Deciding football sequences

Dedicated to the memory of Antal Bege (1962–2012)

Antal IVÁNYI
Eötvös Loránd University,
Faculty of Informatics, Hungary
email: ivanyi.antal2@upcmail.hu

Jon E. SCHOENFIELD
Huntsville, Alabama, USA
email: jonscho@hiwaay.net

Abstract. An open problem posed by the first author [36, 53, 54, 59, 100] is the complexity to decide whether a sequence of nonnegative integer numbers can be the final score of a football tournament. In this paper we propose polynomial time approximate and exponential time exact algorithms which solve the problem.

1 Introduction

Let a, b and n be nonnegative integers ($b \geq a \geq 0$, $n \geq 1$), $\mathcal{T}(a, b, n)$ be the set of directed multigraphs $T = (V, E)$, where $|V| = n$, and each pair of different vertices u, $v \in V$ are connected with at least a and at most b arcs [56, 57]. $T \in \mathcal{T}(a, b, n)$ is called (a, b, n)-*tournament.* $(1, 1, n)$-tournaments are the usual tournaments, and $(0, 1, n)$-tournaments are also called *oriented graphs* or *simple directed graphs* [45, 93]. The set $\mathcal{T}$ is defined by

$$\mathcal{T} = \bigcup_{b \geq a \geq 0, \ n \geq 1} \mathcal{T}(a, b, n).$$

The definition of (undirected) (a, b, n)-*graphs* is similar. The $(0, 1, n)$-graphs are the usual *simple graphs.*

An (a, b, n)-tournament is called *complete,* if the set of permitted results is $\{0 : c, 1 : c - 1, \ldots, c : 0\}$ for all possible c ($a \leq c \leq b$). If some of these results are prohibited, then the tournament is called *incomplete* [55, 56, 57] .

For example football is an incomplete $(2, 3, n)$-tournament since the permitted results are $0 : 3$, $1 : 1$ and $3 : 0$, while $0 : 2$, $1 : 2$, $2 : 0$, and $2 : 1$ are prohibited.

According to this definition $\mathcal{T}$ is the set of the finite directed loopless multigraphs. We remark, that if $a' \leq a \leq b \leq b'$ then an (a, b, n)-tournament is also an (a', b', n)-tournament. The outdegree sequence of an (a, b, n)-tournament we call *the score sequence* of the tournament [45, 93, 95].

Let l, u, and m be integer numbers with $u \geq l$ and $m \geq 1$. The sequence $s = (s_1, \ldots, s_m)$ of integer numbers with $l \leq s_1 \leq \cdots \leq s_m \leq u$ is called (l, u, m)-*regular*. It is well-known that the number of (l, u, m)-regular sequences is

$$R(l, u, m) = \binom{u - l + m}{m}. \tag{1}$$

In this paper we consider only the graph theoretical aspects of the investigated problems, although they have many applications [1, 16, 17, 68, 76, 88, 108]. We analyze only sequential algorithms. The Reader can find parallel results e.g. in [2, 29, 92, 102, 104].

The structure of the paper is as follows. After this introduction in Section 2 we deal with the filtering of potential complete sequences, then in Section 3 describe incomplete sequences. Section 4 contains filtering and Section 5 reconstruction algorithms of potential football sequences. Finally in Section 6 we deal with the enumeration of football sequences.

2   Filtering of potential complete sequences

We are seeking football sequences. Taking into account that a score sequence of an incomplete (a, b, n)-tournament is at the same time a score sequence of the complete (a, b, n)-tournament, the properties of score sequences of complete tournaments allow some filtering among the regular sequences.

In 1953 Landau [75] proved the following popular theorem. About ten proofs are summarized by Reid [95]. Further proofs are in [3, 18, 19, 20, 22, 44, 46, 101, 106, 110].

Theorem 1 (Landau [75]) *A* $(0, n - 1, n)$-*regular sequence* $s = (s_1, \ldots, s_n)$ *is*

the outdegree sequence of some $(1, 1, n)$*-tournament if and only if*

$$\sum_{i=1}^{k} s_i \geq \frac{k(k-1)}{2}, \quad 1 \leq k \leq n, \tag{2}$$

with equality when $k = n$.

Proof. See [64, 75, 86]. □

Moon [85] proved the following generalization of Landau's theorem (we present it in reformulated form). Later Takahashi [107] reproved the theorem.

Theorem 2 (Moon [85]) *A* $(0, b(n-1), n)$*-regular sequence* $s = (s_1, \ldots, s_n)$ *is the score sequence of some* (b, b, n)*-tournament if and only if*

$$\sum_{i=1}^{k} s_i \geq \frac{bk(k-1)}{2}, \ 1 \leq k \leq n,$$

with equality when $k = n$.

Proof. See [85]. □

We define a *point-loss function* P_k $(k = 0, \ldots, n)$ by the following recursion: $P_0 = 0$ and if $1 \leq k \leq n$, then

$$P_k = \max \left(P_{k-1}, \frac{bk(k-1)}{2} - \sum_{i=1}^{k} s_i \right).$$

Now P_k gives a lower bound for the number of lost points in the matches among the teams $T_1, \ldots, T_k$ (not the exact value since the teams $T_1, \ldots, T_k$ could win points against $T_{k+1}, \ldots, T_n$).

Theorem 3 (Iványi [56]) *A* $(0, b(n-1), n)$*-regular sequence* $s = (s_1, \ldots, s_n)$ *is the score sequence of some complete* (a, b, n)*-tournament if and only if*

$$\frac{ak(k-1)}{2} \leq \sum_{i=1}^{k} s_i \leq \frac{bn(n-1)}{2} - P_k - (n-k)s_k \ (1 \leq k \leq n).$$

Proof. See [56]. □

3   Incomplete tournaments

We know only the following three results on the score sequences of incomplete tournaments.

Semicomplete digraphs (semicomplete tournaments) are defined as $(1, 2, n)$-digraphs in which if two vertices are connected with two arcs then these arcs have different directions.

Theorem 4 (Reid, Zhang [96]) *A* $(0, n-1, n)$-*regular sequence* $s = (s_1, \ldots, s_n)$ *is the score sequence of some semicomplete tournament if and only if*

$$\sum_{i=1}^{k} s_i \geq \frac{k(k-1)}{2} \quad and \quad s_k \leq n-1, \quad 1 \leq k \leq n. \tag{3}$$

Proof. See [96]. $\qquad\square$

Antal Bege asked in 1999 [7] how many wins are necessary in a football tournament of n teams to get a strictly monotone score sequence. If $n = 2$ then 1, if $n = 3$ then 1, and if $n = 4$ then 2 are sufficient and necessary. The following assertion gives the general answer.

Theorem 5 (Iványi [55]) *If* $N(n)$ *denotes the minimal number of necessary and sufficient wins for different scores in a football tournament of* n *teams then*

$$N(n) = \left(\frac{3}{2} - \sqrt{2}\right) n^2 + \Theta(n). \tag{4}$$

Recently Berger [10] published the following criterion for special incomplete $(0, 2, n)$-tournaments.

Theorem 6 (Berger [10]) *Sequence* $\sigma = \left(\binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}\right)$ *with* $a_1 \geq \cdots \geq a_n$ *is the score sequence of special incomplete* $(0, 2, n)$-*tournaments—in which* $0:0$, $0:1$, $1:0$, *and* $1:1$ *are the permitted results—if and only if*

$$\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} \min(b_i, k-1) + \sum_{i=k+1}^{n} \min(b_i, k) \tag{5}$$

for all $k = 1, \ldots, n$, *with equality for* n.

Proof. See [10]. $\qquad\square$

Earlier (weaker) results can be found in [23, 41, 42, 99].

4 Filtering of potential football sequences

There are many exact results deciding whether a given sequence is the degree/outdegree sequence of a given type of undirected (e.g. [24, 25, 30, 31, 32, 33, 34, 35, 43, 46, 47, 48, 50, 51, 61, 69, 74, 83, 84, 90, 116, 118]) or directed (e.g. [12, 13, 75, 85, 56, 57, 94, 115]) graphs. Several authors studied the case when the indegree and outdegree sequences are together prescribed [9, 10, 14, 33, 48, 91].

The score sequences of the football tournaments we call *football sequences*. A $(0, 3n - 3, n)$-regular sequence $s = (s_1, \ldots, s_n)$ is called *good* if there exists a football tournament whose score sequence is s, and s is called *bad* otherwise. We denote the football sequences by $f = (f_1, \ldots, f_n)$.

In this section we present approximate algorithms which filter only some part of the bad sequences. Since these filtering algorithms have short running time they help to reduce the expected running time of the exact algorithms.

The filtering algorithms are classified according to their worst running time as constant, linear, and other polynomial type ones.

4.1 Constant time filtering algorithms

The expected running time can be substantially decreased if we can filter some part of the investigated sequences in constant time.

Let $n \geq 2$ and $f = (f_1, \ldots, f_n)$ a football sequence.

Lemma 7 (C1 test) $f_n \neq 3n - 4$.

Proof. If a team wins all matches then its score is $3n - 3$. If not, then it loses at least two points making a draw, so its score is at most $3n - 5$. ☐

Lemma 8 (C2 test) *If* $f_n = 3n - 3$ *then* $f_{n-1} \leq 3n - 6$.

Proof. f_n can be $3n - 3$ only so, that T_n wins all matches. Then the score T_{n-1} is at most $3n - 6$. ☐

Lemma 9 (C3 test) *If* $f_1 = 0$ *then* $f_2 \geq 3$.

Proof. If $f_1 = 0$ then T_1 lost all matches therefore T_2 has at least one win and so f_2 is at least 3. ☐

Lemma 10 (C4 test) *If* $f_1 = f_2 = 1$ *then* $f_3 \geq 6$.

Proof. If $f_1 = f_2 = 1$ then the match of T_1 and T_2 ended with a draw implying that T_3 has at least two wins and so at least six points. $\qquad\square$

Lemma 11 (C5 test) *If $f_n = f_{n-1} = 3n - 5$, then $f_{n-2} \leq 3n - 9$.*

Proof. If the joint score of T_n and T_{n-1} is $3n - 5$ then the result of their match has to be a draw. In this case T_{n-2} lost at least two matches and so $f_{n-2} \leq 3n - 9$. $\qquad\square$

Lemma 12 (C6 test) *If $f_n = 3n - 3$ and $f_{n-1} = 3n - 6$, then $f_{n-2} \leq 3n - 9$.*

Proof. If $f_n = 3n - 3$, then T_n won all matches. In this case the score of T_{n-1} can be $3n - 6$ only then if T_{n-1} loses against T_n but wins all remaining matches. Then T_{n-2} lost at least two matches and so $f_{n-2} \leq 3n - 9$. $\qquad\square$

Lemma 13 (C7 test) *If $f_1 = 0$ and $f_2 = 3$ then $f_3 \geq 6$.*

Proof. See the proof of Lemma 12. $\qquad\square$

Lemma 14 (C8 test) *If $f_1 = 1$ and $f_2 = 2$ then $f_3 \geq 4$.*

Proof. Since T_1 and T_2 gathered points only with draws their match ended with a draw. Therefore T_3 won against T_1 and either won against T_2 or they made a draw, so T_3 has at least 4 points. $\qquad\square$

Lemma 15 (C9 test) *If $f_n = 3n - 5$ and $f_{n-1} = 3n - 7$ then $f_{n-2} \leq 3n - 8$.*

Proof. If $f_n = 3n - 5$ then T_n has a draw and $n - 2$ wins. If $f_{n-1} = 3n - 7$ then T_{n-1} has two draws and $n - 3$ wins, and the match between T_n and T_{n-1} ended with a draw. In this case T_{n-2} has at least a loss and a draw implying $f_{n-2} \leq 3n - 8$. $\qquad\square$

The following program CONSTANT realizes the tests of the previous 9 lemmas. This and later programs are written using the pseudocode conventions described in [27]. In this and in the further pseudocodes input variables are n: the length of the investigated sequence ($n \geq 3$); $s = (s_1, \ldots, s_n)$: a $(0, 3n - 3, n)$-regular sequence; output variable is L: $L = 0$ means that the investigated input is bad, $L = 1$ means that it is good while $L = 2$ shows that the given algorithm could not decide.

CONSTANT(n, s)

```
01 L = 0                                        // line 01: initialization of L
02 if sₙ == 3n − 4                              // line 02–03: C1
03     return L
04 if sₙ == 3n − 3 and sₙ₋₁ ≥ 3n − 5           // line 04–05: C2
05     return L
06 if s₁ == 0 and s₂ ≤ 2                        // line 06–07: C3
07     return L
08 if s₁ == 1 and s₂ == 1 and s₃ ≤ 5           // line 08–09: C4
09     return L
10 if sₙ == 3n − 5 and sₙ₋₁ = 3n − 5 and sₙ₋₂ ≥ 3n − 8  // line 10–11: C5
11     return L
12 if sₙ ==3n − 3 and sₙ₋₂ ==3n − 6 and sₙ₋₃ ≥ 3n − 8  // line 12–13: C6
13     return L
14 if s₁ == 0 and s₂ == 3 and s₃ ≤ 5           // line 14–15: C7
15     return L
16 if s₁ == 1 and s₂ == 2 and s₃ ≤ 3           // line 16–17: C8
17     return L
18 if sₙ == 3n − 5 and sₙ₋₁ ==3n − 7 and sₙ₋₂ ≥ 3n − 8  // line 18–19: C9
19     return L
20 L = 2                                        // line 20–21: these tests can not decide
21 return 2
```

Tables 1, 2, and 3 show the filtering results of CONSTANT. The numbers in the tables show how many sequences are accepted from the sequences accepted by the previous filtering algorithm. The exact results in these tables are printed with bold font (such emphasizing will be used in the later tables too).

The programs are written in C by Loránd Lucz and run on an Inter Core i7 processor (3.4 GHz) with optimization level O3. The running times are given in seconds.

Table 2 shows the filtering results of C4, C5, C6 and C7.

Table 3 shows the filtering results of algorithms C8 and C9, further the number of football sequences (F) and the running time of LINEAR for $n = 1, \ldots, 15$ teams. Column R in Table 1 and column t in Table 3 show that the running time is approximately proportional with the number of the regular sequences.

For example if $n = 2$ then C1, C2 and C3 filter 80 % of the regular and 100 % of the bad sequences. If $n = 3$ then they filter 54 from the 84 regular sequences while C1, $\ldots$, C9 filter 70 sequences which represent 90.90 % of the

n	R	C1	C2	C3
1	1	**1**	**1**	**1**
2	10	7	4	**2**
3	84	63	45	30
4	715	550	414	311
5	6 188	4 823	3 718	2 911
6	54 264	42 636	33 320	26 650
7	480 700	379 753	299 421	242 624
8	4 292 145	3 404 115	2 700 775	2 207 800
9	38 567 375	30 678 375	24 452 220	20 116 030
10	348 330 136	277 722 676	222 146 496	183 629 160
11	3 159 461 960	2 523 716 572	2 024 386 180	1 679 655 640
12	28 760 021 745	23 008 017 396	18 498 140 232	15 394 304 500
13	262 596 783 764	210 345 382 913	169 436 070 190	141 355 053 635
14	2 403 979 904 200	1 927 719 734 500	1 555 302 958 664	1 300 210 775 786
15	22 057 981 462 440	17 704 432 489 590	14 303 680 429 990	11 978 596 958 384

Table 1: Number of $(0, 3n - 3, n)$-regular sequences (R) accepted by C1, C2, and C3 for $n = 1, \ldots, 15$ teams.

n	C4	C5	C6	C7
1	**1**	**1**	**1**	**1**
2	**2**	**2**	**2**	**2**
3	26	22	19	17
4	281	255	237	222
5	2 691	2 501	2 374	2 271
6	24 000	23 373	22 302	21 596
7	227 770	215 227	207 042	200 609
8	2 700 775	2 207 800	2 097 803	1 972 783
9	19 155 258	18 065 694	17 460 916	16 989 609
10	175 138 885	165 526 269	160 206 767	156 070 967
11	1 591 808 376	1 518 385 621	1 471 133 714	1 434 460 309
12	14 605 778 836	13 947 629 921	13 524 714 862	13 196 925 716
13	134 230 657 710	128 305 394 396	124 497 616 840	121 549 435 860
14	1 235 669 598 354	1 181 962 750 733	1 147 511 569 252	1 208 609 923 538
15	11 391 620 617 874	10 903 053 416 141	10 590 098 238 918	10 348 178 700 655

Table 2: Number of $(0, 3n - 3, n)$-regular sequences accepted by C4, C5, C6, and C7 for $n = 1, \ldots, 15$ teams.

bad sequences. If $n = 15$ then the nine constant time algorithms filter 54.73 % of the bad sequences. This is surprisingly high efficiency but smaller than the sum of the individual asymptotic efficiency of the 9 algorithms. The reason is simple: e. g. the sequence $s = (0, 0, 5)$ would be filtered by C1 and C3 too.

n	C8	C9	F	t
2	**2**	**2**	**2**	0.000
3	15	14	**7**	0.000
4	209	203	**40**	0.000
5	2 175	2 133	**355**	0.000
6	20 039	20 510	**3 678**	0.000
7	194 333	191 707	**37 263**	0.016
8	1 795 074	1 772 842	**361 058**	0.062
9	16 524 335	16 332 091	**3 403 613**	0.499
10	154 361 149	150 288 309	**31 653 777**	4.602
11	1 398 051 547	1 383 099 467	**292 547 199**	41.771
12	12 870 899 770	12 737 278 674	**2 696 619 716**	380.984
13	118 612 802 828	117 411 184 292		3 489.299
14	1 094 282 911 155	1 083 421 567 482		34 079.254
15	10 106 678 997 431	10 008 094 941 133		316 965.954

Table 3: Number of $(0, 3n - 3, n)$-regular sequences accepted by C8 and C9, the number of football sequences (F), and the running time (t) of C9 for $n = 1, \dots, 15$ teams.

4.2   Efficiency of the constant time testing algorithms

Using (1) we give the efficiency of the nine constant time filtering algorithms.

Lemma 16 (efficiency of C1) *The ratio of sequences with $s_n = 3n - 4$ among $(0, 3n - 3, n)$-regular sequences is*

$$\frac{\binom{4n-5}{n-1}}{\binom{4n-3}{n}} = \frac{n(3n-3)}{(4n-4)(4n-3)} = \frac{3}{16} + \frac{9}{16(4n-3)} = \frac{3}{16} + o(1). \qquad (6)$$

Proof. The sequences satisfying the given condition are such $(0, 3n - 3, n)$-regular ones, whose lower bound is $l = 0$, upper bound is $u = 3n - 4$, and contain $m = n - 1$ elements. So according to (1) the required ratio is

$$\frac{R(0, 3n - 4, n - 1)}{R(0, 3n - 3, n)} = \frac{n(3n-3)}{(4n-4)(4n-3)} = \frac{3}{16} + o(1). \qquad (7)$$

$\square$

Lemma 17 (efficiency of C2) *The ratio of the sequences satisfying the conditions $s_n = 3n - 3$ and $s_{n-1} \geq 3n - 5$ among the $(0, 3n - 3, n)$-regular sequences is*

$$\frac{37}{256} + o(1). \qquad (8)$$

Proof. Since $R(0, 3n - 3, n - 2)$ sequences satisfy the conditions $s_n = 3n - 3$ and $s_{n-1} = 3n - 3$, the corresponding ratio is

$$\frac{R(0, 3n - 3, n - 2)}{R(0, 3n - 3, n)} = \frac{n(n - 1)}{(4n - 4)(4n - 3)} = \frac{1}{16} + o(1). \tag{9}$$

$R(0, 3n - 4, n - 2)$ sequences satisfy $s_n = 3n - 3$ and $s_{n-1} = 3n - 4$, so the corresponding ratio is

$$\frac{R(0, 3n - 4, n - 2)}{R(0, 3n - 3, n)} = \frac{n(n - 1)(3n - 3)}{(4n - 3)(4n - 4)(4n - 5)} = \frac{3}{64} + o(1). \tag{10}$$

$R(0, 3n - 5, n - 2)$ sequences have the properties $s_n = 3n - 3$ and $s_{n-1} = 3n - 5$, so the corresponding ratio is

$$\frac{R(0, 3n - 5, n - 2)}{R(0, 3n - 3, n)} = \frac{n(n - 1)(3n - 3)(3n - 4)}{(4n - 3)(4n - 4)(4n - 5)(4n - 6)} = \frac{9}{256} + o(1). \tag{11}$$

Summing up the right sides (9), (10), and (11) we get the value (8). □

Lemma 18 (efficiency of C3) *The ratio of the sequences satisfying the conditions $s_1 = 0$ and $s_2 \leq 2$ among the $(0, 3n - 3, n)$-regular sequences is*

$$\frac{37}{256} + o(1). \tag{12}$$

Proof. Similar to the proof of Lemma 8. □

Lemma 19 (efficiency of C4) *The ratio of the sequences satisfying the conditions $s_1 = 1$ and $s_2 = 1$ and $s_3 \leq 5$ among the $(0, 3n - 3, n)$-regular sequences is*

$$\frac{2343}{4^8} + o(1). \tag{13}$$

Proof. Since $R(1, 3n - 3, n - 3)$ sequences satisfy the conditions $s_1 = s_2 = s_3 = 1$ the corresponding ratio is

$$\frac{R(1, 3n - 3, n - 3)}{R(0, 3n - 3, n)} = \frac{n(n - 1)(n - 2)(3n - 3)}{(4n - 3)(4n - 4)(4n - 5)(4n - 6)} = \frac{3}{4^4} + o(1). \tag{14}$$

The sequences with $s_1 = s_2 = 1$ and $s_3 = 2$, $s_1 = s_2 = 1$ and $s_3 = 3$, $s_1 = s_2 = 1$ and $s_3 = 4$, and $s_1 = s_2 = 1$ and $s_3 = 5$ have the asymptotic ratio $3/4^5$, $3/4^6$, $3/4^7$, and $3/4^8$ resp.

The sum of the received five ratios is

$$\frac{3}{4^4} + \frac{3^2}{4^5} + \frac{3^3}{4^6} + \frac{3^4}{4^7} + \frac{3^5}{4^7} = \frac{2343}{4^8}, \tag{15}$$

implying (13). □

Lemma 20 (efficiency of C5) *The ratio of the sequences satisfying the conditions of $s_n = s_{n-1} = 3n-5$ and $s_{n-3} \geq 3n-8$ among the $(0, 3n-3, n)$-regular sequences is*

$$\frac{1575}{4^8} + o(1). \tag{16}$$

Proof. We have to sum the contributions of $R(0, 3n-5, n-2)$, $R(0, 3n-6, n-2)$, $R(0, 3n - 7, n - 2)$, and $R(0, 3n - 8, n - 2)$ sequences:

$$\frac{3^2}{4^5} + \frac{3^3}{4^6} + \frac{3^5}{4^7} + \frac{3^6}{4^8} = \frac{1575}{4^8}, \tag{17}$$

implying (16). □

Lemma 21 (efficiency of C6) *The ratio of the sequences satisfying the conditions of $s_n = 3n-3$, $s_{n-1} = 3n-6$, and $s_{n-2} \geq 3n-8$ among the $(0, 3n-3, n)$-regular sequences is*

$$\frac{999}{4^8} + o(1). \tag{18}$$

Proof. In this case we sum the contributions of $R(0, 3n - 6, n - 3)$, $R(0, 3n - 7, n - 1)$, and $R(0, 3n - 8, n - 1)$ sequences:

$$\frac{3^3}{4^6} + \frac{3^4}{4^7} + \frac{3^5}{4^8} = \frac{999}{4^8}, \tag{19}$$

implying (18). □

Lemma 22 (efficiency of C7) *The ratio of the sequences satisfying the conditions of $s_1 = 0$, $s_2 = 3$, and $s_3 \leq 5$ among the $(0, 3n - 3, n)$-regular sequences is*

$$\frac{999}{4^8} + o(1). \tag{20}$$

Proof. Similar to the proof of Lemma 21. □

Lemma 23 (efficiency of C8) *The ratio of the sequences satisfying the conditions of $s_1 = 1$, $s_2 = 2$, and $s_3 \leq 3$ among the $(0, 3n - 3, n)$-regular sequences is*

$$\frac{63}{4^6} + o(1). \tag{21}$$

Proof. We sum the contributions of $R(2, 3n-3, n-3)$ and $R(3, 3n-3, n-3)$ sequences:

$$\frac{3^2}{4^5} + \frac{3^3}{4^6} = \frac{63}{4^6}, \tag{22}$$

implying (21). □

Lemma 24 (efficiency of C9) *The ratio of the sequences satisfying the conditions of $s_n = 3n-5$, $s_{n-1} = 3n-7$, and $s_{n-2} \geq 3n-7$ among the $(0, 3n-3, n)$-regular sequences is*

$$\frac{3^4}{4^7} + o(1). \tag{23}$$

Proof. Similar to the proof of Lemma 23. □

The cumulated asymptotic efficiency of the constant time algorithms is

$$\frac{3}{16} + \frac{2 \cdot 37}{4^4} + \frac{2343}{4^8} + \frac{1575}{4^8} + \frac{2 \cdot 999}{4^8} + \frac{63}{4^6} + \frac{3^4}{4^7} = \frac{38480}{4^8}. \tag{24}$$

The cumulated efficiency of the nine constant time algorithms is about 58.72 %. According to Table 1 the practical joint efficiency of C1, C2 and C3 is 64.28 % for $n = 3$ and 45.91 % for $n = 14$. According to Table 3 the total practical efficiency of the nine constant time algorithms is 91.67 % for $n = 3$ and 54.93 % for $n = 14$.

The practical cumulated efficiency is smaller than the theoretical one, since some part of the sequences is filtered by several algorithms: e.g. the sequence $s = (0, 0, 5)$ is filtered by C1 and C3 too.

We remark that the algorithms of CONSTANT are sorted on the base of their nonincreasing asymptotic efficiency. We get the same order of the practical efficiency of these algorithms shown on the small values of n.

4.3 Filtering algorithms with linear running time

We investigate the following filtering algorithms whose worst running time is linear: COMPLETE = L1, POINT-LOSSES = L2, REDUCTION0 = L3, REDUCTION1 = L4, DRAW-UNIQUE = L5, BALANCED = L6, DRAW-UNIFORM = L7, DRAW-SORTED-UNIQUE = L8.

4.3.1 Linear filtering algorithm L1 = Complete

The first linear time filtering algorithm $L1 = $ Complete is based on the following special case of Lemma 3 in [56].

Corollary 25 $((2,3,n)$-complete test, [56]) *If* $n \geq 1$ *and* $(f_1, \ldots, f_n)$ *is a football sequence then*

$$2\binom{k}{2} \leq \sum_{i=1}^{k} f_i \leq 3\binom{n}{2} - (n-k)f_k \quad (k = 1, \ldots, n). \tag{25}$$

Basic parameters of Complete are the usual ones, further S: the current sum of the first i elements of s.

```
COMPLETE(n, s)
01 S = 0                                      // line 01: initialization of S
02 for i = 1 to n                             // line 02–06: test
03      S = S + s_i
04      if (S < 2(i/2)) ∨ (S > 3(n/2) − (n − i)s_i) = TRUE
05          L = 0
06          return L
07 L = 2                                       // line 07–08: s is undecided
08 return L
```

4.3.2 Linear filtering algorithm L2 = Point-Losses

The second linear time filtering algorithm $L2 = $ Point-Losses is based on the following assertion which is an extension of Lemma 3 in [56]. The basic idea is, that the small sums of the prefixes of s and the mod 3 remainders of the elements of s signalize lost points.

Lemma 26 *If* $(f_1, \ldots, f_n)$ *is a football sequence then*

$$2\binom{k}{2} \leq \sum_{i=1}^{k} f_i \leq 3\binom{n}{2} - (n-k)f_k - P_k \quad (k = 1, \ldots, n), \tag{26}$$

where $P_0 = 0$ *and*

$$P_k = \max\left(P_{k-1}, 3\binom{k}{2} - \sum_{i=1}^{k} f_i, \left\lceil \frac{\sum_{i=1}^{k}(f_i - 3\lfloor f_i/3\rfloor)}{2} \right\rceil\right). \tag{27}$$

Proof. The sum of the k smallest scores is at least $2\binom{k}{2}$ and at most $3\binom{n}{2}$ minus the following point-losses:

1. the sum of the remaining scores, which is at least $(n-k)f_k$;

2. the point-losses due to draws documented by the mod 3 remainders;

3. the point-losses documented by differences $3\binom{k}{2} - \sum_{i=1}^{k} f_i$;

$\square$

Basic parameters of POINT-LOSSES are the usual ones, further S: the current sum of the first i elements of s, and P: the current value of the point-losses.

POINT-LOSSES(n, s)

```
01 S = P = L = 0                          // line 01: initialization of S, P, and L
02 for k = 1 to n                         // line 02–06: filtering
03      S = S + s_k
04      P = max (P_{k-1}, 3(k choose 2) − S, ⌈(∑_{i=1}^{k}(s_i−3⌊s_i/3⌋))/2⌉)
05      if S > 3(n choose 2) − (n − k)s_k − P
06          return L
07 L = 2
08 return L                               // line 08: s is undecided
```

$$
04 \quad P = \max\left(P_{k-1}, 3\binom{k}{2} - S, \left\lceil \frac{\sum_{i=1}^{k}(s_i - 3\lfloor s_i/3\rfloor)}{2} \right\rceil\right)
$$

$$
05 \quad \textbf{if } S > 3\binom{n}{2} - (n-k)s_k - P
$$

4.3.3 Linear filtering algorithm L3 = REDUCTION0

The third linear test is based on the observation that if the sum of the k smallest scores is minimal then all matches among the first k teams ended by a draw and if the sum of the k largest scores is maximal then the corresponding scores are multiples of 3 and further if $k < n$ then $f_{n-k} \le 3(n-k-1)$.

Lemma 27 *If $n \ge 2$, $1 \le k \le n$, and $f = (f_1, \ldots, f_n)$ is a football sequence then*

1) if the sum of the first k scores is $k(k-1)$ then $f_1 = \cdots = f_k = k-1$ and if further $k < n$ then $f_{k+1} \ge 3k$;

2) if the sum of the last k scores is $3(n-k)k + 3\binom{k}{2}$ then $f_{n-k+1}, \ldots, f_n$ are multiples of 3 and if further $k < n$ then $f_{n-k} \le 3(n-k-1)$.

Proof. If $f_1 + \cdots + f_k = k(k-1)$ then all matches among $T_1, \ldots, T_k$ ended with a draw and these teams lost all matches against the remaining teams implying assertions 1).

If $f_{k+1} + \cdots + f_n = 3(n-k)k + 3\binom{k}{2}$ then $T_{k+1}, \ldots, T_n$ won all matches against the remaining teams and have no draws implying assertion 2. $\qquad\square$

Parameters of REDUCTION0 are the usual ones, further S: the current sum of the i smallest scores; Q: the current sum of the i largest scores; B is a logical variable characterizing the remainders mod 3 of the i largest scores.

REDUCTION0(n, s)

```
01 L = B = S = Q = 0                    // line 01: initialization of L, B, S, and Q
02 for i = 1 to n − 1                   // line 02–12: test of the small scores
03      S = S + s_i
04      if S == i(i − 1)
05          if s_1 < i − 1 ∨ s_i > i − 1
06              return L
07          if s_{i+1} < 3i
08              return L
09 S = S + s_n
10 if S == n(n − 1)
11     if s_1 < n − 1
12         return L
13 for i = n downto 2                   // line 13–25: test of the large scores
14      Q = Q + s_i
15      if s_{i−1} > 3(n − i − 1)
16          return L
17      if s_i − 3⌊s_i/3⌋ > 0
18          B = 1
19      if B == 1
20          return L
21 Q = Q + s_1
22 if s_1 − ⌊s_1/3⌋ > 0
23          B = 1
24          if B == 1
25              return L
26 L = 2                                 // line 26–27: s is undecided
27 return L
```

Even this simple filtering algorithm finds a football sequence: if the condition of line 11 does not hold then the sum of all scores is minimal therefore all matches ended with draw. For the sake of the simplicity of the program we left this sequence undecided.

4.3.4 Linear filtering algorithm L4 = REDUCTION1

The fourth linear test is based on the observations that if the sum of the i smallest scores is $i(i-1)+1$ then either zero or one match among the first i teams ended with a win and if the sum of the i largest scores has near the maximal $3i(n-i)+3i(i-1)/2$ value then among the i maximal scores $i-2$ are multiples of 3 and 2 give 1 as remainder mod 3.

Lemma 28 *If $n \geq 3$, $f = (f_1, \ldots, f_n)$ is a football sequence, $1 \leq k \leq n$ then*
  1) if

$$\sum_{i=1}^{k} f_i = k(k-1) + 1 \tag{28}$$

then
  a) either $f_1 = \cdots = f_{k-1} = k-1$, $f_k = k$, and if $k+1 \leq n$, and $f_{k+1} \geq 3k-2$;
  b) or $f_1 = k-2$, $f_2, \ldots, f_{k-1} = k-1$, $f_k = k+1$, and $f_{k+1} \geq 3k$;
  2) if

$$\sum_{i=1}^{k} f_{n-i+1} = 3k(n-k) + 3\binom{k}{2} - 1 \tag{29}$$

then
  a) $\sum_{i=1,\ f_i-3\lfloor f_i/3 \rfloor=0}^{k} 1 = k-2$;
  b) $\sum_{i=1,\ f_i-3\lfloor f_i/3 \rfloor=1}^{k} 1 = 2$;
  c) $\sum_{i=1,\ f_i-3\lfloor f_i/3 \rfloor=2}^{k} 1 = 0$;
  d) if $n-k > 0$ then $f_{n-k} \leq 3(n-k-1)$.

Proof. 1) If $f_1 + \cdots + f_k = k(k-1) + 1$ then either all matches among T_1, $\ldots$, T_k ended with a draw and these teams lost all but one matches against the remaining teams and T_k made a draw with one of the teams T_k implying assertions a) or T_k won against T_1, the remaining matches among T_1, $\ldots$, T_k ended with a draw and the teams T_{k+1}, $\ldots$, T_n has no draw and won all matches against the first $n-k$ teams implying assertions b).

2) In case 2) of the lemma the teams T_{k+1}, $\ldots$, T_n won all matches against the first $n-k$ teams, and made exactly one draw. $\qquad\square$

Parameters of REDUCTION1 are the usual ones, further S: the current sum of the first i scores; Q: the current sum of the last i scores; L_1 and L_2: logical variables; B is the number of scores giving remainder 1 mod 3; C is the number of scores giving remainder 0 mod 3.

REDUCTION1(n, s)

01 L = B = C = S = Q = 0   // line 01: initialization of L, B, C, S, and Q

```
02 for i = 1 to n − 1                          // line 02–12: test of the small scores
03     S = S + s_i
04     if S == i(i − 1) + 1
05         L_1 = (s_1 == i − 1) ∧ (s_{i−1} == i − 1) ∧ (s_i == i) ∧ (s_{i+1} ≥ 3i − 2)
06         L_2 = (s_1 == i − 2) ∧ (s_2 == i − 1) ∧ (s_{i−1} == i − 1)
                  ∧ (s_i == i + 1) ∧ (s_{i+1} ≥ 3i)
07         if (L_1 == FALSE) ∧ (L_2 == FALSE) == TRUE
08             return L                          // line 07–08: s is not good
09 S = S + s_n
10 if S == n(n − 1) + 1
11     if (s_1 < n − 2) ∧ (s_2 == n − 1) ∧ (s_{n−1} == n − 1) ∧ (s_n == n + 1)
              == FALSE
12         return L
13 for i = n downto 2                           // line 13–35: test of the large scores
14     Q = Q + s_i
15     if s_i − 3⌊s_i/3⌋ == 2
16         return L
17     if s_i − 3⌊s_i/3⌋ == 1
18         B = B + 1
19     if s_i − 3⌊s_i/3⌋ > 0
20         C = C + 1
21     if Q == 3(n − i)i + 3i(i − 1)/2 − 1
22         if s_{n−i} > 3(n − i − 1)
23             return L
24         if (B == 2) ∧ (C == i − 2) == FALSE
25             return L
26 Q = Q + s_1
27 if s_i − 3⌊s_i/3⌋ == 2
28     return L
29 if s_i − 3⌊s_i/3⌋ == 1
30     B = B + 1
31 if s_i − 3⌊s_i/3⌋ > 0
32     C = C + 1
33 if Q == 3n(n − 1)/2 − 1
34     if (B == 2) ∧ (C == i − 2) == FALSE
35         return L
36 L = 2                                        // line 36–37: s is undecided
37 return L
```

4.3.5 Linear filtering algorithm L5 $=$ Draw-Unique

A *draw sequence* $d(s) = (d_1, \ldots, d_n)$ belonging to a $(0, 3(n-1), n)$-regular sequence s accepted by L4 is defined as a sequence of nonnegative integers having the following properties for $i = 1, \ldots, n$:

1. $0 \leq d_i \leq 2$;
2. $d_i = s_i \mod 3$;
3. $d_i \leq \min(s_i, n-1)$;
4. $d_i + 3(n-1-d_i) \geq s_i$,

further

$$\sum_{i=1}^{n} d_i = 2 \left(3 \binom{n}{2} - \sum_{i=1}^{n} s_i \right). \tag{30}$$

A draw sequence $d = (d_1, \ldots, d_n)$ is called $(0, 1, n)$-*graphic* (or simply graphic or good), if there exists a $(0, 1, n)$-graph whose degree sequence is d.

The fifth linear filtering algorithm is based on the following assertion.

Lemma 29 *If a $(2, 3, n)$-regular sequence s has only a unique draw sequence $d(s)$ which is not graphical then s is not football sequence.*

Proof. Since the football sequences have at least one graphical draw sequence, the regular sequences without graphical draw sequence are not football sequences. $\square$

Basic parameters of Draw-Unique are the usual ones, further S: ithe current sum of the elements of s; R: the number of obligatory draws; Dn: the number of the draws in the investigated potential tournament; $d = (d_1, \ldots, d_n)$: d_i is the number of draws allocated to T_i; $r = (r_1, \ldots, r_n)$: r_i is the remainder of $s_i \mod 3$; y: is the current number of allocated draws; x: is the current maximal number of draw packets acceptable by T_i.

Draw-Unique(n, s)

```
01 S = R = L = x = y = 0        // line 01: initialization of S, R, L, x and y
02 for i = 1 to n               // line 02–03: computation of S
03      S = S + s_i
04 Dn = 3(n choose 2) − S       // line 04: computation of Dn
05 for i = 1 to n               // line 05–17: allocation of draws
06      r_i = s_i − 3⌊s_i/3⌋
```

07      $R = R + r_i$

08      $x = \min\left(\frac{s_i - r_i}{3}, \lfloor\frac{n-1-r_i}{3}\rfloor, \lfloor\frac{3(n-1)-2r_i-s_i}{6}\rfloor\right)$

09      $d_i = r_i + 3x$

10      $y = y + d_i$

11 **if** $R > 2Dn$

12   **return** L, d

13 **if** $y < 2Dn$

14   **return** L, d

15 **if** $y \geq 2Dn$

16    $L = 2$

17   **return** L, d

18 sort d in decreasing order by Counting-Sort resulting d'

19 HHL(d')

20 **return** L, d   // line 20: s is undecided

Procedure HHL (Havel-Hakimi-Linear) is described in [60]. We remark that the original Havel-Hakimi algorithm requires in worst case $\Theta(n^2)$ time. Recently Király [70] published a version which uses the data structure proposed by van Emde Boas [71, 114] and requires $O(n \log\log n)$ time. Our algorithm is linear and works also for some multigraphs.

A natural requirement is $d_i \leq n - 1$ but $d_i > n - 1$ can occur only in the cases $s = (0, 2)$ and $s = (1, 2)$ which are filtered by the constant time algorithms.

We get a stronger filtering algorithm Draw-Sorted-Unique using the definition of the uniqueness of the sorted draw sequence. For example in the case of the sequence $s = (3, 3, 3, 5)$ we have three possibilities to allocate two draw packets but only the teams having 3 points can accept a packet therefore we get in each case the bad draw sequence $(3, 3)$.

We remark that the problem of unicity of graphs determined in a unique way by their degree sequences was studied for some graph classes (see e.g. the papers of Tetali [109], Tyskevich [113], and Barrus [6]).

4.3.6   Linear filtering algorithm L6 = Balanced-Lin

The sixth linear filtering algorithm L6 = Balanced-Lin is based on the observation that if the draw sequence is unique, then the victory sequence $w = (w_1, \ldots, w_n)$ and the loss sequence $l = (l_1, \ldots, l_n)$ are also unique. The following assertion gives a necessary condition for the reconstructability of the sequence pair (w, l).

Lemma 30 (Lucz [77]) *If $n \geq 2$, $w = (w_1, \ldots, w_n)$ is the win sequence and $l = (l_1, \ldots, l_n)$ is the loss sequence of a football sequence $f = (f_1, \ldots, f_n)$ with $\sum_{i=1}^{n} f_i > n(n-1)$ then let*

$$w_i = \max_{1 \leq j \leq n} w_j \quad and \quad l_j = \max_{1 \leq i \leq n} l_i. \tag{31}$$

In this case

$$w_i \leq \sum_{j=1}^{i-1} \left\lceil \frac{l_j}{n-1} \right\rceil + \sum_{j=i+1}^{n} \left\lceil \frac{l_j}{n-1} \right\rceil \tag{32}$$

and

$$l_j \leq \sum_{i=1}^{j-1} \left\lceil \frac{w_i}{n-1} \right\rceil + \sum_{i=j+1}^{n} \left\lceil \frac{w_i}{n-1} \right\rceil \tag{33}$$

for $n = 1, \ldots, n$.

Proof. The wins (losses) of the team T_i (T_j) having the maximal number of wins (losses) can be paired with losses (wins) only if there are at least w_i (l_j) teams having at least one loss (win). $\qquad \square$

4.3.7 Linear filtering algorithm L7 = Sport-Uniform

The seventh linear filtering algorithm L7 = Sport-Uniform is connected with a popular concept called in the world of sport *sport matrix*. It is an $n \times 5$ sized matrix containing the basic data of the teams of a tournament. We use the following formal definition of *sport matrix* for n teams.

Definition 31 *Let $n \geq 1$ and $s = (s_1, \ldots, s_n)$ be a $(0, 3(n-1), n)$-regular sequence. Then the sport matrices $S(s)$ corresponding to s are defined by the following properties:*

1. *the size of the matrix is $n \times 5$, its elements are nonnegative integers;*

2. *$w_i + d_i + l_i = n - 1$ for $i = 1, \ldots, n$;*

3. *$3w_i + d_i = s_i$ for $i = 1, \ldots, n$;*

4. *$\sum_{i=1}^{n} w_i = \sum_{i=1}^{n} l_i = \sum_{i=1}^{n} s_i - n(n-1)$;*

5. *$\sum_{i=1}^{n} d_i = 2\left(3\binom{n}{2} - \sum_{i=1}^{n} s_i\right)$.*

We remark that the ith row of the sport matrices contains data of T_i for $i = 1, \ldots, n$: index i, number of wins w_i, number of draws d_i, number of losses l_i and number of points s_i (w_i, d_i and l_i are estimated values). These formal requirements are only *necessary* for S to contain the basic characteristics of some football tournament.

A sequence $s = (s_1, \ldots, s_n)$ is called *sport sequence* if there exists at least one sport matrix corresponding to s.

Another useful concept is the *obligatory sport matrix* belonging to given regular sequence s.

Definition 32 *Let $n \geq 1$ be and $s = (s_1, \ldots, s_n)$ be a $(0, 3(n-1), n)$-regular sequence. Then the obligatory sport matrix $\mathcal{O}(s)$ corresponding to s is defined by the following properties:*

1. *the size of the matrix is $n \times 5$, its elements are nonnegative integers;*

2. $wo_i = \max\left(0, \lceil \frac{s_i - (n-1)}{2} \rceil\right)$ *for $i = 1, \ldots, n$;*

3. $do_i = s_i - 3\lfloor \frac{s_i}{3} \rfloor$ *for $i = 1, \ldots, n$;*

4. $lo_i = \max(0, n - 1 - s_i)$ *for $i = 1, \ldots, n$.* □

The i-th row of the matrix contains the (partially estimated) data of T_i for $i = 1, \ldots, n$: index i, number of obligatory wins wo_i, number of obligatory draws do_i, number of obligatory losses lo_i and number of points s_i (the obligatory values are lower bounds for the correct values, the index and the number of points are exact values).

Definition 33 *We say that the obligatory sport matrix $\mathcal{O}(s)$ of s is extendable to a sport matrix $\mathcal{S}(s)$ corresponding to s if*

1. $\mathcal{O}(s)$ *is a sport matrix belonging to s or*

2. *we can increase some w_i, d_i and l_i values so that the result will be a sport matrix $\mathcal{S}(s)$.*

According to the following assertion we get a linear filtering algorithm using the obligatory sport matrix.

Lemma 34 *The obligatory sport matrix $\mathcal{O}(s)$ belonging to a $(0, 3(n-1), n)$-regular sequence s is unique. If $\mathcal{O}(s)$ is not extendable to a sport matrix $\mathcal{S}(s)$ then s is not a football sequence.*

Proof. The obligatory sport matrix is defined by unique formulas therefore it is unique. If s is a football sequence then its obligatory sport matrix $\mathcal{O}(s)$ contains lower bounds for w_i, d_i, and l_i of any sport matrix $\mathcal{S}(s)$ therefore any sport matrix $\mathcal{S}(s)$ can be constructed by the extension of $\mathcal{O}(s)$. $\qquad\square$

The following SPORT-UNIFORM is a *draw-based* algorithm which at first constructs the obligatory sport matrix belonging to s then tries to extend it to a sport matrix so that it allocates the draw packets in a greedy way as uniformly as possible. If the so received draw sequence is not graphic then the investigated sequence is not good.

The base of the uniform allocation of the draws is the following assertion.

Lemma 35 *If* $n \geq 1$, $d = (d_1, \ldots, d_n)$ *is graphical and* $d_i < d_j$ *then the sequence* d'—*received increasing* d_i *by* 1 *and decreasing* d_j *by* 1—*is also graphical.*

Proof. Let G be a $(0, 1, n)$-graph on vertices V_1, ..., V_n having the degree sequence $d = (d_1, \ldots, d_n)$ in which $d_i < d_j$. Then there exists a vertex V_k which is connected with V_j and not connected with V_i. In G delete the edge between V_j and V_k and add the edge between V_i and V_k. Then the received new graph is graphical with the required degree sequence. $\qquad\square$

This lemma has a useful corollary.

Corollary 36 *If* $n \geq 1$, $s = (s_1, \ldots, s_n)$ *is a* $(2, 3, n)$-*regular sequence, and its uniform draw sequence* $u(s) = (u_1, \ldots, u_n)$ *is not graphical, then* s *is not a football sequence.*

Proof. By the recursive application of Lemma 35 we get that if s has a graphical draw sequence then its uniform draw sequence is also graphical. $\qquad\square$

We remark that the problem of the pairing of the draws has a reach bibliography as the problem of degree sequences of simple graphs [24, 32, 47, 50, 51, 62, 80, 89, 107, 110, 111, 112].

Basic parameters of SPORT-UNIFORM are the usual ones further S: the sum of the elements of s; S_0: auxiliary variable; $wo = (wo_1, \ldots, wo_n)$: wo_i is the number of obligatory wins of T_i; $do = (do_1, \ldots, do_n)$: do_i is the number of obligatory draws of T_i; $lo = (lo_1, \ldots, lo_n)$: lo_i is the number of obligatory losses of T_i; $WO = (WO_0, \ldots, WO_n)$: WO_i is the total number of wins of the first i teams; $DO = (DO_0, \ldots, DO_n)$. DO_i is the total number of draws of the first i teams; $LO = (LO_0, \ldots, LO_n)$. LO_i is the total number of the first i teams; $wm = (wm_1, \ldots, wm_n)$: wm_i is the maximal number of wins of T_i;

$dm = (dm_1, \ldots, dm_n)$: dm_i is the maximal number of draw packets of T_i; $lm = (lm_1, \ldots, lm_n)$: lm_i is the maximal number of losses of T_i; Wn: the number of the wins in the tournament; Ln: the number of the losses in the tournaments; Dn: the number of draws in the tournament; D: the current number of yet not allocated draws; $da = (da_1, \ldots, da_n)$: da_i is the number of allocated to T_i draw packets; $wa = (wa_1, \ldots, wa_n)$: wa_i is the number of allocated wins of T_i; $la = (la_1, \ldots, la_n)$: la_i is the number of allocated losses of T_i; $R = (R_0, R_1, R_2)$: R_i is the number of elements of s giving remainder i mod 3; c: average number of draw packets to allocate for a team.

SPORT-UNIFORM(n, s)

```
01 S₀ = WO₀ = DO₀ = LO₀ = R₀ = R₁ = R₂ = L = 0   // line 01: initialization
02 for i = 1 to n                    // line 02–03: computation of the parameters
03     S = S + sᵢ
```

$04 \quad wo_i = \max\left(0, \lceil \frac{s_i - (n-1)}{2} \rceil\right)$

```
05     WOᵢ = WOᵢ₋₁ + woᵢ
```

$06 \quad do_i = s_i - 3\lfloor \frac{s_i}{3} \rfloor$

```
07     DOᵢ = DOᵢ₋₁ + doᵢ
08     R_{doᵢ} = R_{doᵢ} + 1
09     loᵢ = max(n − 1 − sᵢ, 0)
10     LOᵢ = LOᵢ₋₁ + loᵢ
```

$11 \quad dm_i = \min\left(\frac{s_i - do_i}{3}, n - 1 - do_i, \lfloor \frac{3(n-1) - 2do_i - s_i}{6} \rfloor\right)$

$12 \quad wm_i = \frac{s_i - do_i}{3}$

$13 \quad lm_i = \lfloor \frac{3(n-1) - s_i}{3} \rfloor$

```
14 Wn = Ln = Sₙ − n(n − 1)           // line 14: computation of Wn, Ln
15 Dn = D = 3n(n − 1)/2 − S           // line 15: computation of Dn, D
```

$16 \text{ if } \frac{D - DO_n}{3} > \lfloor \frac{D - DO_n}{3} \rfloor$ // line 16–43: allocation of draw packets

```
17     return L
18 while D > 0
```

$19 \qquad c = \lfloor \frac{D}{R_0 + R_1 + R_2} \rfloor$

```
20         while c ≥ 1
21             R₀ = R₁ = R₂ = 0
22             for i = 1 to n
```

$23 \qquad\qquad\qquad da_i = \min(\frac{s_i - d_i}{3}, c)$

```
24                 dᵢ = doᵢ + 3daᵢ
25                 D = D − 3daᵢ
26                 if dᵢ < dmᵢ
```

$$27 \qquad\qquad R_{do_i} = R_{do_i} + 1$$

$$28 \qquad\qquad c = \left\lfloor \frac{D}{R_0 + R_1 + R_2} \right\rfloor$$

29   **if** $0 < \frac{D}{3} \leq R_0$

30   **for** $i = 1$ **to** n

31   **if** $(D > 0 \wedge do_i == 0) ==$ TRUE

32              $d_i = d_i + 3$

33              $D = D - 3$

34   **if** $R_0 < \frac{D}{3} \leq R_0 + R_1$

35   **for** $i = 1$ **to** n

36   **if** $(D > 0 \wedge do_i == 0 \vee do_i == 1) ==$ TRUE

37              $d_i = d_i + 3$

38              $D = D - 3$

39   **if** $R_0 + R_1 > \frac{D}{3}$

40   **for** $i = 1$ **to** n

41   **if** $D > 0$

42              $d_i = d_i + 3$

43              $D = D - 3$

44 sort d in decreasing order resulting d'

45 HHL(d') // line 44–45: sorting of the draw sequence

46 **return** L, d // line 46: s is undecided (if $L = 2$) or bad (if $L = 0$)

4.3.8   Linear filtering algorithm L8 = DRAW-SORTED-UNIQUE

The fifth linear filtering algorithm DRAW-UNIQUE exploits the fact that some football sequences have unique sport matrix implying the uniqueness of the draw sequence. The eighth linear algorithm L8 = DRAW-SORTED-UNIQUE exploits that the uniqueness of the sport matrix is not necessary to have a unique sorted draw sequence.

Sorted version of a sport matrix $\mathcal{S}(s)$ is denoted by $\overline{\mathcal{S}}(s)$ and is defined by the following property: if $1 \leq i < j \leq n$ then either $d_i' < d_j'$ or $d_i' = d_j'$ and $w_i' < w_j'$ or $d_i' = d_j'$ and $w_i' = w_j'$ and $i' < j'$ (d_i' is the draw value in the i-th row of the sorted matrix and i' is the original index belonging to d_i').

DRAW-SORTED-UNIQUE is based on the following assertion.

Lemma 37 *If* $n \geq 1$, $s = (s_1, \ldots, s_n)$ *is a* $(2, 3, n)$-*regular sequence, the sorted versions of the sport matrices* $\mathcal{S}(s)$ *are identical and their joint draw sequence is not graphical, then* s *is not a football sequence.*

Basic parameters of DRAW-SORTED-UNIQUE are the usual ones, further S: the current sum of the elements of s; D: the number of the draws in

the investigated potential tournament; $d = (d_1, \ldots, d_n)$: d_i is the number of draws allocated to T_i; $do = (do_1, \ldots, do_n)$: do_i is the number of obligatory draws of T_i; DO: the number of the obligatory draws in the tournament; $lo = (lo_1, \ldots, lo_n)$: lo_i is the number of obligatory losses of T_i; LO is the number of obligatory losses in the tournament; $wo = (wo_1, \ldots, wo_n)$: wo_i is the number of obligatory wins of T_i; WO is the number of obligatory wins in the tournament; $dm = (dm_1, \ldots, dm_n)$: dm_i is the maximal number of draw packets which can be accepted by T_i; DM: the sum of the dm_i's; $lm = (lm_1, \ldots, lm_n)$: lm_i is of the maximal number of losses of T_i; LM: the sum of the lm_i's; $wm = (wm_1, \ldots, wm_n)$: wm_i is the maximal number of wins of T_i; WM: the sum of the wm_i's; Wn: the number of the wins in the tournament; Ln: the number of the losses in the tournaments; Dn: the number of draws in the tournament; D: the number of yet not allocated draws; $da = (da_1, \ldots, da_n)$: da_i is the number of allocated to T_i draw packets; $wa = (wa_1, \ldots, wa_n)$: w_i is the number of allocated to T_i wins; $la = (la_1, \ldots, la_n)$: la_i is the number of allocated to T_i losses; h: the maximal number of draw packets assigned to a team; $R = R_{i,j}$: a $3 \times h$ sized matrix, where $R_{i,j}$ gives the number of teams which are able at most i draw packets and having score of form $3k + j$; $A = (A_0, A_1, A_2)$: A_j is the number of scores giving $i \bmod 3$; $B = (B_0, \ldots, B_h)$: B_i is the number of teams which are able to accept at most i draw packets; z: number of draw pockets which the program tries to allocate to all teams; fs: first score among the scores receiving maximal number of draw pockets; Rm: critical value of the remainder (mod 3) of the scores.

Draw-Sorted-Unique(n, s)

01 $S = WO = DO = LO = A_0 = A_1 = A_2 = L = 0$ // line 01: initialization
02 **for** $i = 1$ **to** n // line 02–29: test of the obligatory sport matrix
03      $S = S + s_i$
04      $do_i = s_i - 3\lfloor s_i/3 \rfloor$
05      $DO = DO + do_i$
06      $wo_i = \max(0, \lceil \frac{s_i - (n-1)}{2} \rceil$
07      $WO = WO + wo_i$
08      $lo_i = \max(0, n - 1 - s_i)$
09      $LO = LO + lo_i$
10      $dm_i = \min \left(\frac{s_i - do_i}{3}, \frac{n-1-do_i}{3}, \frac{3(n-1) - 2d_i - s_i}{6} \right)$
11      $DM = DM + dm_i$
12      $wm_i = \min \left(\frac{s_i - do_i}{3}, (N - 1) - DO_i \right)$

13 $\quad WM = WM + wm_i$

14 $\quad lm_i = \min\left(\lfloor\frac{3(n-1)-s_i}{3}\rfloor, n-1-do_i\right)$

15 $\quad LM = LM + lm_i$

16 $Dn = 3\binom{n}{2} - S$

17 $Wn = Ln = S - 2\binom{n}{2}$

18 **if** $DO > 2Dn$

19 $\quad$ **return** L

20 **if** $3DM < 2Dn$

21 $\quad$ **return** L

22 **if** $WO > Wn$

23 $\quad$ **return** L

24 **if** $WM < Wn$

25 $\quad$ **return** L

26 **if** $LO > Ln$

27 $\quad$ **return** L

28 **if** $LM < Ln$

29 $\quad$ **return** L

30 $h = \lfloor(n-1)/3\rfloor$ $\qquad\qquad\qquad$ // line 30–45: preparation of the allocation

31 **for** $i = 0$ **to** h

32 $\quad B_i = 0$

33 $\quad$ **for** $j = 0$ **to** 2

34 $\quad\quad R_{j,i} = 0$

35 **for** $i = 1$ **to** n

36 $\quad R_{do_i,dm_i} = R_{do_i,dm_i} + 1$

37 **for** $i = 1$ **to** h

38 $\quad$ **for** $j = 0$ **to** 2

39 $\quad\quad A_j = A_j + R_{i,j}$

40 $\quad\quad B_i = B_i + R_{i,j}$

41 $q = 0$

42 $A = A_0 + A_1 + A_2$

43 $D = 2Dn - DO$

44 $c = \lfloor\frac{D}{A}\rfloor$

45 $q = q + c$

46 **while** $c \geq 1$ $\qquad\qquad\qquad\qquad$ // line 46–78: allocation of the draws

47 $\quad z = 0$

48 $\quad$ **for** $i = q - c + 1$ **to** q

49 $\quad\quad z = z + iB_i$

50 $\quad\quad D = D - 3z$

```
51                 for i = 0 to 2
52                     for j = q − c + 1 to q
53                         A_i = A_i − R_{i,j}
54                 A = A_0 + A_1 + A_2
55                 c = ⌊D/A⌋
56 if q > 0
57     for i = 1 to n
58         d_i = d_i − 3 min(z, dm_i)
59 if D == 0
60     go to 79
61 fs = −1
62 Rm = 2
63 if D ≤ A_1 + A2
64     Rm = 1
65 if D ≤ A_1
66     Rm = 0
67 for i = 1 to n
68     if (dm_i > q) ∧ (do_i ≤ Rm) == TRUE
69         if fs == −1
70             fs = s_i
71         else if s_i ≠ fs
72                 return L, d
73         if do_i < Rm
74             d_i = d_i + 3
75             D = D − 3
76         if (do_i == Rm) ∧ (D > 0) == TRUE
77             d_i = d_i + 3
78             D = D − 3
```

79 sort d in nonincreasing order resulting d' // line 79–80: sorting of d

80 HHL(d')

81 **return** L, d   // line 81: return the result of HHL

Procedure HHL (HAVEL-HAKIMI-LINEAR) is described in [60]. We remark that the original Havel-Hakimi algorithm requires in worst case $\Theta(n^2)$ time. Recently Király [70] published a quicker algorithm which uses the data structure proposed by van Emde Boas [27, 71, 114] and requires only $O(n \log \log n)$ time. Our algorithm is linear and works also for some multigraphs.

A natural requirement is $d_i \leq n − 1$ but $d_i > n − 1$ can occur only in the cases $s = (0, 2)$ and $s = (1, 2)$ which are filtered by the constant time algorithms.

4.3.9 Efficiency of linear time filtering algorithms

LINEAR is the union of the described linear time algorithm.

```
LINEAR(n, s)
01 L = 0                                    // line 01: initialization of L
02 L1(n, s)                                 // line 02–04: filtering by COMPLETE
03 if L = 0
04    return L
05 L2(n, s)                                 // line 05–07: filtering by LOSSES
06 if L = 0
07    return L
08 L3(n, s)                                 // line 08–10: filtering by REDUCTION0
09 if L = 0
10    return L
11 L4(n, s)                                 // line 11–13: filtering by REDUCTION1
12 if L = 0
13    return L
14 L5(n, s)                                 // line 14–16: filtering by DRAW-UNIQUE
05 if L = 0
16    return L
17 L6(n, s)                                 // line 17–19: filtering by BALANCED
18 if L = 0
19    return L
20 L7(n, s)                                 // line 20–22: filtering by DRAW-UNIFORM
21 if L = 0
22    return L
23 L8(n, s)                                 // line 23–25: filtering by DRAW-SORTED-UNIQUE
24 if L = 0
25    return L
26 L = 1                    // line 26–27: the linear time algorithms can not decide
27 return L
```

Since all included algorithms have linear worst case running time, the total running time of LINEAR is also $O(n)$. Since the best running time of L1 is $O(1)$, therefore the best running time of LINEAR is also $O(1)$.

Tables 4 and 5 show the concrete filtering results of the linear time filtering algorithms. Table 4 contains the number of regular sequences (R), the number of sequences, accepted by C9, L1 = COMPLETE-TEST, L2 = LOSSES and L3 = REDUCTION0.

n	C9	L1	L2	L3 + L4
1	**1**	**1**	**1**	**1**
2	**2**	**2**	**2**	**2**
3	14	12	10	10
4	203	134	94	87
5	2133	1230	901	814
6	20518	10947	8348	7526
7	191707	97427	76526	69349
8	1772842	872234	699344	637735
9	16332091	7851193	6387443	5859125
10	150288309	71001641	58367243	53817029
11	1383099467	644668154	538591486	494427384
12	12737278674	5873396400	4888701306	4544762304
13	117411184292	53669099755	44823480671	41804695971
14	1083421567402	491669304392	411496549436	384847810936

Table 4: Results of filtering by linear tests tests L1, L2, and L3 + L4 for $n = 1, \ldots, 14$ teams.

Table 5 contains the number of sequences accepted by L4 = REDUCTION1, L5 = DRAW-UNIQUE, L6 = BALANCED, L7 = SPORT-UNIFORM and L8 = INNER-DRAW, further the number of the football sequences (F) and the cumulated running time and (the exact values of L7 are bold).

4.4 Quadratic filtering algorithms

In this section the quadratic recursive filtering algorithms Q1 = BALANCED-QUAD, Q2 = REDUCTION-REC-SMALL, and Q3 = REDUCTION-REC-LARGE are described.

4.4.1 Quadratic filtering algorithm Q1 = BALANCED-QUAD

The filtering algorithm Q1 = BALANCED-QUAD is based on the observation that if the draw sequence is unique, then the victory sequence $w = (w_1, \ldots, w_n)$ and the corresponding loss sequence $l = (l_1, \ldots, l_n)$ are also unique, further that the wins (losses) of any subset of teams have to be paired with inner and outer losses (wins). The following assertion gives a necessary condition for the reconstructability of the sequence pair (v, l).

n	$L5 + L6$	$L7 + L8$	F	t
1	**1**	**1**	**1**	0.000
2	**2**	**2**	**2**	0.000
3	**7**	**7**	**7**	0.000
4	46	**40**	**40**	0.000
5	475	365	355	0.000
6	4459	4086	3678	0.015
7	47867	44657	37263	0.047
8	460153	451213	361058	0.437
9	4371783	4348655	3403613	4.196
10	41261057	41166157	31653777	40.217
11	387821927	387416935	292547199	393.280
12	3635039265	3633749149	2696619716	3828.002
13	34011137972	33821636274		37611.185
14	317827900632	316291028902		364978.049

Table 5: Results of filtering by linear tests $L5 + L6$ and $L7 + L8$, further the number of football sequences (F) and the running time of L8 (t) for $1, \ldots, 14$ teams.

Lemma 38 *If* $(a_1, \ldots, a_n)$ *is the monotone nonincreaing win sequence and* $(b_1, \ldots, b_n)$ *is the corresponding loss sequence of a football tournament then*

$$\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} \min(b_i, k-1) + \sum_{i=k+1}^{n} \min(b_i, k) \tag{34}$$

for all $k = 1, \ldots, n,$ *with equality for* n.

Proof. The wins included in the sum of the left side of (34) have to be paired with the "inner losses" (losses among $T_1, \ldots, T_k$) and "outer losses" (losses of $T_1, \ldots, T_k$ in the matches against the remaining teams). $\square$

We remark that this lemma is a consequence of Theorem 3 of the recent paper due to Berger [10] containing a necessary and sufficient condition for some incomplete $(0, 2, n)$-tournaments. As the sequence $(1, 1, 8, 9, 9)$ satisfying 34 shows, in our case (34) is only a necessary condition, since s has a unique sport matrix shown in Table 6 which is not reconstructable.

The paper [33] contains an algorithm for our problem but the algorithm does not terminate for some inputs.

i	w_i	d_i	l_i	s_i
1	0	1	3	1
2	0	1	3	1
3	2	2	0	8
4	3	0	1	9
5	3	0	1	9

Table 6: Unique sport matrix belonging to the sequence $s = (1, 1, 8, 9, 9)$.

The following natural implementation BALANCED-QUAD of Lemma 38 requires quadratic time.

Parameters of BALANCED-QUAD are the usual ones, further $w = (w_1, \ldots, w_n)$: w_i is the number of wins allocated to T_i ($0 \leq w_i \leq n-1$); $l = (l_1, \ldots, l_n)$: l_i is the number of losses allocated to T_i ($0 \leq l_i \leq n-1$); Sw: the current number of the necessary wins; Ss: the maximal number of pairable losses of teams having small indices; Sl: the maximal number of pairable losses of the teams having large indices.

BALANCED-QUAD(n, w, l)

```
01 Sw = L = 0                        // line 01: initialization of Sw and L
02 sort (w, l) nonincreasingly in w using COUNTING-SORT
03 for i = 1 to n                    // line 03–13: counting of wins and losses
04      Ss = Sl = 0
05      Sw = Sw + w_i
06      for j = 1 to i               // line 06–07: small indices
07          Ss = Ss + min(w_j, i − 1)
08      for j = i + 1 to n           // line 08–09: large indices
09          Sl = Sl + min(w_j, i)
10      if Sw > Ss + Sl              // line 10–13: (w, l) is not pairable
11          return L
12 if Sw < Ss + Sl
13      return L
14 else L = 2                        // line 14–15: s is undecided
15      return L
```

We yet did not implemented BALANCED-QUAD.

4.4.2 Quadratic filtering algorithm Q2 = Reduction-Rec-Small

Algorithm Q2 = Reduction-Rec-Small is based on the recursive applica-
tion of Recursive0 and Recursive1. Using Q2 and the next Q3 we shorten
the input sequences and often can filter them.

Parameters are the usual ones, further $e = (e_1, \ldots, e_N)$: work version of the
investigated sequence; n_l: smallest index of not deleted elements of s.

Reduction-Rec-Small(n, s)

```
01 L = S = 0                    // line 01–04: initialization of L, S, n_l, and e
02 for i = 1 to n
03      e_i = s_i
04 n_l = 1
05 while n_l ≤ n
06        S = 0
07        for i = n_l to n
08            S = S + e_i
09            if S == i(i − 1)              // line 09–21: S is minimal
10              if i < n
11                if (e_{n_l} ≠ i − 1) ∨ (e_{n_l+i−1} ≠ i − 1) == TRUE
12                  return L
13                if e_{n_l+i} < 3i
14                  return L
15              if i == n
16                if (e_{n_l} ≠ i − 1) ∨ (e_n ≠ i − 1) == TRUE
17                  return L
18                else L = 1
19                      return L
20              n_l = n_l + i
21              for j = n_l to n
22                  e_j = e_j − 3i
23              go to 05
24            if S == i(i − 1) + 1        // line 22–35: S is minimum plus one
25              if i < n
26                L_1 = (e_{n_l} = i − 1) ∧ (e_{n_l+i−2} = i − 1) ∧ (e_{n_l+i−1} = i)
                       ∧(e_{n_l+i} ≥ 3i − 2)
27                L_2 = (e_{n_l} = i − 2) ∧ (e_{n_l+i−2} = i − 1) ∧ (e_{n_l+i−1} = i + 1)
28                if (L_1 == FALSE) ∧ (L_2 == FALSE) == TRUE
29                  return L          // line 28–29: s is not football sequence
```

```
30              if i == n
31                L_2 = (e_{n_l} = n - 2) ∧ (e_{n_l+1} = n - 1) ∧ (e_{n-1} = n + 1)
32                if L_2 == False   // line 32–33: s is not football sequence
33                  return L
34              n_l = n_l + i
35              for j = n_l to n
36                  e_j = e_j − 3i
37              go to 05
38            Reduction-Rec-Large(n − n_l + 1, e)
39            if L == 0
40              return L
41 if n_u > 0
42    Filter(n_u, e)
43 if L == 0
44    return L
45 L = 2                                    // line 45–46: s is undecided
46 return L
```

Reduction-Rec-Small calls Filter which is a union of the constant and linear time filtering algorithms and Reduction-Rec-Large which is the next quadratic filtering algorithm.

Filter(n, e)

```
01 Constant(n, e)   // line 01–03: filtering by the constant time algorithms
02 if L == 0
03    return L
04 Linear(n, e)            // line 04–06: filtering by the linear time algorithms
05 if L == 0
06    return L
07 L = 2                                    // line 07–08: s is undecided
08 return L
```

4.4.3   Quadratic filtering algorithm Q3

Algorithm Q3 = Reduction-Rec-Large is based on the recursive application of Recursive0 and Recursive1.

Parameters are the usual ones, further $e = (e_1, \ldots, e_N)$: work version of the investigated sequence; n_u: smallest index of the not deleted elements of s; Q: the sum of the i largest scores; B: the number of investigated scores giving

0 remainder mod 3; C: the number of investigated scores giving remainder 1 mod 3; D: the number of investigated scores giving remainder 2 mod 3.

REDUCTION-REC-LARGE(n, e)

```
01 L = Q = B = C = D = 0  // line 01–02: initialization of L, Q, B, C, D, n_u
02 n_u = n
03 while n_u ≤ 1                          // line 03–25: recursive reduction
04         for i = n_u downto 1       // line 04–09: preparing of the filtering
05                 Q = Q + e_i
06                 if e_i − 3⌊e_i/3⌋ == 0
07                     B = B + 1
08                 if e_i − 3⌊e_i/3⌋ == 1
09                     C = C + 1
10                 if e_i − 3⌊e_i/3⌋ == 2
11                     D = D + 1
12                 if Q == 3i(n_u − i) + 3i(i − 1)/2   // line 12–17: Q is maximal
13                     if B ≠ i
14                         return L, n_u
15                     if i > 1
16                         if e_{n_u−i} > 3(n_u − i − 1)
17                         return L, n_u
18                     n_u = n_u − i
19                     go to 03
20                 if Q == 3i(n_u − i) + 3i(i − 1)/2 − 1
21                     if (B == i − 2) ∧ (C == 2) == FALSE
22                         return L, n_u
23                     if i > 1                     // line 20–25: Q is maximum minus 1
24                         if e_{n_u−i} > 3(n_u − i − 1)
25                             return L, n_u
26 L = 2                                          // line 26–27: s is not decided
27 return L, n_u
```

The following Table 7 contains the results of quadratic filtering algorithms.

5   Reconstruction of potential football sequences

In this part we investigate polynomial reconstruction algorithms, as R1 = REDUCTION, R2 = DRAW-UNIFORM-REC, and R3 = DRAW-INNER-REC.

n	Linear	Q2 + Q3	F	t
1	**1**	**1**	**1**	0.000
2	**2**	**2**	**2**	0.000
3	**7**	**7**	**7**	0.000
4	**40**	**40**	**40**	0.000
5	365	**355**	**355**	0.000
6	4086	3760	**3 678**	0.015
7	44657	39417	**37 273**	0.109
8	451213	393072	**361 058**	1.264
9	4348655	3804485	**3 403 613**	15.226
10	41166157	36302148	**31 653 777**	179.249
11	387416935	344012885	**292 547 199**	2066.323
12	3633749149	3246651763	**2 696 619 716**	23429.877
13	33821636274	30405902165		

Table 7: Results of filtering by Linear and quadratic algorithms Q2 + Q3, further the number of football sequences (F) and the running time of Q3 (t) for $n = 1, \ldots, 13$ teams.

5.1   Reconstruction algorithm R1 = Reduction

R1 = Reduction is based on filtering algorithms Reduction0 and Reduction1.

5.2   Reconstruction algorithm R2 = Draw-Uniform-Rec

R2 = Draw-Uniform-Rec is based on filtering algorithms: it tries—using the degree sequence d produced by Sport-Uniform or Draw-Sorted-Unique and using a greedy pairing algorithm "largest wins with largest losses"—to pair the wins and losses.

Parameters of R2 are the usual ones further S: sport matrix computed using the output draw sequence d of Sport-Uniform or Draw-Sorted-Unique and sorted its rows so that either $w_i > w_{i+1}$ or $w_i = w_{i+1}$ and $l_i \leq l_{i+1}$; $d = (d_1, \ldots, d_n)$: draw sequence of S; $\mathcal{M}_{n \times n}$ (result matrix): $\mathcal{M}_{i,j}$ is the number of points received by T_i in the match against T_j; $w = (w_1, \ldots, w_n)$: w_i is the number of wins of T_i; $l = (l_1, \ldots, l_n)$: l_i is the number of losses of T_i.

Draw-Uniform-Rec(n, s, d)

01 **for** $i = 1$ **to** n // line 01–03: initialization of $\mathcal{M}$

```
02      for j = 1 to n
03          𝓜_{i,j} = 0
04 HAVEL-HAKIMI-DRAWS(n, s, d, 𝓜)    // line 04: HHD allocates the draws
05 for i = 1 to n                     // line 05–07: computation of w and l
06      w_i = (s_i − d_i)/3
07      l_i = n − 1 − d_i − w_i
08 for i = n downto 1                 // line 08–24: allocation of wins and losses
09      j = n
10      while (w_i > 0) ∨ (M_{ij} ≠ 1) ∨ (j > 0) ∨ (i ≠ j) ∨ (l_j > 0) == TRUE
11          𝓜_{ij} = 3
12          w_i = w_i − 1
13          l_j = l_j − 1
14          j = j − 1
15      if w_i > 0                    // line 15–17: s is undecided
16        L = 2
17        return L, 𝓜
18 L = 1                              // line 18–19: s is a football sequence
19 return L, 𝓜
```

R2 uses a special version of Havel-Hakimi algorithm called HAVEL-HAKIMI-DRAWS (or shortly HHD). While for the classical Havel-Hakimi algorithm the equal scores are equivalent, in this application we have to distinguish them.

Additional parameters are $d = (d_1, \ldots, d_n)$: a draw sequence produced by DRAW-REC; $\mathcal{M}$: $n \times n$ sized matrix where $\mathcal{M}_{ij}$ is the number of points received by T_i in the match with T_j; $\mathcal{E} = (E_1, \ldots, E_n) = ((e_1, h_1), \ldots, (e_n, h_n))$: current extended and sorted version of d; $H = (h_1, \ldots, h_n)$: h_i is the index of e_i in d; n_l: lower index of the essential part of $\mathcal{E}$; n_u: upper index of the essential part of $\mathcal{E}$; $c = (c_0, \ldots, c_n)$: c_i is the number of i's among $e_{n_l}, \ldots, e_{n_u}$; $C = (C_0, \ldots, C_n)$: C_i is the cumulated number of i's among $e_{n_l}, \ldots, e_{n_u}$.

HAVEL-HAKIMI-DRAWS(n, d, 𝓜)

```
01 n_l = 1                     // line 01–05: initialization of n_l, n_u, and 𝓔;
02 n_u = n
03 for i = n_l to n_u          // line 03–07: initialization of G and n_u;
04      e_i = d_i
05      h_i = i
07 n_u = n
08 for i = 1 to n              // line 08–15: pairing of the draws;
09      COUNTING-SORT-DRAWS(n, i, n_u, 𝓔)        // line 09: sorting
```

```
10      if e_i = 0
11         return M
12      for k = 1 to e_i
13           M_{h_i,h_i+k} = M_{h_i+k,h_i} = 1              // line 13: a draw is fixed
14           e_{i+k} = e_{i+k} + 1
15      while n_u == 0
16             n_u = n_u - h_i
17 return M        // line 17: return the matrix containing the paired draws
```

COUNTING-SORT-DRAWS is a modified version of the well-known linear time sorting algorithm COUNTING-SORT [27].

Additional parameters are $d = (d_1, \ldots, d_n)$: a draw sequence produced by DRAW-UNIFORM-REC; n_l: lower index of the essential part of $\mathcal{E}$; n_u: upper index of the essential part of $\mathcal{E}$; $\mathcal{M}$: $n \times n$ sized matrix where $\mathcal{M}_{ij}$ is the number of points received by T_i in the match with T_j; $\mathcal{E} = (E_1, \ldots, E_n) = ((g_{11}, g_{12}), \ldots, (g_{1n}, g_{2n}))$: current extended and sorted version of d with the corresponding indices; $\mathcal{G}$: the working version of $\mathcal{E}$; n_l : lower index of the essential part of $\mathcal{E}$; n_u: upper index of the essential part of $\mathcal{E}$; $c = (c_0, \ldots, c_{n-1})$: c_i is the number of i's among $g_{1,n_l}, \ldots, g_{1,n_u}$; $C_n = 0$ working variable; $C = (C_0, \ldots, C_{n-1})$: C_i is the number of investigated scores larger or equal with i.

```
COUNTING-SORT-DRAWS(n, d, n_l, n_u, E)
01 for i = n_l to n_u                    // line 01–05: initialization of G and c;
02      g_{1,i} = e_{1,i}
03      g_{2,i} = e_{2,i}
04 for i = 0 to n - 1
05      c_i = 0
06 for i = n_l to n_u                    // line 06-10: computation of the counters
07      c_{g1i} = c_{g1i} + 1
08 C_n = 1
09 for n - 1 downto 0
10      C_i = C_{i+1} + c_i
11 for i = n_l to n_u                    // line 11-16: computation of the new E
12      x = C_{g_{1,i}} + 1
13      e_{1,x} = g_{1,i}
14      e_{2,x} = g_{2,i}
15      C_x = C_x + 1
16 return E
```

The running time of COUNTING-SORT-DRAW is $\Theta(n)$, of HAVEL-HAKIMI-DRAW is $O(n^2)$ and the one of DRAW-UNIFORM-REC is also $O(n^2)$.

As an example let $s = (1, 1, 7, 7)$. Then s has a unique draw sequence $(1, 1, 1, 1)$ and unique sport matrix shown in Table 8.

i	w_i	d_i	l_i	s_i
1	0	1	2	1
2	0	1	2	1
3	2	1	0	7
4	2	1	0	7

Table 8: Unique sport matrix belonging to the sequence $s = (1, 1, 7, 7)$.

According to the relatively quick version HAVEL-HAKIMI-SHIFTING [62] T_1 plays a draw with T_4 and T_2 with T_3 resulting the partial result matrix shown in Table 9.

i	T_1	T_2	T_3	T_4	s_i
1	—	?	?	1	1
2	?	—	1	?	1
3	?	1	—	?	7
4	1	?	?	—	7

Table 9: Partial result matrix belonging to the draws of $s = (1, 1, 7, 7)$.

The partial result matrix containing the draws in Table 9 is not reconstructible since no acceptable result for the match between T_1 and T_2.

If we use the classical Havel-Hakimi algorithm then the draws are between T_1 and T_2, further between T_3 and T_4 and our greedy algorithm DRAW-UNIFORM-REC reconstructs the received partial result matrix.

Another example let $s = (1, 1, 8, 8, 10, 13)$. Then s has a unique draw sequence $(1, 1, 2, 2, 1, 1)$ and a unique sport matrix shown in Table 10.

i	w_i	d_i	l_i	s_i
1	0	1	4	1
2	0	1	4	1
3	2	2	1	8
4	2	2	1	8
5	3	1	1	10
6	4	1	0	13

Table 10: Unique sport matrix belonging to the sequence $s = (1, 1, 8, 8, 10, 13)$.

In this case at first $\mathcal{E} = ((2,3),(2,4),(1,1),(1,2),(1,5),(1,6))$. The draws allocated by HHD are shown in Table 11.

i	T_1	T_2	T_3	T_4	T_5	T_6	s_i
1	—	?	1	?	?	?	1
2	?	—	?	1	?	?	1
3	1	?	—	1	?	?	8
4	?	1	1	—	?	?	8
5	?	?	?	?	—	1	10
6	?	?	?	?	1	—	13

Table 11: Partial result matrix belonging to the draws of $s = (1,1,8,8,10,13)$.

The partial result matrix in Table 11 is not reconstructible since no acceptable result for the match between T_1 and T_2.

5.3 Reconstruction algorithm R3 = Draw-Inner-Rec

Reconstruction algorithm R3 = DRAW-INNER-REC is an improved version of R2: it takes into account the obligatory inner draws.

The base of INNER-DRAWS is the following lemma.

Lemma 39 *If* $n \geq 1$, $f = (f_1, \ldots, f_n)$ *is a football sequence,* $1 \leq k \leq n$ *and*

$$\sum_{i=1}^{k} f_i < 3\binom{k}{2}, \tag{35}$$

then among the teams T_1, ..., T_k *there are at least*

$$\left\lceil \left(3\binom{k}{2} - \sum_{i=1}^{k} f_i \right) / 2 \right\rceil \tag{36}$$

draws.

Proof. If

$$\left\lceil 2 \left(3\binom{k}{2} - \sum_{i=1}^{k} f_i \right) / 2 \right\rceil = q > 0, \tag{37}$$

then the first k teams lost at least q points due to inner draws (or even more, if they gathered points in the matches against the remaining teams). $\qquad\square$

Trying to reconstruct the sequence $s = (1, 1, 8, 8, 10, 13)$ which was the last example of the previous Section 5.2 DRAW-INNER-REC (see Table 10 and 11) recognizes that $s_1 + s_2 = 2$ therefore according to Lemma 39 the obligatory result between T_1 and T_2 is a draw. Then DRAW-INNER-REC finishes the allocation of the draws as it is shown in Table 12.

1	—	1	?	?	?	?	1
2	1	—	?	?	?	?	1
3	?	?	—	1	1	?	8
4	?	?	1	—	?	1	8
5	?	?	1	?	—	?	10
6	?	?	?	1	1?	—	13

Table 12: Partial result matrix belonging to the draws of $s = (1, 1, 8, 8, 10, 13)$ allocated by DRAW-INNER-REC.

Using the matrix of the allocated draws shown in Table 12 DRAW-UNIFORM-REC produces the complete result matrix shown in Table 13. proving that $s = (1, 1, 8, 8, 10, 13)$ is a football sequence.

i	T_1	T_2	T_3	T_4	T_5	T_6	s_i
i	T_1	T_2	T_3	T_4	T_5	T_6	s_i
1	—	1	0	0	0	0	1
2	1	—	0	0	0	0	1
3	3	3	—	1	1	0	8
4	3	3	1	—	0	1	8
5	3	3	1	3	—	0	10
6	3	3	3	1	3	—	13

Table 13: Partial result matrix belonging to the draws of $s = (1, 1, 8, 8, 10, 13)$ allocated by DRAW-INNER-REC.

The algorithm based on this lemma yet is is not implemented.

6   Enumeration of football sequences

There are many publications connected with the generation [5, 52, 58, 100] and enumeration of degree sequences of graphs, e.g. [4, 5, 8, 21, 26, 49, 62, 63, 67, 72, 78, 79, 81, 87, 92, 97, 98, 105, 117]. The problems connected with directed graphs sometimes are considered as problems of orientation of undirected graphs [37, 36, 38, 39].

The enumeration of degree [4, 21, 40, 62, 63] and score [49, 53] sequences also has a reach literature.

The first published enumeration results connected with football score sequences belong to Gábor Kovács, Norbert Pataki, Zoltán Hernyák and Tamás Hegyessy [73] who computed $F(n)$ for $n = 1, \ldots, 8$ in 2002. N. J. A. Sloane in May 2007 determined $F(9)$, then in June 2008 Min Li computed $F(10)$. The newest results were received by J. E. Schoenfield who computed $F(11)$ in September of 2008 and $F(12)$ in December of 2008 [100].

Connected problems are the listing of all degree sequences and sampling of degree sequences [11, 15, 28, 65, 66, 82].

Our basic method is similar as we enumerated the degree sequences of simple graphs [62, 103].

From one side we try to test the elements of the possible smallest set, and from the other side we try to use quick as possible testing and reconstruction algorithms.

A natural idea is to investigate only the nonincreasing sequences of integers having 0 as lower bound and $3(n-1)$ as upper bound. Paul Erdős and Tibor Gallai called such sequences *regular* [32]. The number of such sequences is given by (1).

6.1   Decreasing of the number of the investigated sequences

A useful tool of the enumeration of the number of football sequences is the decreasing of the number of the considered sequences.

In Section 4 we proposed and analyzed filtering of regular sequences with constant, linear and quadratic time algorithms. For 14 teams we excluded more then the half of the regular sequences by the constant time algorithms. For 13 teams the linear and quadratic algorithms left less then 10.58 percent of the regular sequences. In Section 5 the polynomial reconstruction algorithms decreased the fraction of the undecided regular sequences to 4.68 percent of the regular sequences.

6.2   Backtrack filtering and accepting test

This method is due to Antal Iványi [54, 73].

The results of the filtering algorithms are summarized in Table 14.

The running time of the filtering algorithms are presented in Table 15. The times are cumulated and contain the time necessary for the generation of the sequences too.

n	Constant	Linear	Quad	Backtrack = F
1	**1**	**1**	**1**	**1**
2	**2**	**2**	**2**	**2**
3	14	**7**	**7**	**7**
4	203	**40**	**40**	**40**
5	2 133	365	**355**	**355**
6	20 518	4 086	3 760	**3 678**
7	191 707	44 657	39 417	27 263
8	1 772 442	451 213	393 072	361 058
9	16 332 091	4 348 655	3 804 485	3 403 613
10	150 288 309	41 166 157	36 302 148	31 653 777
11	1 383 099467	387 416 935	344 012 885	292 547 199
12	12 737 278 674	3 633 749 149	3 246 651 763	2 696 619 716
13	117 411 154 292	33 821 636 274	30 405 902 165	
14	1 083 421 567 482			

Table 14: Numbers of sequences accepted by constant, linear and quadratic time and BACKTRACK filtering algorithms for $n = 1, \ldots, 14$ teams.

n	Constant	Linear	Quad	Backtrack = F
1	0.000	0.000	0.000	0.000
2	0.000	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	0.015
7	0.016	0.031	0.042	0.172
8	0.046	0.375	0.577	52.603
9	0.468	3.572	5.772	
10	4.134	34.632	54.741	
11	37.612	329.816	525.752	
12	343.575	3 145.494	4 998.831	
13	3 142.469	30 541.260	49 035.625	
14	29 438.094			

Table 15: Running times of constant, linear and quadratic time filtering algorithms for $n = 1, \ldots, 14$ teams.

The individual results of the reconstruction algorithms are summarized in Table 16.

The running times of the reconstruction algorithms are shown in Table 17.

n	R1	R2 + R3	BACKTRACK	F
1	1	0	0	1
2	2	0	0	2
3	6	1	0	7
4	18	22	0	40
5	50	305	0	355
6	137	3 460	81	3 678
7	375	33 993	2 895	37 263
8	1 023	304 349	56 909	361 058
9	2 776	2 576 124		3 403 613
10	7 498	21 453 751		31 653 777
11	20 177	177 819 555		292 547 199
12	54 127	1 476 661 425		2 696 619 716
13	144 708	12 300 060 430		

Table 16: Number of $(0, 3n - 3, n)$-regular sequences reconstructed by reconstruction algorithms R1, R2 + R3 and BACKTRACK for $n = 1, \ldots, 14$ teams.

n	R1	R3	BACKTRACK
2	0.000	0.000	0.000
3	0.000	0.000	0.000
4	0.000	0.000	0.000
5	0.000	0.000	0.000
6	0.000	0.015	0.015
7	0.063	0.109	0.172
8	0.546	1.264	52.603
9	5.491	15.226	
10	53.880	179.249	
11	522.386	2 066.323	
12	4 998.831	23 429.877	
13	49 035.625	261 904.750	

Table 17: Running times of the R1, R3 and BACKTRACK reconstructing algorithms for $n = 1, \ldots, 13$ teams.

6.3 Recursive accepting test

This method is due to Schoenfield [100]. According to this method we compare the sequences of length n passed through the filtering and accepting tests with the good sequences of length $n-1$ whether they can be derive from them.

Since if we omit a team with its results from a football matrix of size $n \times n$, then we get a football matrix of size $(n-1) \times (n-1)$, therefore we regularly delete the *first* elements of the investigated n-length sequences.

Let $n \geq 2$. We suppose that when we enumerate the n-length good sequences then we know the $F(n-1) \times (n-1)$ sized matrix M containing the $(n-1)$-length good sequences in lexicographically increasing order, and also know the vector $(P_0, \ldots, P_k)$, where $k = \lfloor 3(n-1)/2 \rfloor$ and P_i gives the number of $(n-1)$-length good sequences starting with i.

Let start the recursion with $n = 2$. Matrix M_1 contains only one row (0) and P contains one element $P(1) = 1$.

The constant time filtering algorithms accept only the sequences $(0,3)$ and $(1,1)$. At first we omit 0 from the first sequence and state that the remaining sequence (3) can be derived from (0) only if the team having zero points in the shorter sequence wins against the omitted player. So the omitted player has to have zero points. Since the omitted score is exactly zero, $(0,3)$ *is a good sequence.*

Then we delete the first element from the sequence $(1,1)$ and state that the player having zero points has to play a draw with the omitted team. Since it has exactly one point, therefore $(1,1)$ is also *a good sequence* and so $F(2) = 2$.

Now let $n = 3$. Then M_2 contains two rows: $(0,3)$ and $(1,1)$. In this case the filtering algorithms accept only the seven good sequences: $(0,3,6)$, $(0,4,4)$, $(1,1,4)$, $(1,2,4)$, $(1,3,4)$, $(2,2,2)$ and $(3,3,3)$.

At first we delete 0 from $(0,3,6)$ and compare the remaining $(3,6)$ with the known good sequences. There are thee possibilities: the first team of the good sequence received 3, 1 or 0 points against the omitted one. If 3, then the good sequence has to start with 0. There is only one sequence $(0,3)$ requiring two losses for the omitted team. Since the omitted element is exactly zero, $(0,3,6)$ *is a good sequence.*

The second accepted sequence is $(0,4,4)$. Omitting 0 and comparing $(4,4)$ with the good sequences we get, that $(1,1)$ is the only potential ancestor requiring zero points for the deleted team. Since it has exactly zero points, $(0,4,4)$ is also a *good sequence.*

In a similar way we can prove that the remaining five accepted sequences are also good.

When $n = 4$ then M contains seven elements and $P = (1, 3, 6, 7)$.

RECONSTRUCT executes this recursive step. Its additional parameters are $F(n-1)$: the number of $(n-1)$-length good sequences; $\mathcal{M}_{F(n-1)\times(n-1)}$: matrix of good sequences of length $n-1$ (this matrix consists of submatrices containing the good sequences having identical first element; $P = (P_0, \ldots, P_k)$, where $k = k = \lfloor 3(n-1)/2 \rfloor$ and P_i is the number of $n-1$ length football sequences starting with i; $\mathcal{N}_{F(n)\times n}$: matrix of good sequences of length n; $m = (m_1, \ldots, m_{n-1})$: the current reduced version of s; d: the current score of the deleted team.

RECONSTRUCT(n, s, F, M, P)

```
01 L = 1 line 01–02: initialization of L and u
02 u = ⌊3(n − 1)/2⌋
03 if s₂ ≤ u                        // line 03-21: omitted element starts with a loss
04     j ← P_{s₂}
05     while 𝓜_{j,1} == s₂
06             d ← 0
07             k ← 2
08             while k ≤ n
09                     if s_k − 𝓜_{j,k} == 3
10                         d = d + 0
11                         go to 19
12                     if s_k − 𝓜_{j,k} == 1
13                         d = d + 1
14                         go to 19
15                     if s_k − 𝓜_{j,k} == 0
16                         d = d + 3
17                         go to 19
18                     go to 22
19                     k ← k + 1
20     if d == s₁
21         return L
22 if 0 ≤ s₂ − 1                     // line 22-40: omitted element starts with a draw
23     j ← P_{s₂−1}
24     while 𝓜_{j,1} == s₂ − 1
25             d ← 1
26             k ← 2
27             while k ≤ n
28                     if s_k − 𝓜_{j,k} == 1
29                         d = d + 1
```

```
30                    go to 38
31                if s_k − M_{j,k} == 1
32                    d = d + 1
33                    go to 38
34                if s_k − M_{j,k} == 1
35                    d = d + 1
36                    go to 38
37                go to 39
38                k = k + 1
39    if d == s_1
40        return L
41 if 0 ≤ s_2 − 3              // line 41-59: omitted element starts with a win
42        j ← P[s_2 − 3]
43    while M_{j,1} == s_2 − 3
44            d ← 3
45            k ← 2
46            while k ≤ n
47                    if s_k − M_{j,k} == 3
48                        d = d + 3
49                        go to 57
50                    if s_k − M_{j,k} == 1
51                        d = d + 1
52                        go to 57
53                    if s_k − M_{j,k} == 1
54                        d = d + 1
55                        go to 57
56                    go to 58
57                    k ← k + 1
58        if d == s_1
59            return L
60 L = 0
61 return L
```

Table 18 shows the number of regular sequences ($R(n)$, the number of football sequences ($F(n)$, the ratio ($R(n + 1)/R(n)$), the ratio $F(n + 1)/F(n)$, and the ratio ($F(n)/R(n)$) for $n = 1, \ldots, 12$. In this table if $n \geq 2$ then $R(n)$ is decreasing.

Lemma 40 *If n tends to infinity then $R(n + 1)/R(n)$ tends to 256/27.*

n	$R(n)$	$\frac{R(n+1)}{R(n)}$	$F(n)$	$\frac{F(n+1)}{F(n)}$	$\frac{F(n)}{R(n)}$
1	1	10.000	1	2.000	1.0000
2	1	8.400	2	3.500	0.2000
3	84	8.512	7	5.714	0.0833
4	715	8.655	40	8.875	0.0559
5	6188	8.769	355	10.361	0.0574
6	54264	8.859	3678	10.131	0.0678
7	480700	8.929	37263	9.689	0.0775
8	4292145	8.986	361058	9.427	0.0841
9	38567100	9.032	3403613	9.300	0.0883
10	348330136	9.070	31653777	9.242	0.0909
11	3159461968	9.103	292547199	9.217	0.0926
12	28760021745	9.131	2696619716		0.0938
13	262596783864	9.155			
14	240397990420				

Table 18: Number of regular and football sequences and the ratio of these numbers for neighboring numbers of teams

Proof. According to (1)

$$\frac{R(n+1)}{R(n)} = \frac{(4n+1)(4n)(4n-1)(4n-2)}{(n+1)(3n)(3n-1)(3n-2)} = \frac{256}{27} + o(1), \qquad (38)$$

implying the required limit. $\square$

If $n \geq 1$ then in Table 18 $F(n+1)/F(n)$ is nondecreasing. We suppose that it tends to 1.

If $5 \leq n \leq 12$ then $F(n)/R(n)$ is increasing. It is easy to see that

$$\lim_{n \to \infty} \frac{F(n+1)}{F(n)} \leq \frac{R(n+1)}{R(n)}. \qquad (39)$$

The behavior of $F(n)/R(n)$ is a bit surprising since the similar relative density of tournaments score sequences tends to zero (see [21]). We suppose that $F(n)/R(n)$ also tends to zero but the convergence is slow.

References

[1] M. Anholcer, V. Babiy, S. Bozóki, W. W. Koczkodaj, A simplified implementation of the least squares solution for pairwise comparisons matrices. *CEJOR Cent. Eur. J. Oper. Res.* **19,** 4 (2011) 439–444. ⇒131

[2] S. R. Arikati, A. Maheshwari, Realizing degree sequences in parallel, *SIAM J. Discrete Math.* **9,** 2 (1996) 317–338. ⇒131

[3] Ch. M. Bang, H. Sharp, Jr., Score vectors of tournaments. *J. Combin. Theory Ser. B* **26,** 1 (1979) 81–84. ⇒131

[4] T. M. Barnes, C. D. Savage, A recurrence for counting graphical partitions, *Electron. J. Combin.* **2** (1995), Research Paper 11, 10 pages (electronic). ⇒169, 170

[5] T. M. Barnes, C. D. Savage, Efficient generation of graphical partitions, *Discrete Appl. Math.* **78,** 1–3 (1997) 17–26. ⇒169

[6] M. D. Barrus, Havel-Hakimi residues of unigraphs, *Inf. Proc. Letters* **112** (2012) 44–48. ⇒148

[7] A. Bege, Personal communication, Visegrád, 1999. ⇒133

[8] S. Bereg, H. Ito, Transforming graphs with the same degree sequence, in: *Kyoto Int. Conf. on Computational Geometry and Graph Theory,* (ed. H. Ito et al.) LNCS **4535** Springer-Verlag, Berlin, Heidelberg. 2008. pp. 25–32. ⇒169

[9] A. Berger, *Directed degree sequences*, PhD Dissertation, Martin-Luther-Universität Halle-Wittenberg, 2011. ⇒134

[10] A. Berger, A note on the characterization of digraph sequences, *arXiv*, arXiv:1112.1215v1 [math.CO] (6 December 2011) ⇒133, 134, 159

[11] A. Berger, M. Müller-Hannemann, Uniform sampling of digraphs with a fixed degree sequence, in (ed. D. M. Thilikos) *36th Int. Workshop on Graph Theoretic Concepts in Computer Science* (June 28 - 30, 2010, Zarós, Crete, Greece), LNCS **6410** (2010) 220–231. ⇒170

[12] A. Berger, M. Müller-Hannemann, Dag realizations of directed degree sequences, in (ed. O. M. Steffen, J. A. Telle) *Proc. 18th FCT* LNCS **6914** (2011) 264–275. Full version with proofs: Technical Report 2011/5 of University Halle-Wittenberg, Institute of Computer Science. ⇒134

[13] A. Berger, M. Müller-Hannemann, Dag characterizations of directed degree sequences, i Technical Report 2011/6 of University Halle-Wittenberg, Institute of Computer Science. ⇒134

[14] A. Berger, M. Műller-Hannemann, How to attack the NP-complete dag realization problems in practice.
arXiv, arXiv:1203.36v1, 2012. http://arxiv.org/abs/1203.3636 ⇒134

[15] J. K. Blitzstein, P. Diaconis, A sequential importance sampling algorithm for generating random graphs with prescribed degrees. *Internet Mathematics* **6,** 4 (2011) 489–522. ⇒170

[16] S. Bozóki, J. Fülöp, A. Poesz, On pairwise comparison matrices that can be made consistent by the modification of a few elements. *CEJOR Cent. Eur. J. Oper. Res.* **19** (2011) 157–175. ⇒131

[17] Bozóki S., J. Fülöp, L. Rónyai, On optimal completion of incomplete pairwise comparison matrices, *Math. Comput. Modelling* **52** (2010) 318–333. ⇒131

[18] A. Brauer, I. C. Gentry, K. Shaw, A new proof of a theorem by H. G. Landau on tournament matrices. *J. Comb. Theory* **5** (1968) 289–292. ⇒131

[19] A. R. Brualdi, K. Kiernan, Landau's and Rado's theorems and partial tournaments, *Electron. J. Combin.* **16,** (#N2) (2009) 6 pages. ⇒131

[20] A. R. Brualdi, J. Shen, Landau's inequalities for tournament scores and a short proof of a theorem on transitive sub-tournaments, *J. Graph Theory* **38,** 4 (2001) 244–254. ⇒131

[21] J. M. Burns: *The number of degree sequences of graphs* PhD Dissertation, MIT, 2007. ⇒169, 170, 176

[22] A. N. Busch, G. Chen, M. S. Jacobson, Transitive partitions in realizations of tournament score sequences, *J. Graph Theory* **64,** 1 (2010), 52–62. ⇒131

[23] W. Chen, On the realization of a (p,s)-digraph with prescribed degrees, *J. Franklin Institute* **281,** (5) 406–422. ⇒133

[24] S. A. Choudum, A simple proof of the Erdős-Gallai theorem on graph sequences, *Bull. Austral. Math. Soc.* **33** (1986) 67–70. ⇒134, 151

[25] V. Chungphaisan, Conditions for sequences to be r-graphic. *Discrete Math.* **7** (1974) 31–39. ⇒134

[26] J. Cooper, L. Lu, Graphs with asymptotically invariant degree sequences under restriction, *Internet Mathematics* **7,** 1 (2011) 67–80. ⇒169

[27] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* Third edition, The MIT Press/McGraw Hill, Cambridge/New York, 2009. ⇒135, 156, 166

[28] C. I. Del Genio, H. Kim, Z. Toroczkai, K. E. Bassler, Efficient and exact sampling of simple graphs with given arbitrary degree sequence, *PLoS ONE* **5,** 4 e10012 (2010). ⇒170

[29] A. Dessmark, A. Lingas, O. Garrido, On parallel complexity of maximum f-matching and the degree sequence problem. *Mathematical Foundations of Computer Science 1994* (Košice, 1994), LNCS **841,** Springer, Berlin, 1994, 316–325. ⇒131

[30] R. B. Eggleton, Graphic sequences and graphic polynomials: a report, in *Colloq. Math. Soc. J. Bolyai* **10,** North Holland, Amsterdam, 1975, 385–392. ⇒134

[31] R. B. Eggleton, D. A. Holton, Graphic sequences. *Lecture Notes in Mathematics* **10**, Springer Verlag, Berlin, 1979, 1–10. ⇒134

[32] P. Erdős, T. Gallai, Graphs with vertices having prescribed degrees (Hungarian), *Mat. Lapok* **11** (1960) 264–274. ⇒134, 151, 170

[33] P. L. Erdős, I. Miklós, Z. Toroczkai, A simple Havel-Hakimi type algorithm to realize graphical degree sequences of directed graphs. *Electronic J. Combin.* **17,** 1 R66 (2011). ⇒134, 159

[34] P. Erdős, L. B. Richmond, On graphical partitions, *Combinatorica* **13,** 1 (1993) 57–63. ⇒134

[35] L. R. Ford, D. R. Fulkerson, *Flows in Networks.* Princeton University, Press, Princeton, 1962. ⇒134

[36] A. Frank, *Connections in Combinatorial Optimization,* Oxford University Press, Oxford, 2011. ⇒130, 169

[37] A. Frank, On the orientation of graphs. *J. Combin. Theory Ser. B* **28,** 3 (1980) 251–261. ⇒169

[38] A. Frank, A. Gyárfás, How to orient the edges of a graph? In *Combinatorics. Vol. 1* (ed. A. Hajnal and V. T. Sós), North-Holland, Amsterdam-New York, 1978. pp. 353–364. ⇒169

[39] A. Frank, T. Király, Z. Király, On the orientation of graphs and hypergraphs. *Discrete Appl. Math.*, **131,** 2 (2003) 385–400. ⇒169

[40] D. A. Frank, C. D. Savage, J. A. Sellers, On the number of graphical forest partitions, *Ars Combin.* **65** (2002) 33–37. ⇒170

[41] D. R. Fulkerson, Zero-one matrices with zero trace, *Pacific J. Math* **10** (1960) 831–836. ⇒133

[42] D. Gale, A theorem on flows in networks, *Pacific J. Math.* **7** (1957) 1073–1082. ⇒133

[43] A. Garg, A. Goel, A. Tripathi, Constructive extensions of two results on graphs sequences. *Discrete Appl. Math.* **159,** 17 (2011) 2170–2174. ⇒134

[44] J. Griggs, K. B. Reid, Landau's theorem revisited, *Australas. J. Comb.* **20** (1999), 19–24. ⇒131

[45] J. L. Gross, J. Yellen, *Handbook of Graph Theory,* CRC Press, Boca Raton, 2004. ⇒130, 131

[46] B. Guiduli, A. Gyárfás, S. Thomassé, P. Weidl, 2-partition-transitive tournaments. *J. Combin. Theory Ser. B* **72**, 2 (1998) 181–196. ⇒131, 134

[47] S. L. Hakimi, On the realizability of a set of integers as degrees of the vertices of a simple graph. *J. SIAM Appl. Math.* **10** (1962) 496–506. ⇒134, 151

[48] S. L. Hakimi, On the degrees of the vertices of a graph, *F. Franklin Institute,* **279,** 4 (1965) 290–308. ⇒134

[49] H. Harborth, A. Kemnitz, Eine Anzahl der Fussballtabelle. *Math. Semester.* **29** (1982) 258–263. ⇒169, 170

[50] V. Havel, A remark on the existence of finite graphs (Czech), *Časopis Pěst. Mat.* **80** (1955) 477–480. ⇒134, 151

[51] P. Hell, D. Kirkpatrick, Linear-time certifying algorithms for near-graphical sequences. *Discrete Math.* **309,** 18 (2009) 5703–5713. ⇒134, 151

[52] R. Hemasinha, An algorithm to generate tournament score sequences, *Math. Comp. Modelling* **37,** 3–4 (2003) 377–382. ⇒169

[53] G. Isaak, Tournaments and score sequences, in ed. by D. B. West *REGS in Combinatorics*, 2010, No. 7,
http://www.math.uiuc.edu/ west/regs/fifa.html ⇒130, 170

[54] A. Iványi, Testing of football score sequences (Hungarian), in: *Abstracts of 25th Hungarian Conf. on Operation Research* (Debrecen, October 17–20, 2001), MOT, Budapest, 2001, 53–53. ⇒130, 170

[55] A. Iványi, Maximal tournaments. *Pure Math. Appl.* **13,** 1–2 (2002) 171–183. ⇒131, 133

[56] A. Iványi, Reconstruction of complete interval tournaments, *Acta Univ. Sapientiae, Inform.*, **1,** 1 (2009) 71–88. ⇒130, 131, 132, 134, 142

[57] A. Iványi, Reconstruction of complete interval tournaments. II, *Acta Univ. Sapientiae, Math.* **2,** 1 (2010) 47–71. ⇒130, 131, 134

[58] A. Iványi, Directed graphs with prescribed score sequences (ed by S. Iwata), *The 7th Hungarian-Japanese Symposium on Discrete Mathematics and Applications* (Kyoto, May 31 - June 3, 2011, ed by S. Iwata), 114–123. ⇒169

[59] A. Iványi, Deciding the validity of the score sequence of a soccer tournament, in (ed. by A. Frank): *Open problems of the Egerváry Research Group,* Budapest, 2012. http://lemon.cs.elte.hu/egres/open/ ⇒130

[60] A. Iványi, Degree sequences of multigraphs. *Annales Univ. Sci. Budapest., Sect. Comp.* **37** (2012), 195–214. ⇒148, 156

[61] A. Iványi, L. Lucz, Degree sequences of multigraphs (Hungarian), *Alkalm. Mat. Lapok* **29** (2012) (to appear). ⇒134

[62] A. Iványi, L. Lucz, T. F. Móri, P. Sótér, On the Erdős-Gallai and Havel-Hakimi algorithms. *Acta Univ. Sapientiae, Inform.* **3,** 2 (2011) 230–268. ⇒151, 167, 169, 170

[63] A. Iványi, L. Lucz, T. F. Móri, P. Sótér, The number of degree-vectors for simple graphs, in: ed. by N. J. A. Sloane, *The On-Line Encyclopedia of Integer Sequences,* 2011. http://oeis.org/A004251 ⇒169, 170

[64] A. Iványi, S. Pirzada, Comparison based ranking, in: ed. A. Iványi, *Algorithms of Informatics, Vol. 3,* AnTonCom, Budapest 2011, 1209–1258. ⇒132

[65] R. Kannan, P. Tetali, S. Vempala, Simple Markovian-chain algorithms for generating bipartite graphs and tournaments. *Random Struct. Algorithms* **14,** 4 (1999) 293–308. ⇒170

[66] K. Kayibi, M. A. Khan, S. Pirzada, A. Iványi, Random sampling of minimally cyclic digraphs with given imbalance sequence. *Acta Univ. Sapientiae, Math.* (submitted). ⇒170

[67] A. Kemnitz, S. Dolff, Score sequences of multitournaments. *Congr. Numer.* **127** (1997) 85–95. ⇒169

[68] G. Kéri, On qualitatively consistent, transitive and contradictory judgment matrices emerging from multiattribute decision procedures, *CEJOR Cent. Eur. J. Oper. Res.* **19,** 2 (2011) 215–224. ⇒131

[69] H. Kim, Z. Toroczkai, I. Miklós, P. L. Erdős, L. A. Székely, Degree-based graph construction, *J. Physics*: *Math. Theor. A* **42,** 39 (2009), 392001-1-3920001.10. ⇒134

[70] Z. Király, Recognizing graphic degree sequences and generating all realizations, Technical Report of Egerváry Research Group, TR-2011-11, Budapest. Last modification 23 April, 2012. http://www.cs.elte.hu/egres/ ⇒148, 156

[71] Z. Király, *Data Structures* (Lecture notes in Hungarian), Eötvös Loránd University, Mathematical Institute, Budapest, 2012. http://www.cs.elte.hu/~kiraly/Adatstrukturak.pdf ⇒148, 156

[72] D. J. Kleitman, K. J. Winston, Forests and score vectors, *Combinatorica* **1,** 1 (1981) 49–54. ⇒169

[73] G. Zs. Kovács, N. Pataki, Analysis of ranking sequences (Hungarian), Scientific student paper, Eötvös Loránd University, Faculty of Sciences, Budapest 2002. ⇒170

[74] M. D. LaMar, Algorithms for realizing degree sequences of directed graphs. arXiv, (2010). http://arxiv.org/abs/0906.0343. ⇒134

[75] H. G. Landau, On dominance relations and the structure of animal societies. III. The condition for a score sequence, *Bull. Math. Biophys.* **15,** (1953) 143–148. ⇒131, 132, 134

[76] F. Liljeros, C. R. Edling, L. Amaral, H. E. Stanley, Y. Aberg, The web of human sexual contacts. *Nature* **411** (2001) 907–908. ⇒131

[77] L. Lucz, *Analysis of degree sequences of graphs* (Hungarian), MSc Thesis, Eötvös Loránd University, Faculty of Informatics, Budapest, 2012. http://people.inf.elte.hu/lulsaai/diploma ⇒149

[78] L. Lucz, A. Iványi, Testing and enumeration of football sequences, in: *MaCS'12. 9th Joint Conference in Mathematics and Computer Science* (Siófok, Hungary, February 9–12, 2012, ed. by Z. Csörnyei), ELTE IK, Budapest, 2012, 63–63. ⇒169

[79] B. D. McKay, X. Wang, Asymptotic enumeration of tournaments with a given score sequence. *J. Comb. Theory A*, **73,** 1 (1996) 77–90. ⇒169

[80] D. Meierling, L. Volkmann, A remark on degree sequences of multigraphs. *Math. Methods Oper. Res.* **69,** 2 (2009) 369–374. ⇒151

[81] N. Metropolis, P. R. Stein, The enumeration of graphical partitions, *European J. Comb.* **1,** 2 (1980) 139–153. ⇒169

[82] I. Miklós, P. L. Erdős, L. Soukup, Towards random uniform sampling of bipartite graphs with given degree sequence, *arXiv* 1004.2612v3 [math.CO] (14 Sep 2010), http://arxiv.org/pdf/1004.2612v3.pdf ⇒170

[83] J. W. Miller, Reduced criterion for degree sequences, *arXiv*, arXiv:1205.2686v1 [math.CO] 11 May 2012, 18 pages. ⇒134

[84] J. W. Moon, On the score sequence of an n-partite tournament. *Can. Math. Bull.* **5** (1962) 51–58. ⇒134

[85] J. W. Moon, An extension of Landau's theorem on tournaments, *Pacific J. Math.* **13** (1963), 1343–1345. ⇒132, 134

[86] J. W. Moon, *Topics on Tournaments*. Holt, Rinehart and Winston. New York, 1968. ⇒132

[87] T. V. Narayana, D. H. Bent, Computation of the number of score sequences in round-robin tournaments, *Canad. Math. Bull.* **7,** 1 (1964) 133–136. ⇒169

[88] M. Newman, A. L. Barabási, D. J. Watts, *The Structure and Dynamics of Networks.* Princeton University Press, (2006). ⇒131

[89] S. Özkan, Generalization of the Erdős-Gallai inequality. *Ars Combin.* **98** (2011) 295–302. ⇒151

[90] D. Pálvölgyi, Deciding soccer scores and partial orientations of graphs. *Acta Univ. Sapientiae, Math.* **1,** 1 (2009) 35–42. ⇒134

[91] A. N. Patrinos, S. L. Hakimi, Relations between graphs and integer-pair sequences. *Discrete Math.* **15** 4 (1976) 347–358 ⇒134

[92] G. Pécsy, L. Szűcs, Parallel verification and enumeration of tournaments, *Stud. Univ. Babeş-Bolyai, Inform.* **45,** 2 (2000) 11–26. ⇒131, 169

[93] S. Pirzada, *An Introduction to Graph Theory.* Orient BlackSwan, Hyderabad, 2012. ⇒130, 131

[94] S. Pirzada, G. Zhou, A. Iványi, Score lists of multipartite hypertournaments, *Acta Univ. Sapientiae, Inform.* **2,** 2 (2011) 184–193. ⇒134

[95] K. B. Reid, Tournaments: Scores, kings, generalizations and special topics, *Congr. Numer.* **115** (1996) 171–211. ⇒131

[96] K. B. Reid, C. Q. Zhang, Score sequences of semicomplete digraphs, *Bull. Inst. Combin. Appl.* **24** (1998) 27–32. ⇒133

[97] Ø. J. Rødseth, J. A. Sellers, H. Tverberg, Enumeration of the degree sequences of non-separable graphs and connected graphs. *European J. Comb.* **30,** 5 (2009) 1309–1317. ⇒169

[98] F. Ruskey, F. R. Cohen, P. Eades, A. Scott, Alley CATs in search of good homes. *Congr. Numer.* **102** (1994) 97–110. ⇒169

[99] H. J. Ryser, Combinatorial properties of matrices of zeroas and ones, *Canad. J. Math.* **9** (1957) 371–377. ⇒133

[100] J. E. Schoenfield, The number of football score sequences, in: ed. by N. J. A. Sloane, *The On-Line Encyclopedia of Integer Sequences,* 2012. http://oeis.org/A064626 ⇒130, 169, 170, 173

[101] G. Sierksma, H. Hoogeveen, Seven criteria for integer sequences being graphic, *J. Graph Theory* **15,** 2 (1991) 223–231. ⇒131

[102] B. Siklósi, *Comparison of sequential and parallel algorithms solving sport problems.* Master thesis. Eötvös Loránd University, Faculty of Sciences, Budapest, 2001. ⇒131

[103] N. J. A. Sloane, The number of degree-vectors for simple graphs. In (ed. N. J. A. Sloane): *The On-Line Encyclopedia of the Integer Sequences.* 2011. http://oeis.org/A004251 ⇒170

[104] D. Soroker, *Optimal parallel construction of prescribed tournaments, Discrete Appl. Math.* **29,** 1 (1990) 113–125. ⇒131

[105] R. P. Stanley, A zonotope associated with graphical degree sequence, in: *Applied Geometry and Discrete Mathematics, Festschr. 65th Birthday Victor Klee.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science. **4** (1991) 555–570. ⇒169

[106] L. A. Székely, L. H. Clark, R. C. Entringer. An inequality for degree sequences. *Discrete Math.* **103,** 3 (1992) 293–300. ⇒131

[107] M. Takahashi, *Optimization Methods for Graphical Degree Sequence Problems and their Extensions*, PhD thesis, Graduate School of Information, Production and systems, Waseda University, Tokyo, 2007. http://hdl.handle.net/2065/28387 ⇒132, 151

[108] J. Temesi, Pairwise comparison matrices and the error-free property of the decision maker, *CEJOR Cent. Eur. J. Oper. Res.* **19,** 2 (2011) 239–249. ⇒131

[109] P. Tetali, A characterization of unique tournaments. *J. Combin. Theory Ser. B* **72,** 1 (1998) 157–159. ⇒148

[110] A. Tripathi, H. Tyagy, A simple criterion on degree sequences of graphs. *Discrete Appl. Math.* **156,** 18 (2008) 3513–3517. ⇒131, 151

[111] A. Tripathi, S. Vijay, A note on a theorem of Erdős & Gallai. *Discrete Math.* **265,** 1–3 (2003) 417–420. ⇒151

[112] A. Tripathi, S. Venugopalan, D. B. West, A short constructive proof of the Erdős-Gallai characterization of graphic lists. *Discrete Math.* **310,** 4 (2010) 833–834. ⇒151

[113] R. Tyskevich, Decomposition of graphical sequences and unigraphs, *Discrete Math.* **220,** 1–3 (2000) 201–238. ⇒148

[114] P. van Emde Boas, Preserving order in a forest in less than logarithmic time, *Proc. 16th Annual Symp. Found Comp. Sci.* **10** (1975) 75–84. ⇒148, 156

[115] E. W. Weisstein, *Degree Sequence,* From MathWorld—Wolfram Web Resource, 2012. ⇒134

[116] E. W. Weisstein, *Graphic Sequence,* From MathWorld—Wolfram Web Resource, 2012. ⇒134

[117] K. J. Winston, D. J. Kleitman, On the asymptotic number of tournament score sequences. *J. Combin. Theory Ser. A.* **35** (1983) 208–230. ⇒169

[118] I. E. Zverovich, V. E. Zverovich, Contribution to the theory of graphic sequences, *Discrete Math.* **105** (1992) 293–303. ⇒134