Selection Algorithms with Small Groups*

Ke Chen †   Adrian Dumitrescu‡

January 31, 2019

Abstract

We revisit the selection problem, namely that of computing the ith order statistic of n given elements, in particular the classic deterministic algorithm by grouping and partition due to Blum, Floyd, Pratt, Rivest, and Tarjan (1973). Whereas the original algorithm uses groups of odd size at least 5 and runs in linear time, it has been perpetuated in the literature that using smaller group sizes will force the worst-case running time to become superlinear, namely $\Omega(n \log n)$. We first point out that the usual arguments found in the literature justifying the superlinear worst-case running time fall short of proving this claim. We further prove that it is possible to use group size smaller than 5 while maintaining the worst case linear running time. To this end we introduce three simple variants of the classic algorithm, the repeated step algorithm, the shifting target algorithm, and the hyperpair algorithm, all running in linear time.

Keywords: median selection, ith order statistic, comparison algorithm.

1 Introduction

Together with sorting, selection is one of the most widely used procedures in computer algorithms. Indeed, it is easy to find numerous algorithms (documented in at least as many research articles) that use selection as a subroutine. Two classic examples from computational geometry are [24, 27].

Given a sequence A of n numbers (usually stored in an array), and an integer (target) parameter $1 \leq i \leq n$, the selection problem asks to find the ith smallest element in A. Sorting the numbers trivially solves the selection problem, but if one aims at a linear time algorithm, a higher level of sophistication is needed. A now classic approach for selection [7, 15, 20, 30, 33] from the 1970s is to use an element in A as a pivot to partition A into two smaller subsequences and recurse on one of them with a (possibly different) selection parameter i.

The time complexity of this kind of algorithms is sensitive to the pivots used. For example, if a good pivot is used, many elements in A can be discarded; whereas if a bad pivot is used, in the worst case, the size of the problem may be only reduced by a constant, leading to a quadratic worst-case running time. But choosing a good pivot can be time consuming.

Randomly choosing the pivots yields a well-known randomized algorithm with expected linear running time (see e.g., [8, Ch. 9.2], [25, Ch. 13.5], or [28, Ch. 3.4]), however its worst case running time is quadratic in n.

The first deterministic linear time selection algorithm SELECT (called PICK by the authors), in fact a theoretical breakthrough at the time, was introduced by Blum et al. [7]. By using the median

*A preliminary version of this paper appeared in the *Proceedings of the 29th International Symposium on Algorithms and Data Structures*, (WADS 2015), Victoria, Canada, August 2015, Vol. 9214 of LNCS, Springer, pp. 189–199.

†Department of Computer Science, University of Wisconsin–Milwaukee, USA. Email kechen@uwm.edu.

‡Department of Computer Science, University of Wisconsin–Milwaukee, USA. Email dumitres@uwm.edu.

of medians of small (constant size) disjoint groups of A, good pivots that guarantee reducing the size of the problem by a constant fraction can be chosen with low costs. The authors [7, page 451, proof of Theorem 1] required the group size to be at least 5 for the SELECT algorithm to run in linear time. It has been perpetuated in the literature the idea that SELECT with groups of 3 or 4 does not run in linear time: an exercise of the book by Cormen et al. [8, page 223, exercise 9.3-1] asks the readers to argue that "SELECT does not run in linear time if groups of 3 are used".

We first point out that the argument for the $\Omega(n \log n)$ lower bound in the solution to this exercise [9, page 23] is incomplete by failing to provide an input sequence with one third of the elements being discarded in each recursive call in both the current sequence and its sequence of medians; the difficulty in completing the argument lies in the fact that these two sequences are not disjoint thus cannot be constructed or controlled independently. The question whether the original SELECT algorithm runs in linear time with groups of 3 remains open at the time of this writing.

Further, we show that this restriction on the group size is unnecessary, namely that group sizes smaller than 5 can be used by a linear time deterministic algorithm for the selection problem. Since selecting the median in smaller groups is easier to implement and requires fewer comparisons (e.g., 3 comparisons for group size 3 versus 6 comparisons for group size 5), it is attractive to have linear time selection algorithms that use smaller groups. Our main result concerning selection with small group size is summarized in the following theorem.

Theorem 1. *There exist suitable variants of* SELECT *with groups of* 2*,* 3*, and* 4 *running in* $O(n)$ *time.*

Historical background. The interest in selection algorithms has remained high over the years with many exciting developments (e.g., lower bounds, parallel algorithms, etc) taking place; we only cite a few here [2, 6, 10, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 29, 32, 33]. We also refer the reader to the dedicated book chapters on selection in [1, 4, 8, 11, 25, 26] and the more recent articles [3, 23], including experimental work.

Outline. In Section 2, the classic SELECT algorithm is introduced (rephrased) under standard simplifying assumptions. In Section 3, we introduce a variant of SELECT, the *repeated step* algorithm, which runs in linear time with either group size 3 and 4. With groups of 3, the algorithm executes a certain step, "group by 3 and find the medians of the groups", twice in a row. In Section 4, we introduce another variant of SELECT, the *shifting target* algorithm, a linear time selection algorithm with group size 4. In each iteration, upper or lower medians are used based on the current rank of the target, and the shift in the target parameter i is controlled over three consecutive iterations. In Section 5, we introduce yet another variant of SELECT, the *hyperpair* algorithm, a linear time selection algorithm with group size 2. The algorithm performs the "group by pairs" step four times in a row to form hyperpairs. In Section 6, we briefly introduce three other variants of SELECT with group size 4, including one due to Zwick [34], all running in linear time.

In Section 7, we compare our algorithms (with group size 3 and 4) with the original SELECT algorithm (with group size 5) by deriving upper bounds on the exact numbers of comparisons used by each algorithm. We also present experimental results that verify our numeric calculations. In Section 8, we summarize our results and formulate a conjecture on the running time of the original SELECT algorithm from [7] with groups of 3 and 4, as suggested by our study.

2 Preliminaries

Without affecting the results, the following two standard simplifying assumptions are convenient: (i) the input sequence A contains n distinct numbers; and (ii) the floor and ceiling functions are omitted in the descriptions of the algorithms and their analyses. We also assume that all the grouping steps are carried out using the "natural" order, i.e., given a sequence $A = \{a_1, a_2, \ldots, a_n\}$, "arrange A into groups of size m" means that group 1 contains $a_1, a_2, \ldots, a_m$, group 2 contains $a_{m+1}, a_{m+2}, \ldots, a_{2m}$ and so on. Under these assumptions, SELECT with groups of 5 (from [7]) can be described as follows (using this group size has become increasingly popular, see e.g., [8, Ch. 9.2]):

1. If $n \leq 5$, sort A and return the ith smallest number.
2. Arrange A into groups of size 5. Let M be the sequence of medians of these $n/5$ groups. Select the median of M recursively, let it be m.
3. Partition A into two subsequences $A_1 = \{x|x < m\}$ and $A_2 = \{x|x > m\}$ (the order of elements is preserved). If $i = |A_1| + 1$, return m. If $i < |A_1| + 1$, go to step 1 with $A \leftarrow A_1$ and $n \leftarrow |A_1|$. If $i > |A_1| + 1$, go to step 1 with $A \leftarrow A_2$, $n \leftarrow |A_2|$ and $i \leftarrow i - |A_1| - 1$.

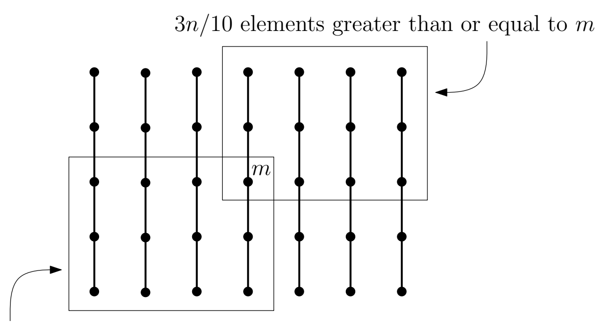

Figure 1: One iteration of the SELECT algorithm with group size 5. At least $3n/10$ elements can be discarded.

Denote the worst case running time of the recursive selection algorithm on an n-element input by $T(n)$. As shown in Figure 1, at least $3 * (n/5)/2 = 3n/10$ elements are discarded at each iteration, which yields the recurrence

$$T(n) \leq T(n/5) + T(7n/10) + O(n). \tag{1}$$

This recurrence is one of the following generic form:

$$T(n) \leq \sum_{i=1}^{k} T(a_i n) + O(n), \text{ where } a_i > 0 \text{ for } i = 1, \ldots, k \text{ and } \sum_{i=1}^{k} a_i \leq 1. \tag{2}$$

It is well-known [8, Ch. 4] (and can be verified by direct substitution) that the solution of (2) is

$$T(n) = \begin{cases} O(n) & \text{if } \sum_{i=1}^{k} a_i < 1, \\ O(n \log n) & \text{if } \sum_{i=1}^{k} a_i = 1. \end{cases} \tag{3}$$

As such, since the coefficients in (1) sum to $1/5 + 7/10 = 9/10 < 1$, we see that the original SELECT algorithm with group size 5 runs in $T(n) = \Theta(n)$ (as it is well-known).

3 The Repeated Step Algorithm

Using group size 3 directly in the SELECT algorithm in [7] yields

$$T(n) \le T(n/3) + T(2n/3) + O(n), \tag{4}$$

which solves to $T(n) = O(n \log n)$. Here a large portion (at least one third) of A is discarded in each iteration but the cost of finding such a good pivot is too high, namely $T(n/3)$. The idea of our *repeated step* algorithm, inspired by the algorithm in [5], is to find a weaker pivot in a faster manner by performing the operation "group by 3 and find the medians" twice in a row (as illustrated in Figure 2). It is worth noting that this method is akin to using the Tukey's ninther [31]. More precisely, M' as defined in step 3 below is the sequence formed by the Tukey's ninthers of groups of 9 elements in A.

Algorithm

1. If $n \le 3$, sort A and return the ith smallest number.
2. Arrange A into groups of size 3. Let M be the sequence of medians of these $n/3$ groups.
3. Arrange M into groups of size 3. Let M' be the sequence of medians of these $n/9$ groups.
4. Select the median of M' recursively, let it be m.
5. Partition A into two subsequences $A_1 = \{x | x < m\}$ and $A_2 = \{x | x > m\}$. If $i = |A_1| + 1$, return m. If $i < |A_1| + 1$, go to step 1 with $A \leftarrow A_1$ and $n \leftarrow |A_1|$. If $i > |A_1| + 1$, go to step 1 with $A \leftarrow A_2$, $n \leftarrow |A_2|$ and $i \leftarrow i - |A_1| - 1$.

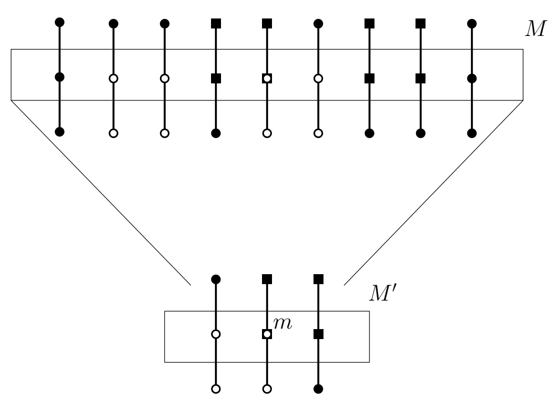

Figure 2: One iteration of the *repeated step* algorithm with groups of 3. Empty disks represent elements that are guaranteed to be smaller than or equal to m. Filled squares represent elements that are guaranteed to be greater than or equal to m.

Analysis. Since elements are discarded if and only if they are too large or too small to be the ith smallest element, the correctness of the algorithm is implied. Regarding the time complexity of this algorithm, we have the following lemma:

Lemma 1. *The repeated step algorithm with groups of* 3 *runs in* $\Theta(n)$ *time on an* n*-element input.*

Proof. By finding the median of medians of medians instead of the median of medians, the cost of selecting the pivot m reduces from $T(n/3) + O(n)$ to $T(n/9) + O(n)$. We need to determine how well m partitions A in the worst case. In step 4, m is guaranteed to be greater than or equal to $2 * (n/9)/2 = n/9$ elements in M. Each element in M is a median of a group of size 3 in A, so it is greater than or equal to 2 elements in its group. All the groups of A are disjoint, thus m is greater than or equal to $2n/9$ elements in A. Similarly, m is smaller than or equal to $2n/9$ elements in A. Thus, in the last step, at least $2n/9$ elements can be discarded. The recursive call in step 4 takes $T(n/9)$ time. So the resulting recurrence is

$$T(n) \leq T(n/9) + T(7n/9) + O(n),$$

and since the coefficients on the right side sum to $8/9 < 1$, by (3), we have $T(n) = \Theta(n)$, as required. □

Note that grouping by 3 twice and finding the median of medians of medians is different from grouping by 9 and finding the median of medians. The number of comparisons required for grouping by 3 twice is $3n/3 + 3n/9 = 12n/9$, while for grouping by 9 the number is $14n/9$ (14 comparisons for selecting the median of 9). The number of elements guaranteed to be discarded is also different: for grouping by 3 twice, at least $2n/9$ elements can be discarded, while for grouping by 9, this number is $5n/18$. So our method trades some of the quality of the pivots for speed (discards fewer elements than the median of 9 approach) by doing fewer comparisons.

4 The Shifting Target Algorithm

In the SELECT algorithm introduced in [7], the group size is restricted to odd numbers, where the median of a group has a privileged symmetric position. For group size 4, depending on the choice of upper, lower, or average median, there are three possible partial orders to be considered (see Figure 3).

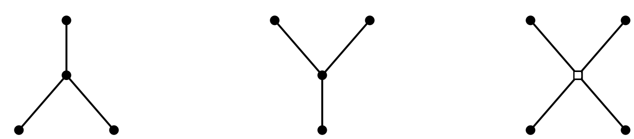

Figure 3: Three partial orders of 4 elements based on the upper (left), lower (middle), and average (right) medians. The empty square represents the average of the upper and lower median, which is not necessarily part of the 4-element sequence.

If the upper (or lower) median is always used, only $2 * (n/4)/2 = n/4$ elements are guaranteed to be discarded in each iteration (see Figure 4), which gives the recurrence

$$T(n) \leq T(n/4) + T(3n/4) + O(n). \tag{5}$$

The term $T(n/4)$ is for the recursive call to find the median of all $n/4$ medians. This recursion solves to $T(n) = O(n \log n)$. Even if we use the average of the two medians, the recursion remains the same since only 2 elements from each of the $(n/4)/2 = n/8$ groups are guaranteed to be discarded.

Observe that if the target parameter satisfies $i \leq n/2$ (resp., $i \geq n/2$), using the lower (resp., upper) median gives a better chance to discard more elements and thus obtain a better recurrence; detailed calculations are given in the proof of Lemma 2. Inspired by this idea, we propose the *shifting target* algorithm as follows:

Algorithm

1. If $n \le 4$, sort A and return the ith smallest number.
2. Arrange A into groups of size 4. Let M be the sequence of medians of these $n/4$ groups. If $i \le n/2$, the lower medians are used; otherwise the upper medians are used. Select the median of M recursively, let it be m.
3. Partition A into two subsequences $A_1 = \{x|x < m\}$ and $A_2 = \{x|x > m\}$. If $i = |A_1| + 1$, return m. If $i < |A_1| + 1$, go to step 1 with $A \leftarrow A_1$ and $n \leftarrow |A_1|$. If $i > |A_1| + 1$, go to step 1 with $A \leftarrow A_2$, $n \leftarrow |A_2|$ and $i \leftarrow i - |A_1| - 1$.

Analysis. Regarding the time complexity, we have the following lemma.

Lemma 2. *The shifting target algorithm with group size* 4 *runs in* $\Theta(n)$ *time on an* n*-element input.*

Proof. We shall prove that in at most three consecutive iterations, the size of the problem is reduced by a large enough fraction so that the resulting recurrence is of the form in (2) with $\sum_{i=1}^{k} a_i < 1$.

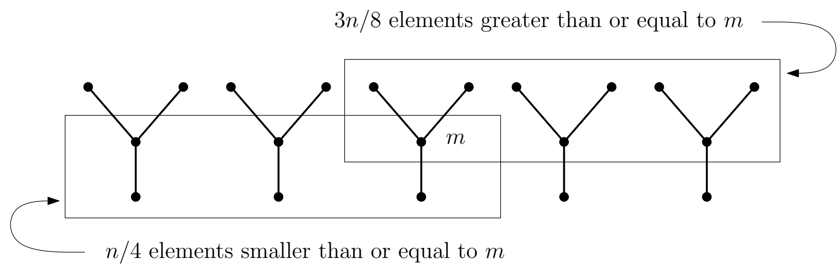

Figure 4: Group size 4 with lower medians used.

If in some iteration, we have $i \le n/4$, then the lower medians are used. Recall that m is guaranteed to be greater than or equal to $2 * (n/4)/2 = n/4$ elements of A. So either m is the ith smallest element in A or at least $3 * (n/4)/2 = 3n/8$ largest elements are discarded (see Figure 4), hence the worst-case running time recurrence is

$$T(n) \le T(n/4) + T(5n/8) + O(n). \tag{6}$$

Observe that in this case the coefficients on the right side sum to $7/8 < 1$, yielding a linear solution, as required.

Now consider the case $n/4 < i \le n/2$, again the lower medians are used. If $|A_1| \ge i$, i.e., the rank of m is higher than i, again at least $3 * (n/4)/2 = 3n/8$ largest elements are discarded and (6) applies. Otherwise, suppose that only $t = |A_1| \ge 2 * (n/4)/2 = n/4$ smallest elements are discarded. Then in the next iteration, $i' = i - t$, $n' = n - t$.

If $i' \le n'/4$, at least $3n'/8$ elements are discarded. The first iteration satisfies recurrence (5) and we can use recurrence (6) to bound the term $T(3n/4)$ from above. We deduce that in two iterations the worst case running time satisfies the recurrence:

$$\begin{aligned} T(n) &\le T(n/4) + T(3n/4) + O(n) \\ &\le T(n/4) + T((3n/4)/4) + T((3n/4) * 5/8) + O(n) \\ &= T(n/4) + T(3n/16) + T(15n/32) + O(n). \end{aligned} \tag{7}$$

Observe that the coefficients on the right side sum to $29/32 < 1$, yielding a linear solution, as required. Subsequently, we can therefore assume that $i' \geq n'/4$. We have

$$\begin{aligned} i'/n' &= (i-t)/(n-t) \leq (i-n/4)/(n-n/4) \\ &\leq (n/2-n/4)/(n-n/4) = 1/3. \end{aligned}$$

Since $1/4 < i'/n' \leq 1/3 \leq 1/2$, the lower medians will be used. As described above, if at least $3n'/8$ largest elements are discarded, in two iterations, the worst case running time satisfies the same recurrence (7).

So suppose that only $t' \geq 2 * (n'/4)/2 = n'/4$ smallest elements are discarded. Let $i'' = i' - t'$, $n'' = n' - t'$. We have

$$\begin{aligned} i''/n'' &= (i'-t')/(n'-t') \leq (i'-n'/4)/(n'-n'/4) \\ &\leq (n'/3-n'/4)/(n'-n'/4) = 1/9. \end{aligned}$$

Since $i''/n'' \leq 1/9 < 1/4$, in the next iteration, at least $3n''/8$ elements will be discarded. The first two iterations satisfy recurrence (5) and we can use recurrence (6) to bound the term $T(9n/16)$ from above. We deduce that in three iterations the worst case running time satisfies the recurrence:

$$\begin{aligned} T(n) &\leq T(n/4) + T(3n/4) + O(n) \\ &\leq T(n/4) + T((3n/4)/4) + T((3n/4) * 3/4) + O(n) \\ &= T(n/4) + T(3n/16) + T(9n/16) + O(n) \\ &\leq T(n/4) + T(3n/16) + T((9n/16)/4) + T((9n/16) * 5/8) + O(n) \\ &= T(n/4) + T(3n/16) + T(9n/64) + T(45n/128) + O(n). \end{aligned}$$

The sum of the coefficients on the right side is $119/128 < 1$, so again by (3), the solution is $T(n) = \Theta(n)$.

By symmetry, the analysis also holds for the case $i \geq n/2$, and the proof of Lemma 2 is complete. □

5 The Hyperpair Algorithm

For completeness, we consider the ultimate group size 2, i.e., each group contains a pair of elements. The upper (resp. lower) median of a pair is the larger (resp. smaller) element in that pair. In the original SELECT algorithm, if pairs were used, only $1 * (n/4)$ elements are guaranteed to be discarded in each iteration, which gives the recurrence

$$T(n) \leq T(n/2) + T(3n/4) + O(n). \tag{8}$$

The term $T(n/2)$ is for the recursive call to find the median of the $n/2$ upper (or lower) medians. However, the above recursion does not yield a solution linear in n. Now, one can make the following adjustment: instead of taking the median of half the input recursively, let the algorithm recursively compute the jth smallest element among the $n/2$ upper medians, where $j = n/6$. Then $2j = n/2 - j = n/3$ elements can be discarded in each iteration, thus the size of the largest remaining recursive call is $n - n/3 = 2n/3$. However, even with this adjustment, the resulting recurrence (9) does not yield a solution linear in n.

$$T(n) \leq T(n/2) + T(2n/3) + O(n). \tag{9}$$

The key for obtaining a linear running time in this setting seems to be to use groups of 2 in a repeated manner. The following algorithm has the same flavor as the repeated step algorithm in section 3 but uses group size 2. Its name, the *hyperpair* algorithm, will be justified in the analysis.

Algorithm

1. If $n \leq 2$, sort A and return the ith smallest number.
2. Arrange A into groups of size 2. Let M_1 be the sequence of upper medians of these $n/2$ pairs.
3. Arrange M_1 into pairs. Let M_2 be the sequence of lower medians of these $n/4$ pairs.
4. Arrange M_2 into pairs. Let M_3 be the sequence of upper medians of these $n/8$ pairs.
5. Arrange M_3 into pairs. Let M_4 be the sequence of lower medians of these $n/16$ pairs.
6. Select the median of M_4 recursively, let it be m.
7. Partition A into two subsequences $A_1 = \{x|x < m\}$ and $A_2 = \{x|x > m\}$. If $i = |A_1| + 1$, return m. If $i < |A_1| + 1$, go to step 1 with $A \leftarrow A_1$ and $n \leftarrow |A_1|$. If $i > |A_1| + 1$, go to step 1 with $A \leftarrow A_2$, $n \leftarrow |A_2|$ and $i \leftarrow i - |A_1| - 1$.

Analysis. In order to calculate the time complexity of this algorithm, we need to estimate how well m partitions the sequence A. Observe that steps 2–5 can be viewed as constructing *hyperpairs*, as in the non-recursive selection algorithm of Schönhage et al. [30]. In their definition, a single element is a hyperpair with itself as the *center*; given two disjoint copies of a hyperpair, we can combine them to form a larger hyperpair by comparing their centers and taking the upper or lower of these as the new center. The hyperpairs P constructed in our algorithm are illustrated in Figure 5. Observe that in P, three elements are guaranteed to be greater than its center c and

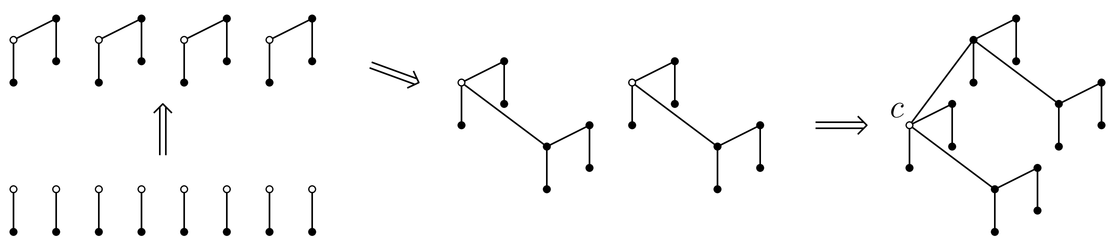

Figure 5: Construction of a hyperpair P with 16 elements; the center of each hyperpair is marked by an empty circle.

three are guaranteed to be smaller than c. We are now ready to establish the time complexity of this algorithm:

Lemma 3. *The hyperpair algorithm runs in $\Theta(n)$ time on an n-element input.*

Proof. Steps 2–5 take $n/2 + n/4 + n/8 + n/16 = 15n/16$ comparisons to form the hyperpairs P. The pivot m is the median of the centers of these $n/16$ hyperpairs. So the cost of selecting the pivot is $T(n/16) + 15n/16$. By the above observation about the center c of P, m is guaranteed to be greater than or equal to $4 * (n/16)/2 = n/8$ elements in A. Similarly, m is guaranteed to be smaller than or equal to $n/8$ elements in A. Thus, in the last step, at least $n/8$ elements can be discarded. The resulting recurrence is

$$T(n) \leq T(n/16) + T(7n/8) + O(n),$$

and since the coefficients on the right side sum to $15/16 < 1$, by (3), we have $T(n) = \Theta(n)$, as required. □

Note that larger hyperpairs can also be used to obtain linear-time algorithms. If the "group into pairs" step is repeated $2k$ times, $k \geq 2$, where upper and lower medians are used alternatively, then $n/2^{2k}$ hyperpairs of size 2^{2k} are built. Each center is guaranteed to be greater than or equal to 2^k elements in its hyperpair and is also guaranteed to be smaller than or equal to 2^k elements in its hyperpair. So using the median of these centers as pivot, at least $2^k * \left(n/2^{2k}\right)/2 = n/2^{k+1}$ elements can be discarded. The resulting recurrence is

$$T(n) \leq T\left(n/2^{2k}\right) + T\left(\left(1 - 1/2^{k+1}\right)n\right) + O(n),$$

where the $O(n)$ term involves $\sum_{j=1}^{2k} n/2^j = n - n/2^{2k}$ comparisons to build the hyperpairs and at most n comparisons to partition the sequence. Since the coefficients on the right side sum to $1 - \left(2^{k-1} - 1\right)/2^{2k} < 1$, by (3), we have $T(n) = \Theta(n)$.

6 Other Variants

A similar idea of repeating the group step (from Section 3) also applies to the case of groups of 4 and yields

$$T(n) \leq T(n/16) + T(7n/8) + O(n),$$

and thereby another linear time selection algorithm with group size 4.

A hybrid algorithm. Yet another variant of SELECT with group size 4 (we refer to it as the hybrid algorithm), can be obtained by using the ideas of both algorithms together, i.e., repeat the grouping by 4 step twice in a row while M contains the lower medians and M' contains the upper medians (or vice versa). Recursively selecting the median m of M' takes time $T(n/16)$. Notice that m is greater than or equal to $3 * (n/16)/2 = 3n/32$ elements in M of which each is greater than or equal to 2 elements in its group in A. So m is greater than or equal to $3n/16$ elements of A. Also, m is smaller than or equal to $2 * (n/16)/2 = n/16$ elements in M of which each is smaller than or equal to 3 elements in its group of A. So m is smaller than or equal to $3n/16$ elements of A, thus the resulting recurrence is

$$T(n) \leq T(n/16) + T(13n/16) + O(n),$$

again with a linear solution, as desired.

Zwick's variant. The fact that the SELECT algorithm can be modified so that it works with groups of 4 in linear time was observed prior to this writing. The following variant, from 2010, is due to Zwick [34]. Split the elements of A into quartets. Find the 2nd smallest element of each quartet (i.e., the lower median), and let M be this subset of $n/4$ elements. Recursively find the $(3/5)(n/4)$th smallest element m of M. Now $(3/5)(n/4)$ groups of A have 2 elements smaller than or equal to m, so m is greater than or equal to $2(3/5)(n/4) = 3n/10$ elements in A. Similarly, $(2/5)(n/4)$ groups of A have 3 elements greater than or equal to m, so m is smaller than or equal to $3(2/5)(n/4) = 3n/10$ elements in A. Thus, the remaining recursive call involves at most $7n/10$ elements, and the resulting recurrence is

$$T(n) \leq T(n/4) + T(7n/10) + O(n).$$

Since $1/4 + 7/10 < 1$, the solution is linear.

7 Comparison of the Algorithms and Experimental Results

To compare our algorithms with the original SELECT algorithm, we first derive upper bounds on the exact numbers of comparisons for each variant in the same manner as in Section 2 of [7]. It should be noted that all recurrent formulas and all proofs do not provide (nor aim to provide) tight bounds or expected number of comparisons. Tighter analytical bounds might exist than those shown. Let now $T(n)$ denote the total number of comparisons performed. For the original SELECT algorithm with group size 5, we have

$$T(n) \leq T(n/5) + T(7n/10) + 6n/5 + n,$$

in which the term $6n/5$ is for computing the $n/5$ medians (each takes at most 6 comparisons) and the term n is for partitioning the sequence around the selected pivot. Solving the recurrence yields $T(n) \leq 22n$. Similarly, for the repeated step algorithm, we have

$$T(n) \leq T(n/9) + T(7n/9) + 3n/3 + 3n/9 + n,$$

and consequently, $T(n) \leq 21n$. For the hybrid algorithm, we have

$$T(n) \leq T(n/16) + T(13n/16) + 4n/4 + 4n/16 + n,$$

and consequently, $T(n) \leq 18n$. For Zwick's algorithm, we have

$$T(n) \leq T(n/4) + T(7n/10) + 4n/4 + n,$$

and consequently, $T(n) \leq 40n$. For the hyperpair algorithm, we have

$$T(n) \leq T(n/16) + T(7n/8) + 15n/16 + n,$$

and consequently, $T(n) \leq 31n$. For the shifting target algorithm, the analysis is more involved; it yields $T(n) \leq 66n$.

We note that sharper upper bounds are possible by taking extra care in avoiding comparisons with known outcomes against the pivot; however, for simplicity of implementation we opted to forego this saving. In order to avoid the overhead of repeated array copying, all the algorithms were implemented in-place, in the sense that, with the exception of the recursion, only $O(1)$ extra space is used in addition to the input array. This requires minor modifications of the algorithms; however, their running time analyses remain unchanged. We carried out 1000 experiments[1] on selecting medians in arrays of 10 million randomly permuted distinct integers. The results are summarized in Table 1.

We observed that the experimental results agree with the worst-case estimates in the number of comparisons, in the sense that they show roughly the same speed ranking. One reason why the experimental speed ranking does not fully match the analytical bounds derived is the existence of other operations performed during the selection process that are unaccounted for by the recurrences, such as data copying (shown in the last two columns of the table as swaps). It is worth noting that optimizations introduced in Section 3 of [7], or others discussed in [3], may be used to reduce the multiplicative constant factors.

[1] The experiments were performed on a desktop with 64bits operating system, 7.8GB memory and Intel® Core™ i7-2600 3.4GHz processor. The C code used can be downloaded at `https://drive.google.com/file/d/0B7USj6ZPkysnMjNwV014RDJGMWc/view?usp=sharing`.

Algorithm	Group	Upper Bound	Average Time	Comparisons		Swaps	
				Average	Max	Average	Max
Hybrid	4	$18n$	364.3ms	4.1	4.2	1.2	1.2
Repeated step	3	$21n$	446.9ms	4.3	4.4	1.8	1.8
Original	5	$22n$	468.9ms	5.7	5.8	1.5	1.5
Hyperpair(4)	2	$31n$	480.6ms	2.9	2.9	3.0	3.0
Zwick's	4	$40n$	541.1ms	6.3	6.3	2.0	2.0
Shifting target	4	$66n$	558.0ms	6.6	6.7	2.0	2.1
Original	4	$O(n \log n)$	560.2ms	6.7	6.7	2.0	2.0
Original	3	$O(n \log n)$	813.4ms	8.2	8.5	3.4	3.5
Hyperpair(6)	2	$127n/3$	452.4ms	2.8	2.8	2.8	2.8
Hyperpair(8)	2	$73n$	456.0ms	2.8	2.8	2.8	2.9
Hyperpair(10)	2	$2047n/15$	458.8ms	2.9	2.9	2.9	2.9

Table 1: Experimental results. The last four columns are values per element. The numbers in parentheses for the hyperpair algorithms indicate the numbers of times the "group into pairs" step is repeated. The "Upper Bound" column shows the leading term in the solution of the corresponding recurrence for the worst-case number of comparisons.

8 Conclusion

The question whether the original selection algorithm introduced in [7] (outlined in Section 2) runs in linear time with group size 3 and 4 remains unsettled. Although the recurrences

$$T(n) \leq T(n/3) + T(2n/3) + O(n), \text{ and}$$
$$T(n) \leq T(n/4) + T(3n/4) + O(n)$$

(see (4) and (5)) for its worst-case running time with these group sizes both solve to $T(n) = O(n \log n)$, we believe that they only give non-tight upper bounds on the worst case scenarios. In any case and against popular belief we think that $\Theta(n \log n)$ is *not* the answer in regard to the time complexity of selection with these group sizes:

Conjecture 1. *The* SELECT *algorithm introduced by Blum et al. [7] runs in $o(n \log n)$ time with groups of* 3 *or* 4*.*

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison–Wesley, Reading, Massachusetts, 1983.

[2] M. Ajtai, J. Komlós, W. L. Steiger, and E. Szemerédi, Optimal parallel selection has complexity $O(\log \log n)$, *Journal of Computer and System Sciences* **38(1)** (1989), 125–133.

[3] A. Alexandrescu, Fast deterministic selection, *Proceedings of the 16th International Symposium on Experimental Algorithms* (SEA 2017), June 2017, London, pp. 24:1–24:19.

[4] S. Baase, *Computer Algorithms: Introduction to Design and Analysis*, 2nd edition, Addison-Wesley, Reading, Massachusetts, 1988.

[5] S. Battiato, D. Cantone, D. Catalano, G. Cincotti, and M. Hofri, An efficient algorithm for the approximate median selection problem, *Proceedings of the 4th Italian Conference on Algorithms and Complexity* (CIAC 2000), LNCS vol. 1767, Springer, 2000, pp. 226–238.

[6] S. W. Bent and J. W. John, Finding the median requires $2n$ comparisons, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (STOC 1985), ACM, 1985, pp. 213–216.

[7] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, Time bounds for selection, *Journal of Computer and System Sciences* **7(4)** (1973), 448–461.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd edition, MIT Press, Cambridge, 2009.

[9] T. H. Cormen, C. Lee, and E. Lin, *Instructor's Manual*, to accompany *Introduction to Algorithms*, 3rd edition, MIT Press, Cambridge, 2009.

[10] W. Cunto and J. I. Munro, Average case selection, *Journal of ACM* **36(2)** (1989), 270–279.

[11] S. Dasgupta, C. Papadimitriou, and U. Vazirani, *Algorithms*, Mc Graw Hill, New York, 2008.

[12] D. Dor, J. Håstad, S. Ulfberg, and U. Zwick, On lower bounds for selecting the median, *SIAM Journal on Discrete Mathematics* **14(3)** (2001), 299–311.

[13] D. Dor and U. Zwick, Finding the αnth largest element, *Combinatorica* **16(1)** (1996), 41–58.

[14] D. Dor and U. Zwick, Selecting the median, *SIAM Journal on Computing* **28(5)** (1999), 1722–1758.

[15] R. W. Floyd and R. L. Rivest, Expected time bounds for selection, *Communications of ACM* **18(3)** (1975), 165–172.

[16] F. Fussenegger and H. N. Gabow, A counting approach to lower bounds for selection problems, *Journal of ACM* **26(2)** (1979), 227–238.

[17] W. Gasarch, W. Kelly, and W. Pugh, Finding the ith largest of n for small i, n, *SIGACT News* **27(2)** (1996), 88–96.

[18] A. Hadian and M. Sobel, Selecting the t-th largest using binary errorless comparisons, *Combinatorial Theory and Its Applications* **4** (1969), 585–599.

[19] C. A. R. Hoare, Algorithm 63 (PARTITION) and algorithm 65 (FIND), *Communications of the ACM* **4(7)** (1961), 321–322.

[20] L. Hyafil, Bounds for selection, *SIAM Journal on Computing* **5(1)** (1976), 109–114.

[21] J. W. John, A new lower bound for the set-partitioning problem, *SIAM Journal on Computing* **17(4)** (1988), 640–647.

[22] D. G. Kirkpatrick, A unified lower bound for selection and set partitioning problems, *Journal of ACM* **28(1)** (1981), 150–165.

[23] D. G. Kirkpatrick, Closing a long-standing complexity gap for selection: $V_3(42) = 50$, in *Space-Efficient Data Structures, Streams, and Algorithms – Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday* (A. Brodnik, A. López-Ortiz, V. Raman, and A. Viola, editors), LNCS vol. 8066, Springer, 2013, pp. 61–76.

[24] D. G. Kirkpatrick and R. Seidel, The ultimate planar convex hull algorithm?, *SIAM Journal on Computing* **15(1)** (1986), 287–299.

[25] J. Kleinberg and É. Tardos, *Algorithm Design*, Pearson & Addison–Wesley, Boston, Massachusetts, 2006.

[26] D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, 2nd edition, Addison–Wesley, Reading, Massachusetts, 1998.

[27] N. Megiddo, Partitioning with two lines in the plane, *Journal of Algorithms* **6(3)** (1985), 430–433.

[28] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, 2005.

[29] M. Paterson, Progress in selection, *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory (SWAT 1996)*, LNCS vol. 1097, Springer, 1996, pp. 368–379.

[30] A. Schönhage, M. Paterson, and N. Pippenger, Finding the median, *Journal of Computer and System Sciences* **13(2)** (1976), 184–199.

[31] J. W. Tukey, The ninther, a technique for low-effort robust (resistant) location in large samples, *Contributions to Survey Sampling and Applied Statistics* (1978), 251–257.

[32] A. Yao and F. Yao, On the average-case complexity of selecting the kth best, *SIAM Journal on Computing* **11(3)** (1982), 428–447.

[33] C. K. Yap, New upper bounds for selection, *Communications of the ACM* **19(9)** (1976), 501–508.

[34] U. Zwick, Personal communication, Sept. 2014.