TESTING OF RANDOM MATRICES

A. Iványi and I. Kátai
(Budapest, Hungary) JULY 2, 2010

Dedicated to Professor Péter Simon on his 60th birthday

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

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

A realization $M = [m_{ij}]$ 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 $[2, 5, 7, 13, 23, 25, 26, 27, 28, 36, 33, 51, 53]$ 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 $n$ ($1 \leq n$) be integers and $X = [x_{ij}]_{1 \leq i \leq m, 1 \leq j \leq n}$ be an $m \times n$ sized matrix of independent random variables having joint uniform distribution

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

(1)

A realization $M = [m_{ij}]$ 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 $M$ 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.

Mathematics Subject Classification 2010: 68Q25
Key words and phrases: random sequences, efficiency of algorithms
2 Test of random vectors

We start with the first step of the testing of \( M \): describe and analyse several algorithms testing the first row of \( M \). The inputs of these algorithms are \( n \) (the length of the first row of \( M \)) and the elements of the first row \( m = (m_{11}, m_{12}, \ldots, m_{1n}) \). For the simplicity we use the notation \( s = (s_1, s_2, \ldots, s_n) \). 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 \((\frac{n}{k})\) by \( B(n,k) \) and the function \( \log_2 n \) by \( \lg n \).

We characterise the running time of the algorithms by the number of necessary assignments and comparisons and denote the running time of algorithm \( \text{Alg} \) by \( T_{\text{worst}}(n, \text{Alg}) \), \( T_{\text{best}}(n, \text{Alg}) \) and \( T_{\text{exp}}(n, \text{Alg}) \) 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

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

where

\[
 \frac{1}{12n+1} < \tau(n) < \frac{1}{12n},
\]

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

Let \( a_k(n) = a_k \) and \( S_i(n) = S_i \) as follows for any \( n \geq 1 \) integer:

\[
 a_k(n) = a_k = \frac{n^k}{k!} \quad (k = 0, 1, 2, \ldots),
\]

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

In (4) and (5) if \( n = k = i = 0 \), then \( n^k = k^i = 0 \). Solving a problem posed by S. Ramanujan [45] Gábor Szegő [47] proved the following connection between \( e^n \) and \( S_0 \).

**Lemma 2** ([47]) For the function \( \sigma(n) \) defined by equation

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

hold \( \sigma(0) = \frac{1}{6} \) and \( \sigma(n) \) tends monotonically decreasing to zero when \( n \) tends to \( \infty \).

The following lemma shows the connection between \( S_i \) and \( S_{i-1} \).
Lemma 3 If $i \geq 1$ is integer then
\[ S_i(n) = n \sum_{k=0}^{i-1} B(i-1,k) S_k - n^{i-1} a_{n-1}. \]  
\[ (7) \]

Proof. Omitting the member belonging to the index $k = 0$ in $S_i$, then simplifying by $k$ and using the substitution $k = j$ we get
\[ S_i = \sum_{k=0}^{n-1} \frac{n^k}{k!} k^i = \sum_{k=1}^{n-1} \frac{n^{k-1}}{(k-1)!} k^{i-1} = n \sum_{j=0}^{n-2} \frac{n^j}{j!} (j+1)^{i-1}. \]  
\[ (8) \]
Completing the sum with the member belonging to index $j = n - 1$ results
\[ S_i = n \sum_{j=0}^{n-1} \frac{n^j}{j!} (j+1)^{i-1} - n^i a_{n-1}. \]  
\[ (9) \]
Now applying the binomial theorem results \[ (8). \]

In this paper we need only the following consequences of Lemma 3.

Lemma 4 If $n \geq 1$ then
\[ S_1 = nS_0 - na_{n-1}, \quad S_2 = S_0(n^2 + n) - 2n^2 a_n, \]  
\[ (10) \]
and
\[ S_3 = S_0(n^3 + 3n^2 + n) - (3n^3 + 2n^2) a_n. \]  
\[ (11) \]

Proof. Using recursively Lemma 3 for $i = 1, 2$ and 3 we get the required formula for $S_1, S_2,$ and $S_3$. \[ \square \]

We remark that \[ (11) \] was proved by Balázs Novák in \[ [40] \] directly.
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 \( \{1, 2, \ldots, n\} \). Let \( m = (m_1, m_2, \ldots, m_n) \) be the input sequence (a realization of \( x \)).

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 \( m \) 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 $T_{\exp}(n, \text{LINEAR})$. 

\[ \text{Testing of random matrices} \]
Theorem 5 The expected running time of Linear is

\[ T_{\text{exp}}(n, \text{Linear}) = n + \sqrt{2\pi n} + \frac{7}{3} - 2 \frac{n!}{n^n} + 2\kappa(n), \]  

(12)

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

Proof. Let

\[ y(n) = y = \max\{k : 1 \leq k \leq n \text{ and } m_1, m_2, \ldots, m_k \text{ are different} \} \]  

(13)

be a random variable characterising the maximal length of the prefix of \( \mathbf{m} \) containing different elements, and

\[ \Pr\{y = k\} = p_k(n) = p_k \]  

(14)

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

If \( y = k \) and \( 1 \leq k \leq n - 1 \), then Linear executes \( k + 1 \) comparisons, and only \( n \) comparisons, if \( y = n \), therefore

\[ C = \sum_{k=1}^{n-1} p_k(k + 1) + p_n n = \sum_{k=1}^{n} p_k(k + 1) - p_n = 1 - \frac{n!}{n^n} + \sum_{k=1}^{n} p_k k, \]  

(15)

where \[ p_k = \frac{n}{n} \ldots \frac{n}{n} \ldots \frac{n}{n} \frac{n-k+1}{n} \frac{n-k}{n} \ldots \frac{n-1}{n} \frac{n}{n} = \frac{n!k}{(n-k)!n^k} \]  

(16)

Substitution of (16) into (15) and substitution of \( n-k \) by \( j \) results

\[ C = 1 - \frac{n!}{n^n} + \sum_{k=1}^{n} \frac{n!k^2}{(n-k)!n^{k+1}} = 1 - \frac{n!}{n^n} + \frac{n!}{n^n} \sum_{j=0}^{n-1} \frac{n^j}{j!} (n-j)^2. \]  

(17)

After simple arithmetic operations we get

\[ C = 1 - \frac{n!}{n^n} + \frac{n!}{n^{n+1}} (n^2S_0 - 2nS_1 + S_2), \]  

(18)

from where using Lemma 4 results

\[ C = 1 - \frac{n!}{n^n} + \frac{n!}{n^n} \left[ S_0(n^2 - 2n^2 + n^2 + n) + (2n^2a_{n-1} - 2n^2a_n) \right], \]  

(19)

and finally we get

\[ C = 1 - \frac{n!}{n^n} + \frac{n!}{n^n} S_0. \]  

(20)
Using Szegö’s result (Lemma [2]) we have

$$C = 1 - \frac{n!}{n^n} + \frac{n!}{n^n} \left( \frac{e^n}{2} - \frac{n^n}{3} \frac{n!}{n!} - \sigma(n) \frac{n^n}{n!} \right) = \frac{2}{3} - \frac{n!}{n^n} + \frac{\sqrt{2\pi n}}{n^n} \left[ \frac{e^n}{2} - \sigma(n) \right].$$ \hspace{1cm} (21)

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

$$C = \frac{2}{3} - \frac{n!}{n^n} + \frac{1}{n^n} \left( \frac{n}{e} \right)^n \sqrt{2\pi n} \left[ 1 + (e^{\tau(n)} - 1) \right] \left[ \frac{e^n}{2} - \sigma(n) \right].$$ \hspace{1cm} (22)

The product \( P \) of the expressions in the square brackets is

$$P = \frac{e^n}{2} + \frac{e^n}{2} (e^{\tau(n)} - 1) - \sigma(n) e^{\tau(n)},$$ \hspace{1cm} (23)

therefore

$$C = \sqrt{\frac{\pi n}{2}} + \frac{2}{3} - \frac{n!}{n^n} + \kappa(n),$$ \hspace{1cm} (24)

where

$$\kappa(n) = \kappa_1(n) - \kappa_2(n) = \sqrt{\frac{\pi n}{2}} \left( e^{\tau(n)} - 1 - \frac{2\sigma(n) e^{\tau(n)}}{e^n} \right).$$ \hspace{1cm} (25)

It remained to show the monotonity of \( \kappa(n) \). Before to do it let us compute some concrete values.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( C )</th>
<th>( \sqrt{\pi n/2} + 2/3 )</th>
<th>(-n!/n^n)</th>
<th>( \kappa(n) )</th>
<th>( \delta(n) = \kappa(n) - n!/n^n )</th>
<th>( \sigma(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000</td>
<td>1.9199</td>
<td>-1.0000</td>
<td>0.0800</td>
<td>-0.9200</td>
<td>0.026188</td>
</tr>
<tr>
<td>2</td>
<td>2.0000</td>
<td>2.4391</td>
<td>-0.5000</td>
<td>0.0609</td>
<td>-0.4371</td>
<td>0.013931</td>
</tr>
<tr>
<td>3</td>
<td>2.6667</td>
<td>2.8374</td>
<td>-0.2222</td>
<td>0.0506</td>
<td>-0.1716</td>
<td>0.009504</td>
</tr>
<tr>
<td>4</td>
<td>3.1250</td>
<td>3.1732</td>
<td>-0.0937</td>
<td>0.0455</td>
<td>-0.0482</td>
<td>0.007205</td>
</tr>
<tr>
<td>5</td>
<td>3.4720</td>
<td>3.4692</td>
<td>-0.0384</td>
<td>0.0412</td>
<td>+0.0028</td>
<td>0.005800</td>
</tr>
<tr>
<td>6</td>
<td>3.7590</td>
<td>3.7366</td>
<td>-0.0154</td>
<td>0.0378</td>
<td>+0.0224</td>
<td>???</td>
</tr>
<tr>
<td>7</td>
<td>4.0120</td>
<td>3.9826</td>
<td>-0.0061</td>
<td>0.0355</td>
<td>+0.0294</td>
<td>???</td>
</tr>
<tr>
<td>8</td>
<td>4.2430</td>
<td>4.2126</td>
<td>-0.0025</td>
<td>0.0339</td>
<td>+0.0314</td>
<td>???</td>
</tr>
<tr>
<td>9</td>
<td>4.4570</td>
<td>4.4266</td>
<td>-0.0009</td>
<td>0.0313</td>
<td>+0.0304</td>
<td>???</td>
</tr>
<tr>
<td>10</td>
<td>4.6600</td>
<td>4.6300</td>
<td>-0.0004</td>
<td>0.0303</td>
<td>+0.0299</td>
<td>???</td>
</tr>
</tbody>
</table>

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

The monotonity of \( \kappa(n) \) is equivalent with

$$\gamma(n) = \frac{\kappa(n) + 1}{\kappa(n)} = \frac{\kappa_1(n) + 1 + \kappa_2(n) + 1}{\kappa_1(n) + \kappa_2(n)} < 1 \quad \text{for} \quad n = 1, 2, \ldots, n - 1.$$ \hspace{1cm} (26)

Numerical results in Table 1 show that this inequality holds for \( 1 \leq n \leq 7 \).
Since all $\kappa$ functions are positive for all permitted $n$’s, therefore $\kappa_2(n)$ can be omitted from (29). 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

$$\frac{2\sigma(n)e^{\tau(n)}}{e^n} < 2 \cdot 0.0058 \cdot 1.02 < \frac{0.012}{n^2} \quad \text{for } n \geq 10.$$  \hfill (27)

Using (28) and the Lagrange remainder of the Taylor series of the function $e^x$ we have

$$\frac{\kappa(n+1)}{\kappa(n)} < \frac{\sqrt{n+1}}{\sqrt{n}} \frac{\tau(n+1) + \tau^2(\xi_{n+1})/2}{\tau(n) + \tau^2(\xi_n)/2 - \frac{0.012}{n^2}}, \quad \text{where } 0 < \xi_{n+1} < n+1 \text{ and } 0 < \xi_n < n,$$

therefore using Lemma 1 we get

$$\frac{\kappa_1(n+1)}{\kappa_1(n)} = \frac{\sqrt{n+1}}{\sqrt{n}} \frac{12n}{12n + 1} \left(1 + \frac{13}{12n} \right) < 1$$ \hfill (29)

Now multiplying the nominator and denominator by $(12n)^2$ results

$$\frac{\kappa_1(n+1)}{\kappa_1(n)} = \frac{\sqrt{n+1}}{\sqrt{n}} \frac{12n}{12n + 1} \left(1 + \frac{13}{12n} \right) < 1.$$ \hfill (30)

Since

$$\frac{1}{12n} + \frac{1}{2} \left( \frac{1}{12n} \right)^2 - \frac{0.012}{n^2} < 12n + 11 + \frac{13}{12n},$$ \hfill (31)

(30) and (31) imply

$$\frac{\kappa_1(n+1)}{\kappa_1(n)} < \frac{\sqrt{144n^3 + 144n^2}}{144n^3 + 264 \cdot n^2 + 137 + \frac{169}{144n}} < 1$$ \hfill (32)

finishing the proof of the monotony of $\kappa(n)$.

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 $C_{exp}(n, \text{LINEAR})$. Therefore

$$T_{exp}(n, \text{LINEAR}) = n + 1 + 2C_{exp}(n, \text{LINEAR}).$$ \hfill (33)

Substituting (24) into (33) results the required (12). \hfill ■

We remark, that (17) is equivalent with

$$C = 1 - \frac{n!}{n^n} + \frac{n - 1}{n} + \frac{n - 1}{n} \frac{n - 2}{n} + \cdots + \frac{n - 1}{n} \frac{n - 2}{n} \cdots \frac{1}{n},$$ \hfill (34)

demonstrating the close connection with the function

$$Q(n) = C - 1 + \frac{n!}{n^n},$$ \hfill (35)
studied by several authors, e.g. in [6, 24, 32]. The monotony of $\kappa_n$ was published in [24] without proof.

We can observe in Table 1 that $\delta(n) = \kappa(n) - \frac{n!}{n^n}$ is increasing from $n = 1$ to $n = 7$, but for larger $n$ is decreasing. Taking into account

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

we can prove – using the same arguments as in the proof of Theorem 5 – the following assertion.

**Theorem 6** The expected running time of Linear is

$$T_{exp}(n, \text{Linear}) = n + \sqrt{2 \pi n} + \frac{7}{3} + \delta(n), \tag{37}$$

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

$$\delta(n+1) > \delta(n) \text{ for } 1 \leq n \leq 7 \text{ and } \delta(n+1) < \delta(n) \text{ for } n \geq 8. \tag{38}$$

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 (24) and (35) we get

$$\kappa(n) = C(n) - \sqrt{\frac{\pi n}{2}} - \frac{2}{3} - \frac{n!}{n^n} = Q(n) - \sqrt{\frac{\pi n}{2}} + \frac{1}{3}. \tag{39}$$

Substituting the power series

$$Q(n) = \sqrt{\pi n} 2 - \frac{1}{3} + \frac{1}{12} \frac{\pi}{2n} - \frac{14}{135} + \frac{1}{288} \frac{\pi}{2n^3} + O(n^{-2}) \tag{40}$$

of $Q(n)$ cited by D. E. Knuth [32, Equation (25) on page 120] into (39) and using

$$\frac{1}{n^{k/2}} - \frac{1}{(n+1)^{k/2}} = \Theta \left( \frac{1}{n^{1+k/2}} \right) \tag{41}$$

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

$$\kappa(n+1) - \kappa(n) = \frac{\sqrt{\pi}}{12 \sqrt{2}} \left( \frac{1}{\sqrt{n}} - \frac{1}{\sqrt{n+1}} \right) + O(n^{-2}), \tag{42}$$

implying

$$\kappa(n+1) - \kappa(n) = \frac{\sqrt{\pi}}{12 \sqrt{2}} \frac{1}{\sqrt{n} \sqrt{n+1} \sqrt{n+1}} + O(n^{-2}), \tag{43}$$

guaranteeing the existence of the required $n_0$. 

2.2 Running time of algorithm BACKWARD

BACKWARD compares the second \((m_2)\), third \((m_3)\), \ldots, last \((m_n)\) 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 \(A_{exp}(n, \text{BACKWARD}) = 1 + \frac{n!}{n^n}, \text{since only the good realizations require the second assignment.}\)

The expected running time is

\begin{equation}
T_{exp}(n, \text{BACKWARD}) = n + \sqrt{\frac{\pi n}{8} + \frac{4}{3}} - \alpha(n),
\end{equation}

where \(\alpha(n) = \frac{k(n)}{2} + \frac{n!}{n^n} \frac{n+1}{2}\) monotonically decreasing tends to zero when \(n\) tends to infinity.

**Proof.** Let \(y(n)\) as defined in (13), \(p_k\) as defined in (14), and let

\[
z = \{q: 1 \leq q \leq k \text{ and } m_1, m_2, \ldots, m_k \text{ are different and } m_{k+1} = m_q \mid y = k\},
\]

be a random variable characterising the index of the first repeated element of \(m\).

Let

\[
q_i(k, n) = \Pr\{z = i \mid y = k\} \quad (k = 1, 2, \ldots, n; \ i = 1, 2, \ldots k)
\]

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

BACKWARD executes \(B(k, 2)\) comparisons among the elements \(m_1, m_2, \ldots, m_k\), and \(m_{k+1}\) 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

\[C = \sum_{k=1}^{n-1} p_k \left( B(k, 2) + \sum_{i=1}^{k} iq_i \right) + p_n B(n, 2),\]

where

\[q_i(n, k) = q_i = \frac{1}{k} \quad (i = 1, 2, \ldots, k; \ k = 1, 2, \ldots, n).\]

Adding a new member to the first sum we get

\[C = \sum_{k=1}^{n} p_k \left( B(k, 2) + \sum_{i=1}^{k} iq_i \right) - p_n \sum_{i=1}^{n} iq_i.\]
Testing of random matrices

Using the uniform distribution (48) of \( z \) we can determine its contribution to \( C \):

\[
\sum_{i=1}^{k} q_{i} = \sum_{i=1}^{k} \frac{i}{k} = \frac{k+1}{2}.
\]  

(50)

Substituting the probability (14) and the contribution in (50) into (49) we have

\[
C = \sum_{k=1}^{n} \frac{n!}{(n-k)!n^{k+1}} \frac{k^3 + k}{2} - \frac{n!}{n^n} \frac{n+1}{2}.
\]  

(51)

The substitution \( n-k = j \) results

\[
C = \frac{n!}{2n^{n+1}} \left[ \sum_{j=0}^{n-1} j^{n} \left( (n-j)^3 + (n-j) \right) \right] - \frac{n!}{n^n} \frac{n+1}{2},
\]  

further

\[
C = \frac{n!}{2n^{n+1}} \left[ \sum_{j=0}^{n-1} j^{n} \left( -j^3 + 3j^2 n - j(3n^2 + 1) + (n^3 + n) \right) \right] - \frac{n!}{n^n} \frac{n+1}{2},
\]  

implying

\[
C = \frac{n!}{2n^{n+1}} \left[ -S_3 + 3nS_2 - (3n^2 + 1)S_1 + (n^3 + n)S_0 \right] - \frac{n!}{n^n} \frac{n+1}{2}.
\]  

(53)

Using Lemma 4 we get

\[
C = \frac{n!}{2n^{n+1}} \left[ 3nS_0(n+1) - S_0(n^2 + 3n + 1) - (3n^2 + 1)S_0 + (n^3 + n)S_0 \right] + \beta(n),
\]  

(55)

where

\[
\beta(n) = \frac{n!}{2n^{n+1}} \left[ -(3n^3 + 2n^2)a_n - 6n^2a_n + (3n^3 + n)a_{n-1} \right] - \frac{n!}{n^n} \frac{n+1}{2}.
\]  

(56)

The result of simple arithmetic operations is

\[
C = \frac{n!}{2n^{n+1}} \left[ (2n^2 + n) - nS_0 \right] - \frac{n!}{n^n} = n + \frac{1}{2} - \frac{1}{2} \frac{n!}{n^n} \frac{S_0}{n} - \frac{n!}{n^n} \frac{n+1}{2}
\]  

(57)

We have seen in the proof of Theorem 5 that

\[
\frac{n!}{n^n} S_0 = \sqrt{\frac{\pi n}{2}} - \frac{1}{3} + \kappa(n).
\]  

(58)

Using this result we get

\[
C = n - \sqrt{\frac{\pi n}{2}} + \frac{2}{3} - \frac{1}{2} \kappa(n) - \frac{n!}{n^n} \frac{n+1}{2}.
\]  

(59)

Taking into account \( A_{exp}(n, \text{BACKWARD}) = 1 + \frac{n!}{n^n} \) and (59) we get (44).

Let us compute some concrete values. The following Table 4 represents some concrete numerical results. It is worth to remark that \( \frac{n!}{n^n} \frac{n+1}{2} = \Theta n \sqrt{\pi n} \), while \( \kappa n = \Theta \frac{1}{\sqrt{n}} \), therefore \( \kappa n \) decreases much quicker than the other expression.
Table 2: Values of 

\[ n - \sqrt{\pi n/8} + 2/3, \frac{n!}{n} \frac{n+1}{2}, -\kappa(n), \alpha(n) = \frac{\kappa(n)}{2} + \frac{n! n+1}{2} \]

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
\( n \) & \( C \) & \( n - \sqrt{\pi n/8} + 2/3 \) & \( \frac{n!}{n} \frac{n+1}{2} \) & \( -\kappa(n) \) & \( \alpha(n) = \frac{\kappa(n)}{2} + \frac{n! n+1}{2} \) \\
\hline
1 & 0.00000 & 0.20667 & 1.00000 & -0.08002 & 1.04004 \\
2 & 1.00000 & 0.78044 & 0.75000 & -0.06088 & 0.88127 \\
3 & 2.11111 & 2.58127 & 0.44444 & -0.05144 & 0.47016 \\
4 & 3.15625 & 3.41336 & 0.23438 & -0.04545 & 0.25711 \\
5 & 4.12960 & 4.26542 & 0.11520 & -0.04124 & 0.13582 \\
6 & 5.05864 & 5.13168 & 0.05401 & -0.03806 & 0.07304 \\
7 & 5.96645 & 6.00869 & 0.02448 & -0.03552 & 0.04224 \\
8 & 6.86668 & 6.89421 & 0.00818 & -0.03346 & 0.02754 \\
9 & 7.76616 & 7.78670 & 0.00468 & -0.03171 & 0.02054 \\
\hline
\end{tabular}
\end{center}

2.3 Running time of algorithm FORWARD

FORWARD compares the first \( (m_1) \), second \( (m_2) \), \ldots, 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 \( C_{\text{best}}(n, \text{FORWARD}) = 1 = \Theta(1) \), and \( C_{\text{worst}}(n, \text{FORWARD}) = B(n, 2) = \Theta(n^2) \).

The next assertion gives the expected running time.

**Theorem 8** The expected running time of the algorithm FORWARD is

\[ T_{\text{exp}}(n, \text{FORWARD}) = n + \Theta(\sqrt{n}). \]

**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 \( s \). It uses the procedure \( \text{RAN}(k) \) 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 9** The expected running time of RANDOM is

\[ T_{\text{exp}}(n, \text{RANDOM}) = \Theta(n). \]

**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 \cite{14} page 109–110 or \cite{50}) the expected number of the necessary comparisons is

\[
C_{\text{good}} = n \sum_{i=1}^{B_n} \frac{1}{i} = \Theta(n \log n). \tag{61}
\]

If RANDOM gets a bad input, then

\[
\sum_{i=0}^{\infty} \left( \frac{n - 1}{n} \right)^i 1 = 1 + \sum_{i=1}^{n} (1 - \frac{1}{i}) = n - O(\log n) = \Theta(n) \tag{62}
\]

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

\[
C_{\text{exp}}(n, \text{RANDOM}) \leq \frac{n!}{n^n} \Theta(n \log n) + \frac{n^n - n!}{n^n} \Theta(n) = \Theta(n). \tag{63}
\]

\[\Box\]

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 \( \mathbf{m} \) are equal, so \( C_{\text{best}}(n, \text{TREE}) = 1 = \Theta(1) \).

Using the known fact that the expected height of a random search tree is \( \Theta(\log n) \) and Lemma 11 we can get that the order of the expected running time is \( \sqrt{n} \log n \).

**Theorem 10** The expected running time of TREE is

\[
T_{\text{exp}}(n, \text{TREE}) = \Theta(\sqrt{n} \log n).
\]

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 11** The expected running time of GARBAGE is

\[
T_{\text{exp}}(n, \text{GARBAGE}) = \Theta(\sqrt{n}).
\]
2.7 Running time of algorithm BUCKET

BUCKET divides the interval \([1, n]\) into \(m = \sqrt{n}\) subintervals \(B_1, B_2, \ldots, B_m\), where \(B_j = [(j-1)m+1, jm]\), and sequentially puts the elements of \(S\) into the corresponding interval (or bucket, due to some similarity to bucket sort [13]): if \(s_i/m = r\), 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 \(g = \text{TRUE}\)).

BUCKET handles an array \(Q[1 : m, 1 : m]\) (where \(m = \lceil\sqrt{n}\rceil\)) and puts the element \(s_i\) into the \(r\)th row of \(Q\), and it tests using linear search whether \(s_j\) appeared earlier in the corresponding bucket. The elements of the vector \(c = (c_1, c_2, \ldots, c_m)\) are counters, where \(c_j (1 \leq j \leq m))\) shows the 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 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 \(T_{\text{best}}(n, \text{BUCKET}) = m + 7 = \Theta(n).\) 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 \(m^2(m-1)/2 + 3m^2\) operations. Lines 01, 02, and 09 require 1 assignment per line, and the assignment in line 04 is repeated \(m\) times. So \(T_{\text{worst}}(n, \text{BUCKET}) = m^2(m-1)/2 + 3m^2 + m + 3 = \Theta(n^{3/2}).\)

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 12** Let \(\beta_j (j = 1, 2, \ldots, m)\) be a random variable characterizing the number of elements in the bucket \(B_j\) at the moment of the first repetition. Then

\[
E(\beta_j) = \frac{\pi}{2} - \delta(n),
\]

where \(\delta(n) = \frac{n}{\sqrt{n}} - \frac{1}{3}\sqrt{n}\) and \(\delta(n)\) tends monotonically decreasing to zero when \(n\) tends to infinity.

**Proof.** Due to the symmetry it is sufficient to prove (64) for \(j = 1\).

Let \(m\) be a positive integer and \(n = m^2\). Let \(y(n) = y\) be the random variable defined in (13) and \(p_k(n)\) be the probability defined in (14).

Let \(A_i (i = 1, 2, \ldots, n)\) be the event that the number \(i\) appears in \(S\) before the first repetition and \(Y_i\) be the indicator of \(A_i\). Then using the theorem of the full probability we have

\[
E(\beta_1) = \sum_{i=1}^{m} Y_i(n) = \sum_{i=1}^{m} \Pr\{A_i\} = m \Pr\{A_1\}
\]

and

\[
\Pr\{A_1\} = \Pr\{1 \in (s_1, s_2, \ldots, s_k) | y = k\} = \sum_{k=1}^{n} \frac{k}{n} = \frac{1}{n} \sum_{k=1}^{n} p_k k.
\]
Comparing (66) with (13) and (35) we get the close connection
\[ \Pr\{A_1\} = \frac{Q_n}{n}. \] (67)
resulting
\[ E\{\beta_1\} = \sqrt{\frac{\pi}{2}} - \delta(n), \] (68)
where
\[ \delta(n) = \frac{\kappa(n)}{\sqrt{n}} - \frac{1}{3\sqrt{n}}. \] (69)
We omit the proof of the monotonicity of \( \delta(n) \), since it is similar to the proof of Theorem 8.

The following table 3 shows some concrete values.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( E{\beta_1} )</th>
<th>( \frac{\pi}{2} )</th>
<th>( \frac{1}{3\sqrt{n}} )</th>
<th>( \frac{\kappa(n)}{\sqrt{n}} )</th>
<th>( \frac{\kappa(n)}{\sqrt{n}} \frac{1}{3\sqrt{n}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.999981</td>
<td>1.253314</td>
<td>0.333333</td>
<td>-0.08002</td>
<td>-0.253333</td>
</tr>
<tr>
<td>2</td>
<td>1.??????</td>
<td>1.253314</td>
<td>0.???????</td>
<td>-0.0?????</td>
<td>-0.136226</td>
</tr>
<tr>
<td>3</td>
<td>1.143918</td>
<td>1.253314</td>
<td>0.111111</td>
<td>-0.01715</td>
<td>-0.109396</td>
</tr>
<tr>
<td>4</td>
<td>1.171117</td>
<td>1.253314</td>
<td>0.083333</td>
<td>-0.01136</td>
<td>-0.082197</td>
</tr>
<tr>
<td>5</td>
<td>1.197473</td>
<td>1.253314</td>
<td>0.066666</td>
<td>-0.00825</td>
<td>-0.065841</td>
</tr>
<tr>
<td>6</td>
<td>1.198428</td>
<td>1.253314</td>
<td>0.055555</td>
<td>-0.00669</td>
<td>-0.054886</td>
</tr>
<tr>
<td>7</td>
<td>1.206202</td>
<td>1.253314</td>
<td>0.047619</td>
<td>-0.00507</td>
<td>-0.047112</td>
</tr>
<tr>
<td>8</td>
<td>1.221698</td>
<td>1.253314</td>
<td>0.041616</td>
<td>-0.00???</td>
<td>-0.0?????</td>
</tr>
<tr>
<td>9</td>
<td>1.225169</td>
<td>1.253314</td>
<td>0.037033</td>
<td>-0.00463</td>
<td>-0.0?????</td>
</tr>
<tr>
<td>10</td>
<td>1.???????</td>
<td>1.253314</td>
<td>0.033333</td>
<td>-0.00317</td>
<td>-0.0?????</td>
</tr>
</tbody>
</table>

Table 3: Values of \( E\{\beta_1\} \), \( \frac{\pi}{2} \), \( \frac{1}{3\sqrt{n}} \), \( \frac{\kappa(n)}{\sqrt{n}} \) and \( \frac{\kappa(n)}{\sqrt{n}} \frac{1}{3\sqrt{n}} \) for \( n = 1, 2, \ldots, 10 \)

**Theorem 13** The expected running time of **Bucket** is
\[ T_{\text{exp}}(n, \text{Bucket}) = \Theta(\sqrt{n}). \] (70)

**Proof.** Let \( n \geq 2 \) be a positive integer and let \( X_1, X_2, \ldots, X_n \) be independent random variables having join uniform distribution on the set \( \{1, 2, \ldots, n\} \). Let \( s_1, s_2, \ldots, s_n \) 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 set \( \{(r - 1)n/m + 1, (r - 1)n/m + 2, \ldots, rn/m\} \), then **Bucket** sequentially compares \( s_i \) with the elements in the bucket \( B_r \) and finishes, if it finds a collision, or puts \( s_i \) into the bucket, if \( s_i \) differs from all elements in \( B_r \).
Let $y(n)$ be the random variable, defined in (12). Let $H_r(n)$ denote the number of the elements put into $B_r$ before the first repetition. Then this bucket requires

$$Z_n(H_r) = \sum_{l=1}^{y_n(H_r)} y_n(H_r) (y_n(H_r) - 1).$$

(71)

Then according to the theorem of the total probability we have

$$P(A_1) = \sum_{k=1}^{n} P(A_1 \mid T = k) P(T = k) = \sum_{k=1}^{n} \frac{k}{n} \frac{n!k}{(n-k)!n^{k+1}}.$$  

(72)

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 14 The expected running time of Matrix is

$$T_{\text{exp}}(n, \text{Matrix}) = \Theta(\sqrt{n}).$$

(73)

Proof. The ???

4 Summary

Table 4 summarises the basic properties of the running times of the investigated algorithms.

<table>
<thead>
<tr>
<th>Index and Algorithm</th>
<th>$T_{\text{best}}(n)$</th>
<th>$T_{\text{worst}}(n)$</th>
<th>$T_{\text{exp}}(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. BACKWARD</td>
<td>$\Theta(1)$</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(n)$</td>
</tr>
<tr>
<td>2. FORWARD</td>
<td>$\Theta(1)$</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(n)$</td>
</tr>
<tr>
<td>3. LINEAR</td>
<td>$\Theta(n)$</td>
<td>$\Theta(n)$</td>
<td>$n + \Theta(\sqrt{n})$</td>
</tr>
<tr>
<td>4. RANDOM</td>
<td>$\Theta(1)$</td>
<td>$\Theta(n^2\log n)$</td>
<td>$\Theta(n)$</td>
</tr>
<tr>
<td>5. TREE</td>
<td>$\Theta(1)$</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(\sqrt{n} \log n)$</td>
</tr>
<tr>
<td>6. GARBAGE</td>
<td>$\Theta(1)$</td>
<td>$\Theta(n)$</td>
<td>$\Theta(\sqrt{n})$</td>
</tr>
<tr>
<td>7. BUCKET</td>
<td>$\Theta(\sqrt{n})$</td>
<td>$\Theta(n\sqrt{n})$</td>
<td>$\Theta(\sqrt{n})$</td>
</tr>
<tr>
<td>8. MATRIX</td>
<td>$\Theta(1)$</td>
<td>$\Theta(n\sqrt{n})$</td>
<td>$\Theta(\sqrt{n})$</td>
</tr>
</tbody>
</table>

Table 4: 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 \( s \)) and \( s \left(= (s_1, s_2, \ldots, s_n)\right) \), a sequence of nonnegative integers with \( 1 \leq s_j \leq n \) for \( 1 \leq j \leq n \) 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 \( v = (v_1, v_2, \ldots, v_n) \), 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**\((n, s)\)
01 \( g \leftarrow \text{True} \)
02 for \( i \leftarrow 1 \) to \( n \)
03 do \( v[i] \leftarrow 0 \)
04 for \( i \leftarrow 1 \) to \( n \)
05 do if \( v[s[i]] > 0 \)
06 then \( g \leftarrow \text{False} \)
07 return \( g \)
08 else \( v[s[i]] \leftarrow v[s[i]] + 1 \)
09 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 \( s \) with the previous elements until the first collision or until the last pair of elements.

**BACKWARD**\((n, s)\)
01 \( g \leftarrow \text{True} \)
02 for \( i \leftarrow 2 \) to \( n \)
03 do for \( j \leftarrow i - 1 \) downto \( 1 \)
04 do if \( s[i] = s[j] \)
05 then \( g \leftarrow \text{False} \)
06 return \( g \)
07 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**\((n, s)\)
01 \( g \leftarrow \text{True} \)
02 \( \text{for } i \leftarrow 1 \text{ to } n-1 \)
03 \( \text{do for } j \leftarrow i+1 \text{ to } n \)
04 \( \text{do if } s[i] = s[j] \)
05 \( \text{then } g \leftarrow \text{False} \)
06 \( \text{return } g \)
07 \( \text{return } g \)

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 \( \text{Ran}(k) \) \[^1\] generating a random integer value distributed uniformly in the interval \([1, k]\).

\[
\text{RANDOM}(n, s)
\]
01 \( g \leftarrow \text{True} \)
02 \( \text{while } g = \text{True} \)
03 \( \text{do } i \leftarrow \text{Ran}(n^2) \)
04 \( j \leftarrow (\text{Ran}(n^2 - 1) + j) \pmod{n^2} \)
05 \( \text{if } s_i = s_j \)
06 \( \text{then } g \leftarrow \text{False} \)
07 \( \text{return } g \)

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).

\[
\text{TREE}(n, s)
\]
01 \( g \leftarrow \text{True} \)
02 \( \text{let } s[1] \text{ be the root of a tree} \)
03 \( \text{for } i \leftarrow 2 \text{ to } n \)
04 \( \text{if } [s[i] \text{ is in the tree} \)
05 \( \text{then } g \leftarrow \text{False} \)
06 \( \text{return } \)
07 \( \text{else } \text{insert } s[i] \text{ in the tree} \)
08 \( \text{return } g \)

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 $v = (v_1, v_2, \ldots, v_n)$. Interesting is that $v$ is used without initialisation, that is its initial values can be arbitrary integer numbers.

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

GARBAGE($n, s$)

01 $g \leftarrow \text{True}$
02 for $i \leftarrow 1$ to $n$
03 \hspace{1em} do if $v[s[i]] < i$ and $s[v[s[i]]] = s[i]$
04 \hspace{2em} then $g \leftarrow \text{False}$
05 \hspace{1em} else $v[s[i]] \leftarrow i$
06 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 $r$th 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 $c = (c_1, c_2, \ldots, c_m)$ are counters, where $c_j$ ($1 \leq j \leq m$) shows the number of elements of the $i$th row.

For the simplicity we suppose that $n$ is a square.

BUCKET($n, s$)

01 $g \leftarrow \text{True}$
02 $m \leftarrow \sqrt{n}$
03 for $j \leftarrow 1$ to $m$
04 \hspace{1em} do $c[j] \leftarrow 1$
05 for $i \leftarrow 1$ to $n$
06 \hspace{1em} do $r \leftarrow \lceil s[i]/m \rceil m$
07 \hspace{2em} for $j \leftarrow 1$ to $c[r] - 1$
08 \hspace{3em} do if $s[i] = Q[r, j]$
09 \hspace{4em} then $g \leftarrow \text{False}$
10 \hspace{2em} return $g$
11 \hspace{1em} else $Q[r, c[r]] \leftarrow s[i]$
12 \hspace{2em} $c[r] \leftarrow c[r] + 1$
13 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($n, \mathcal{M}$)

01 $g \leftarrow \text{True}$
02 BUCKET($n, r_1$)
03 if $g = \text{False}$
A. Iványi and I. Kátai

18

04 then return $g$
05 for $i \leftarrow 2$ to $n$
06 do $\text{Bucket}(n, r_i)$
07 if $g = \text{False}$
08 then return $g$
09 for $j \leftarrow 1$ to $n$
10 do $\text{Bucket}(n, c_j)$
11 if $g = \text{False}$
12 then return $g$
13 return True

Acknowledgements. The authors thank Tamás F. Móri, and Péter Burcsi, teachers of Eötvös Loránd University for the useful consultation, and Gábor Monostori, student of the same university for proposing the algorithm GARBAGE.

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

References

http://www.combinatorics.org/ ⇒[1]


http://www.bentham.org/open/tomatj/openaccess2.htm ⇒[1]


  [http://www.bolyai150.hu/eng/?page=page&name=Organizers](http://www.bolyai150.hu/eng/?page=page&name=Organizers) ⇒[1]


http://www.sciencedirect.com/science/journal/0012365X ⇒[I]

http://www.springerlink.com/content/1432-0541/? ⇒[I]

http://journals.cambridge.org/action/displayJournal?jid=JAZ ⇒[I]

http://www.sciencedirect.com/science/journal/0012365X ⇒[I]


http://ajc.maths.uq.edu.au/ ⇒[I]

http://www.math.cudenver.edu/~tvis/Research/cmj174-181.pdf ⇒[I]

MR2574850,
http://www.sciencedirect.com/science/journal/0012365X ⇒[I]

http://www.jstor.org/journals/00029890.html ⇒[I]


http://www.combinatorics.org/ ⇒[I]


http://www.combinatorics.org/  ⇒ I


http://www.sciencedirect.com/science/journal/0012365X  ⇒ I

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