Variations on the Tournament Problem

Fabrizio Luccio ✉
University of Pisa, Italy

Linda Pagli ✉
University of Pisa, Italy

Nicola Santoro ✉
Carleton University, Ottawa, Canada

Abstract

In 1883, Lewis Carrol wrote a newspaper article to criticize how the second best player was determined in a tennis tournament, and to suggest how such a task could be done correctly. This article has been taken by Donald Knuth as the inspiration for efficiently determining the smallest t elements of a totally ordered set of size n using k-comparisons. In the ensuing research, optimal algorithms for some low values of k and t have been established, by Knuth and Aigner; for $k = 2$ and $t \leq 3$, a few new bounds have been established for special values of n. Surprisingly, very little else is known on this problem, in spite of its illustrious pedigree and its relationship to other classical problems (e.g., selection and sorting with k-sorters). Enticed by the undeniable beauty of the problem, and the obvious promise of fun, we have joined the investigative quest. The purpose of this paper is to share some new results obtained so far. We are glad to report advances in two directions.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Computing methodologies → Parallel algorithms; Mathematics of computing → Discrete mathematics

Keywords and phrases algorithms, parallel algorithms, tournament, selection, ranking

Digital Object Identifier 10.4230/LIPIcs.FUN.2024.20

Funding This research has been supported in part by the Natural Sciences and Engineering Research Council of Canada under the Discovery Grant program.

1 Introduction

Lewis Carrol, the poet, mathematician, photographer, and beloved author of *Alice's Adventures in Wonderland*, was also known to be a tennis enthusiast and to have followed many important matches. He did observe that, in the single-elimination tournaments usually adopted for tennis matches, the selection of the winner was fair while that of the runner-up was not [4]. Indeed, in this type of tournaments, the player defeated in the final match is declared second, with the implicit meaning of "second best in the tournament", while s/he is the true second only with a probability close to half. If, for instance, the true second belongs to the same part of the board as the champion, s/he does not have any possibility to emerge. Lewis Carrol also explained with an example how to determine the true second and third, without however specifying a real algorithm [4]; furthermore, his proposal required a high number of matches, most of which are actually not needed for this determination. A well known presentation and discussion of Lewis Carroll's proposal has been provided by Donald Knuth [5].

In the game of tennis a match involves two players; in general, the problem of determining the ranking of the players in tournaments can be formulated as the following combinatorial problem: given a set of n elements and a comparison operation on k elements (k-comparison) that returns the linear ordering of them, we want to calculate the number of k-comparisons needed to find the top t elements. We shall call this problem `t-TopFinding in k-TrackRacing` (t-T-k-T); see Figure 1.

Let $C(n,k,t)$, with $t < n$, be the number of k-comparisons of an algorithm that determines the first t elements among n (*upper bound*), and let $S(n,k,t)$ be the minimum number of k-comparisons required to solve the problem (*lower bound*). For $k = 2$ and $t = n$ the problem corresponds to traditional sorting with binary comparisons. Optimal algorithms for some low values of k and t have been established by Knuth [5] and, a decade later, by Aigner [1], namely:

▶ **Fact 1** ([5]).

$$\begin{aligned} C(n,2,1) &= S(n,2,1) = n-1 \\ C(n,2,2) &= S(n,2,2) = n + \lceil \log_2 n \rceil - 2 \\ S(n,k,1) &= \lceil \frac{n-1}{k-1} \rceil \end{aligned}$$

▶ **Fact 2** ([1]). *Let $n = 2^s + r$, where $0 \le r \le 2^{s-1}$. Then*

$$C(n,2,3) = S(n,2,3) = \begin{cases} (n-3) + \lceil \log_2 n \rceil & \text{if } r = 0, \\ (n-3) + \lceil \log_2 n \rceil + 1 & \text{if } 1 \le r \le 2^{s-2}, \\ (n-3) + \lceil \log_2 n \rceil + 2 & \text{otherwise.} \end{cases}$$

For $k = 2$ and $t \le 3$, a few new bounds have been established for special values of n [6, 3]. Surprisingly, very little else is known on this problem, in spite of its relationship to classical problems of selection and sorting (see the nice review by Iványi and Fogarasi [8]) especially using k-comparators (called k-sorters, e.g. [2, 9])

Enticed by the noble pedigree of the problem, the undeniable beauty of the challenge, and the obvious promise of fun, we have started to examine t-T-k-T. The purpose of this paper is to share some new results obtained so far. We are glad to report advances in two directions.

In Section 2 of this paper we discuss our study of the problem, for any value of n and k, when $t = 2$. We show that a standard two-phases tournament can be improved when $(n-1)/(k-1)$ is not an integer, and find a new lower bound $S(n,k,2)$ that matches the upper bound $C(n,k,2)$ for all values of n.

In Section 3 we take a different approach which has so far received little attention. We consider the k-comparison problem in a parallel setting, where the most significant function to consider is the number $R(n,k,t,p)$ of *rounds* played in parallel by a certain number of groups of k competitors, where p is now the maximum number of available facilities (racetracks, or tennis courts, or football fields, etc.). For minimum R we then minimize $C(n,k,t,)$. We study the problem thoroughly for k and t up to three, conjecturing what happens for larger values.

Figure 1 A horse race: a real life instance of t-T-k-T.

2 Winner and runner-up using k comparisons

2.1 An introductory example

Consider n horses running on a track of k lanes, among which we want to establish the first $t = 2$ with minimal number of races. After a race, a linear ordering is established among the k horses on the track, based only on the finishing order (race time is not considered). For $n = 25$ and $k = 5$ lanes in a track, we can find the winner and runner-up by applying an immediate generalization of the 2-phases known tournament algorithm for $k = 2$. In the first phase, the horses are divided into $n/k = 5$ groups of five horses each, and each group runs in a race corresponding to a 5-comparison. Then the five winners run in a final race to establish the champion, with a total of six races. The second phase takes place with a second tournament between all the horses who were beaten directly (i.e. in the same race) by the champion and ranked second in that race. Since there are five such horses, one additional race is sufficient to award the second place, for a total of sevens races.

2.2 The upper bound

Let us review what is known on $C(n, k, 2)$ for arbitrary values of n and k, and propose a non-standard tournament organization by which the value of $C(n, k, 2)$ can be lowered for particular pairs n, k.

As stated in Fact 1, $\lceil \frac{n-1}{k-1} \rceil$ k-comparisons are needed in general to establish the top element. In a standard two-phases tournament for arbitrary k, the top-element appears in $\lceil log_k n \rceil$ k-comparisons, from which $\lceil log_k n \rceil$ elements are ranked as a possible second and additional $\lceil \lceil (log_k n \rceil - 1)/(k-1) \rceil$ k-comparisons are used to establish the real second one. As we have seen in the introductory example above, for $n = 25$ and $k = 5$ we have $\lceil log_k n \rceil = 2$, $\lceil \lceil log_k n \rceil - 1/(k-1) \rceil = 1$, and a further k-comparison is sufficient to award second place. Therefore for a standard two-phases tournament we have:

▶ **Fact 3.** $C(n, k, 2) = \lceil (n-1)/(k-1) \rceil + \lceil (\lceil log_k n \rceil - 1)/(k-1) \rceil$.

This upper bound is tight if $(n-1) \mod (k-1) = 0$. However, should $(n-1)$ and $(k-1)$ be coprime, in a k-comparison of the first phase, not all k entry points are needed; these "free" entry points could be used to partially contribute to the second phase where the second element is determined.

For example, for $n = 27$ and $k = 5$ we perform $\lceil n - 1/(k-1) \rceil = 7$ comparisons for the top-element, but one of them has two not exploited entry points that can be used in the search for the second element. As shown in Figure 2, we can re-organize the initial phase of a tournament on $n = 27$ elements, postponing the last of the seven 5-comparisons where the winner of the first twenty-five elements is compared to elements number 26 and 27, to include the elements x and y that *must* compete for the second place. So the total number of k-comparisons is seven instead of eight, as we would get from Fact 3 if a standard two-phases algorithm is applied.

We now transform the above considerations into a precise algorithmic form, and we establish an upper bound for $C(n, 2, 2)$ and any value of n and k with $n > k$.

Let $r = (n-1) mod (k-1)$; then the number p of entry points of a k-comparison left free in the first tournament is $p = 0$ if $r = 0$, $p = k - r - 1$ otherwise. Let $n' = \lceil log_k n \rceil$; let $n'' = n' - p - 1$ if $n' > p$, $n'' = 0$ otherwise.

If $r \neq 0$ and $n' \leq p$, the free entry points of the first phase can absorb all the remaining elements among which the second element has to be determined. This is the case of the latter example where $n = 27$, $k = 5$, $\lceil log_k n \rceil = 3$, $r = 2$, and the selection of the first and second element can be performed in a single phase. We have:

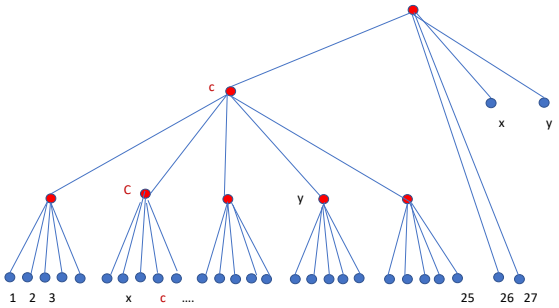

Figure 2 Tournament for $n = 27$ and $k = 5$. The selection of the first and second element can be accomplished in the first phase. The last comparison includes elements $26, 27, x$ and y. x is the second in the first k-comparison of the champion c, and y is the second in the second comparison of c.

▶ **Theorem 1.** *Let $n > k \geq 2$. Then*

$$C(n,k,2) = \begin{cases} \lceil (n-1)/(k-1) \rceil + \lceil (n'-1)/(k-1) \rceil & \text{if } r = 0, \\ \lceil (n-1)/(k-1) \rceil + \lceil n''/(k-1) \rceil & \text{otherwise.} \end{cases}$$

Proof. Case $r = 0$ is known. If $r \neq 0$ run the first tournament with $n + p$ elements with $\lceil (n-1)/(k-1) \rceil$ comparisons. If $n' \leq p + 1$, then $n'' = 0$ and no other comparisons is needed. Otherwise, $n'' = n' - p - 1$ and a second tournament is run with $n - p$ elements. ◀

2.3 The lower bound

We now show a matching lower bound $S(n,k,2)$ for determining the first and second elements, recalling the definition of r and p of the previous section. .

▶ **Theorem 2.** *Let $n > k \geq 2$. Then $S(n,k,2) = C(n,k,2)$.*

Proof. First of all, note that for $k = 2$ the lower bound is the known formula $S(n,2,2) = n + \lceil \log_2 n \rceil - 2$ (see Fact 1). Note that this formula has been proved by [5] selecting the second, once the first has been found. We follow the same approach.
Further note that, for $r \neq 0$ and $p = 0$, upper and lower bounds match.

Let $k > 2$; we shall consider the cases $r = 0$ and $0 < r < k$, separately.

Case $r = 0$.
First observe that $\lceil (n-1)/(k-1) \rceil$ is the necessary number of k-comparisons for the selection of the top element. Let us then determine the minimal number of additional k-comparisons required to determine the second element. In order to do so, we need not to include any comparison among elements whose relation can be derived, by transitivity, from previous k-comparisons. In order to minimize the number of elements ordered by transitivity, it is sufficient to impose the following *balancing rule*: if, in a k-comparison, it results that x_j is less than x_l, $1 \leq j \neq l \leq k$, then the number of elements already known to be less than x_j,

are less or equal than the number of elements already known to be less or equal than x_l. Let us now define as $L(i)$ the maximum number of elements established to be less than the champion after i k-comparisons; this knowledge can be acquired either directly, because the elements are compared with the top one in a k-comparisons, or by the transitive property. From the balancing rule, we have the following recursive definition:

$$L(i) = \begin{cases} k-1 & \text{if } i = 1 \\ kL(i-1) + k - 1 & \text{otherwise} \end{cases}$$

whose closed formula is $L(i) = k^i - 1$.

Since the elements less than the champion are $n-1$, there exists a $\hat{i}$ such that $L(\hat{i}) = k^{\hat{i}} - 1 = n - 1$. The value $\hat{i} = \lceil log_k n \rceil$ represents the number of k-comparisons done by the champion to win the competition. The second element has to be established among all the elements classified at the second place in a direct k-comparison with the champion, whose number is precisely $\hat{i}$. Therefore the second element cannot be determined in less than $\lceil \lceil (log_k n \rceil - 1)/(k-1) \rceil$ additional comparisons, for a total of $\lceil (n-1)/(k-1) \rceil + \lceil (n'-1)/(k-1) \rceil$.

Case $0 < r < k$.
In this case, exactly $p = k - r - 1$ entry points are empty and can be exploited to compare elements to determine the second element. If $n' - 1 \leq p$, all these elements can be accommodated in the empty entry points and the second element can be selected directly in the block of comparisons needed to establish the first element. However, if $n' > p$, at least $n'' = n' - p - 1$ elements require a second phase, requiring an additional $\lceil (n''-1)/(k-1) \rceil$ k-comparisons, fora total of $\lceil (n-1)/(k-1) \rceil + \lceil n''/(k-1) \rceil$. ◀

Let consider, for instance, the case $n = 47$ $k = 4$. We have $n' = 3$. According to the theorem we have $S(47, 4, 2) = \lceil 46/3 \rceil + \lceil (3 - 2 - 1)/3 \rceil = 16$. Computing the two phases separatedy we would have been obtained a number of comparisons equal to: $\lceil 47/3 \rceil + \lceil (3-1)/3 \rceil = 16 + 1 = 17$.
It is very easy to modify the tournament algorithm of Sec.2 in a way that the case that the approximation $\lceil (n-1)/(k-1) \rceil$ is not exact is also considered, in order to obtain a matching upper bound.

3 t-T-k-T in parallel

Everything we have discussed so far reflects a standard computing approach. Let us now look at the problem from a slightly different point of view, that is, the various operations required may take very different times to be executed. In particular this is the case of sports competitions such as horse racing, tennis tournaments, final stages of football championships, where comparisons of indexes such as those appearing in **while** or **do** instructions needed for deciding the order of the events take incomparably less time than the comparison in **if** $A[i] > A[j]$, where $A[i]$ and $A[j]$ are two players competing in the game.

In fact the complexity of most sorting and searching algorithms is evaluated in order of magnitude by counting the number of comparisons between elements of the dataset, and this is due to the observation that the number of all other operations is of the same order as those, and each of them requires time comparable to the others. Instead we will now evaluate the number of single sports matches because they dominate the overall time. In particular we consider the *k-comparison problem* in a parallel setting, where a new function to consider

is the number $R(n, k, t, p)$ of *rounds* played in parallel by a certain number of groups of k competitors, where p is a new parameter representing the maximum number of available facilities (racetracks, or tennis courts, or football fields). Our primary goal is minimizing the number R of rounds (e.g. days) the tournament lasts, imagining that each competitor can play a maximum of one game per day. For minimum R we then minimize the total number $C(n, k, t)$ of comparisons made (e.g., for better conservation of the facilities). As you shall see, we study the problem in full for k and t up to three, leaving as a conjecturing what happens for larger values.

Note that the k-comparison problem was discussed in depth in the literature without any real attention to parallel processing. This aspect was considered in [10] in a much wider context, evaluating the time complexity in order of magnitude as a function of n and p instead of counting the exact number comparisons and rounds as we do. And then it was discussed in [7], with focus on the overhead communication time for different kinds of interconnection networks. Our approach is completely different. Let's start with some clarifications on what was said in the previous sections.

The results of Fact 1 are valid for any value $p \geq 1$. Another result implicitly discussed in the previous section must be better specified, namely:

▶ **Fact 4.** *$R(n, k, 1, p) = \lceil \log_k n \rceil$ is an upper and a lower bound of R for $p \geq n/k$.*

Note that the upper bound in Fact 4 is demonstrated directly by referring to the standard tournament algorithm (see the proof of Theorem 2), and the lower bound is also proved by equivalent reasoning although this is generally not underlined. In fact, for $k = 2$ the adversary model used in the proof implies that in an optimal algorithm with $n - 1$ comparisons, when the final winner f is engaged in a new comparison with another competitor g the following must happen. If f and g have already been recognised as superior to other participants of sets F and G respectively, f wins the comparison with g only if $|F| \geq |G|$ and $F \cap G = \Phi$. Then the computation must proceed essentially as in a tournament or equivalent, where f and g are the temporary winners of two independent sub-problems solved on disjoint subsets of participants. This reasoning can be immediately extended to $k > 2$ where an optimal algorithm requires $\lceil \frac{n-1}{k-1} \rceil$ k-comparisons to find the winner f, and when this is engaged in any k-comparison with $k - 1$ competitors all of them must be top elements of disjoint subsets of cardinality less or equal the one of f.

From now on we will assume that n is a power of k without affecting the core of the problem. In fact we can add fictitious competitors who, by default, would lose all matches with the others, until reaching a number of competitors equal to the next power of k. This does not affect Fact 4 where the integer approximation is no more needed. For $p < n/k$, instead, the bound of Fact 4 must be increased because it will not be possible to execute in parallel all the comparisons due in certain rounds. As an example we limit the calculation of these bounds to $k = 2$ as it is not particularly interesting.

▶ **Fact 5.** *For $n = 2^q$, and for $2^i \leq p < 2^{i+1}$ with $0 \leq i \leq q - 1$, $R(n, 2, 1, p) = 2^{q-i} + i - 1$ is an upper bound of R.*

Proof. Consider the tournament algorithm, where some rounds are divided into several consecutive sub-rounds if needed.
(1) The two limit cases $i = 0$ and $i = q - 1$ are immediate. For $i = 0$ we have $2^0 \leq p < 2^1$ and $R(n, 2, 1, 1) = 2^q - 1$, in fact only one comparison can be made in each round. For $i = q - 1$ we have $2^{q-1} \leq p < 2^q$ and $R(n, 2, 1, 1) = q$, so that the $\log_2 n$ rounds indicated in Fact 4 are executed.

(2) For a generic value of i with $0 < i < q-1$, some rounds of the tournament must be performed as a sequence of sub-rounds. In fact, the $2^{q-1}, 2^{q-2}, \ldots, 2^{i+1}$ comparisons needed in rounds $1, 2, \ldots, q-i-1$, must be divided into $2^{q-1-i}, 2^{q-2-i}, \ldots, 2^1$ sub-rounds played in the $p = 2^i$ facilities. The total number of these sub-rounds is $2^{q-i} - 2$, to which the number $q - (q-i-1) = i+1$ of undivided rounds must be added, and the bound $2^{q-i} + i - 1$ follows. ◀

For example, for $n = 2^5$ and $2^2 \leq p < 2^3$ we have $i = 2$ hence $R(2^5, 2, 1, 2^2) = 2^3+2-1 = 9$. Giving a matching lower bound is open.

We now study the best way to determine in parallel the first elements of the game for the values $n = 2^q$, $k = 2$, $p \geq n/2$, and t=2 (for $t = 1$ the problem is solved in Fact 3), and for $n = 3^q$, $k = 3$, $p \geq n/3$, and $t = 3$. We give bounds on the function R, and then on C when the former are satisfied. Then the solution for $k > 3$ and $t > 3$ is discussed as a conjecture.

3.1 The case k=2

We start computing $R(n, 2, 2, n/2)$ and then $C(n, 2, 2)$, for $n = 2^q$. Based on the considerations made above on the proof of Fact 4 we will use a parallel tournament to calculate the winner, adding new comparisons in different rounds to calculate the second too. Recall that in a binary tree representing a tournament, the vertices in each round (i.e. level of the tree) are labelled with match winners who are known at the end of the matches, so the results cannot be used before the next round. We have:

▶ **Theorem 3.** *$R(n, 2, 2, n/2) = \log_2 n + 1$ is an upper and a lower bound of R, for n power of two.*

Proof. *(i) Upper bound.* Based on the inductive application of the following reasoning. Consider the section of a tournament shown in figure 3-above (solid edges) where s is possible final winner, as it will be known after the comparison between p and s is made in round r_{i+2}. At round r_{i+1}, n, m are the possible candidates for the second position, where m is the maximum element among the ones in the sub-tournanent leaded by p int round r_i. A symmetric situation holds for the pair t, q in the sub-tournament lead by s in the round r_{i+1}. Comparisons n vs m and t vs q can be made in round r_{i+2} along with p vs s, and these three results will be usable from round ri_{i+3} onwards.

The same situation now occurs for the element s which is the winner of the round r_{i+2}, and will compete in the round r_{i+3} for the final victory with the winner of another section of the tournament not shown. The looser p at level r_{i+2}, and the maximum element in the sub-tournament lead by s in round r_{i+1}, will compete for second position. In the example provided, such a maximum element is q, as determined with the comparison q vs t in the round r_{i+2}. Note that the comparison n vs m in the round r_{i+2} is actually useless, but this was not predictable in that round where the outcome of s vs p was not known.

To see how our algorithm works, refer to figure 3-below. Rounds r_1 and r_2 are like the ones of a standard tournament for the winner ($t = 1$). Starting to round r_3 a subset of competitors are compared to determine the final winner, and for each of these comparisons, two more comparisons are created to determine the final runner-up as shown in the figure. The basis of the induction is shown in the first three rounds, where the runner-up competitors in each sub-tournament of round r_3 are the pairs of elements defeated by the current winner in rounds r_1 and r_2.

As usual the winner is decided in round $r_{\log_2 n}$ (r_4 in the example). In the same round two more comparisons are made to decide the two candidates for second place which, is decided in the next round $r_{\log_2 n+1}$. Note that in each round the number of comparisons is less than or equal to $p = n/2$ (see the next corollary).

(ii) Lower bound. To prove that $\log_2 +1$ rounds are needed note that, as we said before, the winner can be determined in $\log_2 n$ rounds only using a standard tournament or equivalent algorithm. Two candidates x, y for the first position are compared in the last round, and all others are divided into two subsets X, Y where so far no element of one has been compared directly or indirectly with an element of the other. Clearly the loser between x and y, say x, can be the runner-up, but to make this decision x must be compared to at least one element of Y, say z, since no element of Y except y has ever been compared to the elements of X. However the x vs z comparison cannot be done before knowing that x lost to y, so an additional round is required after the comparison x vs y. ◀

From the algorithm reported in Theorem 3 (i) we have with simple calculations:

▶ **Corollary 4.** *If the first and the second elements are determined in* $\log_2 n + 1$ *rounds,* $C(n, 2, 2) = 3n/2 - 2$ *is an upper bound of* C*, for* n *power of two.*

Note that the upper bound of Corollary 4 is higher than that of Fact 1, due to the requirement to proceed with a minimum number of parallel rounds. Finding a corresponding lower bound is an open problem.

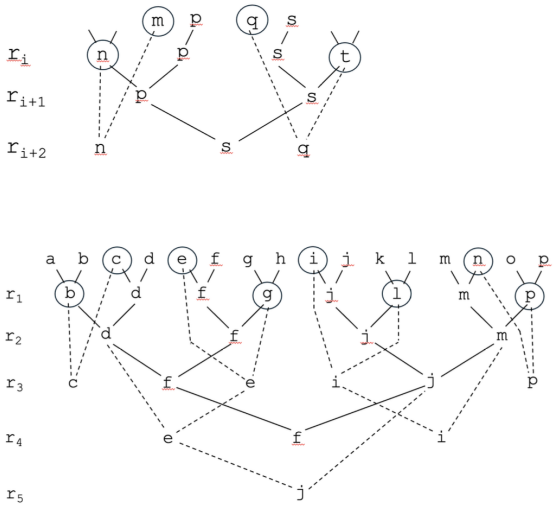

Figure 3 Above: Section of a scoreboard (solid edges). s is a possible winner. n, m and q, t are the pairs for a possible runner-up before the comparison between p and s is made. Below: Scoreboard of a tournament of 5 rounds for 2^4 competitors. f is the winner, and j is the runner-up.

For $p = n/2$ the comparisons required in each round of the algorithm in Theorem 3 are done in the round itself. Since it is not possible to select some participants to engage in further comparisons before the results of the current round are known, it is not necessary to allow $p > n/2$ parallel comparisons. For $p < n/2$, however, the initial rounds must be divided into parts and a new value for R must be determined as done in Fact 3. Let's leave out the boring and uninteresting calculations involved.

3.2 The case k=3 and beyond

We now compute $R(n, 3, 3, n/3)$ and then $C(n, 3, 3)$, for $n = 3^q$. Again we use a parallel tournament to calculate the winner, adding new comparisons in different rounds to calculate the second and the third too. For better understanding, in the scoreboard the vertices in

each round will be labeled $x - y - z$ according to the resulting order in the comparison among x, y, z (in particular the winner is shown in bold). Again this order is known at the end of the comparison, so the result cannot be used before the next round. The proof of the following theorem is an extension of the one given for Theorem 3. We have:

▶ **Theorem 5.** *$R(n, 3, 3, n/3) = \log_3 n + 2$ is an upper and a lower bound of R, for n power of three.*

Proof. *(i) Upper bound.* Consider the section of a tournament shown in figure 3-above. In each round the comparisons for the possible first, second, and third final elements are shown with thick-solid lines, thin-solid lines, and dashed lines, respectively. First turn your attention to the participants a to i in the left section of the figure. Once the result $d - a - g$ of round r_{i+1} is known, the candidates for the final second and third positions are limited to $e - a$ and $f - b - g$ respectively, and these two comparisons are scheduled for round r_{i+2}, where they will be carried out together with the comparison among d, z, j for the first place. This last comparison produces the ordering $z - d - j$, so now d can be at most the second finisher and will be considered for this position in the next round r_{i+3}, competing with v which emerges as possible second from the section of the tournament led by z in the previous round r_{i+1}.

The second position occupied by d in round r_{i+2} also has an important consequence on the competition for the third place. Indeed, the possible second e now becomes a possible third, and will compete with x and j which similarly emerge from the section of the tournament led by z in the round r_{i+1}. Note that no element in the tournament section led by j in round r_{i+1}, except j itself, can now compete for second or third position because j was third in the comparison $z - d - j$ in round r_{i+2}. Furthermore, the possible third f that emerged in round r_{i+2} in the comparison $f - b - g$ is now abandoned, and that comparison has no further consequences.

All in all, in each round $r_{j \geq i+1}$, for each comparison for the first position, there is one comparison for the second position and one for the third position in the round r_{j+1}. The winner is decided in round $r_{\log_3 n}$. In the same round four more comparisons are made, two of which to decide the two candidates for second place, and the other two to decide the three candidates for the third place. The holders of the second and third place will be respectively determined in the rounds $r_{\log_3 n+1}$ and $r_{\log_3 n+2}$ as shown in figure 4-below. Note that in each round the number of comparisons is less than or equal to $p = n/3$ (see the next corollary).

(ii) Lower bound. As in the proof of theorem 3, note that $\log_3 n$ rounds are needed for determining the winner (Fact 4), and this may happen only using a tournament or equivalent algorithm. Refer to figure 3. In the round $r_{\log_3 n}$ three candidates a, b, c remain for the first position and are compared in this round. All the others are divided into three subsets A, B, C where so far no element of one has been compared directly or indirectly with an element of the others. Suppose the comparison for first place results in $b - a - c$. Then b is the final winner, and the second place can be assigned to a or to at least one element, say v, belonging to B. Note that v cannot belong to C, as c was third in the comparison with b and a and all elements of C are inferior to c. So at least one comparison, say $a - v$, must be made in the round $r_{\log_3 n} + 1$ to assign second place. Clearly the winner a in this comparison is the second of the tournament and the loser v may be the third, but in this round there is at least another competitor for third place, coming from one of the three sections of the tournament led by a, b, and c in the round $r_{\log_3 n}$. So another round $r_{\log_3 n+2}$ is needed to award the third place. ◀

From the algorithm reported in Theorem 5 we have:

▶ **Corollary 6.** *If the first, second, and third elements are determined in* $\log_3 n + 1$ *rounds,* $C(n, 3, 3) = 13n/18 + 1/2$ *is an upper bound of* C*, for* n *power of three.*

Finding a corresponding lower bound for $C(n, 3, 3)$ is an open problem.

We can now speculate on competitions with k-ary comparisons, with $k > 3$. We believe that the algorithm presented in the upper-bound of the Theorems 3 and 5 can be immediately extended, adding comparisons from the third round onwards. In particular, adding binary, ternary, ... k-ary comparisons in each round to determine the candidates for the second, third, ... k-th place. Since the proof would probably be long and tedious, we pose the following:

Strong Conjecture. $R(n, t, t, n/t) = \log_t n + t - 1$ *is an upper and a lower bound of* R*, for* n *power of* t*.*

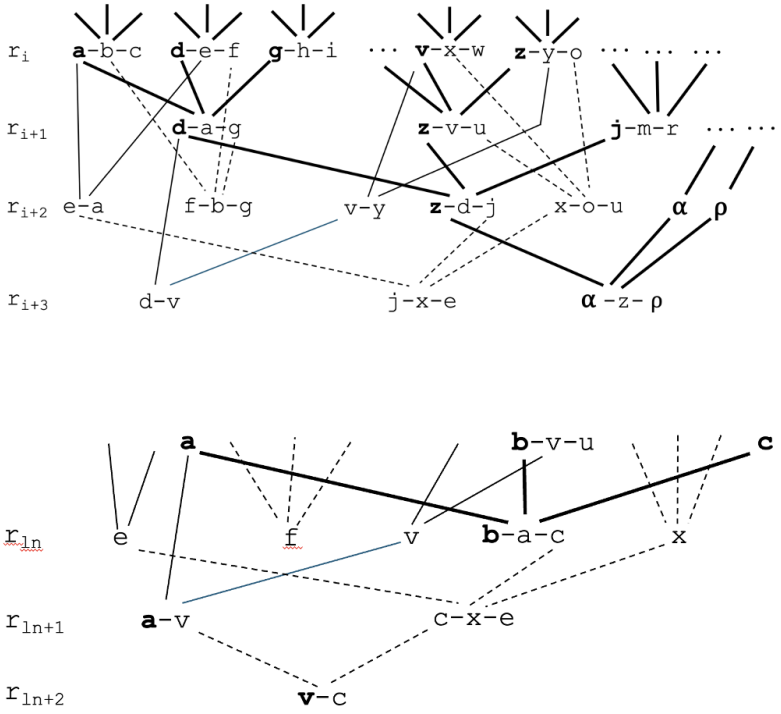

Figure 4 Above: Section of a scoreboard for 3^q competitors. Thick-solid, thin-solid, and dashed lines indicate comparisons for the first, second, and third place respectively. Below: The final three rounds.

References

1 Martin Aigner. Selecting the top three elements. *Discrete Apllied Math.*, 4:247–267, 1982.

2 Richard Beigel and John Gill. Sorting n objects with a k-sorter. *IEEE Transactions on Computers*, 39:714–716, 1990.

3 D.Kirkpatrick. *Closing a Long-standing Complexity Gap for Selection*, volume 8066. LNTCS Springer, 2013.

4 Charles L. Dodgson. The fallacies of lown tennis tournaments. *St. James Gazette*, 1:5–6, 1883.

5 Donald E.Knuth. *The Art of Computer Programming, Vol 3.* Addison Wesley, 1973.

6 Jutta Eusterbrock. Errata to "selecting the top three elements by m. aigner: A result of a computer-assisted proof search". *Discrete Applied Math.*, 41:131–137, 1993.

7 Susumu Horiguci and Willard L. Miranker. On parallel algorithm for finding the maximum value. *Parallel Computing*, 10:101–108, 1989.

8 Andras Ivanyi and Norbert Fogarasi. On partial sorting in restricted rounds. *Acta Univ. Sapientiae, Informatica*, 9:17–34, 2017.

9 Bruce Parker and Ian Parberry. Costructinfg sorting networks from k-sorters. *Information Processing Letteers*, 33:157–162, 1989.

10 Yo Shiloach and Uzi Vishkin. Finding the maximum, merging, and sorting in a parallel computational model. *Journal of Algorithms*, 2(88):88–102, 1981.