Select with Groups of 3 or 4 Takes Linear Time

Ke Chen * Adrian Dumitrescu†

September 29, 2014

Abstract

We revisit the selection problem, namely that of computing the ith order statistic of n given elements, in particular the classical deterministic algorithm by grouping and partition due to Blum, Floyd, Pratt, Rivest, and Tarjan (1973). While 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 groups of 3 or 4 will force the worst-case running time to become superlinear, namely $\Omega(n \log n)$. We first point out that the arguments existent 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 3 or 4 while maintaining the worst case linear running time. To this end we introduce two simple variants of the classical algorithm, the repeated step algorithm and the shifting target algorithm, both 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 procedure in computer algorithms. Indeed, it is easy to find hundreds if not thousands of algorithms (documented in at least as many research articles) that use selection as a subroutine. A classical example is [23].

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. Trivially sorting solves the selection problem, but if one aims at a linear time algorithm, a higher level of sophistication is needed. A now classical approach for selection [6, 14, 17, 26, 28] 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; while 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., [7, Ch. 9.2], [21, Ch. 13.5], or [24, 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. [6]. By using the median of medians of small (constant size) disjoint groups of A, good pivots that guarantee reducing the

*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.

size of the problem by a constant fraction can be chosen with low costs. The authors [6, 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. [7, 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 [8, 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 3 or 4 can be used to obtain a deterministic linear time 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* 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, 5, 9, 11, 12, 13, 14, 15, 16, 18, 19, 25, 27, 28]. We also refer the reader to the dedicated book chapters on selection in [1, 3, 7, 10, 21, 22] and the recent article [20].

Outline. In Section 2, the classical 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 both 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 conclude with some remarks and a conjecture on the running time of the original SELECT algorithm from [6] with groups of 3 and 4.

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. Under these assumptions, SELECT with groups of 5 (from [6]) can be described as follows (using this group size has become increasingly popular, see e.g., [7, 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 $A_2 = \{x|x > m\}$ (order of other elements is preserved). If $|A_1| = i - 1$, return m. Otherwise if $|A_1| > i - 1$, go to step 1 with $A \leftarrow A_1$ and $n \leftarrow |A_1|$. Otherwise go to step 1 with $A \leftarrow A_2$, $n \leftarrow |A_2|$ and $i \leftarrow i - |A_1| - 1$.

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 $(n/5)/2 * 3 = 3n/10$ elements are discarded at each iteration, which yields the recurrence

$$T(n) \leq T(n/5) + T(7n/10) + O(n).$$

Since the coefficients sum to $1/5 + 7/10 = 9/10 < 1$, the recursion solves to $T(n) = \Theta(n)$ (as it is well-known).

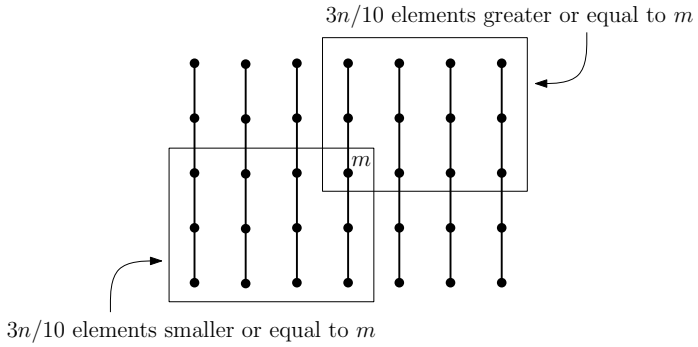

Figure 1: One iteration of the SELECT algorithm with group size 5. At least $3n/10$ elements can be discarded.

3 The Repeated Step Algorithm

Using group size 3 directly in the SELECT algorithm in [6] yields

$$T(n) \leq T(n/3) + T(2n/3) + O(n), \tag{1}$$

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 [4], 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).

Algorithm

1. If $n \leq 3$, sort A and return the it 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 $|A_1| = i - 1$, return m. Otherwise if $|A_1| > i - 1$, go to step 1 with $A \leftarrow A_1$ and $n \leftarrow |A_1|$. Otherwise go to step 1 with $A \leftarrow A_2$, $n \leftarrow |A_2|$ and $i \leftarrow i - |A_1| - 1$.

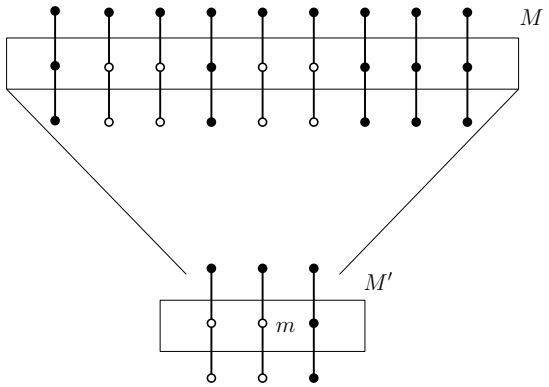

Figure 2: One iteration of the *repeated step* algorithm with groups of 3. Empty disks represent elements that are guaranteed to be smaller or equal to m.

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 follows. 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 or equal to $(n/9)/2 * 2 = n/9$ elements in M. Each element in M is a median of a group of size 3 in A, so it is greater or equal to 2 elements in its group. All the groups of A are disjoint, thus m is at least greater or equal to $2n/9$ elements in A. Similarly, m is at least smaller 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 $8/9 < 1$, we have $T(n) = \Theta(n)$, as required. □

4 The Shifting Target Algorithm

In the SELECT algorithm introduced in [6], the group size is restricted to odd numbers in order to avoid the calculation of the average of the upper and lower median. For group size of 4, depending on the choice of upper, lower or average median, there are three possible partial orders to be considered (see Figure 3).

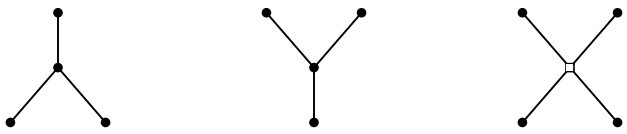

Figure 3: Three partial orders of 4 elements based on the upper (left), lower (middle) and average (right) medians. The empty square represents the average of the upper and lower median which is not necessarily part of the 4-element sequence.

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

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

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 $n \leq 4$, 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 $i \leq n/2$, 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 $|A_1| = i - 1$, return m. Otherwise if $|A_1| > i - 1$, go to step 1 with $A \leftarrow A_1$ and $n \leftarrow |A_1|$. Otherwise go to step 1 with $A \leftarrow A_2$, $n \leftarrow |A_2|$ and $i \leftarrow i - |A_1| - 1$.

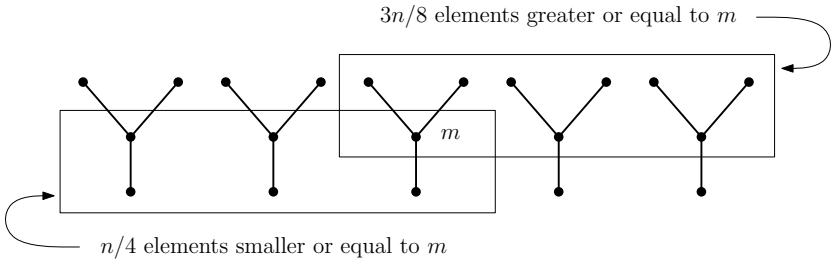

Figure 4: Group size 4 with lower medians used.

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

$$T(n) \leq T(n/4) + T(5n/8) + O(n). \tag{3}$$

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

If $i' \leq n'/4$, at least $3n'/8$ numbers are discarded. The first iteration satisfies recurrence (2) and we can use recurrence (3) to bound the term $T(3n/4)$ from above. We deduce that in two iterations the worst case running time satisfies the recurrence:

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

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 numbers are discarded, in two iterations, the worst case running time satisfies the same recurrence (4).

So suppose that only $t' \geq (n'/4)/2 * 2 = n'/4$ smallest numbers 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 $i''/n'' < 1/4$, in the next iteration, at least $3n''/8$ numbers will be discarded. The first two iterations satisfy recurrence (2) and we can use recurrence (3) 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 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 Concluding Remarks

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.

Yet another variant of SELECT with group size 4 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 or equal to at least $(n/16)/2 * 3 = 3n/32$ elements in M of which each is greater or equal to 2 elements in its group in A. So m is greater or equal to at least $3n/16$ elements of A. Also, m is smaller or equal to at least $(n/16)/2 * 2 = n/16$ elements in M of which each is smaller or equal to 3 elements in its group of A. So m is smaller or equal to at least $3n/16$ 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.

Update. The fact that the SELECT algorithm can be modified so that it works with groups of 4 in linear time was observed prior to our writing of this note. The following variant, from 2010, is due to Zwick [29]. Split the elements in 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 or equal to m, so m is greater or equal to at least $2(3/5)(n/4) = 3n/10$ elements in A. Similarly, $(2/5)(n/4)$ groups of A have 3 elements greater or equal to m, so m is smaller or equal to at least $3(2/5)(n/4) = 3n/10$ 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.

Final comment. The question whether the original selection algorithm introduced in [6] (outlined in Section 2) runs in linear time with group size 3 and 4 remains open. 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 (1) and (2)) 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:

Conjecture 1. *The* SELECT *algorithm introduced by Blum et al. [6] 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] S. Baase, *Computer Algorithms: Introduction to Design and Analysis*, 2nd edition, Addison-Wesley, Reading, Massachusetts, 1988.

[4] 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.

[5] 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.

[6] 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.

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

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

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

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

[11] 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.

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

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

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

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

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

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

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

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

[20] 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.

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

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

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

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

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

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

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

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

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