The Taming of Two Alley CATs

D. Recoskie[*] J. Sawada[†]

January 27, 2012

Abstract

Round robin tournaments are used in a wide variety of competition settings and especially in recreational sports leagues. The outcomes of such tournaments can be modelled by an orientation of the edges of a complete graph, where each vertex corresponds to a unique competitor. A score sequence is an non-decreasing sequence of the out-degrees of such a graph. In this paper we analyze two previously known algorithms that exhaustively generate all feasible score sequences for a given number of competitors n. In both cases, we prove that the algorithms run in Constant Amortized Time: they are CAT.

Keywords: Score sequence, round robin tournament, constant amortized time, generation algorithms

1 Introduction

Round robin tournaments, where every player involved is matched against every other player exactly once, are used in many competition settings. The outcome of such a tournament can be represented by a sequence of integers corresponding to the number of matches each player won. For example, the winter olympic ice hockey tournament makes use of a round robin preliminary round. In the Vancouver 2010 winter olympics, teams were separated into groups of four for the round robin portion. The results involving the Canadian men's team were as follows: the US - 3 wins (beat Canada, Norway, and Switzerland), Canada - 2 wins (beat Norway and Switzerland, Switzerland - 1 win (beat Norway), Norway - 0 wins. The increasing sequence corresponding to this outcome is 0123 (Norway, Switzerland, Canada, US). The women's group had an identical sequence of 0123 (Slovakia, Switzerland, Sweden, Canada). These increasing sequences of integers representing the outcome of a round robin tournament are called *score sequences*.

A *tournament graph* is a complete graph in which each edge is given an orientation. Such a graph can be used to model the outcome of a round robin tournament, where each vertex corresponds to a unique competitor, and the direction of the edge between two vertices indicates the outcome. Using this graphical model, a score sequence corresponds to the non-decreasing sequence of the vertex out-degrees. See Figure 1 for an example of a tournament graph with 5 vertices and its corresponding score sequence.

Score sequences have been studied previously in [11, 3, 1, 9, 7]. Enumeration sequences were given by [12]; they correspond to sequence A000571 in *The On-Line Encyclopedia of Integer Sequences* [16]. Research has gone into areas such as: generating score sequences for transitive tournaments [4] and general oriented

[*]School of Computer Science, University of Guelph, Canada. Research supported by NSERC. E-mail: drecoski@uoguelph.ca

[†]School of Computer Science, University of Guelph, Canada. Research supported by NSERC. E-mail: jsawada@uoguelph.ca

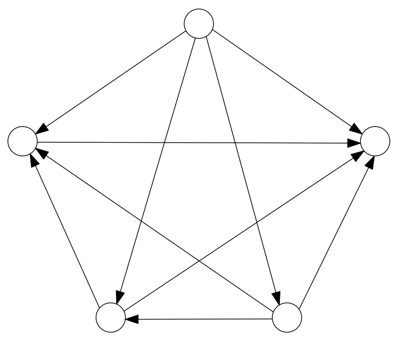

Figure 1: A tournament graph with five vertices. The score sequence corresponding to this graph is 01234.

graphs [2], finding subgraphs in tournaments with specified score sequences [5], bounding the number of score sequences of a given length [17], and generating all score sequences of a given length [6, 13, 15, 14]. The generation algorithms in [15, 14] are conjectured to run in *constant amortized time* (CAT), i.e., the computation time of the algorithm is proportional to the number of score sequences generated. Experimental evidence supports this conjecture, but the authors were unable to provide a formal analysis. As a result, these algorithms are said to be *alley-CATs*.

The main result of this paper is to provide a theoretical analysis proving that the two algorithms to generate score sequences presented in [15] are CAT. Section 2 revisits the first algorithm, which is based on a characterization of Landau [10], and then gives a formal analysis. Section 3 applies an optimization to the second algorithm, which is based on a partition approach, and then gives a formal analysis. Section 4 outlines some avenues for future research.

It is worth mentioning that sporting events are not the only application of round robin tournaments. Many other scenarios can be viewed as a tournament. For example, various types of animals develop a 'pecking order' in which for every pair of animal, one is dominant. Another example is consumer preference among different brands. It is usual for an individual to have a favourite for every pair of brands that they regularly purchase [8]. It is evident that the study of round robin tournaments extends beyond that of simple athletic or game tournaments.

2 Algorithm based on Landau's characterization

The first algorithm for generating score sequences in [15] is based on the following characterization:

THEOREM 1 *[10] A non-decreasing sequence $s_1, s_2, ..., s_n$ of integers is a score sequence of a tournament if and only if*

$$\sum_{i=1}^{k} s_i \geq \binom{k}{2}$$

for $k = 1, 2, \ldots, n$ with equality for $k = n$.

This theorem leads to a method to extend prefixes of score sequences one character at a time. An *n-prescore sequence* is any prefix of a score sequence of length n.

```
procedure GenScore($k, sum$)

    if $k = n$ then
        $s_n := \binom{n}{2} - sum$
        PrintIt()
    else
        for $i :=$ Min($s_{k-1}, \binom{k}{2} - sum$) to $\lfloor \frac{\binom{n}{2} - sum}{n-k+1} \rfloor$ do
            $s_k := i$
            GenScore($k+1, sum+i$)
end.
```

Figure 2: Algorithm to generate all score sequences of length n from [15].

THEOREM 2 *[15] If $s_1, s_2, \ldots, s_{k-1}$ is an n-prescore sequence then $s_1, s_2, \ldots, s_{k-1}, s_k$ is an n-prescore sequence if and only if*

$$max(s_{k-1}, \binom{k}{2} - p_k) \le s_k \le \lfloor \frac{\binom{n}{2} - p_k}{n-k+1} \rfloor,$$

where $p_k = s_1 + s_2 + \cdots + s_{k-1}$, and $s_0 = 0$.

This theorem leads directly to the algorithm GenScore(k, sum) given in Figure 2, where the parameter sum maintains the value p_k and the function PrintIt() outputs the score sequence $s_1, s_2, \ldots, s_n$. The initial call is GenScore($1, 0$) with s_0 initialized to 0. A sample computation tree for this algorithm is given in Figure 3 for $n = 5$.

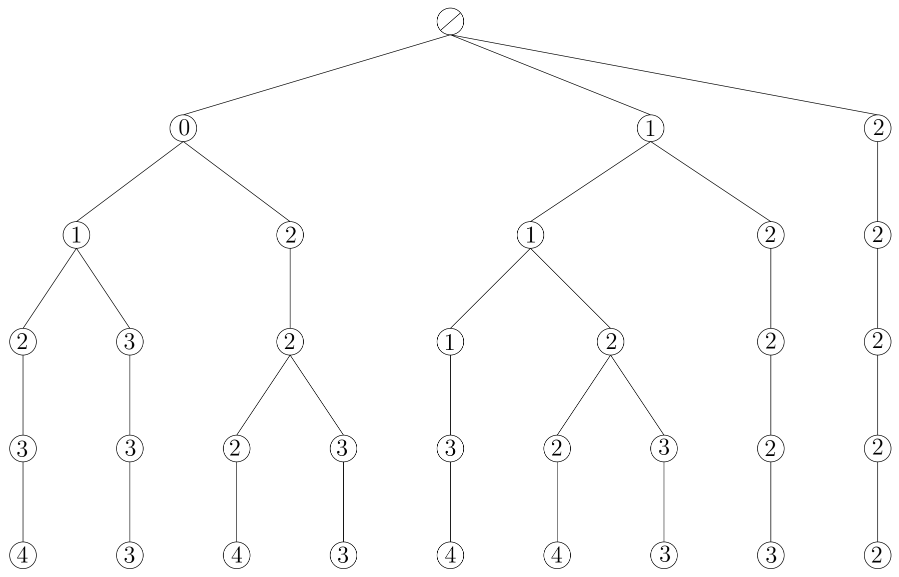

Figure 3: Computation tree for GenScore when $n = 5$. Each path from the root to a leaf corresponds to a score sequence.

2.1 Analysis

In [15], the discussion around analyzing the algorithm for GenScore revolves around finding tight bounds for both the number of score sequences and for the overall size of the computation tree. Each of these problems individually appear to be very difficult. Thus, we take a different approach of directly comparing the number of internal nodes in the computation tree to the number of leaves which correspond to the score sequences. Since each recursive call is the result of a constant amount of work, by showing that the number of internal nodes is proportional to the number of leaves we prove the algorithm is CAT.

For our discussion, an internal node of a computation tree with a single child is said to be a *1-node*; if it has 2 or more children it is called a *2+ node*. Observe that the number of 2+ nodes is bounded by the number of leaves since each 2+ node can be uniquely mapped to a leaf. Such a mapping can be obtained by first visiting the rightmost child of a 2+ node, and then successively following the left child until a leaf is reached. Thus, it remains to show that the number of 1-nodes is proportional to the number of leaves. First, we present some technical results.

LEMMA 1 *If $s_1, s_2, \ldots, s_k$ is a length k score sequence and $k < n-1$, then $s_1, s_2, ..., s_k+1$ is an n-prescore sequence.*

PROOF: Let $\sigma = s_1, s_2, \ldots, s_{k-1}$. If σ, s_k is a length k-score sequence, then $s_k < k$. The number of remaining edges to assign to the n-score sequence is $k + (k+1) + \cdots + (n-1)$. One possible assignment of those edges are: $\sigma, s_k, k, k+1, \ldots, n-1$. Now since $k < n-1$ and $s_k < k$, another possible assignment is $\sigma, s_k+1, k, k+1, \ldots, n-2, n-2$. Therefore, σ, s_k+1 is a valid n-prescore sequence. □

LEMMA 2 *If $\sigma = s_1, s_2, \ldots, s_{k-1}$ corresponds to a 1-node in the computation tree for GenScore, where $k < n-1$, then $s_{k-1} > \binom{k}{2} - \sum\limits_{i=1}^{k-1} s_i$.*

PROOF: (By Contradiction) Let σ be a 1-node, $p_k = \sum\limits_{i=1}^{k-1} s_i$, and $x = \binom{k}{2} - p_k$. Suppose $s_{k-1} \leq x$. Then σ, x will be a child of σ by Theorem 2. Observe that σ, x is a length k score sequence since $p_{k+1} = \binom{k}{2}$. Thus by Lemma 1, $\sigma, x+1$ must also be a child of σ. This is a contradiction because σ was assumed to be a 1-node. Therefore $s_{k-1} > \binom{k}{2} - \sum\limits_{i=1}^{k-1} s_i$. □

LEMMA 3 *If $\sigma = s_1, s_2, ..., s_{k-1}$ corresponds to a 1-node in the computation tree for GenScore, where $k < n-1$, then it will be its parent's right most child.*

PROOF: (By Contradiction) Let σ be a 1-node and $p_k = \sum\limits_{i=1}^{k-1} s_i$. Suppose σ has a sibling immediately to its right. Note that this sibling will have the form σ, s_k+1. Clearly, $max(s_{k-1}, \binom{k}{2} - p_k) < max(s_{k-1}+1, \binom{k}{2} - p_k - 1)$ since $s_{k-1} > \binom{k}{2} - p_k$ by Lemma 2. Since σ is a 1-node, by Theorem 2

$$max(s_{k-1}, \binom{k}{2} - p_k) = \lfloor\frac{\binom{n}{2} - p_k}{n-k+1}\rfloor \geq \lfloor\frac{\binom{n}{2} - p_k - 1}{n-k+1}\rfloor \geq max(s_{k-1}+1, \binom{k}{2} - p_k - 1).$$

But $max(s_{k-1}, \binom{k}{2} - p_k) < max(s_{k-1}+1, \binom{k}{2} - p_k - 1)$, which leads to a contradiction. Therefore, σ is it parent's right most child. □

An immediate consequence of the previous lemma is that any 1-node at a depth less than $n-2$ is the only 1-node of its siblings. It is obvious that the number of 1-nodes which have at least one sibling are proportional to the number of 2+ nodes (i.e. they can be mapped uniquely to their parent). Therefore, it remains to consider 1-nodes which have no siblings, i.e., their parent is also a 1-node. We will refer to these nodes as *1,1-nodes*.

LEMMA 4 *Any 1,1-node must be of the form $s_1, s_2, \ldots, s_c, a^b$, where $b > 1$.*

PROOF: Let $s_1, s_2, \ldots, s_k$ be a 1,1-node and $p_k = \sum_{i=1}^{k-1} s_i$. Since $s_1, s_2, \ldots, s_{k-1}$ is a 1-node, by Lemma 2, $s_{k-1} > \binom{k}{2} - p_k$. Furthermore, by Theorem 2, $s_k = max(s_{k-1}, \binom{k}{2} - p_k)$. Therefore $s_{k-1} = s_k$, and $s_1, s_2, \ldots, s_k$ is of the form $s_1, s_2, \ldots, s_c, a^b$, where $b > 1$. □

LEMMA 5 *The number of 1,1-nodes in the computation tree for GenScore at a depth less than $n-2$ are proportional to the number of 2+ nodes.*

PROOF: We can represent any 1,1-node at level $k, k < n-2$, by σ, a^g, where $a < n$, and $\sigma{=}s_1, s_2, ..., s_{k-g}$ is a n-prescore sequence with no element larger than $a-1$. Observe that $g > 1$ by Lemma 4. We will partition all the 1,1-nodes into distinct equivalence classes based on their form. Let σ, a^b be a 1,1-node at a depth less than $n-2$, where $b > 1$, every element of σ is less than a, and there does not exists a 1,1-node at a depth less than $n-2$ of the form σ, a^e, $e > b$. Any other 1,1-node of the form $\sigma, (a-1)^c a^d$, $d > 1$, $c + d \leq b$, will belong to the equivalence class of σ, a^b. These equivalence classes are mutually exclusive and exhaustive. We will ignore the case where $1 < b < 4$, since any 1,1-nodes in those equivalence classes can be mapped to the 2+ node σ. See Figure 4 for an example equivalence class.

An upper bound on the number of 1,1 nodes in the equivalence class of σ, a^b is $\sum_{i=1}^{b} i = \frac{b(b+1)}{2} = \frac{1}{2}b^2 + \frac{1}{2}b$. We will now show that the lower bound of non-1,1-nodes in this equivalence class is proportional to $\frac{1}{2}b^2 + \frac{1}{2}b$. Now, $\sigma, (a-1)^c a^{b-c}$, $2 < c < b$, will not be a 1,1-node, since both $\sigma, (a-1)^c a^{b-c+1}$ and $\sigma, (a-1)^c a^{b-c}(a+1)$ will be valid prescore sequences. Furthermore, any node of the form $\sigma, (a-1)^c a^d$, $c > \lfloor \frac{b}{2} \rfloor$, $c + d < b$, $d > 0$, must not be a 1,1-node because, similarly, $\sigma, (a-1)^c a^{d+1}$ and $\sigma_{k-b}, (a-1)^c a^d (a+1)$ exist. It follows that the number of nodes guaranteed not to be 1,1-nodes from this equivalence class is at least $b - 3 + \sum_{i=1}^{\lfloor \frac{b}{2} \rfloor - 2} i = b - 3 + \frac{(\lfloor \frac{b}{2} \rfloor - 2)(\lfloor \frac{b}{2} \rfloor - 1)}{2} \geq b - 3 + \frac{(\frac{b-1}{2} - 2)(\frac{b-1}{2} - 1)}{2} = \frac{1}{8}b^2 - \frac{9}{8}$. Thus the number of 1,1-nodes in the equivalence class is proportional to the number of non-1,1-nodes that can be found in this equivalence class. Since 1-nodes are proportional to 2+ nodes, the number of 1,1-nodes is proportional to the number of 2+ nodes. The non-1,1-nodes in this equivalence class are unique from those in any other equivalence class, so the total number of 1,1-nodes in the computation tree must be proportional to the total number of 2+ nodes. □

This result now immediately gives us the following theorem.

THEOREM 3 *The algorithm GenScore to list all score sequences of length n runs in constant amortized time.* □

3 Algorithm based on a partition approach

The second algorithm for score sequences outlined in [15] uses an approach similar to a classical approach for integer partitions: consider those partitions that contain some maximal part p and those that do not. For

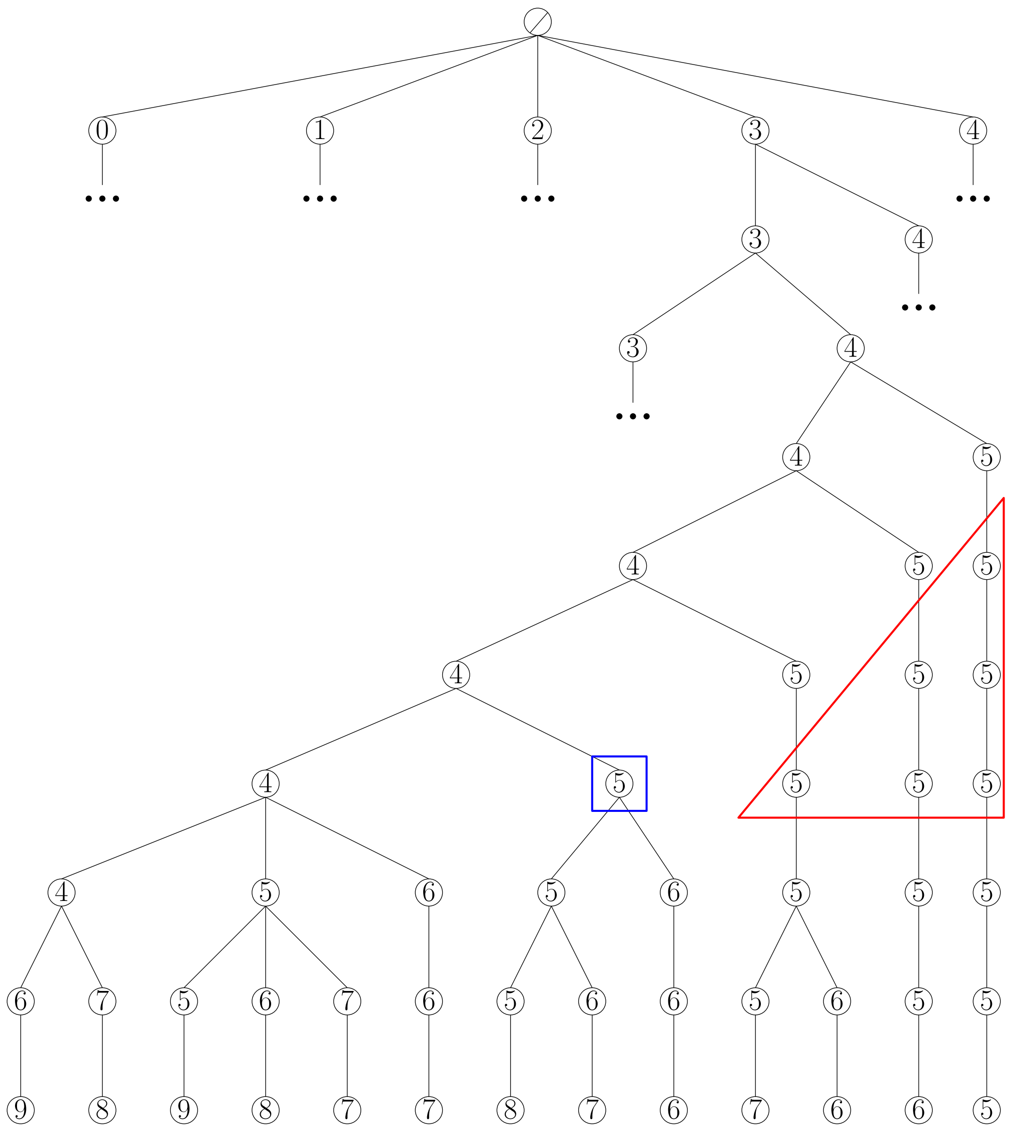

Figure 4: An equivalence class for $n = 10$. Here, σ, a^b is 3345555 with $\sigma = 334$ and $b = 4$. The 1,1-nodes in the triangle belong to the equivalence class. The node in the box is the 2+ node found from this equivalence class.

```
procedure GenScore2($m, k, p$: integer)

    if $k = 2$ then
        for $i :=$ Min$(m, p)$ down to $\lceil \frac{m}{2} \rceil$ do
            $a_2 := i$
            $a_1 := m - i$
            PrintIt()
    else if $m - p < \binom{k-1}{2}$ then GenScore2$(m, k, p - 1)$
    else if $p - 1 < \lceil m/k \rceil$ then
        $a_k := p$
        GenScore2$(m - p, k - 1, p)$
    else
        $a_k := p$
        GenScore2$(m - p, k - 1, p)$
        GenScore2$(m, k, p - 1)$
end.
```

Figure 5: An partition-based approach to generate all score sequences of length n that contains an optimization from the version in [15].

score sequences, we want to partition the sum of the degrees $m = \binom{n}{2}$ into $k = n$ parts where the largest part is $p = n - 1$. In addition, we must satisfy the constraints laid out in Theorem 1. This set, denoted $\mathbf{P}(m, k, p)$, can be defined recursively as follows [15]:

$$\mathbf{P}(m,k,p) = \begin{cases} m & \text{if } k = 1 \\ \mathbf{P}(m,k,p-1) & \text{if } k > 1 \text{ and } m - p < \binom{k-1}{2} \\ \mathbf{P}(m-p,k-1,p) \circ p & \text{if } k > 1 \text{ and } p - 1 < \lceil m/k \rceil \\ \mathbf{P}(m-p,k-1,p) \circ p \cup \mathbf{P}(m,k,p-1) & \text{otherwise,} \end{cases}$$

where the notation $\mathbf{S} \circ x$ denotes the appending of the character x to each sequence in the set $\mathbf{S}$. The set of score sequences of length n is precisely the set $\mathbf{P}(\binom{n}{2}, n, n-1)$. A simple pseudocode derived from this recurrence is illustrated in Figure 5. This pseudocode contains one additional optimization not apparent from the recurrence. Instead of the base case being $k = 1$, we set the base case to $k = 2$ and use a simple **for** loop to handle all possibilities for the first two elements in the score sequence. Specifically, for a given $\mathbf{P}(m, 2, p)$, the first two elements of a score sequence must be of the form $a_2 = i$ and $a_1 = m - i$ where i ranges from the minimum of m and p down to $\lceil \frac{m}{2} \rceil$. Observe that this simple optimization may save a chain of recursive calls that do not actually extend the score sequence in the cases where $p > m$.

A sample computation tree for GenScore2 when $n = 5$ is illustrated in Figure 6. The nodes marked with a * are the recursive calls that do not prepend an element to the prescore sequence. In this figure, each score sequence is obtained by following a path from a leaf back to the root, where the * nodes are ignored.

3.1 Analysis

By reversing a score sequence $\sigma = s_1 s_2, \ldots, s_n$ for a tournament graph G and replacing each s_i with $n - s_i$, we obtain an increasing sequence of the in-degrees for the vertices of G. If the direction of the arcs in G are reversed, then note that this resulting sequence will also be a score sequence for a tournament graph. If we apply this operation to the algorithm GenScore2, then observe that the sequences generated will be in the exact same lexicographic order as the algorithm GenScore. Moreover, if we apply this operation

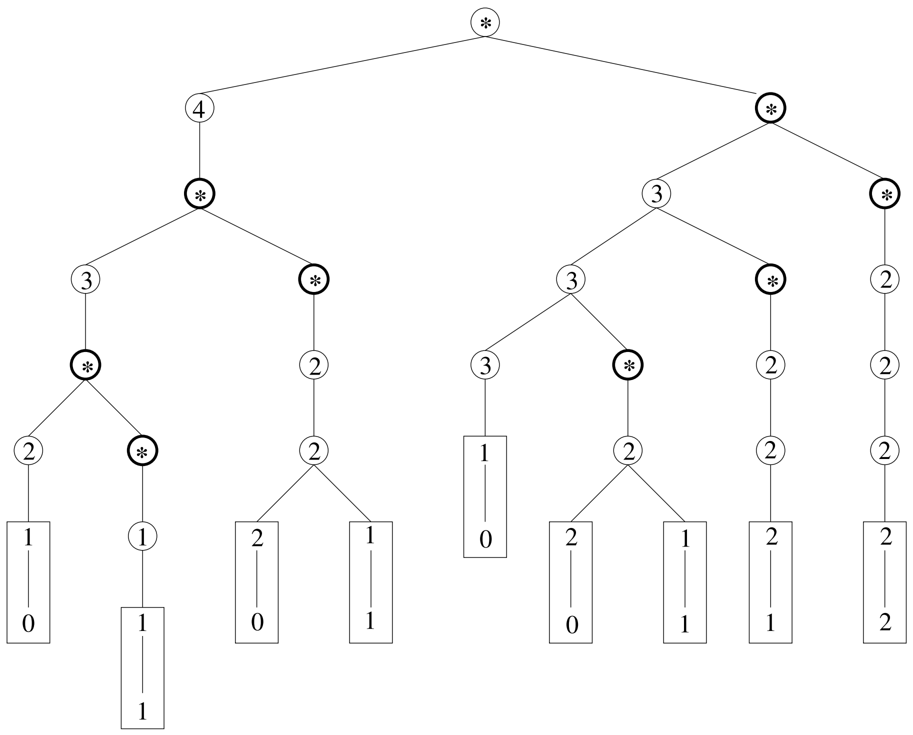

Figure 6: Computation tree for GenScore2 when $n = 5$. Each path from a leaf to the root corresponds to a score sequence.

to the computation tree of GenScore2 and then successively contract each edge between a * node (except for the root) and its parent, then the resulting tree will be identical to the computation tree for GenScore. This operation effectively removes all the * nodes except for the root. By performing these actions on the computation tree in Figure 6, the outcome will be the tree in Figure 3. Thus, to prove that the algorithm GenScore2 is CAT, we need only show that the number of * nodes in its computation tree is proportional in number to the non * nodes.

Let $\mathbf{CompTree}(n)$ denote the set of all nodes in the computation tree for GenScore2 and a given n. Each node u can be uniquely represented by a sequence $a_1, a_2, \ldots, a_i, *^j$, where $a_1, a_2, \ldots a_i$ represents the sequence of non * labels on the path from the root to u and the number of * nodes on the path from the node with label a_i to u is equal to j. Using this representation, the nodes of $\mathbf{CompTree}(n)$ can be partitioned into the following 3 sets:

- ▷ $\mathbf{T}$ = the set of all nodes of the form $a_1, a_2, \ldots, a_i$,
- ▷ $\mathbf{U}$ = the set of all $u = a_1, a_2, \ldots, a_i, *^j$ where $j > 0$ and u has a child in $\mathbf{T}$,
- ▷ $\mathbf{V}$ = the set of all $u = a_1, a_2, \ldots, a_i, *^j$ where $j > 0$ and u has no child in $\mathbf{T}$.

The set $\mathbf{T}$ corresponds to the nodes in the computation tree with a non * label, and when $i = n$, the sequences correspond to the reversals of n-score sequences. The sets $\mathbf{U}$ and $\mathbf{V}$ partition the nodes with a * label. By mapping each node $u \in \mathbf{U}$ to its child in $\mathbf{T}$ it is easy to observe that $|\mathbf{U}| \leq |\mathbf{T}|$. Thus, we need only show that $|\mathbf{V}|$ is proportional to $|\mathbf{T}|$. We do this by partitioning the nodes further by a common length and prefix $\alpha = a_1, \ldots, a_i$. Let:

- ▷ $\mathbf{T}(\alpha)$ = the set of all $\alpha, a_{i+1}, a_{i+2} \in \mathbf{T}$,
- ▷ $\mathbf{V}(\alpha)$ = the set of all $\alpha, a_{i+1}, *^j \in \mathbf{V}$.

Since the base case for GenScore2 occurs when $k = 2$, when we are partitioning $\mathbf{V}$ by $\alpha = a_1, \ldots, a_i$, we can assume that $i < n - 2$. Figure 7 illustrates a partial computation tree highlighting the nodes belonging to $\mathbf{V}(\alpha)$ and $\mathbf{T}(\alpha)$ for a given α.

LEMMA 6 *If* $\alpha = a_1, \ldots, a_i \in \mathbf{T}$, *then* $|\mathbf{V}(\alpha)| \leq 8|\mathbf{T}(\alpha)|$.

PROOF: We divide this proof into two steps. First we find an upper bound for $|\mathbf{V}(\alpha)|$, and then we find a lower bound for $|\mathbf{T}(\alpha)|$.

Upper bound for $|\mathbf{V}(\alpha)|$**.** Let x denote the largest value such that $\alpha, x \in \mathbf{T}$. Furthermore, let m be the largest integer such that $\alpha, (x - m) \in \mathbf{T}$. Observe that in general if α, x and $\alpha, (x - j)$ are both in $\mathbf{T}$ for any $j > 1$, then so is $\alpha, (x - j')$ for $0 < j' < j$. Thus, for each $0 \leq j \leq m$ we determine exactly how many nodes the prefix $\alpha, (x - j)$ contributes to $\mathbf{V}(\alpha)$.

First consider $j = 0$. Let y denote the largest value such that $\alpha, x, y \in \mathbf{T}$ which implies that $\alpha, x, (y+1) \notin \mathbf{T}$. This means there are $t = x - y - 1$ nodes in $\mathbf{V}(\alpha)$ with prefix α, x. For any $0 < j < \lceil t/2 \rceil$, observe that:

- ▷ $\alpha, (x - j) \in \mathbf{T}$,
- ▷ $\alpha, (x - j), (y + j) \in \mathbf{T}$,
- ▷ $\alpha, (x - j), (y + j + 1) \notin \mathbf{T}$.

The latter 2 sequences are easy to verify since the sum of their elements is the same as the corresponding sequences when $j = 0$ (i.e., the parameters m and k will be the same for their respective recursive call, but p is larger). The statement that $\alpha, (x - j) \in \mathbf{T}$ is slightly less obvious, but the conditions in the algorithm can be verified using the fact that $\alpha, x, y \in \mathbf{T}$. Thus, for $0 \leq j < \lceil t/2 \rceil$, there are $t - 2j$ nodes in $\mathbf{V}(\alpha)$ with prefix $\alpha, (x - j)$. For any $\lceil t/2 \rceil \leq j \leq m$, it follows directly from the algorithm and the fact that $\alpha, x, y \in \mathbf{T}$ that $\alpha, (x - j), (x - j) \in \mathbf{T}$ and hence any such prefix $\alpha, (x - j)$ will not contribute to the set $\mathbf{V}(\alpha)$. From this analysis we have:

$$|\mathbf{V}(\alpha)| = \sum_{j=0}^{\lceil t/2 \rceil - 1} (t - 2j) \leq t^2/2.$$

As an illustration of this analysis, see Figure 7, where $x = 9, y = 2$ and $t = 6$. For the α in this example there are $6 + 4 + 2 = 12$ nodes in $\mathbf{V}(\alpha)$.

Lower bound for $|\mathbf{T}(\alpha)|$**.** We have already demonstrated that $\alpha, x, y \in \mathbf{T}$. Let b, β denote the lexicographically largest sequence such that α, x, y, b, β is the reversal of an n-score sequence. Note that $b \leq y$. For $0 \leq j < \lceil t/2 \rceil$, we have also demonstrated that $\alpha, (x - j), (y + j) \in \mathbf{T}$. Since the sum of the elements of each of these sequences is the same as α, x, y, then it will also be the case that the reversal of $\alpha, (x - j), (y + j), b, \beta$ is also an n-score sequence. If $j \geq 2$, then $\alpha, (x - j), (y + j - 1), b + 1, \beta$ will also be a valid n-score sequence since $b + 1 \leq y + j - 1$. Similarly if $j \geq 4$, then $\alpha, (x - j), (y + j - 2), b + 2, \beta$ will also be a valid n-score sequence. Following this pattern, observe that the number of nodes in $\mathbf{T}(\alpha)$ with the prefix $\alpha, (x - j)$ is at least $\lceil (j + 1)/2 \rceil$ for $0 \leq j < \lceil t/2 \rceil$. Thus,

$$|\mathbf{T}(\alpha)| \geq \sum_{j=0}^{\lceil t/2 \rceil - 1} \lceil (j + 1)/2 \rceil \ \geq \ t^2/16.$$

Thus $|\mathbf{V}(\alpha)| \leq 8|\mathbf{T}(\alpha)|$. □

This previous result now immediately gives us the following theorem.

THEOREM 4 *The algorithm GenScore2 to list all score sequences of length n runs in constant amortized time.* □

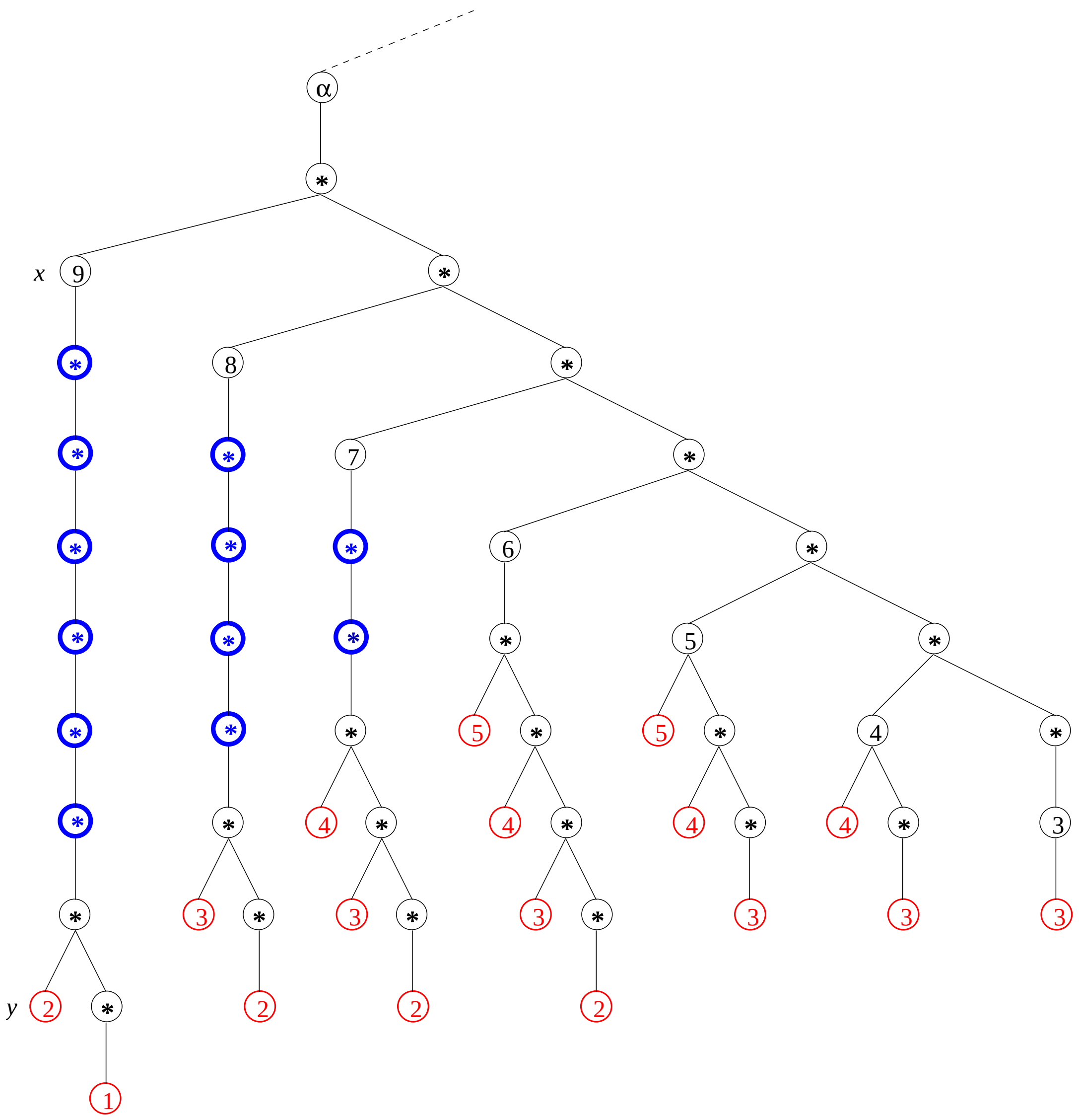

Figure 7: Partial computation tree that arises from a recursive call to GenScore2(12,4,10) for some prefix α. The $*$ nodes in bold are the precisely those that belong to $\mathbf{V}(\alpha)$. The leaves are precisely the nodes in $\mathbf{T}(\alpha)$.

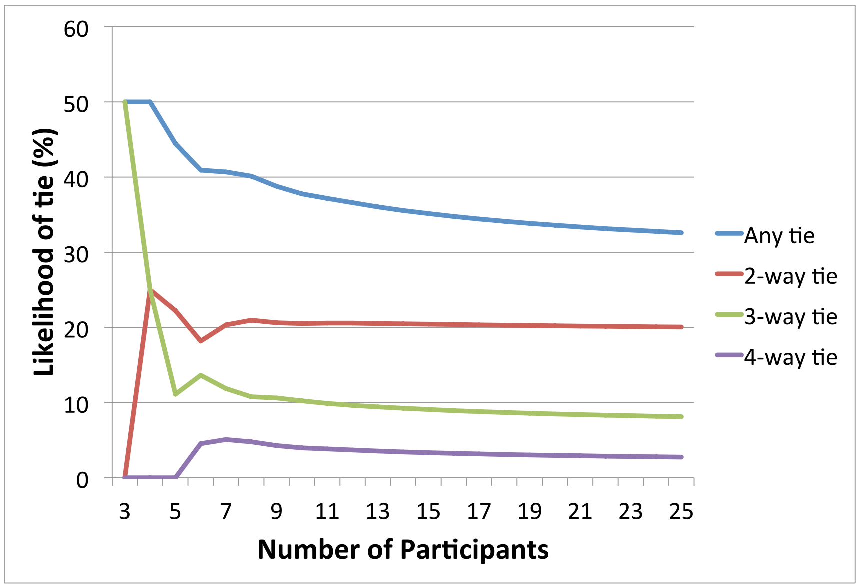

Figure 8: Likelihood a score sequence contains a first place tie in round robin tournaments. Also the likelihood that the first place tie is between precisely 2, 3, or 4 competitors.

4 Summary and future work

We have shown that the two score sequence generation algorithms presented in [15] run in constant amortized time. Some interesting avenues of future research include:

1. providing an analysis for the remaining alley-cat from [15] for degree sequences,
2. developing a model and sequence for double round-robin tournaments (as used in the English Premier League football system),
3. developing a model and sequence to allow ties within a competition,
4. proving the asymptotic behaviour of the likelihood of first place ties.

For the latter question we look at small values of n to determine the likelihood of a first place tie. A tournament will end with a first place tie if the last two elements in its corresponding score sequence are the same. For example, the score sequence 0222 represents a three-way first place tie in a four player tournament. See Figure 8 for a summary of first place tie outcomes for various sized tournaments. The data suggests that as the number of participants increases, the percentage of score sequences with first place ties approaches 30%. Ties can be broken down by the number of players involved in the tie. Figure 8 shows the likelihood of 2, 3, and 4-way ties. Judging from the data, as the number of players increase the likelihood of a specific type of tie approaches a constant.

References

[1] Peter Avery. Condition for a tournament score sequence to be simple. *Journal of Graph Theory*, 4(2):157–164, 1980.

[2] Peter Avery. Score sequences of oriented graphs. *Journal of Graph Theory*, 15(3):251–257, 1991.

[3] Richard A. Brualdi and Jian Shen. Landau's inequalities for tournament scores and a short proof of a theorem on transitive sub-tournaments. *Journal of Graph Theory*, 38(4):244–254, 2001.

[4] Arthur H. Busch, Guantao Chen, and Michael S. Jacobson. Transitive partitions in realizations of tournament score sequences. *Journal of Graph Theory*, 64(1):52–62, 2010.

[5] Zhicheng Gao, Brendan D. McKay, and Xiaoji Wang. Asymptotic enumeration of tournaments with a given score sequence containing a specified digraph. *Random Struct. Algorithms*, 16(1):47–57, 2000.

[6] S. V. Gervacio. Score sequences: lexicographic enumeration and tournament construction. *Discrete Math.*, 72:151–155, December 1988.

[7] J. Griggs and K. B. Reid. Landau's theorem revisited. *Australas Journal of Combinatorics*, 20:19–24, 1999.

[8] Frank Harary and Leo Moser. The theory of round robin tournaments. *The American Mathematical Monthly*, 73(3):pp. 231–246, 1966.

[9] Frank Harary and Ed Palmer. On the problem of reconstructing a tournament from subtournaments. *Monatshefte fr Mathematik*, 71:14–23, 1967. 10.1007/BF01299955.

[10] H. Landau. On dominance relations and the structure of animal societies: III The condition for a score structure. *Bulletin of Mathematical Biology*, 15:143–148, 1953. 10.1007/BF02476378.

[11] J. W. Moon. An extension of Landau's theorem on tournaments. *Pacific Journal of Mathematics*, 13(4):1343–1345, 1963.

[12] T. V. Narayana and D. H. Best. Computation of the number of score sequences in round-robin tournaments. *Canad. Math. Bull.*, 7:133–136, 1964.

[13] T. V. Narayana, R. M. Mathsen, and Sarangi J. An algorithm for generating partitions and its applications. *Journal of Combinatorial Theory*, 11:54–61, 1971.

[14] Hemasinha R. An algorithm to generate tournament score sequences. *Mathematical and Computer Modelling*, 37(3):377–382, 2003.

[15] Frank Ruskey, Robert Cohen, Peter Eades, and Aaron Scott. Alley cats in search of good homes. *Congressus Numerantium*, 102:97–110, 1994.

[16] N Sloane. The on-line encyclopedia of integer sequences, sequences a000571/m1189 and a068029.

[17] Kenneth J. Winston and Daniel J. Kleitman. On the asymptotic number of tournament score sequences. *J. Comb. Theory, Ser. A*, 35(2):208–230, 1983.