR requires $n+1$ assignments in lines 01 and 03, plus assignments in line 08. The expected number of assignments in line 08 is the same as $C_{exp}(n, \textsc{Linear})$. Therefore

$$T_{exp}(n, \textsc{Linear}) = n + 1 + 2C_{exp}(n, \textsc{Linear}). \tag{45}$$

Substituting (44) into (45) results the required (40). ■

The monotonity of $\kappa(n)$ was proven in the proof of Lemma (7). We remark, that (44) is equivalent with

$$C = 1 - \frac{n!}{n^n} + 1 + \frac{n-1}{n} + \frac{n-1}{n}\frac{n-2}{n} + \cdots + \frac{n-1}{n}\frac{n-2}{n}\cdots\frac{1}{n}, \tag{46}$$

demonstrating the close connection with the function

$$Q(n) = C - 1 + \frac{n!}{n^n}, \tag{47}$$

studied by several authors, e.g. in [6, 24, 32].

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

Table 1 shows the concrete values of the functions of the previous proof for $1 \leq n \leq 10$. We can observe in Table 1 that $\delta(n) = \kappa(n) - \frac{n!}{n^n}$ is increasing from $n = 1$ to $n = 8$, but for larger n is decreasing. Taking into account

$$\frac{n!}{n^n} = \left(\frac{n}{e}\right)^2 e^{\tau(n)}\sqrt{2\pi n}e^{\tau n} < \frac{\sqrt{2\pi n}}{e^n}e^1/(12n) < \frac{2.7\sqrt{n}}{e^n} < \frac{0.012}{n^2} \tag{48}$$

we can prove – using the same arguments as in the proof of Lemma 7 – the following assertion.

Theorem 9 *The expected running time of* Linear *is*

$$T_{exp}(n, \text{Linear}) = n + \sqrt{2\pi n} + \frac{7}{3} + \delta(n), \tag{49}$$

where $\delta(n)$ tends to zero when n tends to infinity, further

$$\delta(n+1) > \delta(n) \text{ for } 1 \leq n \leq 7 \text{ and } \delta(n+1) < \delta(n) \text{ for } n \geq 8. \tag{50}$$

If we wish to prove only the existence of some threshold index n_0 having the property that $n \geq n_0$ implies $\delta(n+1) < \delta(n)$, then we can use the following shorter proof.

Using (44) and (47) we get

$$\kappa(n) = C(n) - \sqrt{\frac{\pi n}{2}} - \frac{2}{3} - \frac{n!}{n^n} = Q(n) - \sqrt{\frac{\pi n}{2}} + \frac{1}{3}. \tag{51}$$

Substituting the power series

$$Q(n) = \sqrt{\frac{\pi n}{2}} - \frac{1}{3} + \frac{1}{12}\frac{\pi}{2n} - \frac{14}{135n} + \frac{1}{288}\frac{\pi}{2n^3} + O(n^{-2}) \tag{52}$$

of $Q(n)$ cited by D. E. Knuth [32, Equation (25) on page 120] into (51) and using

$$\frac{1}{n^{k/2}} - \frac{1}{(n+1)^{k/2}} = \Theta\left(\frac{1}{n^{1+k/2}}\right) \tag{53}$$

for $k = 1,\ 2,\ 3$ and 4 we get

$$\kappa(n) - \kappa(n+1) = \frac{\sqrt{\pi}}{12\sqrt{2}}\left(\frac{1}{\sqrt{n}} - \frac{1}{\sqrt{n+1}}\right) + O(n^{-2}), \tag{54}$$

implying

$$\kappa(n) - \kappa(n+1) = \frac{\sqrt{\pi}}{12\sqrt{2}} \frac{1}{\sqrt{n}\sqrt{n+1}(\sqrt{n} + \sqrt{n+1})} + O(n^{-2}), \tag{55}$$

guaranteeing the existence of the required n_0.

2.2 Running time of algorithm BACKWARD

BACKWARD compares the second (s_2), third (s_3), ..., last (s_n) element of the realization with the previous elements until the first collision or until the last pair of elements.

Taking into account the number of the necessary comparisons in line 04 of BACKWARD, we get $C_{best}(n, \text{BACKWARD}) = 1 = \Theta(1)$, and $C_{worst}(n, \text{BACKWARD}) = B(n, 2) = \Theta(n^2)$. The number of assignments is 1 in the best case (in line 01) and is 2 in the worst case (in lines 01 and in line 05). The expected number of assignments is $A_{exp}(n, \text{BACKWARD}) = 1 + \frac{n!}{n^n}$, since only the good realizations require the second assignment.

The next assertion gives the expected running time.

Theorem 10 *The expected running time of the algorithm* BACKWARD *is*

$$T_{exp}(n, \text{BACKWARD}) = n + \sqrt{\frac{\pi n}{8}} + \frac{4}{3} - \alpha(n), \tag{56}$$

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

Proof. Let $y(n)$ as defined in (41), p_k as defined in (16), and let

$$z = \{q : 1 \le q \le k \text{ and } s_1, s_2, \ldots, s_k \text{ are different and } s_{k+1} = s_q \mid y = k\}, \tag{57}$$

be a random variable characterising the index of the first repeated element of $\mathbf{s}$.

Let

$$q_i(k, n) = q_i = \Pr\{z = i | y = k\} \quad (k = 1,\ 2,\ \ldots,\ n;\ i = 1,\ 2,\ \ldots k) \tag{58}$$

and let $C_{exp}(n, \text{BACKWARD})$ denoted by C.

BACKWARD executes $B(k,2)$ comparisons among the elements $s_1,\ s_2,\ \ldots,\ s_k$, and s_{k+1} requires at least 1 and at most k comparisons (with exception of case $k = n$ when additional comparisons are not necessary). Therefore using the theorem of the full probability we have

$$C = \sum_{k=1}^{n-1} p_k \left(B(k,2) + \sum_{i=1}^{k} i q_i \right) + p_n B(n,2), \tag{59}$$

where

$$q_i(n,k) = q_i = \frac{1}{k} \quad (i = 1,\ 2,\ \ldots,\ k;\ k = 1,\ 2, \ldots,\ n). \tag{60}$$

Adding a new member to the first sum we get

$$C = \sum_{k=1}^{n} p_k \left(B(k,2) + \sum_{i=1}^{k} i q_i \right) - p_n \sum_{i=1}^{n} i q_i. \tag{61}$$

Using the uniform distribution (60) of z we can determine its contribution to C:

$$\sum_{i=1}^{k} q_i i = \sum_{i=1}^{k} \frac{i}{k} = \frac{k+1}{2}. \tag{62}$$

Substituting the contribution in (62) into (61), and taking into account Lemma 6 and Lemma 7 we have

$$C = \frac{1}{2} R_2(n) - \frac{1}{2} R_0(n) - \frac{n!}{n^n} \frac{n+1}{2}. \tag{63}$$

Now Lemma 6 and Lemma 7 result

$$C = n - \sqrt{\frac{\pi n}{8}} + \frac{2}{3} - \frac{1}{2}\kappa(n) - \frac{n!}{n^n} \frac{n+1}{2}. \tag{64}$$

Taking into account $A_{exp}(n, \text{BACKWARD}) = 1 + \frac{n!}{n^n} - \frac{n!}{n^n}\frac{n+1}{2}$ we get (56). ■

The following Table 2 represents some concrete numerical results. It is worth to remark that $\frac{n!}{n^n}\frac{n+1}{2} = \Theta(\frac{n\sqrt{n}}{e^n})$, while $\kappa(n) = \Theta(\frac{1}{\sqrt{n}})$, therefore $\kappa(n)$ decreases much slower than the other expression.

2.3 Running time of algorithm FORWARD

FORWARD compares the first (m_1), second (m_2), ..., last but one (m_{n-1}) element of the realization with the next elements until the first collision or until the last pair of elements.

Taking into account the number of the necessary comparisons in line 04 of FORWARD, we get $C_{best}(n, \text{FORWARD}) = 1 = \Theta(1)$, and $C_{worst}(n, \text{FORWARD}) = B(n,2) = \Theta(n^2)$.

The next assertion gives the expected running time.

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

Theorem 11 *The expected running time of the algorithm* Forward *is*

$$T_{exp}(n, \text{Forward}) = n + \Theta(\sqrt{n}). \tag{65}$$

Proof. ... ■

2.4 Running time of algorithm RANDOM

Random generates random pairs of elements and tests them until it finds two identical elements or it tested all the possible pairs of $\mathbf{s}$. It uses the procedure Ran(k) [14] generating a random integer value distributed uniformly in the interval $[1, k]$.

Random needs only $1 = \Theta(1)$ time in the best case. In the worst case its running time can be arbitrary large but the probability of a large running time is small.

Theorem 12 *The expected running time of* Random *is*

$$T_{exp}(n, \text{Random}) = n + \Theta(\lg n). \tag{66}$$

Proof. Algorithm Random can get two types of input: it gets a good input with probability $n!/n^n$ and a bad input with probability $(n^n - n!)/n^n$.

In the case of a good input the algorithm needs $n(n-1)/2$ different comparisons to observe that the investigated input is good. Using the known solution of the coupon collector's problem (see [14, page 109–110] or [50]) the expected number of the necessary comparisons is

$$C_{exp,\text{Random}} = n \sum_{i=1}^{B(n,2)} \frac{1}{i} = \Theta(n \lg n). \tag{67}$$

If RANDOM gets a bad input, then

$$\sum_{i=0}^{\infty} \left(\frac{n-1}{n}\right)^i i = 1 + \sum_{i=1}^{n} \left(1 - \frac{1}{i}\right) = n - \Theta(\lg n) \tag{68}$$

is an upper bound of its expected running time, and so

$$C_{exp}(n, \text{RANDOM}) = \frac{n!}{n^n} B(n,2)\Theta(n \lg n) + \frac{n^n - n!}{n^n} (n - \Theta(n)) = n\Theta(n). \tag{69}$$

■

2.5 Running time of algorithm TREE

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).

The worst case running time of TREE appears when the input contains different elements in increasing or decreasing order. Then the result is $\Theta(n^2)$. The best case is when the first two elements of $\mathbf{s}$ are equal, so $C_{best}(n, \text{TREE}) = 1 = \Theta(1)$.

Using the known fact that the expected height of a random search tree is $\Theta(\lg n)$ and Lemma 1 we can get that the order of the expected running time is $\sqrt{n} \log n$.

Theorem 13 *The expected running time of* TREE *is*

$$T_{exp}(n, \text{TREE}) = \Theta(\sqrt{n} \lg n). \tag{70}$$

2.6 Running time of algorithm GARBAGE

This algorithm is similar to LINEAR, but it works without the setting zeros into the elements of a vector requiring linear amount of time.

Beside the cycle variable i GARBAGE uses as working variable also a vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$. Interesting is that $\mathbf{v}$ is used without initialisation, that is its initial values can be arbitrary integer numbers.

Lemma 14 *The expected running time of* GARBAGE *is*

$$T_{exp}(n, \text{GARBAGE}) = \Theta(\sqrt{n}). \tag{71}$$

2.7 Running time of algorithm BUCKET

BUCKET divides the interval $[1, n]$ into $m = \sqrt{n}$ subintervals $I_1, I_2, \ldots, I_m$, where $I_j = [(j-1)m+1, jm]$, and sequentially puts the elements of $\mathbf{s}$ into the bucket B_j (we use the word bucket due to some similarity to bucket sort [14]): if $\lceil s_i/m \rceil = r$, then s_i belongs to B_r). BUCKET works until the first repetition (stopping with $g = \text{FALSE}$, or up to the processing of the last element s_n (stopping with $g = \text{TRUE}$).

$\textsc{Bucket}$ handles an array $Q[1:m, 1:m]$ (where $m = \lceil\sqrt{n}\rceil$ and puts the element s_i into the rth row of Q, and it tests using linear search whether s_j appeared earlier in the corresponding bucket. The elements of the vector $\mathbf{c} = (c_1, c_2, \ldots, c_m)$ are counters, where c_j $(1 \le j \le m))$ shows the actual number of elements in B_j.

For the simplicity let us suppose that m is a positive integer and $n = m^2$.

In the best case $s_1 = s_2$. Then $\textsc{Bucket}$ executes 1 comparisons in line 08, m assignments in line 04, and 1 assignment in line 01, 1 in line 02, 2 in line 06, and 1 in line 08, 11 and 12, therefore $T_{best}(n, \textsc{Bucket}) = m + 7 = \Theta(\sqrt{n})$. The worst case appears, when the input is bad. Then each bucket requires $1 + 2 + \cdots + m - 1 = B(n-1, 2)$ comparisons in line 08, further $3m$ assignments in lines 06, and 12, totally $\frac{m^2(m-1)}{2} + 3m^2$ operations. Lines 01, 02, and 09 require 1 assignment per line, and the assignment in line 04 is repeated m times. So $T_{worst}(n, \textsc{Bucket}) = \frac{m^2(m-1)}{2} + 3m^2 + m + 3 = \Theta(n^{3/2})$.

In connection with the expected behaviour at first we show that the expected number of elements in a bucket has a constant bound which is independent from n.

Lemma 15 *Let b_j $(j = 1,\ 2,\ \ldots,\ m)$ be a random variable characterising the number of elements in the bucket B_j at the moment of the first repetition. Then*

$$E\{b_j\} = \sqrt{\frac{\pi}{2}} - \mu(n) \quad \textit{for } j = 1,\ 2,\ \ldots,\ m, \tag{72}$$

where

$$\mu(n) = \frac{1}{3\sqrt{n}} - \frac{\kappa(n)}{\sqrt{n}}, \tag{73}$$

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

Proof. Due to the symmetry it is sufficient to prove (73) for $j = 1$.

Let m be a positive integer and $n = m^2$. Let $y(n) = y$ be the random variable defined in (41) and $p_k(n)$ be the probability defined in (16).

Let $A_i(k)$ $(i = 1,\ 2,\ \ldots,\ n; k = 1,\ 2,\ \ldots,\ n)$ be the event that at the condition $y = k$ the number i appears in $\mathbf{s}$ before the first repetition and $Y_i(k)$ be the indicator of $A_i(k)$. Then using the theorem of the full probability and the symmetry of the buckets we have

$$\mathrm{E}\{b_1\} = \sum_{i=1}^{m} Y_i(n) = \sum_{i=1}^{m} \Pr\{A_i(k)\} = m\Pr\{A_1(k)\} \tag{74}$$

and

$$\Pr\{A_1(k)\} = \Pr\{1 \in \{s_1,\ s_2,\ \ldots,\ s_k\}|y = k\} = \sum_{k=1}^{n} p_k \frac{k}{n} = \frac{1}{n}\sum_{k=1}^{n} p_k k = \frac{1}{n} R_1. \tag{75}$$

Using Lemma (7) we get

$$\mathrm{E}\{b_1\} = m\frac{1}{n}R_1 = \frac{m}{n}\left(\sqrt{\frac{\pi n}{2}} - \frac{1}{3} + \kappa(n)\right), \tag{76}$$

resulting (73) and (74).

We omit the proof of the monotonity of $\delta(n)$, since it is similar to the corresponding part in the proof of Theorem 11. ■

The following Table 3 shows some concrete values.

n	$\mathrm{E}\{b_1\}$	$\sqrt{\frac{\pi}{2}}$	$\frac{1}{3\sqrt{n}}$	$\frac{\kappa(n)}{\sqrt{n}}$	$\mu(n) = \frac{1}{3\sqrt{n}} - \frac{\kappa(n)}{\sqrt{n}}$
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 3: Values of $\mathrm{E}\{b_1\}$, $\sqrt{\frac{\pi}{2}}$, $\frac{1}{3\sqrt{n}}$, $\frac{\kappa(n)}{\sqrt{n}}$, and $\mu(n) = \frac{1}{3\sqrt{n}} - \frac{\kappa(n)}{\sqrt{n}}$ for $n = 1,\ 2,\ \ldots,\ 10$

Lemma 16 *Let f_n be a random variable characterising the number of comparisons executed in connection with the first repeated element. Then*

$$E\{f_n\} = \frac{1}{2} + \sqrt{\frac{\pi}{8}} - \eta(n), \tag{77}$$

where

$$\eta(n) = ????, \tag{78}$$

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

Proof. Let $p(i,j,k)$ be the probability of the event that there are k different elements before the first repetition the repeated element belongs to B_j and B_j contains i elements in the moment of the first repetition. Then

$$P(i,j,k) = \binom{m}{i}\binom{n-m}{k-i} k! \frac{i}{b} n^{k+i}). \tag{79}$$

Dividing the inner sum by $\binom{n}{k}$ we get the expected value of the square of a random variable ξ having hipergeeometric distribution with parameters n, m, and k. It is known that

$$\mathrm{E}\{\xi^2\} = (\mathrm{E}\{\xi\})^2 - Var\{\xi^\},\tag{80}$$

therefore

$$\mathrm{E}\{f_n\} = ???, \tag{81}$$

$$=???, \tag{82}$$

$$=\frac{1}{2}+\frac{\pi}{8}+??? \tag{83}$$

We get the monotonity of $\eta(n)$ as earlier. ■

Theorem 17 *The expected running time of* Bucket *is*

$$T_{exp}(n, \textsc{Bucket} = \left(1+3\sqrt{\frac{\pi}{2}}\right)\sqrt{n}+2+\phi(n), \sqrt{\frac{\pi}{2}}+\rho(n)+E\{f_n\}, \tag{84}$$

where

$$\phi(n) =, \tag{85}$$

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

Proof. Let $\mathbf{s} = (s_1,\ s_2,\ \ldots,\ s_n)$ be the input sequence of the algorithm Bucket. Bucket processes the input sequence using $m = \sqrt{n}$ buckets $B_1,\ B_2,\ \ldots, B_n$: it investigates the input elements sequentially and if the i-th input element s_i belongs to the interval $[(r-1)m+1, (r-1)m+2, \ldots,\ rm]$, then it sequentially compares s_i with the elements in the bucket B_r and finishes, if it finds a collision, or puts s_i into B_r, if s_i differs from all elements in B_r.

Let $y(n)$ be the random variable, defined in (12), and $p(k)$ the probability defined in (13). Let b_i be the random variable defined in Lemma 15, and $C_j(n)$ $(j = 1,\ 2,\ \ldots,\ m)$ be a random variable characterising the number of comparisons executed in B_j before the processing of the first repeated element, , and $C(n)$ a random variable characterising the number of necessary comparisons executed totally. Then due to the symmetry we have

$$\mathrm{E}\{C(n)\} = \mathrm{E}\left\{\sum_{j=1}^{m} C_j(n)\right\} + E\{f(n)\} = m\mathrm{E}\{C_1(n)\} + E\{f(n)\}. \tag{86}$$

The probability of the event $A(i_1, i_2, n, k)$ that the elements i_1 and i_2 $(1 \le i_1,\ i_2 \le m)$ will be compared before the processing of the first repeated element at the condition that $y = k$ and $2 \le k \le n$ equals to

$$\Pr\{A(i_1, i_2, n, k) | y = k \text{ and } 2 \le k \le n\} = \frac{\binom{n-2}{k-2}}{\binom{n}{k}} = \frac{k(k-1)}{n(n-1)}, \tag{87}$$

Since there are $\binom{m}{n}$ possible comparisons among the elements of the interval $[1, m]$, we have

$$E\{C_1(n)\} = \sum_{k=1}^{n} p_k \frac{k(k-1)}{n(n-1)} \binom{m}{2} = \frac{m(m-1)}{2n(n-1)} \left(\sum_{k=1}^{n} p_k k^2 - \sum_{k=1}^{n} p_k k\right), \tag{88}$$

from where using Lemma 6 and Lemma 7 we get

$$E\{C_1(n)\} = \frac{n-\sqrt{n}}{2n^2-2n}(R_2-R_1) = \frac{1}{2n+2\sqrt{n}}\left[2n-2\left(\sqrt{\frac{\pi n}{2}}-\frac{1}{3}+\kappa(n)\right)\right]. \quad (89)$$

This equality implies

$$E\{C_1(n)\} = 1-\frac{1}{n+\sqrt{n}}\left(\sqrt{n}-\sqrt{\frac{\pi n}{2}}+\frac{1}{3}-\kappa(n)\right). \quad (90)$$

As illustration we compute again some concrete values (see Table 4).

n	$\mathrm{E}\{C_1(n)\}$	$a(n)\left(\sqrt{n}-\sqrt{\frac{\pi n}{2}}+\frac{1}{3}-\kappa(n)\right)$	$\frac{a(n)}{n+\sqrt{n}}$	$1-\frac{a(n)}{n+\sqrt{n}}$
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 4: Values of $\mathrm{E}\{C_1(n)\}$, $\sqrt{\frac{\pi n}{2}}-\frac{1}{3}+\kappa(n)$, $a(n)=2n-2(\sqrt{\frac{\pi n}{2}}-\frac{1}{3}+\kappa(n))$, and $\frac{a(n)}{2n+2\sqrt{n}}$ for $n=1,\ 2,\ \ldots,\ 10$

From (83) taking into account (79) we get

$$E\{C(n)\} = \sqrt{n}-1-\sqrt{\frac{\pi}{2}}+\frac{1}{\sqrt{n}+1}\left(\kappa(n)-\sqrt{\frac{\pi}{2}}-\frac{4}{3}\right)+E\{f(n)\}, \quad (91)$$

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

Bucket requires 2 assignments in lines 01 and 02, $\sqrt{n}$ assignments in line 04, $R_1(n)$ assignments in line 06, $C+E\{f(n)\}$ assignments in line 08, $1-p_n$ expected assignment in line 09 and $2R_1$ assignments in lines 11 and 12 before the first repeated element, and $2\mathrm{E}\{f(n)\}-1$ assignments after the first repeated element.

Therefore the expected number of assignments of Bucket is

$$A_{exp}(n,\text{Bucket}) = A = 2+3R_1(n)+\mathrm{E}\{C(n)\}+1-\frac{n!}{n^n}+E\{f(n)\}. \quad (92)$$

Substituting $R_1(n)$ and $\mathrm{E}\{C(n)\}$ we get

$$A = 3\left(\sqrt{\frac{\pi n}{2}}-\frac{1}{3}+\kappa(n)\right)+\left(\sqrt{n}-1-\sqrt{\frac{\pi}{2}}\right)+\phi(n), \quad (93)$$

where

$$\phi(n) = \frac{\frac{2}{3} + \frac{\pi}{2} + \kappa(n)}{\sqrt{n}+1} + 3 + \mathrm{E}\{f(n)\} \tag{94}$$

implying

$$A = \sqrt{n}\left(1 + 3\sqrt{\frac{\pi}{2}}\right) + 1 - \sqrt{\frac{\pi}{2}} + \frac{\frac{2}{3} + \frac{\pi}{2} + \kappa(n)}{\sqrt{n}+1} + \mathrm{E}\{f(n)\}. \tag{95}$$

Summing up the expected number of comparisons in (84) and assignments in (88) we get the final formula (77). ■

3 Test of random arrays

Now let $A = [1 : n, 1 : n]$ be a two-dimensional random array. The array A is called *good,* if its all lines (rows and columns) contain a permutation of the elements $1,\ 2,\ \ldots,\ n$.

Theorem 18 *The expected running time of* MATRIX *is*

$$T_{exp}(n, \text{MATRIX}) = \sqrt{n} - O(1) + \lambda(n), \tag{96}$$

where $\lambda(n)$ *tends to zero when* n *tends to infinity.*

Proof. The **???** ■

4 Summary

Table 5 summarises the basic properties of the running times of the investigated algorithms.

Index and Algorithm	$T_{best}(n)$	$T_{worst}(n)$	$T_{exp}(n)$
1. BACKWARD	$\Theta(1)$	$\Theta(n^2)$	$\Theta(n)$
2. FORWARD	$\Theta(1)$	$\Theta(n^2)$	$\Theta(n)$
3. LINEAR	$\Theta(n)$	$\Theta(n)$	$n + \Theta(\sqrt{n})$
4. RANDOM	$\Theta(1)$	$\Theta(n^2 \lg n)$	$\Theta(n)$
5. TREE	$\Theta(1)$	$\Theta(n^2)$	$\Theta(\sqrt{n} \lg n)$
6. GARBAGE	$\Theta(1)$	$\Theta(n)$	$\Theta(\sqrt{n})$
7. BUCKET	$\Theta(\sqrt{n})$	$\Theta(n\sqrt{n})$	$\Theta(\sqrt{n})$
8. MATRIX	$\Theta(1)$	$\Theta(n\sqrt{n})$	$\Theta(\sqrt{n})$

Table 5: The running times of the investigated algorithms in best, worst and expected cases

We used in our calculations the RAM computation model [11, 14]. If the investigated algorithms run on real computers then we have to take into account also the limited capacity of the memory locations and the increasing execution time of the elementary arithmetical and logical operations.

5 Pseudocodes of the algorithms

The inputs of the following seven algorithms are n (the length of the sequence $\mathbf{s}$) and $\mathbf{s} = (s_1, s_2, \ldots, s_n)$, a sequence of nonnegative integers with $1 \le s_i \le n$ for $1 \le i \le n$) in all cases. The output is always a logical variable g (its value is TRUE, if the input sequence is good, and FALSE otherwise).

The working variables are usually the cycle variables i and j.

5.1 Definition of algorithm LINEAR

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 $v[s_i] > 0$ (signalising a repetition), then stops, otherwise adds 1 to v_k.

```
LINEAR(n, s)
g ← TRUE
for i ← 1 to n
    do v[i] ← 0
for i ← 1 to n
    do if v[s[i]] > 0
          then g ← FALSE
               return g
          else v[s[i]] ← v[s[i]] + 1
return g
```

5.2 Definition of algorithm BACKWARD

BACKWARD compares the second (i_2), third (i_3), $\ldots$, last (i_n) element of the realization $\mathbf{s}$ with the previous elements until the first collision or until the last pair of elements.

```
BACKWARD(n, s)
g ← TRUE
for i ← 2 to n
    do for j ← i − 1 downto 1
           do if s[i] = s[j]
                 then g ← FALSE
                      return g
return g
```

5.3 Definition of algorithm FORWARD

FORWARD compares the first (i_1), second (i_2), $\ldots$, 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.

```
FORWARD(n, s)
```

```
g ← TRUE
for i ← 1 to n − 1
    do for j ← i + 1 to n
        do if s[i] = s[j]
            then g ← FALSE
                return g
return g
```

5.4 Definition of algorithm Random

Random generates random pairs of elements and tests them until it finds two identical elements or it tested all the possible pairs of **s**. It uses the procedure Ran(k) [14] generating a random integer value distributed uniformly in the interval $[1, k]$.

Random(n, **s**)

```
g ← TRUE
while g = TRUE
    do i ← RAN(n^2)
       j ← (RAN(n^2 − 1) + j)  (mod n^2)
       if s_i = s_j
          then g ← FALSE
return g
```

5.5 Definition of algorithm Tree

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).

Tree(n, **s**)

```
g ← TRUE
let s[1] be the root of a tree
for i ← 2 to n
    if [s[i] is in the tree
       then g ← FALSE
            return
       else insert s[i] in the tree
return g
```

5.6 Definition of algorithm Garbage

This algorithm is similar to Linear, but it works without the setting zeros into the elements of a vector requiring linear amount of time.

Beside the cycle variable i GARBAGE uses as working variable also a vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$. Interesting is that $\mathbf{v}$ is used without initialisation, that is its initial values can be arbitrary integer numbers.

The algorithm GARBAGE was proposed by Gábor Monostori [37].

GARBAGE($n, \mathbf{s}$)

```
g ← TRUE
for i ← 1 to n
    do if v[s[i]] < i and s[v[s[i]]] = s[i]
          then g ← FALSE
               return g
          else v[s[i]] ← i
return g
```

5.7 Definition of algorithm BUCKET

BUCKET handles the array $Q[1 : m, 1 : m]$ (where $m = \lceil\sqrt{n}\rceil$ and puts the element s_i into the rth row of Q, where $r = \lceil s_i/m \rceil$ and it tests using linear search whether s_j appeared earlier in the corresponding row. The elements of the vector $\mathbf{c} = (c_1, c_2, \ldots, c_m)$ are counters, where c_j $(1 \le j \le m)$ shows the number of elements of the ith row.

For the simplicity we suppose that n is a square.

BUCKET($n, \mathbf{s}$)

```
g ← TRUE
m ← √n
for j ← 1 to m
    do c[j] ← 1
for i ← 1 to n
    do r ← ⌈s[i]/m⌉m
       for j ← 1 to c[r] − 1
           do if s[i] = Q[r, j]
                 then g ← FALSE
                      return g
                 else Q[r, c[r]] ← s[i]
                      c[r] ← c[r] + 1
return g
```

5.8 Definition of algorithm MATRIX

MATRIX is based on BUCKET **???**

For the simplicity let us suppose that n is a square.

MATRIX($n, \mathcal{M}$)

```
g ← TRUE
BUCKET(n, r_1)
```

```
if g = FALSE
   then return g
for i ← 2 to n
     do BUCKET(n, r_i)
          if g = FALSE
             then return g
for j ← 1 to n
     do BUCKET(n, c_j)
          if g = FALSE
             then return g
return TRUE
```

Acknowledgements. The authors thank Tamás F. Móri for proving Lemma 15 [39], Péter Burcsi for useful information on references, both are teachers of Eötvös Loránd University, Balázs Novák for the proof of (14), Gábor Monostori for proposing the algorithm GARBAGE, both are students of the same university.

The research was supported by by the project TÁMOP-4.2.1/B-09/1/KMR-2010-0003 of Eötvös Loránd University.

References

[1] **Adams P., Bryant D., and Buchanan M.,** Completing partial Latin squares with two filled rows and two filled columns. *Electron. J. Combin.* **15** (1) (2008), Research paper 56, 26 pages. MR2398848 (2009h:05035). http://www.combinatorics.org/. ⇒1

[2] **Anisiu M.-C., and Iványi A.,** Two-dimensional arrays with maximal complexity. *Pure Math. Appl.* (*PU.M.A.*) **17** (3–4) (2006), 197–204. MR2481414 (2010b:68124), http://www.bke.hu/puma/Contents.htm. ⇒1

[3] **Bailey R. A., Cameron P. J., and Connelly R.,** Sudoku, gerechte designs, resolutions, affine space, spreads, reguli, and Hamming codes. *Amer. Math. Monthly,* **115** (5) (2008), 383–404. MR2408485. http://www.math.cornell.edu/∼connelly/bcc_sudokupaper.pdf. ⇒1

[4] **Behrens W. U.,** Feldversuchsanordnungen mit verbessertem Ausgleich der Bodenunterschiede, *Zeitschrift für Landwirtschaftliches Versuchs- und Untersuchungswesen,* **2** (1956), 176–193. ⇒1

[5] **Brett S., Hurlbert G., and Jackson B.,** Preface [Generalisations of de Bruijn cycles and Gray codes]. *Discrete Math.*, **309** (17) (2009), 5255–5258. MR2548538, http://www.sciencedirect.com/science/journal/0012365X. ⇒1

[6] **Breusch R., and Gould H. W.** The truncated exponential series. *American Mathematical Monthly* **75** (9) (1968) 1019–1021. MR1535130. ⇒7

[7] **Bruen A. A., and Mollin R. A.,** Cryptography and shift registers. *Open Math. J.*, **2** (2009), 16–21. MR2529768 (2010g:9409), http://www.bentham.org/open/tomatj/openaccess2.htm. ⇒ 1

[8] **Buchanan H. L. II, and Ferencak M. N.,** On completing Latin squares. *J. Combin. Math. Combin. Comput.*, **34** (2000), 129–132. MR1772791 (2001b:05042). ⇒ 1

[9] **Burnett G. J., and Coffman, E. G.** Combinatorial problem related to interleaved memory systems. *Journal of ACM* **20** (1973) 39–45. MR0329045 (48 #7387) ⇒ 1

[10] **Cameron P. J.,** Sudoku - an alternative history. Talk to the Archimedeans. Queen Mary University of London, February 2007. http://www.maths.qmul.ac.uk/ pjc/slides/beamer/sudoku31.pdf. ⇒ 1

[11] **Casanova H., Legrand A., and Robert Y.** *Parallel Algorithms.* CRC Press, Boca Raton, 2009. MR2450469 (2009i:68106) ⇒ 17

[12] **Chen Z.,** Heuristic reasoning on graph and game complexity of sudoku. 6 pages. ARXIV, 2010. http://arxiv.org/PS_cache/arxiv/pdf/0903/0903.1659v1.pdf. ⇒ 1

[13] **Cooper J., and Heitsch, C.,** The discrepancy of the lex-least de Bruijn sequence. *Discrete Math.,* **310** (6-7) (2010), 1152–1159. MR2579847, http://www.sciencedirect.com/science/journal/0012365X. ⇒ 1

[14] **Cormen T. H., Leiserson C. E., Rivest R. L., and Stein C.**, *Introduction to Algorithms.* Third edition. The MIT Press, 2009. MR2572804 ⇒ 2, 11, 12, 17, 19

[15] **Crook J. F.,** A pencil-and-paper algorithm for solving Sudoku puzzles. *Notices Amer. Math. Soc.,* **56** (2009), 460–468. MR2482305 (2010c:05022). http://www.ams.org/notices/200904/tx090400460p.pdf ⇒ 1

[16] **Dahl G.,** Permutation matrices related to Sudoku. *Linear Algebra Appl.,* **430** (2009), 2457–2463. MR2508304 (2010d:05022). http://www.sciencedirect.com/science/journal/00243795 ⇒ 1

[17] **Dénes J., and Keedwell, A. D.,** *Latin Squares. New Developments in the Theory and Applications.* North-Holland, Amsterdam, 1991. ⇒ 1

[18] **Easton T., and Parker R. G.,** On completing Latin squares. *Discrete Appl. Math.*, **113** (2–3) (2001), 167–181. MR1857774 (2002j:05029), http://www.sciencedirect.com/science/journal/0166218X. ⇒ 1

[19] **Euler R.,** On the completability of incomplete Latin squares. *European J. Combin.* **31** (2010), 535–552. MR2565345. http://www.sciencedirect.com/science/journal/01956698. ⇒ 1

[20] **Hajirasouliha I., Jowhari H., Kumar R., and Sundaram R.,** On completing Latin squares. *Lecture Notes in Comput. Sci.*, **4393** (STACS2007), 524–535. Springer, Berlin, 2007. MR2362490 (2008k:68122). http://www.springerlink.com/content/105633/. ⇒ 1

[21] **Hellerman H.** *Digital Computer System Principles.* Mc Graw Hill, New York, 1967. ⇒ 1, 3

[22] **Heppes A., and Révész P.,** A new generalization of the concept of latin squares and orthogonal latin squares and its application to the design of experiments (in Hungarian). *Magyar Tud. Akad. Mat. Int. Közl.,* **1** (1956), 379–390. ⇒ 1

[23] **Horváth M., and Iványi, A.,** Growing perfect cubes. *Discrete Math.,* **308** (19) (2008) 4378–4388. MR2433864 (2009d:05036), http://www.sciencedirect.com/science/journal/0012365X. ⇒ 1

[24] **Iványi A., and Kátai I.,** Estimates for speed of computers with interleaved memory systems, *Annales Univ. Sci. Budapest., Sectio Mathematica*, **19** (1976), 159–164. ⇒ 1, 6, 7

[25] **Iványi A.,** On the *d*-complexity of words. *Ann. Univ. Sci. Budapest., Sect. Comput.* **8** (1987), 69–90 (1988). MR0974183 (90d:68063), http://compalg.inf.elte.hu/∼tony/Kutatas/PerfectArrays/On%20the%20d-complexity....pdf. ⇒ 1

[26] **Iványi A., and Tóth Z.,** Existence of de Bruijn words. *Second Conference on Automata, Languages and Programming Systems* (Salgótarján, 1988), 165–172, DM, 88-4, Karl Marx Univ. Econom., Budapest, 1988. MR1062451 (91k:05016) ⇒ 1

[27] **Iványi A.,** Construction of infinite de Bruijn arrays. *Discrete Appl. Math.* **22** (3) (1988/89), 289–293. MR0983252 (90d:05077), http://www.sciencedirect.com/science/journal/0166218X. ⇒ 1

[28] **Iványi A.,** Construction of three-dimensional perfect matrices. (Twelfth British Combinatorial Conference, Norwich, 1989). *Ars Combin.* **29C** (1990), 33–40. ⇒ 1

[29] **Iványi A., and Kátai I.,** Quick testing of random variables. In: *Proceedings of ICAI'2010* (Eger, January 27-30). To appear. http://icai.ektf.hu/index.php?p=11. ⇒ 1

[30] **Iványi A., and Kátai I.,** Testing of uniformly distributed vectors. In: *Abstracts of Bolyai150* (Budapest, August 28-30). Submitted. http://www.bolyai150.hu/eng/?page=page&name=Organizers. ⇒ 1

[31] **Iványi A., and Novák B.,** Testing of random sequences by simulation. In: *Abstracts of 8th MACS* (Komárnó, July 14–17, 2010). http://www.selyeuni.sk/macs/. ⇒ 1

[32] **Knuth D. E.,** *The Art of Computer Programming. Vol. 1. Fundamental Algorithms* (third edition). Addison–Wesley, 1997. MR0378456 (51 #14624). ⇒ 7, 9

[33] **Kuhl J. S., and Denley T.,** On a generalization of the Evans conjecture. *Discrete Math.* **308** (20) (2008), 4763–4767. MR2438179 (2009i:05047), http://www.sciencedirect.com/science/journal/0012365X. ⇒ 1

[34] **Kumar S. R., Russell A., and Sundaram R.** Approximating Latin square extensions. *Algorithmica* **24** (2) (1999), 128–138. MR1678015 (99m:68088). http://www.springerlink.com/content/1432-0541/. ⇒ 1

[35] **Lorch J.** Mutually orthogonal families of linear Sudoku solutions. *J. Aust. Math. Soc.,* **87** (3) (2009), 409–420. MR2576574. http://journals.cambridge.org/action/displayJournal?jid=JAZ. ⇒ 1

[36] **Matamala M., and Moreno E.,** Minimum Eulerian circuits and minimum de Bruijn sequences. *Discrete Math.,* **309** (17) (2009), 5298–5304. MR2548544, http://www.sciencedirect.com/science/journal/0012365X. ⇒ 1

[37] **Monostori G.,** Personal communication. Budapest, May 2010. ⇒ 20

[38] **Moon T. K., Gunther J. H., and Kupin J. J.** Sinkhorn solves Sudoku. *IEEE Trans. Inform. Theory*, **55** (4) (2009), 1741–1746. MR2582761 ⇒ 1

[39] **Móri T.,** Personal communication. Budapest, April 2010. ⇒ 21

[40] **Novák B.** Analysis of Sudoku algorithms (in Hungarian). MSc thesis. Eötvös Loránd University, Budapest, 2010. ⇒ 3

[41] **Öhman L.-D.** A note on completing Latin squares. *Australas. J. Combin.,* **45** (2009), 117–123. MR2554529. http://ajc.maths.uq.edu.au/. ⇒ 1

[42] **Pedersen R. M., Vis T. L.** Sets of mutually orthogonal Sudoku Latin squares. *College Math. J.,* **40** (3) (2009), 174–180. MR2512645 (2010d:05020) http://www-math.cudenver.edu/∼ tvis/Research/cmj174-181.pdf ⇒ 1

[43] **Penne R.,** A note on certain de Bruijn sequences with forbidden subsequences. *Discrete Math.*, **310** (4) (2010), 966–969. MR2574850, http://www.sciencedirect.com/science/journal/0012365X. ⇒ 1

[44] **Provan J. S.,** Sudoku: strategy versus structure. *Amer. Math. Monthly*, **116** (8) (2009), 702–707. http://www.jstor.org/journals/00029890.html ⇒ 1

[45] **Ramanujan S.** Question 294. *J. Indian Math. Society,* **3** (1928) 128–128. ⇒ 2

[46] **Sander T.,** Sudoku graphs are integral. *Electron. J. Combin.,* **16** (1) (2009), Note 25, 7 pages. MR2529816. http://www.combinatorics.org/. ⇒ 1

[47] **Szegő G.,** Über einige von S. Ramanujan gestellte Aufgaben. *J. London Math. Society,* **3** (1928), 225–232. JFM 54.0231.01 (old Zbl) See also in *Collected Papers of Gábor Szegő* (ed. by R. Askey), Birkhäuser, Boston, 1982. Volume 2, 141–152. ⇒ 2

[48] **Thom D.,** SUDOKU ist NP-vollständig. PhD Dissertation. Stuttgart, 2007. ⇒ 1

[49] **Vaughan E. R.,** The complexity of constructing gerechte designs. *Electron. J. Combin.,* **16** (1) (2009), paper R15, pp. 8. MR2475538 (2009k:05041). http://www.combinatorics.org/. ⇒ 1

[50] **Wikipedia**, Coupon collector's problem. http://en.wikipedia.org/wiki/Coupon_collector%27s_problem. ⇒ 11

[51] **Xu X., Cao Y., Xu J.-M., and Wu Y.,** Feedback numbers of de Bruijn digraphs. *Comput. Math. Appl.,* **59** (4) (2010), 716–723. MR2575561 http://www.ccsenet.org/journal/index.php/jmr/article/viewFile/3732/3336 ⇒ 1

[52] **Xu C., and Xu W.,** The model and algorithm to estimate the difficulty levels of Sudoku puzzles. *J. Math. Res.,* **11** (2) (2009), 43–46. http://www.ccsenet.org/journal/index.php/jmr/article/viewFile/3732/3336 ⇒ 1

[53] **Zhang W., Liu S., and Huang H.,** An efficient implementation algorithm for generating de Bruijn sequences. *Computer Standards & Interfaces,* **31** (6) (2009), 1190–1191. http://www.sciencedirect.com/science/journal/0012365X ⇒ 1

Iványi A., and Kátai I.,
Department of Computer Algebra
Eötvös Loránd University
H-1117 Budapest
Pázmány Péter sétány 1/C
Hungary
tony@compalg.inf.elte.hu
katai@compalg.inf.elte.hu