Séminaire Lotharingien de Combinatoire **47** (2002), Article B47h

Computing Tournament Sequence Numbers efficiently with Matrix Techniques

Erich Neuwirth

Department of Statistics and Decision Support Systems
University of Vienna
erich.neuwirth@univie.ac.at

Abstract

We give a new, "almost explicit" formula for tournament numbers, representing them as upper left elements of the n^{th} power of a matrix with an explicit formula for elements of the original matrix. Using this representation, we show how to compute tournament numbers in time complexity $O(n^6)$.

1 Main results

Tournament sequences are defined as sequences $(t_1, t_2, ...)$ with $t_1 = 1$ and $t_i < t_{i+1} \le 2t_i$. Defining T_k-sequences as sequences with $(t_1, t_2, ...)$ with $t_1 = k$ and $t_i < t_{i+1} \le 2t_i$ we see that tournament sequences are T_1-sequences. Now let $T(n,k)$ denote the number of T_k-sequences of length n. It is obvious that $T(1,k) = 1$ for all k. To compute $T(n,k)$ for $n > 1$ we note that T_k-sequences of length n are produced by taking $t_1 = k$, and appending all T_i-sequences of length $n-1$ which are "admissible" extensions of a sequence starting with k. It is obvious that $k+1 \le i \le 2k$ are the only possible choices for the first element of the extension sequence. Therefore, we have the recursion

$$T(n,k) = \sum_{j=k+1}^{2k} T(n-1,j). \tag{1}$$

We call T the tournament matrix.

The tournament numbers we want to compute are $T(n) = T(n,1)$, and $T(n)$ is the number of tournament sequences of length n.
The values for $T(n)$ for $n = 1, 2, \ldots 16$ are

1, 1, 2, 7, 41, 397, 6377, 171886, 7892642, 627340987, 87635138366, 21808110976027, 9780286524758582, 7981750158298108606, 11950197013167283686587, 33046443615914736611839942.

Capelli and Narayana introduced tournament sequences in [2]. Further results about $T(n)$ can be found in Sloane's On-Line Encyclopedia of Integer Sequences [7] (sequence A008934). Tournament numbers have been studied by Torelli (see [9]) in connection with the Goldbach conjecture. Kleber and Cook studied algorithmic aspects of computing $T(n)$ extensively in [5]. They showed that $T(n)$ can be computed in time complexity $O(n^6)$ by transforming the original recursive equations, which by themselves would lead to exponential time complexity.

In this paper, we present an "almost explicit" formula for computing $T(n)$, and we show that this formula computes $T(n)$ with time complexity $O(n^6)$, similar to the Kleber-Cook results. We then discuss why for practical purposes our method works noticeably faster than their method.

Here are our two main results.

Theorem 1 *Let $n \geq 2$. Then $T(n)$ is the upper left element of the matrix power C_{n-1}^{n-1} where C_m is the $m \times m$-matrix with elements*

$$\begin{aligned} C(i,j) &= 2^{2i-j-2}\left(\binom{i-1}{j-i+1} + 4\binom{i-1}{j-i} + 4\binom{i-1}{j-i-1}\right), \\ &\qquad \text{for } i \neq j \text{ and } i-1 \neq j; \\ C(i+1,i) &= 2^i - 1; \\ C(i,i) &= 2^{i-2}(i+3) - 1; \end{aligned} \tag{2}$$

for $1 \leq i,\ j \leq m$.

Here are the first few lines and columns of this matrix:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & \dots \\ 1 & 4 & 4 & 1 & 0 & 0 & 0 & \dots \\ 0 & 3 & 11 & 13 & 6 & 1 & 0 & \dots \\ 0 & 0 & 7 & 27 & 38 & 25 & 8 & \dots \\ 0 & 0 & 0 & 15 & 63 & 104 & 88 & \dots \\ 0 & 0 & 0 & 0 & 31 & 143 & 272 & \dots \\ 0 & 0 & 0 & 0 & 0 & 63 & 319 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Theorem 2 *Using the representation of Theorem 1, $T(n)$ can be computed in time complexity $O(n^6)$.*

2 Matrix preliminaries

The key for applying matrix techniques to our problem is the observation that Equation (1) can be interpreted in the following way:
Let $\mathbf{U}$ be the matrix with elements $T(i,j)$, $i \geq 1$, $j \geq 1$. Then each row of $\mathbf{U}$ can be obtained by multiplying the previous row with the matrix W from the right, where W has the elements $W(i,j) = 1$ for $j < i \leq 2j$ and $W(i,j) = 0$ everywhere else. Writing the matrices $\mathbf{U}$ and W elementwise, we have

$$\begin{pmatrix} T(2,1) & T(2,2) & T(2,3) & T(2,4) & T(2,5) & \dots \\ T(3,1) & T(3,2) & T(3,3) & T(3,4) & T(3,5) & \dots \\ T(4,1) & T(4,2) & T(4,3) & T(4,4) & T(4,5) & \dots \\ T(5,1) & T(5,2) & T(5,3) & T(5,4) & T(5,5) & \dots \\ T(6,1) & T(6,2) & T(6,3) & T(6,4) & T(6,5) & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} = \begin{pmatrix} T(1,1) & T(1,2) & T(1,3) & T(1,4) & T(1,5) & \dots \\ T(2,1) & T(2,2) & T(2,3) & T(2,4) & T(2,5) & \dots \\ T(3,1) & T(3,2) & T(3,3) & T(3,4) & T(3,5) & \dots \\ T(4,1) & T(4,2) & T(4,3) & T(4,4) & T(4,5) & \dots \\ T(5,1) & T(5,2) & T(5,3) & T(5,4) & T(5,5) & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \dots \\ 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & 1 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$\mathbf{U}$ and W are infinite matrices, and for infinite matrices multiplication is not always well defined. Matrix multiplication for infinite matrices is, however, well defined, if either all the rows of the left factor have only finitely many nonzero elements, or if all the columns of the right factor have only finitely many nonzero elements. We call a matrix with rows with only finitely many nonzero elements a *matrix with finite rows*, and we call a matrix with columns with only finitely many nonzero elements a *matrix with finite columns.*

In our case, matrix W has finite rows and finite columns, and therefore rewriting recursion (1) as matrix equation is possible. Furthermore, any infinite matrix can be multiplied with W from the left and from the right.

In the rest of this paper, we will use matrix techniques for operations on recursively defined functions. These techniques have been pioneered by Kemeny in [4] and extensively used by Neuwirth in [6].

For our proof, we will need some "helper matrices". We will use infinite matrices.
All our matrices start with row 1 and column 1.
$\delta_{i,j}$ is the standard Kronecker symbol, defined by $\delta_{i,i} = 1$ and $\delta_{i,j} = 0$ for$i \neq j$.
$\mathcal{I}$ defined by $\mathcal{I}(n,k) = \delta_{n,k}$ is the identity matrix.
$\mathcal{S}$ defined by $\mathcal{S}(n,k) = \delta_{n+1,k}$ is a matrix with 1s just above the main diagonal and 0s everywhere else.

$\mathcal{T}$ defined by $\mathcal{T}(n,k) = \delta_{n,k+1}$ is a matrix with 1s just below the main diagonal and 0s everywhere else.
$\mathcal{O}_n$ defined by $\mathcal{O}_n(m,k) = \delta_{m,k}$ for $m \leq n$ and $\mathcal{O}_n(m,k) = 0$ for $m > n$ is a "cut off" identity matrix.
$\mathcal{J}$ defined by $\mathcal{J}(n,k) = 1$ for $n \geq k$ and $\mathcal{J}(n,k) = 0$ for $n < k$ is a lower triangular matrix with 1s everywhere on and below the main diagonal.
$\mathcal{B}$ defined by $\mathcal{B}(n,k) = \binom{n-1}{k-1}$ is the triangular matrix consisting of the binomial coefficients.
We note that with the exception of $\mathcal{S}$ all these matrices are lower triangular.
M' denotes the transpose of M for any matrix M.
We will need the following identities for these matrices:

$$\mathcal{S}\mathcal{T} = \mathcal{I} \tag{3}$$

$$(\mathcal{I} - \mathcal{T})^{-1} = \mathcal{J} \tag{4}$$

For a (lower) triangular matrix D we have

$$\mathcal{O}_n D = \mathcal{O}_n D \mathcal{O}_n \tag{5}$$

Furthermore we also have

$$\mathcal{O}_n \mathcal{O}_m = \mathcal{O}_{\min(m,n)} \tag{6}$$

$$\mathcal{O}_n \mathcal{S}^k = \mathcal{S}^k \mathcal{O}_{n+k} \tag{7}$$

and

$$\mathcal{J}\mathcal{B} = (\mathcal{I} - \mathcal{T})^{-1}\mathcal{B} = \mathcal{S}\mathcal{B}\mathcal{T} \tag{8}$$

The last equation is just "the matrix way" of rewriting $\sum_{i=0}^{n} \binom{i}{k} = \binom{n+1}{k+1}$.

$\overline{\mathcal{B}} = \mathcal{B}^{-1}$, the inverse of $\mathcal{B}$, has the elements $\overline{\mathcal{B}}(n,k) = \binom{n-1}{k-1}(-1)^{n+k}$, so it may be described as "the binomials with a checkerboard sign pattern".

For a triangular matrix D the matrix $\mathcal{O}_n D \mathcal{O}_n$ is the "left upper $n \times n$-block" of the matrix. To compute the upper left block of the power of a triangular matrix, we only need the block of the same size of the original matrix. Translated into our matrix techniques, this becomes the following lemma:

Lemma 3 *Let D be triangular. Then* $\mathcal{O}_n D^k \mathcal{O}_n = (\mathcal{O}_n D \mathcal{O}_n)^k$.

PROOF. This is an immediate consequence of Equation (5). □

The situation becomes more complicated when we allow matrices with some diagonals above the main diagonal having nonzero elements. We will study a special class of matrices.

Definition 4 *A matrix M is called d-supertriangular if $M(n,k) = 0$ for $k > n + d$.*

Lemma 5 *M is d-supertriangular iff there exists a triangular matrix D with $M = S^d D$.*

PROOF. Trivial. $\square$

Lemma 6 *For a d-supertriangular matrix M we have $\mathcal{O}_n M = \mathcal{O}_n M \mathcal{O}_{n+d}$.*

PROOF. According to Lemma 5 we have $M = \mathcal{S}^d D$ with a triangular matrix D.
Therefore, using Equation (7) we have $\mathcal{O}_n M = \mathcal{O}_n S^d D = S^d \mathcal{O}_{n+d} D$.
Using Equation (5) we have $S^d \mathcal{O}_{n+d} D = S^d \mathcal{O}_{n+d} D \mathcal{O}_{n+d}$.
Using Equation (7) again we have $S^d \mathcal{O}_{n+d} D \mathcal{O}_{n+d} = \mathcal{O}_n \mathcal{S}^d D \mathcal{O}_{n+d} = \mathcal{O}_n M \mathcal{O}_{n+d}$.
$\square$

Lemma 7 *For a d-supertriangular matrix M we have*
$\mathcal{O}_n M^k \mathcal{O}_n = \mathcal{O}_n (\mathcal{O}_{n+(k-1)d} M \mathcal{O}_{n+(k-1)d})^k \mathcal{O}_n$.

PROOF. Using Lemma 6 we have
$\mathcal{O}_n M^k \mathcal{O}_n = \mathcal{O}_n M M^{k-1} \mathcal{O}_n = \mathcal{O}_n M \mathcal{O}_{n+d} M^{k-1} \mathcal{O}_n$.
Applying Lemma 6 repeatedly from left to right following this pattern we have
$\mathcal{O}_n M^k \mathcal{O}_n = \mathcal{O}_n M \mathcal{O}_{n+d} M \mathcal{O}_{n+2d} M \mathcal{O}_{n+3d} \cdots \mathcal{O}_{n+(k-1)d} M \mathcal{O}_{n+kd} \mathcal{O}_n$.
Replacing the terms $\mathcal{O}_{n+id}$ by $\mathcal{O}_{n+(k-1)d} \mathcal{O}_{n+id} \mathcal{O}_{n+(k-1)d}$ for $i \leq k-1$ and replacing the last term $\mathcal{O}_n$ by $\mathcal{O}_{n+(k-1)d)} \mathcal{O}_n$ we have

$$\begin{aligned}\mathcal{O}_n M^k \mathcal{O}_n = \mathcal{O}_n &\mathcal{O}_{n+(k-1)d} M \mathcal{O}_{n+(k-1)d} \mathcal{O}_{n+d}\\ &\cdot \mathcal{O}_{n+(k-1)d} M \mathcal{O}_{n+(k-1)d} \mathcal{O}_{n+2d}\\ &\cdot \mathcal{O}_{n+(k-1)d} M \mathcal{O}_{n+(k-1)d} \mathcal{O}_{n+3d} \cdots\\ &\cdot \mathcal{O}_{n+(k-1)d} M \mathcal{O}_{n+kd} \mathcal{O}_{n+(k-1)d} \mathcal{O}_n\end{aligned}$$

Since $\overline{M} = \mathcal{O}_{n+(k-1)d} M \mathcal{O}_{n+(k-1)d}$ is d-supertriangular and $\mathcal{O}_{n+kd} \mathcal{O}_{n+(k-1)d} = \mathcal{O}_{n+(k-1)d}$ we have
$\mathcal{O}_n M^k \mathcal{O}_n = \mathcal{O}_n \overline{M} \mathcal{O}_{n+d} \overline{M} \mathcal{O}_{n+2d} \overline{M} \mathcal{O}_{n+3d} \cdots \overline{M} \mathcal{O}_n$.
Applying Lemma 6 repeatedly (from right to left) we have
$\mathcal{O}_n M^k \mathcal{O}_n = \mathcal{O}_n M_d \mathcal{O}_{n+d} \overline{M} \mathcal{O}_{n+2d} \overline{M} \mathcal{O}_{n+3d} \cdots \overline{M} \mathcal{O}_n = \mathcal{O}_n \overline{M}^k \mathcal{O}_n$
which finishes the proof. $\square$

We also need the following technical lemma, translating certain recursion equations into matrix equations and deriving a recursion for a matrix product.

Lemma 8 *Let $F(n,k)$ be defined by*

$$\begin{aligned} F(1,k) &= \delta_{1,k}; \\ F(n,1) &= \alpha_0 F(n-1,1), && \text{for } n \geq 2; \\ F(n,k) &= \alpha_0 F(n-1,k) + \alpha_1 F(n-1,k-1), && \text{for } n \geq 2 \text{ and } k \geq 2. \end{aligned} \tag{9}$$

Let furthermore $G(n,k)$ for all $n \geq 2$ be defined by

$$\begin{aligned} G(n,1) &= \beta_0 G(n-1,1); \\ G(n,2) &= \beta_0 G(n-1,2) + \beta_1 G(n-1,1); \\ G(n,k) &= \beta_0 G(n-1,k) + \beta_1 G(n-1,k-1) \\ &\quad + \beta_2 G(n-1,k-2), && \text{for } k \geq 3. \end{aligned} \tag{10}$$

Then $H = FG$ satisfies the equations

$$\begin{aligned} H(1,k) &= G(1,k), && \text{for all } k \geq 1; \\ H(n,1) &= (\alpha_0 + \alpha_1\beta_0) H(n-1,1), && \text{for } n \geq 2; \\ H(n,2) &= (\alpha_0 + \alpha_1\beta_0) H(n-1,2) + \\ &\quad \alpha_1\beta_1 H(n-1,1), && \text{for } n \geq 2; \\ H(n,k) &= (\alpha_0 + \alpha_1\beta_0) H(n-1,k) + \\ &\quad \alpha_1\beta_1 H(n-1,k-1) + \\ &\quad \alpha_1\beta_2 H(n-1,k-2), && \text{for } n \geq 2 \text{ and } k \geq 3. \end{aligned} \tag{11}$$

PROOF. The recursion for F can be rewritten as $\mathcal{S}F = \alpha_0 F + \alpha_1 F\mathcal{S}$. Defining the matrix $B = \beta_0\mathcal{I} + \beta_1\mathcal{S} + \beta_2\mathcal{S}^2$ the recursion for G can be written as $\mathcal{S}G = GB$. Since F has the $(1,0,0,\ldots)$ as its first row, we have $H(1,k) = G(1,k)$.

Using the matrix representation, we have

$$\begin{aligned} \mathcal{S}FG &= (\alpha_0 F + \alpha_1 F\mathcal{S})G = \alpha_0 FG + \alpha_1 FGB \\ &= \alpha_0 FG + \alpha_1\beta_0 FG + \alpha_1\beta_1 FG\mathcal{S} + \alpha_1\beta_2 FG\mathcal{S}^2 \end{aligned}$$

and this is the matrix form of Equation (11). □

We need the following row sum inequality for recursively defined functions.

Lemma 9 *Let $F(n,k)$ be defined for $n \geq 1$ and $k \geq 1$ such that only finitely many of the $F(1,k)$ are nonzero and $F(n,k) = \sum_{j=0}^{\min(d,k-1)} \alpha_j F(n-1,k-j)$. Then we have $\sum_j F(n,j) \leq (\sum_j |F(1,j)|)(\sum_{l=0}^{d} |\alpha_l|)^{n-1}$.*

PROOF. The assertion is trivial for $n = 1$.
For $n > 1$ we have $\sum_j F(n,j) \leq \sum_j |F(n,j)| \leq \sum_j \sum_{l=0}^{d} |F(n-1,j-l)||\alpha_l|$.
□

Remark 10 Later, we will also use the following fact: Let the recursively defined matrices $F_i(n,k)$ (defined for $1 \leq i \leq N$) all fulfill the same recursion, $F_i(n,k) = \sum_{j=0}^{\min(p,k-1)} \alpha_j F_i(n-1,k-j)$.
Then the linear combination $F(n,k) = \sum_{i=1}^{N} \gamma_i F_i(n,k)$ also fulfills the recursion $F(n,k) = \sum_{j=0}^{\min(p,k-1)} \alpha_j F(n-1,k-j)$. We will use this fact with "shifted" functions, defined for $i > 0$ by $F_i(n,k) = F_0(n,k-i)$ for $k \geq i$ and $F_i(n,k) = 0$ for $k < i$.

Remark 11 We will also make use of the following well known fact:
Let $p_d(x)$ be a polynomial of degree d. Then for arbitrary integers k and $n > d$ we have $\sum_{i=0}^{n} p_d(k+i)\binom{n}{i}(-1)^i = 0$.
This is another way of expressing the fact that a polynomial of degree d has vanishing finite differences of any order higher than d.
In matrix notation, we can rewrite this fact in the following way: Let X be a matrix with rows generated by polynomials, i.e., $X(n,k) = p_{n-1}(k)$, with p_n being polynomials of degree n. Then $X\mathcal{B}'^{-1} = X(\mathcal{B}^{-1})'$ is a (lower) triangular matrix.

3 The proof

We start with a few more or less technical lemmata, which finally will result in a proof of Theorems 1 and 2. The main technique used is the translation of recursion equations into matrix equations.

Lemma 12 *Let the infinite matrix W be defined by*
$W(n,k) = 1$ for $k+1 \leq n \leq 2k$ and $W(n,k) = 0$ everywhere else. Then the tournament matrix $\mathbf{U}$ satisfies the equation

$$\mathcal{S}\mathbf{U} = \mathbf{U}W. \qquad (12)$$

PROOF. Equation (12) is just a matrix version of Equation (1). □

Lemma 13 $\mathbf{V} = \mathbf{U}\mathcal{B}'^{-1}$ *is a lower triangular matrix, and the first columns of* $\mathbf{V}$ *and* $\mathbf{U}$ *are equal.*

PROOF. As Cook and Kleber have shown in [5], $T(n,k)$ can be written as $T(n,k) = p_{n-1}(k)$ with p_n being polynomials of degree n. Using Remark 11 proves the first part of our lemma. Since the first column of $\mathcal{B}'^{-1}$ is $(1,0,0,0,\ldots)'$, the second assertion of the lemma is trivial. □

Lemma 14 $\mathbf{V}$ *defined in Lemma 13 satisfies the equation* $\mathcal{S}\mathbf{V} = \mathbf{V}D$ *with* $D = \mathcal{B}'W\mathcal{B}'^{-1}$*, and* D *is 1-supertriangular.*

PROOF. Since we have $\mathcal{S}\mathbf{U} = \mathbf{U}W$ we also have $\mathcal{S}\mathbf{U}\mathcal{B}'^{-1} = \mathbf{U}W\mathcal{B}'^{-1}$.
By Lemma 13 $\mathbf{V} = \mathbf{U}\mathcal{B}'^{-1}$ is lower triangular. Therefore, multiplying it with the upper triangular matrix $\mathcal{B}'$ from the right is a well defined operation and we have $\mathbf{V}\mathcal{B}' = \mathbf{U}$.
Now we want to apply the associative law, which in general is not applicable for multiplication of infinite matrices. Nevertheless, the associative law can be applied if all the matrix products involved are well defined. Therefore we have $\mathcal{S}\mathbf{V} = \mathcal{S}\mathbf{U}\mathcal{B}'^{-1} = \mathbf{U}W\mathcal{B}'^{-1} = (\mathbf{U}(\mathcal{B}'^{-1}\mathcal{B}'))(W\mathcal{B}'^{-1}) = ((\mathbf{U}\mathcal{B}'^{-1})\mathcal{B}')(W\mathcal{B}'^{-1}) = (\mathbf{U}\mathcal{B}'^{-1})(\mathcal{B}'(W\mathcal{B}'^{-1})) = \mathbf{V}D$. □

Lemma 15 *For* D *defined in Lemma 14 we have* $D = C'$ *with* C *defined by Equations 2.*

PROOF. Using matrix notation we note that W can be written as
$W = \mathcal{J}'(\mathcal{I}_2 - \mathcal{I})$ with $\mathcal{I}_2(n,k) = \delta_{n,2k}$. Therefore we have
$C = D' = (\mathcal{B}'W\mathcal{B}'^{-1})' = (\mathcal{B}'\mathcal{J}'(\mathcal{I}_2 - \mathcal{I})\mathcal{B}'^{-1})' = \mathcal{B}^{-1}(\mathcal{I}_2' - \mathcal{I})\mathcal{J}\mathcal{B}$.
Using Remark 11 we have
$C = \mathcal{B}^{-1}(\mathcal{I}_2' - \mathcal{I})\mathcal{S}\mathcal{B}\mathcal{T} = (\mathcal{B}^{-1}\mathcal{I}_2'\mathcal{S}\mathcal{B} - \mathcal{B}^{-1}\mathcal{S}\mathcal{B})\mathcal{T}$.
Now we use Lemma 8 to simplify $\mathcal{B}^{-1}\mathcal{S}\mathcal{B}$ and $\mathcal{B}^{-1}\mathcal{I}_2'\mathcal{S}\mathcal{B}$.
Setting $\alpha_0 = -1$, $\alpha_1 = 1$, $\beta_0 = 1$, $\beta_1 = 1$, $G(1,1) = 1$ and $G(1,2) = 1$ allows us to compute the recursion equation for $H_1 = \mathcal{B}^{-1}(\mathcal{S}\mathcal{B})$. The resulting array H_1 in this case has $H_1(1,1) = 1$, $H_1(1,2) = 1$, $H_1(1,k) = 0$ for $k \geq 2$ and fulfills the simple recursion $H_1(n,k) = H_1(n-1,k-1)$. Therefore, we have $H_1(n,k) = \delta_{n,k} + \delta_{n+1,k}$.
To get the recursion equation for $\mathcal{B}^{-1}\mathcal{I}_2'\mathcal{S}\mathcal{B}$ from Lemma 8 we note that $\mathcal{I}_2'\mathcal{S}\mathcal{B}$ is the matrix with rows $3,5,7,...$ from the binomial triangle $\mathcal{B}$. Therefore, to apply Lemma 8, we set $G(1,k) = \binom{2}{k-1}$, $\beta_0 = 1$, $\beta_1 = 2$ and $\beta_2 = 1$. For F we use $\alpha_0 = -1$ and $\alpha_1 = 1$. Then $H_2 = FG$ for $n \geq 2$ and $k \geq 3$ fulfills the recursion

$$H_2(n,k) = 2H_2(n-1,k-1) + H_2(n-1,k-2) \tag{13}$$

Additionally, we have $H_2(1,1)=1$, $H_2(1,2)=2$, $H_2(1,3)=1$, and $H_2(1,k)=0$ for $k\geq 4$. It is easy to see that $H_3(n,k)=2^{2n-k-1}\binom{n-1}{k-n}$ satisfies Equation (13) with H_2 replaced by H_3 and therefore (using Remark 10) any linear combination $H(n,k)=\sum_{i=0}^{d}\gamma_i H_3(n,k-n-i)$ satisfies Equation (13) with H_2 replaced by H.
Since we have $H_3(1,k)=\delta_{1,k}$, H_4 defined by
$H_4(n,k)=H_3(n,k)+2H_3(n,k-1)+H_3(n,k-2)$ satisfies $H_4(1,k)=G(1,k)$, the boundary conditions for H_2. Since H_4 also fulfills the (adapted) recursion (13), we have $H_2=H_4$.
Rewriting $H_2=H_4$ we have

$$\begin{aligned}H_2(n,k) &= H_3(n,k)+2H_3(n,k-1)+H_3(n,k-1)\\ &= 2^{2n-k-1}\binom{n-1}{k-n}+2.2^{2n-k}\binom{n-1}{k-n-1}+2^{2n-k+1}\binom{n-1}{k-n-2}\\ &= 2^{2n-k-1}\left(\binom{n-1}{k-n}+4\binom{n-1}{k-n-1}+4\binom{n-1}{k-n-2}\right)\end{aligned}$$

Combining our formulas for $H_2\mathcal{B}^{-1}\mathcal{I}_2'\mathcal{S}\mathcal{B}$ and $H_1\mathcal{B}^{-1}\mathcal{S}\mathcal{B}$ we have $C=(H_2-H_1)\mathcal{T}$. Using the fact that multiplying any matrix with $\mathcal{T}$ from the right is the same as "cutting off" the first column of this matrix, and combining $C=(H_2-H_1)\mathcal{T}$ with the elementwise representations for H_1 and H_2 this gives Equations (2) for C. $\square$

Now we are finally ready to prove Theorem 1.

Proof of Theorem 1
According to Lemma 13 the $T(n)$ are the first column of $\mathbf{V}$. So $T(n)$ is the upper left element of $\mathcal{S}^{n-1}\mathbf{V}$. Since $\mathcal{S}\mathbf{V}=\mathbf{V}C'$ we have $S^{n-1}\mathbf{V}=\mathbf{V}(C')^{n-1}$. Since $\mathbf{V}$ is lower triangular and $\mathbf{V}(1,1)=1$, the upper left element of $\mathbf{V}(C')^{n-1}$ is also the upper left element of $(C')^{n-1}$, and, by transposing, the upper left element of C^{n-1}. Applying Lemma 7 with $n=1$, $d=1$, and $k=n-1$ finishes the proof of Theorem 1.

Proof of Theorem 2
We will discuss computing $T(n+1)$ instead of computing $T(n)$ because then the equations become somewhat simpler.
We note that to compute $T(n+1)$ it is not necessary to compute the complete matrix C_n^n. It is sufficient to compute the first column of C_n^n. Defining a vector sequence v_i by v_0=e_1 (the vector $(1,0,0,0,\ldots)'$ of length n) and $v_{i+1}=C_n v_i$ we see that v_n is the first column of C_n^n and $T(n+1)$ is the first element of v_n.

The following algorithm computes v_k:

(1) For $1\leq i\leq n$ let $v_0(i)=\delta_{1,i}$
(2) For $j=1,2,\ldots n$ let $v_j(i)=\sum_{l=1}^{n}C(i,l)v_{j-1}(l)$

Each single execution of step (2) involves n^2 multiplications and $n(n-1)$ additions, and this step is repeated n times. So the total number of multiplications and additions for all repeated executions of step (2) is $O(n^3)$.
To be able to perform these steps we have to calculate the n^2 elements of matrix C. C essentially contains sums of binomial coeffients $\binom{N}{k}$ with $N \leq 2n$ multiplied by powers of 2, and therefore can be calculated in time n^3.

Using the fact that $C = (H_2 - H_1)\mathcal{T}$ we see that $C(i,j) \leq H_2(i, j+1)$ for all i and j.
Applying Lemma 9 and the recursion 13 to H_2 gives
$\sum_j H_2(n,j) \leq 8 \cdot 3^{n-1}$ and therefore trivially $\max_{i,j \leq n} H_2(i,j) \leq 8 \cdot 3^{n-1}$.
Therefore we also have $\sum_j C(n,j) \leq 8 \cdot 3^{n-1}$ and $\max_{i,j \leq n} C(i,j) \leq 8 \cdot 3^{n-1}$.
Therefore, by induction we have $v_k(i) \leq 8^{k-1} 3^{(n-1)(k-1)}$ and as a consequence $v_k(i) \leq 8^{n-1} 3^{(n-1)(n-1)} \leq 3^{n^2}$. So all the multiplications in step (2) have one factor bounded by $8 \cdot 3^{n-1} \leq 3^{n+1}$ and the other factor bounded by 3^{n^2}, and for all additions the two summands are bounded by 3^{n^2}.

To give an estimate for the upper bound for the time needed to compute $T(n)$, we will use naive bit complexity (as introduced by Bach and Shallit in [1]), i.e., we assume that the time needed for addition of large numbers x and y is bounded by $O(\max(\log(x), \log(y)))$, i.e., the number of digits of x and y, and that the time needed for multiplication of large numbers x and y is bounded by $O(\log(x)\log(y))$.
So, to compute the n^{th} tournament number, we need $O(n^3)$ operations with naive bit complexity bounded by $O(n^3)$ for each of these operations, and therefore naive bit complexity for computation is $O(n^6)$. □

Remark 16 The derivation of our complexity bound uses naive bit complexity, assuming $O(\log(x)\log(y))$ as the bound for the multiplication of two numbers x and y. Faster methods for multiplying large numbers are known. Using Karatsuba's multiplication algorithm (see [3]) we get a bound of $O(n^{4+\log 3})$, or approximately $O(n^{5.6})$ for complexity, and using Strassen's method (see [8]) we even could get $O(n^5 \log n \log\log n)$.

4 Practical considerations

For practical computation purposes it is convenient to use matrix computation as implemented in modern computer algebra systems. Matrix calculations are implemented much faster than loops for calculating sums of products. Here is Mathematica code implementing our algorithm.

```
TourCoeff[n_, k_] :=
  2^(2*n - k - 2)*(Binomial[n - 1, k - n + 1] +
```

```
          4*Binomial[n - 1, k - n] +
          4*Binomial[n - 1, k - n - 1]) -
          KroneckerDelta[n, k] -
          KroneckerDelta[n, k + 1]

CoeffMat[N_] := Transpose[Table[TourCoeff[i, j],
                      {i, 1, N}, {j, 1, N}]]

Tour[n_] :=
  Module[{C}, C = CoeffMat[n - 1];
    v = Table[KroneckerDelta[i, 1], {i, 1, n - 1}];
    Do[v = v.C, {i, 1, n - 1}]; v[[1]]]
```

And here is Mathematica code implementing the algorithm of Kleber and Cook.

```
TourKleb[1, k_] := TourKleb[1, k] = 1
TourKleb[n_, k_] /; k <= n :=
  TourKleb[n, k] =
    If[k > 1, TourKleb[n, k - 1], 0] - TourKleb[n - 1, k] +
      TourKleb[n - 1, 2*k - 1] + TourKleb[n - 1, 2*k]
TourKleb[n_, k_] /; k > n :=
  TourKleb[n, k] =
     Sum[TourKleb[n, k - j]*Binomial[n, j]*(-1)^(j + 1),
      {j, 1, n}]
TourKleb[n_] := TourKleb[n, 1]
```

Computing $T(n)$ for selected values of n needed the following time (in seconds) on a PC with a 1 GHz Pentium processor, Windows 98, and Mathematica 4.1

n	100	150	200	300
Kleber-Cook	24	145	2344	
Neuwirth	2	16	88	6107

We did not compute $T(300)$ with the Kleber-Cook method since that would have taken far too long.
These results might be somewhat surprising since the Kleber-Cook algorithm and our algorithm have the same asymptotic complexity. Two facts at least partially explain why our algorithm is much faster: $T(n,k)$ increases in k, and the Kleber-Cook algorithm computes $T(n,k)$ for $k = 1, 2, \dots 2n$. Our algorithm only uses numbers occurring in $\mathbf{V}$ as defined in Lemma 13 and the elements of $\mathbf{V}$ are much smaller than the elements of $\mathbf{U}$. When calculating the product sums in the recursion equations, we calculate $(n-1)^2$ sums of $n-1$ terms each in our algorithms. The Kleber-Cook algorithm has $n-1$

steps and in step i computes i sums for 3 terms and i sums of i terms. So this algorithm computes fewer sums than our algorithm at the beginning and more sums at the end. At the end, however, the numbers are larger and therefore the computations are more time-consuming. Additionally, rewriting product sums as matrix product speeds up computation considerably.

References

[1] E. Bach and J. Shallit, *Algorithmic Number Theory.* MIT Press, 1996.

[2] P. Capelli and T. V. Narayana, On Knock-Out Tournaments. *Canad. Math. Bull.* **13** (1970), 105-109.

[3] A. Karatsuba and Yu. Ofman, Multiplication of Many-Digital Numbers by Automatic Computers. *Doklady Akad. Nauk SSSR* **145** (1962) 293-294. Translation in *Physics-Doklady* **7** (1963), 595-596.

[4] J. G. Kemeny, Matrix representation for combinatorics, *J. Comb. Theory Ser. A* **36**, (1984), 279-306.

[5] M. Cook and M. Kleber, Tournament sequences and Meeussen sequences, *Electron. J. Combin.* **7** (2000), Research Paper 44, 16 pp. (electronic).

[6] E. Neuwirth, Recursively defined combinatorial functions: extending Galton's board. *Discrete Math.* **239** (2001), 33-51.

[7] N. J. A. Sloane, Sloane's On-Line Encyclopedia of Integer Sequences. `http://www.research.att.com/~njas/sequences/`

[8] A. Schönhage and V. Strassen, Schnelle Multiplikation großer Zahlen. *Computing* **7** (1971), 281-292.

[9] M. Torelli, Increasing Integer Sequences and Goldbach's Conjecture. *Preprint*, (1996).