Analysis of An Approximate Median Selection Algorithm

Domenico Cantone
Università di Catania, Dipartimento di Matematica e Informatica,
Viale A. Doria 6, I-95125 Catania, Italy
e-mail:cantone@dmi.unict.it

Micha Hofri
Department of Computer Science, WPI
100 Institute Road, Worcester MA 01609-2280
e-mail:hofri@cs.wpi.edu

August 10, 2006

Abstract

We present analysis of an efficient algorithm for the approximate median selection problem that has been rediscovered many times, and easy to implement. The contribution of the article is in precise characterization of the accuracy of the algorithm. We present analytical results of the performance of the algorithm, as well as experimental illustrations of its precision.[*]

1. Introduction

In this paper we present an efficient algorithm for the approximate median selection problem, and its analysis. The algorithm can be used on data in an array, and it works then *in-place*, requiring no extra space. It can be used to process a read-once stream of values, and then, by the time n items have been processed, the amount of storage it needs is in $\Theta(\log n)$.

The algorithm is not new, we found. In fact, it seems to have been rediscovered many times. Rousseeuw and Bassett exclaim in [12] that each of them discovered it independently, and several other expositions with the same basic idea have been published. The earliest sources for it we have found are [13] and [14]. Our contribution is in advancing its analysis beyond what has been shown so far.

[*]An early version of the work was presented in CIAC 2000 Rome, Italy, [1]

The usefulness of such an algorithm is evident for all applications where it is sufficient to find an approximate median, for example in some heap-sort variants (cf. [11], [7], [2]), for regression in $L1$ metric, or for median-filtering in image representation. Several such applications are described in some detail in [12]. A different type of applications is the planning of database queries; the very thorough [9] gives an interesting view of such needs, and has further references.

While we discuss the algorithm to some extent, the main interest and the focus in the paper is on its analysis, and the implied engineering decisions. The only earlier analyses we know of are in [12], and to some extent [4], and their point of view is rather different from ours. Certain comments in [14] suggest that some analysis was done, but none is given. In addition, the analysis of the precision is largely new, and of independent interest. The analysis sheds light on the merits of the various possible settings of the main design parameter of the algorithm, the size b of the subsets of which it finds the true median, on the way to produce its approximate median of all the data.

Most discussions of the algorithm in the literature refer to it as a method to estimate the median of a distribution underlying the data. We adopt the more immediate objective of finding the item in a given set which is the median: the number of elements in the set which are smaller than it is, and the number that are larger than it is are equal (to within one, for an even set size). This introduces no distributional assumptions or concerns about independence. Having said that, for the asymptotic analysis we show that we can and need to use meaningfully such assumptions.

In Section 2 we present the algorithm. Section 3 provides a perfunctory analysis of its run-time. In Section 4 we establish the soundness of the method. To do so we present a probabilistic analysis of the precision of its median selection, providing both precise (and ultimately intractable), and asymptotic versions. Section 5 provides computational results for refinements which are beyond our analyses. Section 6 concludes the paper with suggested directions for additional research.

2. The Algorithm

We distinguish two cases, when the input is in an array in main storage, or when the algorithm receives the data one entry at a time (and then it need not be aware of the amount it will process).

In the first case the algorithm works *in situ*, and for efficiency may perform minor changes of the order of the data: it swaps the selected b-median with the element in the middle position (if necessary), so that at every stage the elements that are still candidates are equally spaced, without using any extra storage. In the second type, the algorithm requires a sequence of arrays of b positions. By the time it has processed n entries it has required $\log_b n$ arrays. We omit here the machinery of creating new arrays as the process continues.

Typical values for b are small odd numbers, such as 3,5,7....

```
set **size** to $n$ and **step** to $1$;
while **size** > 1
  set m to $\lfloor \textbf{size}/b \rfloor$;
  find $b$-median in m blocks, of $b$
  elements separated by **step**;
  place the median in the middle
     position of the block.
  set **size** to $\lfloor \textbf{size}/b \rfloor$ and **step**
  to $b\times$ **step**;
```

algo

```
while new input $t$ is available
   set $i=0$
A: insert $t$ in $B_i$
   if $B_i$ is full
     set $t$ to median($B_i$)
     clear $B_i$
     increment $i$
     continue at A
```

Figure 2: Approximate median selection: in an array and over an input stream

We also omit in both cases, except a brief mention, the activities needed for an input size which is not an integral power of b. For the first algorithm, this is a minor issue: it deals with $b\lfloor (n/b) \rfloor$ entries in the first pass, discarding at most $b-1$. If we do this at every pass the maximal 'loss' is $(b-1)\log_b n$, which is typically minor, but with some careful attention to detail, most such elements can be regrouped, for a much smaller loss. This is especially important at the last few phases. One way to avoid the difficulty is suggested later, and that is to stop the process as soon as **size** reaches a threshold value t, which can be several times larger than b, and find the exact median of the remaining terms, possibly using Hoare's QUICK-SELECT. If this number is even, say $2k$, we would choose between k and $k+1$ with equal likelihood.

When the algorithm processes a stream of data, and the input ends with some of the bases loaded, the situation is different: it is the same number as above, but those values in the high order arrays represent a large amount of input. A simple approach is to give each value in array B_i a relative weight of b^i, and find the 'weighted' median of the set, possibly adapting the above QUICK-SELECT. The small size of the set guarantees that the extra work, even with the additional bookkeeping required for the weights, is negligible compared with the main pass.

An earlier version of the work, presented in [1], went into greater detail on the efficient implementation of the array algorithm.

3. Performance Analysis

We discuss the performance costs assuming we use b-medians, and then look at the numbers for the first few odd integers as candidates for b. The performance costs are as usual in space and time, and for this algorithm the space costs are one of its strengths: they are quite modest.

3.1 Space requirements

The array processing algorithm should not be charged for the array space, and beyond that uses a handful of variables. The number of variables does not depend on the main design parameter b, which is one of them. We can justifiably say that this algorithm has no space costs.

The stream processing version shows a different situation. By the time it has read in n items, the algorithm has allocated $\lceil \log_b n \rceil$ buffers of size b. Each comes with an index or counter, to keep track of its state, for a total of $(b+1)\lceil \log_b n \rceil$ storage positions. The dependence of this formula on b for a value of n which we find representative of a moderately large application, $n = 10^8$ is given in the following table:

b	3	5	7	9
$(b+1)\lceil \log_b n \rceil$	68	69	76	84

Table 1: Space requirement for approximate median selection of a stream of 10^8 entries spb

We believe the numbers in this table support our describing the space requirements as modest. It would be easy to accommodate on most embedded system, in instruments or sensors.

There are two run-time cost components: element comparisons and moves (or swaps). We shall follow tradition and pay more attention to the first component.

3.2 Run-time considerations

Except that the context is quite different, there is very little difference between the operation of the algorithm on n terms in an array, or processing n arriving entries (disregarding input management). The number of term comparisons the two make is the same; but there is some difference in the number of term moves. In an array, as explained above, the fraction of selected b medians which is moved is $(b-1)/b$, for an overall of $n(b-1)/b^2$ expected moves in the first round and an approximate total of $\frac{n(b-1)}{b} \times \sum_{j=1}^{r} b^{-j} \approx \frac{n}{b}$ moves, assuming $n = b^r$. When processing a stream, if we assume the initial placement of an element in the first array is part of the input process, then we only have to account for the move of selected b-medians, only, but this time they are all moved, for a total of n/b. Details of implementation may swamp this difference. The total number of b-medians selected is approximately, assuming $n = b^r$, given by $n \times \sum_{j=1}^{r} b^{-j} \approx \frac{n}{b-1}$.

The number of comparisons is the more interesting run-time cost component. The known values for the number of comparisons required to find the median of b numbers, for $b \in (3,5,7,9)$, are in the first two rows of Table 2.

b	3	5	7	9
$\overline{V}_m(b)$	2.667	5.867	9.305	13.187
$V_m(b)$	3	6	10	14
C_b	1.333	1.4667	1.3293	1.6484

Table 2: The number of comparisons associated with approximate median selection vmb

Following [8] we denote by $V_m(b)$ the minimum number of comparisons needed to find the median of b numbers in the worst case. $\overline{V}_m(b)$ is the *average* number of comparisons needed for this feat. Most of the numbers are from [8, p. 217], except the value given for $\overline{V}_m(9)$. This entry is not guaranteed to be the correct cost of the mean-optimal algorithm. That algorithm is not yet known. For our analysis, where many b-medians are computed, the significance of the worst-case bound is unclear; we surely want the smallest possible mean value. The reason we bring the values of $V_m(b)$ is to show how close the optimal mean is to the upper bound, and to justify the value we give for $\overline{V}_m(9)$. It was obtained as the average cost of the algorithm which is *worst-case optimal*, as given in [10]. For other values of b, where the optimal algorithms are known for both objective functions, they differ. We however expect the difference would be small enough for the given figure to be used here reasonably.

The total comparison-cost is then given by $n\overline{V}_m(b)/(b-1) \overset{\text{def}}{=} C_b n$. The values of C_b are given in the last row of Table 2; because of the slight super-linear increase of $\overline{V}_m(b)$, we expected the larger buffers to be somewhat more expensive, but curiously, the charmed number 7 is here the least expensive of this set. Some authors have considered much larger buffers, and in [4] there is even some discussion of the limit $b \to \infty$; algorithmically, there seems to be small reason to consider such extremes. Finding the median of 11 or 15 items with anything close to the mean-optimal cannot be a simple feat. In fact, the optimal algorithms are not known yet for arrays larger than seven. We do not consider this matter as closed, however, since as we shall see later, increasing the size of the buffer affects the accuracy of its selection in a significant way.

4. Analysis of the Selection Accuracy

The accuracy with which the algorithm computes its result derives from two factors, with unequal significance: the deterministic exclusion of extreme values from consideration, and the probabilistic effect of the repeated preference given to intermediate values. We deal with them in this order.

4.1 Range of selection

It is obvious that not all the input array elements can be selected by the algorithm — *e.g.*, the smallest one is discarded in the first stage. The discussion may be easier to follow assuming the algorithm operates on an array, but it holds for the other version with no change. Let the array size be n, and denote by $v(n)$ the number of elements from the lower end (or the upper one, since the algorithm has bilateral symmetry) of the input, which are excluded deterministically: they will never be selected. The interpretation is that if we denote by x the output of the algorithm from n elements, then

$$v(n) < rank(x) < n - v(n) + 1. \tag{1}$$ bounds

We see (*e.g.*, by observing the tree built by the algorithm) that a single (the first) round weeds out the smallest m values, when $b = 2m+1$. To survive two rounds it is needed for an entry to be at least the $(m+1)^2$th smallest, and in general, we have the recurrence

$$v(b) = m, \qquad v(n) \geq (m+1)[v(n/b)+1] - 1. \tag{2}$$

Moreover, when $n = b^r$, the equality holds. The solution of the following recurrence, for $n = b^r$ follows:

$$v(n) = (m+1)[v(n/b)+1] - 1, \quad v(b) = m \quad \Longrightarrow \quad v(n) = (m+1)^{\log_b n} - 1.$$

While this may seem a nontrivial number, we see that the ratio of $v(n)$ to n is not encouraging:

$$\frac{v(n)}{n} = \frac{(m+1)^r - 1}{(2m+1)^r} \approx \left(\frac{m+1}{2m+1}\right)^r \approx \left(\frac{1}{2}\right)^r = \frac{1}{n^{\log_b 2}}.$$

The fraction which is deterministically eliminated decreases exponentially in r (and sublinearly in n. For $b = 7$ the rate is $1/n^{0.35621}$). Thus for this $b = 7$ and $n = 7^9 \approx 40M$, we find just barely over one half of a percent[†] are trimmed from each end of the sample. Hence our claim that this factor only makes a marginal contribution to the accuracy of the selection.

When n is not exactly b^r the calculation is more awkward (a detailed example for $b = 3$ is given in [1]), but the results are very close to the above and, qualitatively, the same.

The true state of affairs, as we now proceed to show, is much better: while the possible range of choice is wide, the algorithm zeroes in, with overwhelming probability, on a very small neighborhood of the true median.

[†]The exact numbers are $2 \times 262,143$ out of 40,353,607.

4.2 Probabilities of Selection

The purpose of this analysis is to be able to derive the following probability distribution:

$$P(z) = \Pr[zn < rank(x) < (1-z)n+1], \tag{3}$$ pfunc

for $0 \le z \le 1/2$. This describes the closeness of the selected value to the true median.

The first part of the analysis is combinatorial, and quite heavy; we only do it here for the smallest candidate for b, three, and consider n which is a power of three.

Definition 1.: Let $q^{(r)}_{a,d}$ be the number of permutations, out of the $n! = 3^r!$ possible ones, in which the entry which is the a^{th} smallest in the set is: (1) selected, and (2) becomes the d^{th} smallest in the next set, which has $\frac{n}{3} = 3^{r-1}$ entries.

It will turn out that this quite narrow look at the selection process is all we need to characterize it completely.

The reader may find it best, when following the derivation, to imagine the data arranged in a way which is somewhat different than the one actually used by the algorithm: View a permutation as an arrangement of the first n natural numbers in $\frac{n}{3} = 3^{r-1}$ successive triplets, that we index by j. The jth triplet, in positions $(3j-2,\ 3j-1,\ 3j),\quad 1 \le j \le \frac{n}{3}$, provides one locally selected median-of-three, or three-median, that continues to the next stage. Only such permutations where the integer a is the d^{th} smallest three-median are admissible (that is, contribute to $q^{(r)}_{a,d}$).

We count admissible permutations in the following steps:

(1) Count such permutations where the three-medians come out partially sorted, in increasing order. By partial sorting we mean that the leftmost $d-1$ three-medians are smaller than a and the rightmost $\frac{n}{3}-d$ are larger than a.

(2) Account for this restriction: multiply by the number of rearrangements of each permutation constructed as in step (1).

Step (2) is easy to dispose of: step (1) fixes the position of the triplet where a is the three-median, and allows $(d-1)!(\frac{n}{3}-d)!$ orders of the other triplets. Hence step (2) will contribute the factor $\frac{(n/3)!}{(d-1)!(\frac{n}{3}-d)!} = \frac{n}{3}\binom{\frac{n}{3}-1}{d-1}$.

To do (1) we need to account for the partial sortedness of the three-medians, and notice in which ways our arrangements restrict the steps that need to be done (we show below they imply certain restrictions on in-triplet ordering). The relative order of three-medians requires the following:
A—in each of the leftmost $d-1$ triplets as above, numbered $j \in [1,\ d)$, there are two values smaller than a ("small values"). This guarantees that each three-median there is smaller than a. We call them *small triplets*. One more small value must land in the d^{th} triplet.

B—in each triplet numbered $j \in (d,\ \frac{n}{3}]$, there are two values larger than a (“large values”). This guarantees that each three-median there is larger than a. We call them *large triplets*. One more large value is in triplet d.

This guarantees the partial sortedness of the permutation. Our assumption that we use the first n natural numbers implies numerical constraints between a, d, n:

$$a - 1 \geq 2(d-1) + 1 \quad \Longrightarrow \quad a \geq 2d, \tag{4}$$ $\boxed{\texttt{E1}}$

$$n - a \geq 2(\frac{n}{3} - d) + 1 \quad \Longrightarrow \quad a \leq \frac{n}{3} + 2d - 1. \tag{5}$$ $\boxed{\texttt{E2}}$

This also leads to the same $v(n)$, the possible range of the median-selection-procedure we derived.

Counting the ways we can do the above is best done by viewing the arrangement in stages.

First we place the element a in triplet d (say, in location $3d$). Then we select and distribute $d-1$ pairs (and a singleton for triplet d) of small values, in the leftmost $d-1$ triplets of places. These elements can be selected in $\binom{a-1}{2d-1}$ ways. Then we scatter them around, which we can do in $(2d-1)!$ ways, for a total of $(a-1)!/(a-2d)!$ arrangements. Similarly for $\frac{n}{3} - d$ pairs of large values (and one in triplet d), in $\binom{n-a}{2(\frac{n}{3}-d)+1}$ ways times $(2(\frac{n}{3} - d) + 1)!$, or in $(n-a)!/(\frac{n}{3} - a + 2d - 1)!$ ways.

To visualize the arguments below assume that at this time, each such pair occupies the two leftmost positions in each triplet (triplet d is now filled up).

This distribution, the first stage of step (1), creates therefore

$$\frac{(a-1)!(n-a)!}{(a-2d)!(\frac{n}{3} - a + 2d - 1)!} \tag{6}$$

arrangements.

Next, we are left with $\frac{n}{3} - 1$ elements, $a - 2d$ of them are small, and the rest are large. They have to be distributed into the positions left open in $\frac{n}{3} - 1$ triplets (all except triplet number d). Here appears a complication, the only one in the entire procedure.

It is best shown via an example: Suppose $a = 20,\ d = 5$ and we look at a small triplet. Further, assume the triplet has so far the entries 1 and 2, from the first distribution.

We compare two possibilities.

In one, we now put there the element 3, one of the $a - 2d = 10$ surviving small values. Like this triplet 1,2,3 we also get the 2,1,3, if the first distribution reversed the order of the pair. The other four permutations of this three values arise when the first distribution selected either of the pairs 1,3 or 2,3 for this location, and the set is completed in the second step by inserting 2 or 1 respectively. Conclusion: each such insertion accounts for exactly one ordered triplet.

In the second possibility we put there a surviving large value, say 25. In the same way we now have the triplets, 1,2,25 and 2,1,25. The other possible positions of the "25," unlike the first possibility, cannot arise via the way we did the initial distribution. Hence we should multiply the count of such permutations by 3; in other words: each such insertion accounts for exactly three triplets. This observation, that at this step we need to distinguish between small and large values and where they land, leads to the need of using an additional parameter. We select it to be the number of small values, out of the $a-2d$, that get to be inserted into "small" triplets, those in the range $1\dots d-1$, and denote it by i. Further, call a small triplet into which we put a small value 'homogeneous,' and let it be 'heterogeneous' if we put there a large value (and similarly for each of the rightmost $\frac{n}{3}-d$ triplets which gets a large or small value, respectively). With this notation we shall have, for a fixed i,

i	small homogeneous triplets
$\frac{n}{3}+d-a+i$	large homogeneous triplets
$d-i-1$	small heterogeneous triplets
$a-2d-i$	large heterogeneous triplets

We need to choose which of the small, and which of the large triplets would be, say, heterogeneous, and this introduces a factor of $\binom{d-1}{i}\binom{\frac{n}{3}-d}{a-2d-i}$.

Next comes the choice of the numbers, out of the $\frac{n}{3}-1$ available, that go into the small and large triplets. Since these need to be put in all possible orders, the selection factors cancel, and we are left with $(a-2d)!(\frac{n}{3}-a+2d-1)!$.

Finally we need to multiply by the factors $6\times 3^{a-d-2i-1}$.

The factor 6 accounts for the possible number of ways we can order the dth triplet (since so far there has been no constraint on the locations of its elements), and the next for the contribution of the possible orders of the heterogeneous triplets, as shown above.

Combining it all, with the contribution of step (2) above, we have

$$q^{(r)}_{a,d} = 2n(a-1)!(n-a)!\binom{\frac{n}{3}-1}{d-1}3^{a-d-1}\times\sum_i\binom{d-1}{i}\binom{\frac{n}{3}-d}{a-2d-i}\frac{1}{9^i}. \qquad (7) \quad \boxed{\text{Eq}}$$

From relations (4–5) we see that $q^{(r)}_{a,d}$ is nonzero for $0\le a-2d\le\frac{n}{3}-1$ only. The sum is expressible as a Jacobi polynomial, $\left(\frac{8}{9}\right)^{a-2d}P^{(u,v)}_{a-2d}\left(\frac{5}{4}\right)$, where $u=3d-a-1, v=\frac{n}{3}+d-a$, but this does not appear to confer any advantage.

Let $p^{(r)}_{a,d}$ be the probability that item a gets to be the d^{th} smallest among those selected for the next stage.

Since the $n! = 3^r!$ permutations are assumed to be equally likely, we have $p^{(r)}_{a,d} = q^{(r)}_{a,d}/n!$:

$$p^{(r)}_{a,d} = \frac{2}{3}\frac{3^{-d}\binom{\frac{n}{3}-1}{d-1}}{3^{-a}\binom{n-1}{a-1}} \times \sum_i \binom{d-1}{i}\binom{\frac{n}{3}-d}{a-2d-i}\frac{1}{9^i} = \frac{2}{3}\frac{3^{-d}\binom{\frac{n}{3}-1}{d-1}}{3^{-a}\binom{n-1}{a-1}} \times [z^{a-2d}](1+\frac{z}{9})^{d-1}(1+z)^{\frac{n}{3}-d}. \tag{8}$$

Eps

This allows us to calculate the function we need: The probability $P^{(r)}_a$, of starting with an array of $n = 3^r$ numbers, and having the a^{th} smallest element ultimately chosen as the approximate median. It is given by

$$P^{(r)}_a = \sum_{d_r} p^{(r)}_{a,d_r} P^{(r-1)}_{d_r} = \sum_{d_r,d_{r-1},\cdots,d_3} p^{(r)}_{a,d_r} p^{(r-1)}_{d_r,d_{r-1}} \cdots p^{(2)}_{d_3,2}\,, \qquad 2^{j-1} \le d_j \le 3^{j-1} - 2^{j-1} + 1. \tag{9}$$

Ep

Some telescopic cancellation occurs when the explicit expression for $p^{(r)}_{a,d}$ is used here, and we get

$$P^{(r)}_a = \left(\frac{2}{3}\right)^r \frac{3^{a-1}}{\binom{n-1}{a-1}} \sum_{d_r,d_{r-1},\cdots,d_3} \prod_{j=2}^{r} \sum_{i_j \ge 0} \binom{d_j - 1}{i_j}\binom{3^{j-1} - d_j}{d_{j+1} - 2d_j - i_j}\frac{1}{9^{i_j}}. \tag{10}$$

Epp

As above, each d_j takes values in the range $[2^{j-1} \;..\; 3^{j-1} - 2^{j-1} + 1]$, $d_2 = 2$, and $d_{r+1} \stackrel{\text{def}}{=} a$ (we could let all d_j take all positive values, and the binomial coefficients would produce nonzero values for the required range only). The probability $P^{(r)}_a$ is nonzero for $\nu(n) < a < n - \nu(n) + 1$ only.

This distribution has resisted our attempts to provide a direct analytical characterization of its behavior. The examples in §4.3 give a fairly good idea of its behavior, but to gain analytical insight we needed to develop in §4.4 an approach that uses the large-sample behavior of this algorithm. We obtain there a function which is the limit distribution of the selected median, and while it appears remarkably close to the Gaussian, it is not! its tails are heavier.

4.3 Numerical examples

examp

The key relation (3) in terms of these probabilities is given by

$$\sum_{\lfloor zn \rfloor < a < \lceil (1-z)n \rceil + 1} P^{(r)}_a \qquad \text{where } 0 \le z < \frac{1}{2}, \tag{11}$$

but in view of these unwieldy expressions we chose to present the effectiveness of the algorithm by considering directly the bias of the returned approximate median, $D_n \stackrel{\text{def}}{=} X_n - \mathcal{M}_d(n)$, where $\mathcal{M}_d(n)$ is the true median rank, $(n+1)/2$. Since $E[D_n] = 0$, by symmetry, it is $|D_n|$ which is of interest. We computed the statistics of this absolute value using the probabilities in Eq.(10).

We denote the mean of $|D_n|$ by μ_d, and the standard deviation of D_n by σ_d. (This is then also $E[D_n^2]$; if we wanted the variance of $|D_n|$ we would compute it as $\sigma_d^2 - \mu_d^2$.)

We obtained the results in Table 3; note the trend in the two rightmost columns. In the next subsection we show that these ratios approach limits.

The ratio $\mu_d/\mathcal{M}_d(n)$ relativizes the expected error of the approximate median selection algorithm. The trend of this ratio $\mu_d/\mathcal{M}_d(n)$ can be then seen as the improvement of the selection effectiveness with increasing (initial) array size n. It is apparent in the table, and shown later, that this ratio decreases as $n^{\log_3 2-1} = (2/3)^r = 1/n^{0.36907}$.

n	$r=\log_3 n$	μ_d	σ_d	$\mu_d/\mathcal{M}_d(n)$	$\mu_d/n^{\log_3 2}$	$\sigma_d/n^{\log_3 2}$
9	2	0.428571	0.654654	0.107143	0.107143	0.163663
27	3	1.475971	1.892344	0.113536	0.184496	0.236543
81	4	3.617240	4.563487	0.090431	0.226077	0.285218
243	5	8.096189	10.194222	0.066911	0.253006	0.318569
729	6	17.377167	21.872372	0.047739	0.271518	0.341756
2187	7	36.427027	45.839609	0.033328	0.284586	0.358122
6561	8	75.255332	94.679474	0.022944	0.293966	0.369842

Table 3: Statistics of the median selection bias $|D_n|$ as function of array size tty

In [1] we show experimental results with this algorithm for larger arrays, and its sensitivity to threshold settings.

4.4 Asymptotic analysis

asymp

Since we find the probability-mass function (PMF) given in Eq.(10) hard to appreciate, we did what analysts do: went asymptotic. But in order to do this, we need to show that it is possible to approximate that PMF with another distribution, that is easier to handle, in a meaningful way.

Let $\Xi = (\xi_1, \xi_2, \ldots, \xi_n)$ be n independent identical $U(0,1)$ variates. We denote these values, when sorted, by $\xi_{(1)}, \xi_{(2)}, \ldots, \xi_{(n)}$. The ranks of the elements of Ξ form a permutation of the integers 1 through n. If we used our Section 2 algorithm on this permutation, and it returned the result X_n, then using the algorithm on Ξ itself would return the order statistic $\xi_{(X_n)}$, which we denote by Y_n. Since the ξ_i are independent, unlike the elements of a permutation, it is much easier to calculate using these variates.

We first show that the Y_n provide a useful approximation as we claimed. More precisely, we use the distribution of $Y_n - \frac{1}{2}$ to approximate the distribution of D_n/n.

The probability density function of the order statistic $\xi_{(k)}$ of $U(0,1)$ is given by

$$f_k(x) = \binom{nk}{k} x^{k-1}(1-x)^{n-k}, \tag{12}$$

hence

$$E[\xi_{(k)}] = \frac{k}{n+1}, \qquad V[\xi_{(k)}] = \frac{k(n+1-k)}{(n+1)^2(n+2)}. \tag{13}$$

Therefore, the mean square distance between $Y_n - \frac{1}{2}$ and D_n/n, over all the permutations for which, say, $X_n = k$ (which implies $Y_n = \xi_{(k)}$), can be estimated as follows:

$$E_k\left[\left(Y_n - \frac{1}{2}\right) - \left(\frac{X_n - \frac{n+1}{2}}{n}\right)\right]^2 = E_k\left[Y_n - \frac{X_n - 1/2}{n}\right]^2 = E\left[\xi_{(k)} - \frac{k-1/2}{n}\right]^2 \lesssim \frac{1}{4n}, \tag{14}$$ $\boxed{\text{Ea}}$

where the last expectation is over the variability of the the order statistic only. Since the bound is independent of k, it holds for all samples, and in terms of locating the median, over all permutations. While we have no tractable mathematical expression for the variance of the approximate median X_n, we can use the numerical results that were computed from the probability density function of Eq. (10). Table 3 suggests that $V(|D_n|/n)$ is in $\Omega(n^{\log_3 4-2})$; since $\log_3 4 \approx 1.262$, and then $n^{\log_3 4-2} \approx n^{-0.732}$ which is significantly larger than the bound in Eq.(14), we can say that $Y_n - 1/2$ is a good approximation for D_n/n; in particular, their distributions converge (to each other) as $n \to \infty$. While the table was computed under the assignment $b = 3$, similar behavior is expected for larger b as well. We shall be able to quantify this claim later on.

Next we derive a recurrence for the distribution of $Y_n - 1/2$. We continue to consider only values of n which are powers of b and define:

$$F_r(x) \equiv \Pr(Y_n - 1/2 \le x), \qquad -1/2 \le x \le 1/2, \qquad n = b^r. \tag{15}$$

We start the recursion with $F_0(x) = x + 1/2$. Since Y_{bn} is obtained by taking the median of $b = 2m+1$ independent values each of which has the distribution of $Y_n - 1/2$, denoted by $F_r(\cdot)$, then $Y_{bn} - 1/2$ has the distribution $F_{r+1}(\cdot)$, equal to the probability that at least $m+1$ of the Y_n are smaller than x.

$$F_{r+1}(x) = \Pr(Y_{bn} \le x + 1/2) = \sum_{j=m+1}^{2m+1} \binom{b}{j} F_r^j(x)(1-F_r(x))^{2m-j+1} \overset{\text{def}}{=} \Phi_b(F_r(x)), \tag{16}$$ $\boxed{\text{E18}}$

where $\Phi_b(t) = \sum_{j=m+1}^{b} \binom{b}{j} t^j (1-t)^{b-j}$. This is exactly the recurrence obtained in [12]. They were considering a general underlying distribution, whereas our needs are well met by the basic uniform distribution $U(0,1)$.

Note that the transformation $\Phi_b(\cdot)$ transforms a distribution into a distribution on the same interval, since it transforms the values of the distribution at the endpoints, 0 and 1, into themselves, and has a positive derivative throughout $(0,1)$. Iterations of $\Phi_b(\cdot)$, denoted with a parenthesized integer for the iteration order, preserve this property, and it implies that the sequence $F_r(x) = \Phi_b^{(r)}(F_0(x))$ converges. The limit function $F(x)$ satisfies the equation $F(x) = \Phi_b(F(x))$, and it is not an interesting one: it has the values 0 and 1, in the subintervals [−1/2,0) and (0,1/2] respectively. At the origin and the end points it has the same values as $F_0(\cdot)$.

While this is a consistent result, it is hardly useful. It merely tells us that as n increases $Y_n - \frac{1}{2}$ (and D_n/n) converge to zero, as Table 3 suggests strongly. To look in more detail in the way this happens we need to create a non-degenerate version of the distribution, and to achieve this we introduce a change of scale.

A clue to the range of possible scales is given by Eq.(14), which implies that $\mu^{2r}E\left[\left(Y_n - \frac{1}{2}\right) - D_n/n\right]^2 \longrightarrow 0$ as r (and $n = 3^r$) increase, so long as μ is not too large. A simple calculation shows we need $\mu \in [0, \sqrt{3})$. With such a μ we could still track $\mu^r(D_n/n)$ with $\mu^r(Y_n - 1/2)$ effectively.

The needed rescaling follows naturally from the following lemma, which we prove here, since its details are essential in the continuation.

lem1 **Lemma 1.:** [6] Let $a \in (0,\infty)$ and a mapping ϕ of $[0,a]$ into $[0,a]$ be given, extended by defining $\phi(x) = x$ for $x > a$. Assume

(*i*) $\phi(0) = 0$
(*ii*) $\phi(a) = a$
(*iii*) $\phi(x) > x$, for all $x \in (0,a)$.
(*iv*) $\phi'(0) = \mu > 1$, and continuous there; $\phi(\cdot)$ is continuous and strictly increasing on $[0,a)$.
(*v*) $\phi(x) < \mu x, \quad x \in (0,a)$.

Then

$$\text{as } r \longrightarrow \infty, \quad \phi_r(x/\mu^r) \longrightarrow \psi(x), \quad x \ge 0 \tag{17}$$ A1

where $\phi_r(\cdot)$ is the rth iterate of $\phi(\cdot)$. The function $\psi(x)$ is well defined and strictly monotonic increasing for all x. It increases from 0 to a, and satisfies the equation $\psi(\mu x) = \phi(\psi(x))$.

Proof. From Property (v): $\phi(x/\mu^{r+1}) < x/\mu^r$,
Since iteration preserves monotonicity, $\phi_{r+1}(x/\mu^{r+1}) = \phi_r(\phi(x/\mu^{r+1})) < \phi_r(x/\mu^r)$.
Since $\phi(\cdot)$ and its iterates are nonnegative, this monotonic decrease implies convergence. We denote the limit by $\psi(\cdot)$.

We note that the properties of $\psi(x)$ depend on the behavior of $\phi(\cdot)$ near $x = 0$. In particular, since $\phi'(x)$ is continuous at $x = 0$, $\psi(\cdot)$ is continuous throughout. Since it is bounded, the convergence is uniform on $[0,\infty]$. Hence, since $\phi(\cdot)$ and all its iterates are strictly monotonic, so is $\psi(\cdot)$ itself.

To find more information about $\psi(\cdot)$, we use the monotonicity we showed and (v) again, to produce

$$\psi(x) \;<\; \phi_r(x/\mu^r) \;<\; \phi(x/\mu) \;<\; x. \tag{18}$$ Eac

For any positive x there is a minimal r such that $x/\mu^r \le a$. Since $\phi_r(\cdot)$ maps $[0,a]$ to itself, then Eq.(18) implies $\psi(x) \le a$, for all $x \ge 0$. We now show that $\psi(\cdot)$ achieves the value a. We create an increasing sequence $\{x_j, j \ge 1\}$, such that

$$x_{j-1}/\mu^{j-1} < x_j/\mu^j \longrightarrow a^-, \tag{19}$$ Eae

that is, the values x_j/μ^j increase as well, and approach a from below. Property (iii) iterates to $\phi_r(x) > x$, hence also $\phi_j(x_j/\mu^j) > x_j/\mu^j \longrightarrow a$. Since $\psi(\infty) = \lim_{j\to\infty} \psi(x_j)$, this limit achieves the value a. □

The transformation $\Phi_b(x)$ does not quite satisfy the properties claimed for ϕ in Lemma 1, but it is close; we only need to shift its argument. We define $G_r(x) = F_r(x) - 1/2$, and the functions $G_r(x)$ are iterated as follows

$$G_{r+1}(x) = F_{r+1}(x) - \frac{1}{2} = \Phi_b(F_r(x)) - \frac{1}{2} = \Phi_b(G_r(x)+1/2) - \frac{1}{2} \overset{\text{def}}{=} \theta_b(G_r(x)), \tag{20}$$ gr1

and $\theta_b(x)$ is a suitable candidate for the transformation $\phi(x)$ on the interval $[0,1/2]$. An explicit expression for the kernel $\theta_b(x)$ is then given by

$$\theta_b(x) = \sum_{j=m+1}^{b} \binom{b}{j} \left(\frac{1}{2}+x\right)^j \left(\frac{1}{2}-x\right)^{b-j} - \frac{1}{2}, \qquad -\frac{1}{2} \le x \le \frac{1}{2}. \tag{21}$$ gr4

Here are the polynomials $\theta_b(x)$ for the small values of b we have been considering.

m	b	$\theta_b(x)$
1	3	$\frac{3}{2}x - 2x^3$
2	5	$\frac{15}{8}x - 5x^3 + 6x^5$
3	7	$\frac{35}{16}x - \frac{35}{4}x^3 + 21x^5 - 20x^7$
4	9	$\frac{315}{128}x - \frac{105}{8}x^3 + \frac{189}{4}x^5 - 90x^7 + 70x^9$

Table 4: The iteration kernel $\theta_b(x)$—first few b values tsb

Of major interest is the derivative of the polynomial at $x = 0$. Extracting the coefficient of x in $\theta_b(x)$ we find

$$\theta_b'(x)|_{x=0} = [x^1]\theta_b(x) = \binom{2m}{m}\frac{b}{4^m}. \qquad (22) \quad \boxed{\texttt{gr8}}$$

For larger values of b the last right-hand side is very close to $b/\sqrt{m\pi}$.

We can now state and prove our main asymptotic result

$\boxed{\texttt{T3}}$ **Theorem 1.** [6] Let $n = b^r$, $b = 2m+1$, $m, r \in \mathbb{N}$, and denote by X_n the approximate median of a random permutation of $1, \ldots, n$ (with all permutations assumed equally likely) computed with any of the algorithms of Figure 2. Then a random variable X exists, such that

$$\mu^r \frac{X_n - \frac{n+1}{2}}{n} \quad \longrightarrow \quad X, \qquad (23) \quad \boxed{\texttt{Elimit}}$$

where X has the distribution $F(\cdot)$, determined by the equations

$$F(x) \equiv G(x) + 1/2, \qquad G(\mu x) = \theta_b(G(x)), \qquad -\infty < x < \infty \qquad (24) \quad \boxed{\texttt{Eab}}$$

The distribution function $F(\cdot)$ is strictly increasing throughout.

Proof. We need only verify that $\theta_b(x)$ satisfies the conditions put on ϕ in Lemma 1, with $a = 1/2$:

(i) $\theta_b(0) = 0$
(ii) $\theta_b(1/2) = 1$
(iii) $\theta_b(x) > x$, for all $x \in (0, 1/2)$.
(iv) $\theta_b'(0) = \mu > 1$, and continuous there; $\theta_b(\cdot)$ is continuous and strictly increasing on $[0, 1/2)$.
(v) $\theta_b(x) < \mu x, \quad x \in (0, 1/2)$.

Property (i) is manifest in Eq.(21), since $\sum_{j=m+1}^{2m+1}\binom{2m+1}{j} = \frac{1}{2} \times 2^{2m+1}$.
Property (ii) is best seen in Eq.(21) as well, which at $x = 1/2$ has only the term '$j = b$' survive in the sum.
For property (iv), to show that $\mu_b > 1$ we extracted the coefficient given in Eq.(22). The rest follows from $\theta_b(x)$ being a polynomial.
Properties (iii, v), combined as $x < \theta_b(x) < \mu x$ for $x \in (0, 1/2)$, can be verified by inspection. Property (iii) is intuitively clear, once stated in probabilistic terms: it claims that the probability a binomial random variable $B(2m+1, \frac{1}{2} + x)$ exceeds m is larger than its single event probability $\frac{1}{2} + x$, for $x > 0$. For $x = 0$ it achieves equality.

Hence $\theta_b(x\mu^r)$, as defined in Eq.(20), converges to a function $G(x)$ which satisfies Eq.(24). $\square$

There is no obvious way to "solve" Eq. (24), but it can still be very informative. First, we see that $G_0(x)$ and $\theta_b(x)$ are odd functions, hence all the $G_r(x)$, as well as $G(x)$ are necessarily odd as well. We can write then $F(x) = \frac{1}{2} + \sum_{k\geq 1} f_k x^{2k-1}$. This power series expansion can be computed by extracting successive coefficients from Eq. (24). This requires some care, when done numerically, since the signs of the f_k alternate, but for $b = 3$ we could carry it out, using MAPLE and the recurrence

$$f_j = \frac{2}{\mu(1-\mu^{2j-2})} \sum_{i=1}^{j-2} f_i \sum_{k=1}^{j-i} f_k f_{j-i-k+1}, \qquad j \geq 2 \qquad f_1 = 1, \qquad b = 3, \qquad \mu = 3/2. \tag{25}$$

Ebrecur

The value of f_1 is obtained by extracting the coefficient of x in both sides of Eq. (24). The recurrence was iterated to produce Table 5, which contains a few of the coefficients (for some of the results below we needed significantly higher-order f_k).

k	f_k
1	$1.00000000000000 \times 10^{+00}$
2	$-1.06666666666667 \times 10^{+00}$
3	$1.05025641025641 \times 10^{+00}$
4	$-8.42310905468800 \times 10^{-01}$
5	$5.66391554459281 \times 10^{-01}$
6	$-3.29043692201665 \times 10^{-01}$
7	$1.69063219329527 \times 10^{-01}$
8	$-7.82052123482121 \times 10^{-02}$
9	$3.30170547707520 \times 10^{-02}$
10	$-1.28576608229956 \times 10^{-02}$
20	$-4.33903859413399 \times 10^{-08}$
30	$-3.20126276232555 \times 10^{-15}$
40	$-1.94773425996709 \times 10^{-23}$
60	$-4.03988860877434 \times 10^{-42}$
80	$-5.63050454255617 \times 10^{-63}$
100	$-1.88810747562091 \times 10^{-85}$
125	$2.50253570235335 \times 10^{-115}$
150	$-8.16299422374440 \times 10^{-147}$

Table 5: Coefficients for the expansion $G(x) = \sum_{k\geq 1} f_k x^{2k-1}$.

fuy

Curiously, initial use of these coefficients to compute values for $F(x)$ produced values for the distribution so close to a centered Normal distribution with standard deviation of $\sigma = 1/\sqrt{2\pi}$, that we suspected this to be the true limiting distribution; the largest *relative* difference between $F(x)$ and $N(0,\sigma^2)$ over the interval $[0,1.3963]$—which covers approximately 3.5 standard deviations—is close to 0.14% (and with a slightly higher value for σ decreases further, to 0.05%). We found such a conclusion quite surprising, since this limiting process shows no reason why it would lead to a Gaussian distribution — and indeed, it is not! The coefficients of the $N(0,\sigma^2)$ do not satisfy anything like the recirrence in Eq. (25).

Equation (23) can be written as $D_n \longrightarrow Xn/\mu^r = X \times n^{1-\log_b \mu}$. An immediate corollary is that, asymptotically, this determines the rate of growth of the moments of D_n (or of $|D_n|$) with n: every time r is raised by 1 (n multiplied by b), we find that μ_d and σ_d get multiplied by $b^{1-\log_b \mu} = b/\mu$. In the case $b = 3$, μ_d and σ_d should double. The numbers in Table 3 show this, and also how for small values of n the rate of increase is even faster, as the influence of the "taboo" values, as shown in §4.1, decreases.

Bounding the function $F(x)$

While the numbers in Table 5 are not very useful in giving us a direct idea about the behavior of the function $F(x)$, it turns out that combined with the properties of the distribution, given in Theorem 1, they can lead to quite definite statements.

We continue with $b = 3$, taking advantage of the availability of Table 5; similar calculations for higher b would be much more complicated, because the kernels are higher-degree polynomials, but numerical experimentation suggests they exhibit similar behavior. Using Theorem 1 we find that $F(\cdot)$, the limiting distribution of $\mu^r D_n/n$, satisfies

$$F(x\mu) = 3F^2(x) - 2F^3(x), \quad b = 3, \quad \mu = \frac{3}{2}. \qquad (26) \quad \boxed{\texttt{E29}}$$

Let $g(x)$ for positive x be the complementary probability function, $g(x) \equiv 1 - F(x)$, and the last equality means that $g(\cdot)$ also satisfies the equation

$$g(x\mu) = 3g^2(x) - 2g^3(x) \Longrightarrow 3g(x\mu) = (3g(x))^2 \left(1 - \frac{2}{3}g(x)\right). \qquad (27) \quad \boxed{\texttt{E30}}$$

Since $0 < g(x) < 1$ for all finite x, we find

$$\frac{1}{3}(3g(x))^2 < 3g(x\mu) < (3g(x))^2. \qquad (28) \quad \boxed{\texttt{E32}}$$

The right-hand side inequality, written for $h(x) \equiv 3g(x)$, produces when iterated k times $h\left(x\mu^k\right) < (h(x))^{2^k}$. Taking logarithm of both sides, we write $\ln h\left((\mu)^k x\right) < 2^k \ln h(x)$.
Define

$$t \equiv x\mu^k \implies k = \frac{\lg(t/x)}{\lg \mu}$$

where we used binary logarithm, since we have k in 2^k, and now we can write

$$\ln h(t) \le q(x)t^{\nu} \implies h(t) \le e^{q(x)t^{\nu}}, \quad \nu \equiv \frac{1}{\lg \mu}, \quad q(x) \equiv x^{-\nu} \ln h(x). \qquad (29) \quad \boxed{\texttt{fio}}$$

We now view x as a parameter; t is the argument of $h(\cdot)$, the two are related through the value of k (which need not be an integer, since $h(\cdot)$ is a smooth function).

The function $q(x)$ is negative. To have a tight bound we would like to select a value of x which minimizes its value. We denote it by $q_0 = q(x_0)$, where x_0 is selected to make that negative coefficient as low as it can go. Since we do not know yet $F(x)$ well enough to locate x_0 analytically, we did so numerically, with the help of the coefficients in Table 5, and obtained that beyond $x_0 = 2.2$ the function $q(x)$ becomes remarkably flat,[‡] and the value of q_0 is approximately $-3.88112230\ldots$. For $b = 3$, the numerical value of the parameter ν is $1.70951129\ldots$, and this value is inherent in the equation that $F(x)$ satisfies. The relation (29) for the limit in n, and for all $x > 0$, is now

$$\ln h(t) < -c \exp\left(\ln t \frac{\ln 2}{\ln 3/2}\right) \Longrightarrow h(t) < e^{-ct^{\nu}}, \quad c \approx 3.88112230, \quad \nu \approx 1.70951129. \qquad (30) \quad \boxed{\text{E34}}$$

From the left-hand side of relation (28) we obtain a similar inequality, $g(\mu x) < g^2(x)$, which leads us to a bound of the form $\ln g(t) > p(x)t^{\nu}$; same ν, and trying for the best bound we look for a value of x, call it x_1 where the negative $p(x) = x^{-\nu} \ln g(x)$ is simply $q(x) - \ln 3x^{-\nu}$, clearly as x increases the value of $p(x)$ approaches that of $q(x)$ and we can peg its desired value at the same as q_0.

We have proven

Corollary 1.: *The tails of the distribution of the random variable X defined in equation* (23) *satisfy*

$$\frac{1}{9}e^{-ct^{\nu}} < 1 - F(t) < \frac{1}{3}e^{-ct^{\nu}}, \quad c \approx 3.88112230, \quad \nu \approx 1.70951129, \qquad t \to \infty. \qquad (31) \quad \boxed{\text{E38}}$$

We remark that the tails of the Normal distribution $N(0, 1/2\pi)$ decay faster; they satisfy, for large x, $1 - \Phi_\sigma(x) \approx e^{-x^2/2} / (2\pi x)$, and it is remarkable that the parameters in equation (31) are such that over the entire range of practical interest the two distributions are hardly distinguishable.

5. Conclusion

We have presented an approximate median finding algorithm, and an analysis of its characteristics.

Both can be extended. In particular, the algorithm can be adapted to select an approximate k^{th}-element, for any $k \in [1, n]$, as has been shown in [3]. The analysis of Section 4 needs to be extended to higher values of b, both in the combinatorial calculation and in the asymptotic bounds; in particular, estimates of the variance and higher moments are needed.

[‡]which means that the exponential is a very good estimate of $h()$ there.

Acknowledgments

The derivations of Section 4.4 are largely based on material provided by Svante Janson; the same approach was suggested by Uwe Rösler.

References

BCCCH00 [1] S. Battiato, D. Cantone, D. Catalano, G. Cincotti, and M. Hofri. An Efficient Algorithm for the Approximate Median Selection Problem. 4th CIAC 2000, Rome , Italy. March 13, 2000. Published in the conference proceedings, LNCS 1767, 226–238, Springer Heidelberg, 2000.

CC02 [2] D. Cantone and G. Cincotti. QuickHeapsort: an efficient mix of classical sorting algorithms. *Theoretical Computer Science*, **285**(1):2542, 2002.

CCH06 [3] D. Cantone, G. Cincotti and M. hofri. An Efficient Approximate Algorithm for the *k*-Selection Problem Technical Repoert #.... Department of Computer Science, Wpi, August 2006.

CL93 [4] Chao, M. T. and Lin, G. D (1993). The Asymptotic Distributions of the Remedian. J. of Statistical Planning and Inference 37, 1-11.

HM95 [5] C. Hurley and Reza Modarres: Low-Storage quantile estimation, *Computational Statistics*, 10:311–325, 1995.

SJ99 [6] Svante Janson – Private communication, June 1999.

kataj [7] J. Katajainen. *The Ultimate Heapsort*, DIKU Report 96/42, Department of Computer Science, Univ. of Copenhagen, 1996.

Kn99 [8] D.E. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, 2nd Ed. 1999.

MRL99 [9] Gurmeet S. Manku, Sridhar Rajagopalan, Bruce G. Lindsay. Random Sampling Techniques for Space Ecient Online Computation of Order Statistics of Large Datasets. *SIGMOD Record*, **28** #2, 251–262, 1999. (Proc. 1999 ACM SIGMOD Conference).

No74 [10] Kohei Noshita. Median selection of 9 elements in 14 comparisons. *Information Processing Letters* 3(1):18–12 (1974).

Ros97 [11] L. Rosaz. Improving Katajainen's Ultimate Heapsort, Technical Report N.1115, Laboratoire de Recherche en Informatique, Université de Paris Sud, Orsay, 1997.

RB90 [12] P.J. Rousseeuw and G.W. Bassett: The remedian: A robust averaging method for large data sets. *Jour. Amer. Statist. Assoc*, 409:97–104, 1990.

Tu78 [13] Tukey, J. W. (1978). The ninther, a technique for low-effort robust location in large samples. Contributions to survey sampling and applied statistics, H. A. David ed. PP. 251-257.

We78 [14] B. W. Weide. Space efficient on-line selection algorithm. *Proceedings of the 11th symposium of Computer Science and Statistics, on the interface*, 308–311. (1978).