Chapter 34

Order Statistics

This chapter presents the problem of computing the order statistics of a sequence and a randomized algorithm for this problem.

1 The Order Statistics Problem

Definition 34.1 (Order Statistics Problem)**.** Given a sequence, an integer k where $0 \leq k < |a|$, and a comparison operation $<$ that defines a total order over the elements of the sequence, find the k^{th} ***order statistics***, i.e., k^{th} smallest element (counting from zero) in the sequence.

Reduction to Sorting. We can solve this problem by reducing it to sorting: we first sort the input and then select the k^{th} element. Assuming that comparisons require constant work, the resulting algorithm requires $O(n \lg n)$ work, but we wish to do better: in particular we would like to achieve linear work and $O(\lg^2 n)$ span.

2 Randomized Algorithm for Order Statistics

This section presents a randomized algorithm for computing order statistics that uses the contraction technique: it solves a given problem instance by reducing it a problem instance whose size is geometrically smaller in expectation.

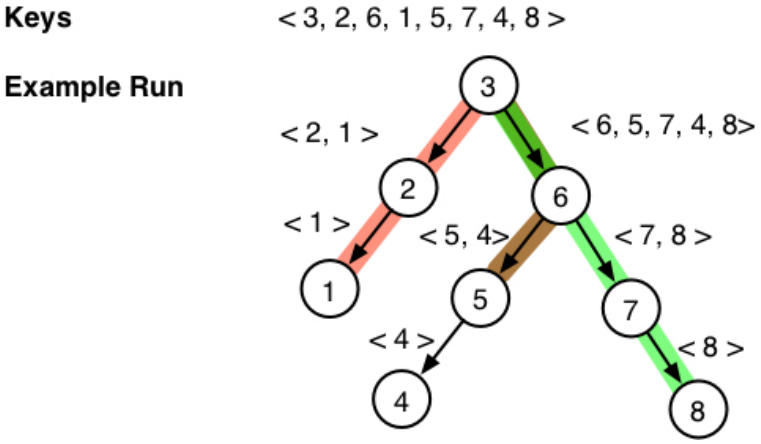
Algorithm 34.2 (Contraction-Based Select)**.** For the purposes of simplicity, let's assume that sequences consist of unique elements and consider the following algorithm that uses randomization to contract the problem to a smaller instance. The algorithm divides the input into left and right sequences, ℓ and r, and figures out the side k^{th} smallest must be in, and explores that side recursively. When exploring the right side, r, the algorithm adjusts

the parameter k because the elements less or equal to the pivot p are being thrown out (there are $|a| - |r|$ such elements).

```
1   select a k =
2   let
3     p = pick a uniformly random element from a
4     ℓ = ⟨ x ∈ a | x < p ⟩
5     r = ⟨ x ∈ a | x > p ⟩
6   in
7     if (k < |ℓ|) then select ℓ k
8     else if (k < |a| − |r|) then p
9     else select r (k − (|a| − |r|))
10  end
```

Example 34.1. Example runs of `select` illustrated by a ***pivot tree.*** For illustrative purposes, we show all possible recursive calls being explored down to singleton sequences. In reality, the algorithm explores only one path.

- The path highlighted with red is the path of recursive calls taken by `select` when searching for the first-order statistics, $k = 0$.
- The path highlighted with brown is the path of recursive calls taken by `select` when searching for the fifth-order statistics, $k = 4$.
- The path highlighted with green is the path of recursive calls taken by `select` when searching for the eight-order statistics, $k = 7$.



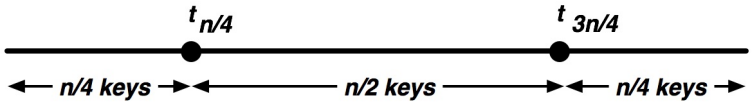
3 Analysis

We analyze the work and span of the randomized algorithm for order statistics and show that the `select` algorithm on input of size n performs $O(n)$ work in expectation and has $O(\lg^2 n)$ span with high probability.

4 Intuitive Analysis

Before we cover the more precise analysis, let's develop some intuition by considering the probability that we split the input sequence more or less evenly.

Observation. Recall that the rank of an element in a sequence is the position of the element in the corresponding sorted sequence. Consider the rank of the pivot selected at a call to `select`. If the selected pivot has rank greater than $n/4$ and less than $3n/4$, then the size of the input passed to the next recursive call is at most $3n/4$. Because all elements are equally likely to be selected as a pivot the probability that the selected pivot has rank greater than $n/4$ and less than $3n/4$ is $\frac{3n/4-n/4}{n} = 1/2$. The figure below illustrates this.



Need for Luck. The observation above implies that at each recursive call, the instance size decreases by a constant fraction of $3/4$ with probability $1/2$. Thus if we are lucky, we successfully decrease the input size by a constant fraction.

But what if we are unlucky? Consider two successive recursive calls, the probability that the instance size decreases by $3/4$ after two calls is the probability that it decreases in either, which is at least $1 - \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$. More generally, after $c > 1$ such successive calls, the probability that the size decreases by a factor of $\frac{3}{4}$ is $1 - \frac{1}{2^c}$.

Note that $\lim_{c\to\infty} 1 - \frac{1}{2^c} = 1$. For $c = 10$, this probability is 0.999 and thus we are highly likely to get lucky after a small number of calls. Thus intuitively speaking, we expect `select` to complete in $c \lg n$ depth (recursive calls) for some constant c. Because the algorithm performs linear work and logarithmic span to filter the input sequence and then recurs, we can write the total work as a geometrically decreasing sum totaling up to $O(n)$ and span is $O(\lg^2 n)$.

4.1 Complete Analysis

We now present an analysis that makes precise described in the intuitive analysis.

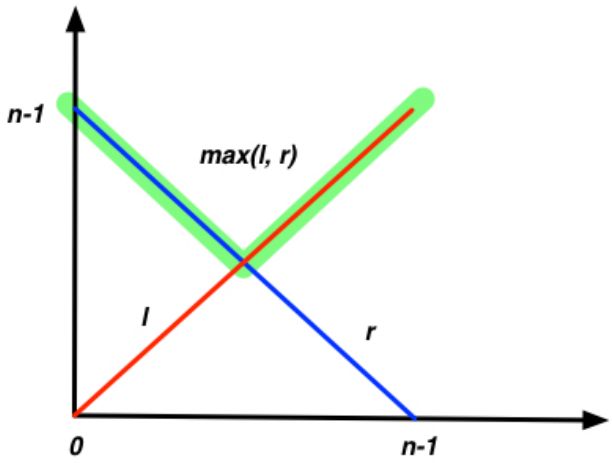
The Recurrences. Let $n = |a|$ and consider the partition of a into ℓ and r. Define

$$X(n) = \frac{\max\{|\ell|, |r|\}}{n}$$

as the fractional size of the larger side. Because *filter* requires linear work and logarithmic span, we can write the work and span of the algorithm as

$$W(n) \leq W(X(n) \cdot n) + O(n)$$
$$S(n) \leq S(X(n) \cdot n) + O(\lg n).$$

Bounding the Expected Fraction. For the analysis, we will bound $\mathbf{E}[X(n)]$, i.e., the expected fraction of the instance size solved by the recursive call. Because all elements are equally likely to be chosen, we can calculate the size of ℓ and size of r as a function of the ***rank*** of the pivot, i.e., its position in the sort of the input sequence. If the pivot has rank i, then ℓ has length i and r has length $n - i - 1$. The drawing illustrates the sizes of ℓ and r and their maximum as a function of the rank of the pivot.



Since the algorithm chooses the pivot uniformly randomly, i.e., with probability $1/n$, we can write the expectation for $X(n)$ as

$$\mathbf{E}[X(n)] = \frac{1}{n}\sum_{i=0}^{n-1}\frac{\max\{i, n-i-1\}}{n} \leq \frac{1}{n}\sum_{j=n/2}^{n-1}\frac{2}{n}\cdot j \leq \frac{3}{4}$$

(Recall that $\sum_{i=x}^{y} i = \frac{1}{2}(x+y)(y-x+1)$.)

Important. Note that expectation bound holds for all input sizes n.

The calculation of $\mathbf{E}[X(n)]$ tells us that in expectation, $X(n)$ is a smaller than 1. Thus when bounding the work we should have a nice geometrically decreasing sum that adds up to $O(n)$. But it is not quite so simple, because the constant fraction is only in expectation. For example, we could get unlucky for a few contraction steps and leading to little or no reduction in the size of the input sequence.

We next show that that even if we are unlucky on some steps, the expected size will indeed go down geometrically. Together with the linearity of expectations this will allow us to bound the work.

Theorem 34.1 (Expected Size of Input)**.** Starting with size n, the expected size of a in algorithm `select` after d recursive calls is at most $\left(\frac{3}{4}\right)^d n$.

Proof. The proof is by induction on the depth of the recursion d. In the base case, $d = 0$ and the lemma holds trivially. For the inductive case assume that the lemma holds for some $d \geq 0$. Consider now the $(d+1)^{th}$ recursive call. Let Y_d be the random variable denoting the instance size and let Z_d denote the pivot chosen, at the depth d^{th}. For any value of y and z, let $f(y, z)$ be the fraction of the input reduced by the choice of the pivot at position z for an input of size y. We can write the expectation for the input size at $(d+1)^{st}$ call as

$$\begin{aligned} E[Y_{d+1}] &= \textstyle\sum_{y,z} y f(y,z) \mathbf{P}_{Y_d, Z_d}(y,z) \\ &= \textstyle\sum_y \sum_z y f(y,z) \mathbf{P}_{Y_d}(y) \mathbf{P}_{Z_d \mid Y_d}(z \mid y) \\ &= \textstyle\sum_y y \mathbf{P}_{Y_d}(y) \sum_z f(y,z) \mathbf{P}_{Z_d \mid Y_d}(z \mid y) \\ &\leq \textstyle\sum_y y \mathbf{P}_{Y_d}(y) \mathbf{E}[X(y)]. \\ &\leq \textstyle\frac{3}{4} \sum_y y \mathbf{P}_{Y_d}(y). \\ &\leq \textstyle\frac{3}{4} \mathbf{E}[Y_d]. \end{aligned}$$

Note that we have used the bound

$$\mathbf{E}[X(y)] = \sum_z f(y,z) \mathbf{P}_{Z_d \mid Y_d}(z \mid y) \leq \frac{3}{4},$$

which we established above.

We thus conclude that $\mathbf{E}[Y_{d+1}] \leq \frac{3}{4}\mathbf{E}[Y_d]$, which this trivially solves to the bound given in the theorem, since at $d = 0$ the input size is n. □

Remark. Note that the proof of this theorem would have been relatively easy if the successive choices made by the algorithm were independent but they are not, because the size to the algorithm at each recursive call depends on prior choices of pivots.

4.1.1 Work Analysis

We now have all the ingredients to complete the analysis.

The work at each level of the recursive calls is linear in the size of the input and thus can be written as $W_{\texttt{select}}(n) \leq k_1 n + k_2$, where n is the input size. Because at least one element, the pivot, is taken out of the input for the recursive call at each level, there are at most n levels of recursion. Thus, by using the theorem below, we can bound the expected

work as

$$\begin{aligned}
\mathbf{E}\left[W_{\texttt{select}}(n)\right] &\leq \sum_{i=0}^{n}(k_1\mathbf{E}\left[Y_i\right] + k_2) \\
\mathbf{E}\left[W_{\texttt{select}}(n)\right] &\leq \sum_{i=0}^{n}(k_1 n\left(\frac{3}{4}\right)^i + k_2) \\
&\leq k_1 n\left(\sum_{i=0}^{n}\left(\frac{3}{4}\right)^i\right) + k_2 n \\
&\leq 4k_1 n + k_2 n \\
&\in O(n).
\end{aligned}$$

Note. As we shall see in subsequent chapters, many contraction algorithms have the same property that the problem instances go down by an expected constant factor at each contraction step.

4.1.2 Span Analysis

Because the span at each level is $O(\lg n)$ and because the depth is at most n, we can bound the span of the algorithm by $O(n \lg n)$ in the worst case. But we expect the average span to be a lot better because chances of picking a poor pivot over and over again, which would be required for the linear span is unlikely. To bound the span in the expected case, we shall use bound on the expected input size established above, and bound the number of levels in `select` with high probability.

A High-Probability Bound for Span. Consider depth $d = 10 \lg n$. At this depth, the expected instance size upper bounded by

$$n\left(\frac{3}{4}\right)^{10 \lg n}.$$

With a little math this is equal to $n \times n^{-10 \lg(4/3)} \approx n^{-3.15}$.

By Markov's inequality , if the expected size is at most $n^{-3.15}$ then the probability of that the size is at least 1 is bounded by

$$\mathbf{P}\left[Y_{10 \lg n} \geq 1\right] \leq \frac{E[Y_{10 \lg n}]}{1} = \frac{1}{n^{3.15}} \leq \frac{1}{n^3}.$$

In applying Markov's inequality, we choose 1, because we know that the algorithm terminates for that input size. We have therefore shown that the number of steps is $O(\lg n)$ with high probability, i.e., with probability $1 - \frac{1}{n^3}$. Each step has span $O(\lg n)$ so the overall span is $O(\lg^2 n)$ with high probability.

Note that by increasing the constant factor from 10 to 20, we could decrease the probability to $n^{-7.15}$, which is extremely unlikely: for $n = 10^6$ this is 10^{-42}.

An Expected Bound for Span. Using the high probability bound, we can bound the expected span by using the Total Expectations Theorem .

For brevity let the random variable Y be defined as $Y = Y_{10 \lg n}$,

$$\begin{array}{rcl} \mathbf{E}[S] & = & \sum_y \mathbf{P}_Y(y)\mathbf{E}[S \mid Y = y]. \\ & = & \sum_{y \leq 1} \mathbf{P}_Y(y)\mathbf{E}[S \mid Y = y] + \sum_{y > 1} \mathbf{P}_Y(y)\mathbf{E}[S \mid Y = y] \\ & \leq & 1 \cdot O(\lg^2 n) + \frac{1}{n^3} O(n) \\ & = & O(\lg^2 n). \end{array}$$

The expectation calculation above has two terms.

- The first term considers the case of $y < 1$. In this case, we know that the span is $\Theta(\lg^2 n)$, because the span of each recursive call is $\Theta(\lg n)$, as dominated by the span of the *filter* operation. The probability that $y < 1$ is at most 1.

- The second part considers the case of $y \geq 1$. We know that this case happens with probability at most $\frac{1}{n^3}$ and thus can afford to use a loose bound on the expected span, e.g., total expected work, which is $O(n)$. As a result, this part contributes only a constant to the total.

Exercise 34.1. Prove that the pivot tree has $O(\lg n)$ height, and is therefore balanced, with high probability.

Alternative Span Analysis. Let the random variable $X(n) = \max\{|\ell|, |r|\}$, which is the size of larger subsequence. The span of *select* is upper bounded by the sizes of these larger subsequences. Because we use *filter* to compute the subsequences we have the following recurrence for span for input size n

$$S(n) = S(X(n)) + O(\lg n).$$

For the analysis, we shall condition the span on the random variable denoting the size of the maximum half and apply Theorem 32.1.

$$\mathbf{E}[S(n)] = \sum_{m=n/2}^{n} \mathbf{P}[X(n) = m] \cdot \mathbf{E}[S(n) \mid (X(n) = m)].$$

The rest is algebra

$$\begin{aligned}
\mathbf{E}[S(n)] &= \sum_{x=n/2}^{n} \mathbf{P}[X(n)=x] \cdot \mathbf{E}[S(n) \mid (X(n)=x)] \\
&\leq \mathbf{P}\left[X(n) \leq \frac{3n}{4}\right] \cdot \mathbf{E}\left[S(\frac{3n}{4})\right] + \mathbf{P}\left[X(n) > \frac{3n}{4}\right] \cdot \mathbf{E}[S(n)] + c \cdot \lg n \\
&\leq \frac{1}{2}\mathbf{E}\left[S(\frac{3n}{4})\right] + \frac{1}{2}\mathbf{E}[S(n)] + c \cdot \lg n \\
&\implies \mathbf{E}[S(n)] \leq \mathbf{E}\left[S(\frac{3n}{4})\right] + 2c \lg n.
\end{aligned}$$

This is a recursion in $\mathbf{E}[S(\cdot)]$ and solves easily to $\mathbf{E}[S(n)] = O(\lg^2 n)$.