Parallel And Sequential Data Structures and Algorithms

Analysis of Quicksort

Learning Objectives

- Prove high probability bounds on the span of Quicksort and expected bounds on the work of quicksort.
- Illustrate some general algorithm analysis and probability techniques.

Quicksort

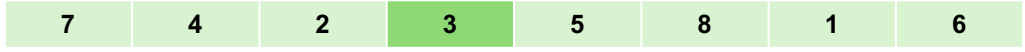
```
fun quicksort(S : sequence<T>) -> sequence<T>:
  if length(S) <= 1:
    return S
  p = a uniformly random element of S
  L = filter(fn x => (x < p), S)
  M = filter(fn x => (x = p), S)
  R = filter(fn x => (x > p), S)
  (SortedL, SortedR) = parallel (quicksort(L), quicksort(R))
  return SortedL + M + SortedR
```

Note that this implementation is parallel, and can handle sequences that have duplicate elements.

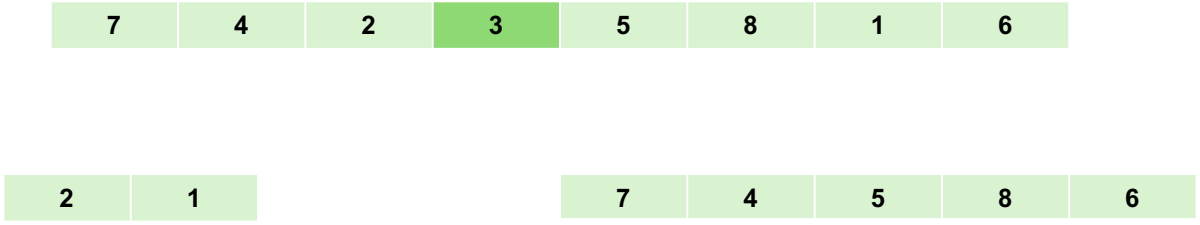
Sample Run of Quicksort

7	4	2	3	5	8	1	6

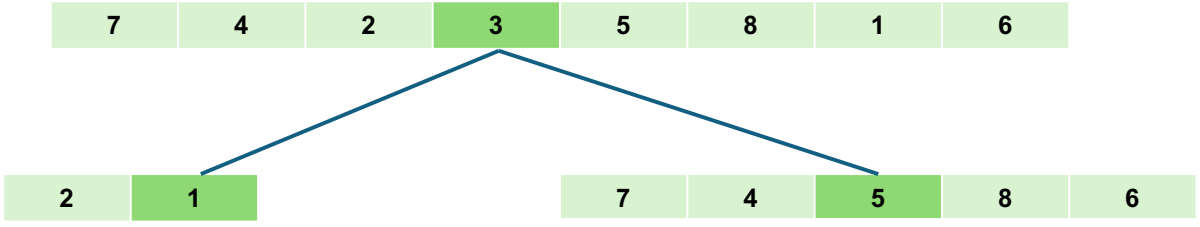
Sample Run of Quicksort



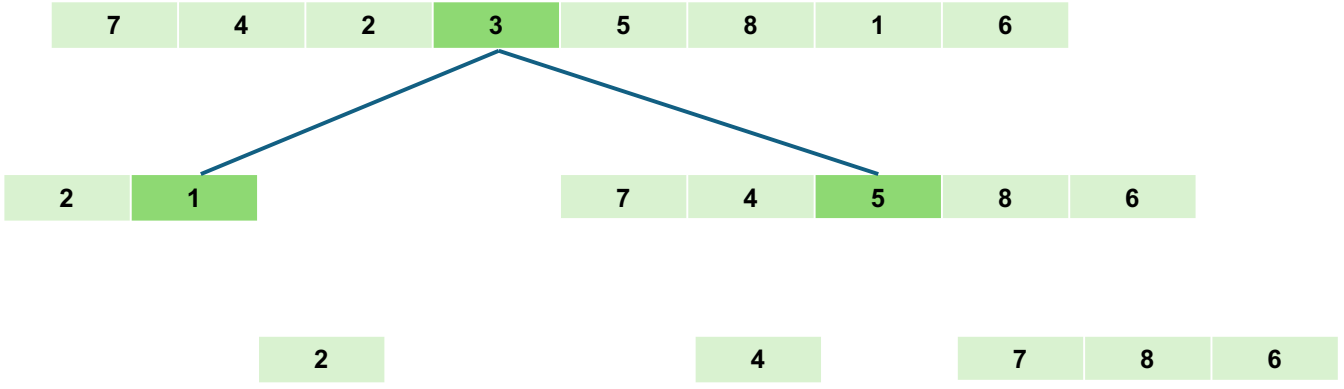
Sample Run of Quicksort



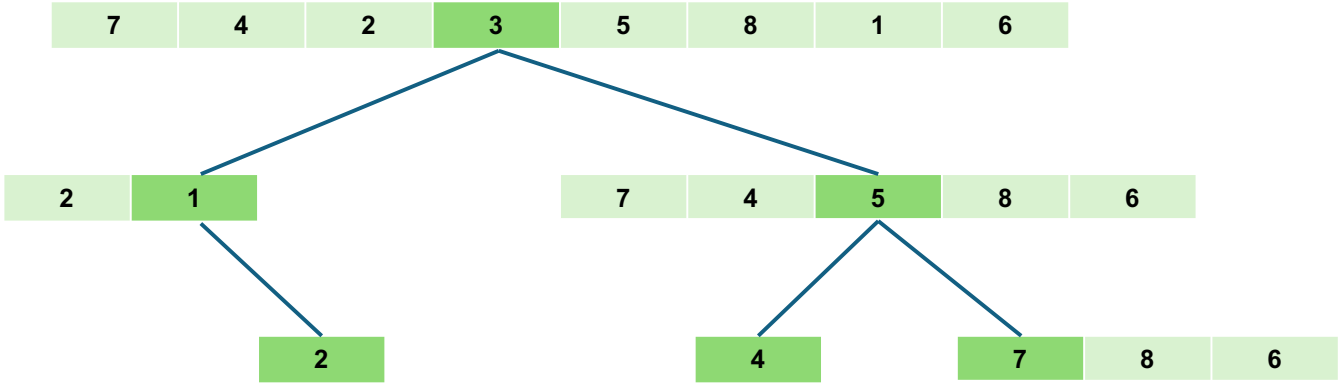
Sample Run of Quicksort



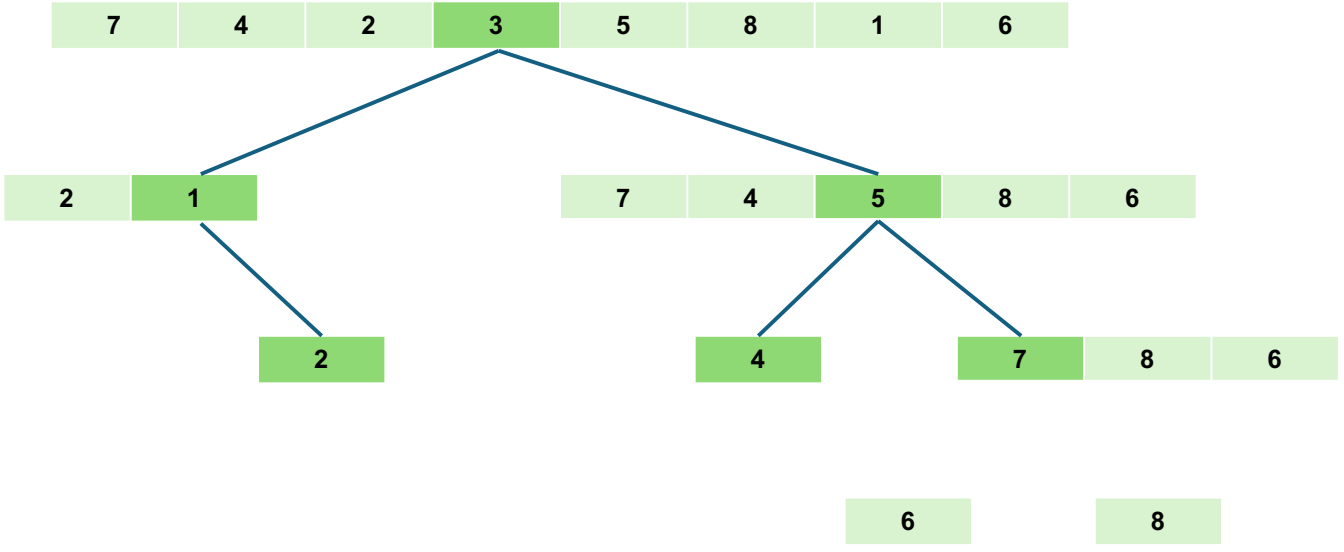
Sample Run of Quicksort



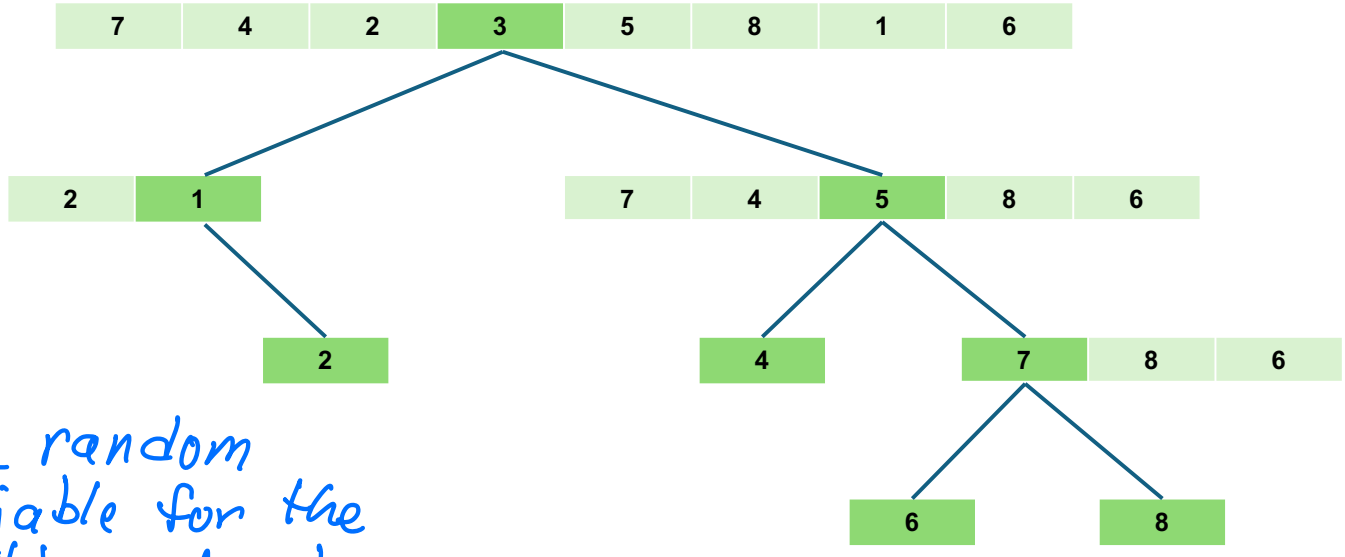
Sample Run of Quicksort



Sample Run of Quicksort

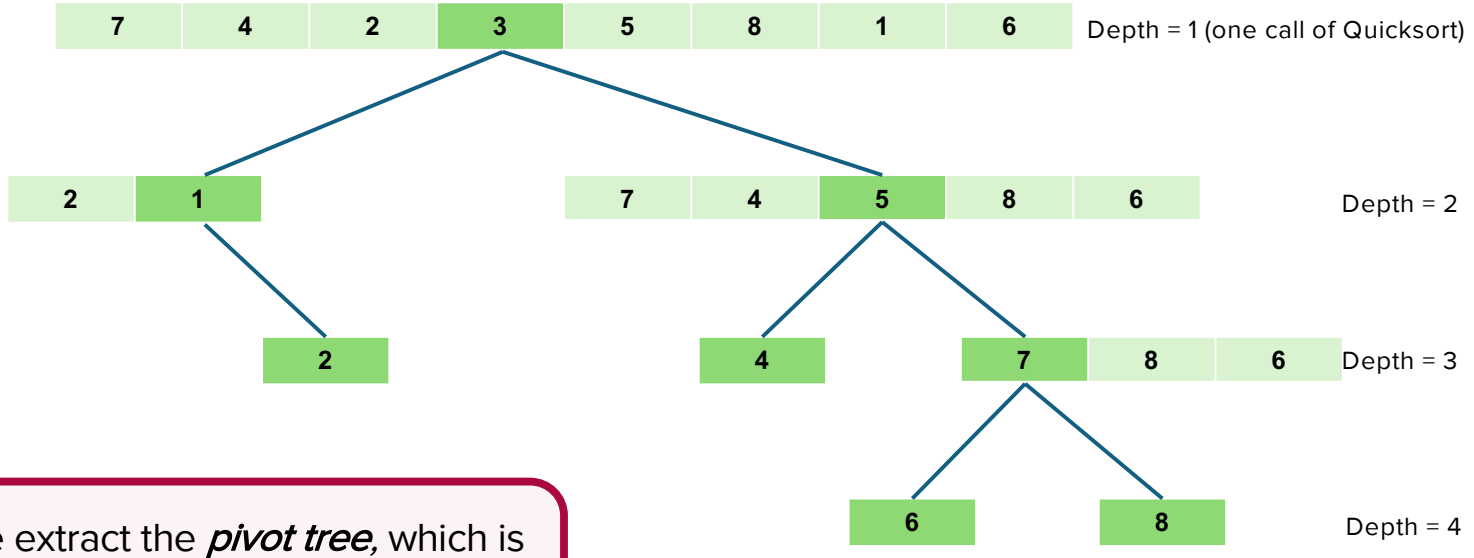


Sample Run of Quicksort



R(8,0) = the random variable for the depth of the smallest element (see page 21)

Sample Run of Quicksort



From this we extract the ***pivot tree,*** which is a binary tree in which every element of the input sequence occurs as a node.

Work/Span Analysis

- The span of quicksort is $O((\underbrace{\log n}_{\text{This } \log n \text{ comes from } \texttt{filter}}) * (depth\ of\ the\ tree))$
- The work of quicksort is $O(\underbrace{n}_{\text{This } n \text{ comes from } \texttt{filter}} * (depth\ of\ the\ tree))$

Therefore, our analysis for work/span is reduced to finding the **depth of the tree**.

Proving Quicksort Bounds with High Probability

The Quicksort Depth Theorem

Theorem (Quicksort Depth Theorem): Let $D(n)$ be a random variable that is the recursion depth of a run of quicksort on a sequence of length n of unique elements. Then $D(n) \in O(\log n)$ with high probability. Or, more specifically, $D(n) \leq 10 \log_2 n$ with high probability.

Proof: We will come back to this proof later, but first we must go back to a previous theorem…

Max Preserves "with high probability"

Theorem (Max Preserves w.h.p.): Let $S(n)$ be a non-negative random variable, and let $T(n) = \max(S(n), \ldots, S(n))$, where there are n (not necessarily independent) copies of $S(n)$ in the max. If $S(n) \leq f(n)$ w.h.p. then $T(n) \leq 2f(n)$ w.h.p.

Therefore, if $S(n) \in O(f(n))$ w.h.p. then so is $T(n)$.

Max Preserves with high probability

Max Preserves w.h.p.: Let $S(n)$ be a non-negative random variable, and let $T(n) = \max(S(n), \ldots, S(n))$, where there are n (not necessarily independent) copies of S(n) in the max. If $S(n) \leq f(n)$ w.h.p. then $T(n) \leq 2f(n)$ w.h.p. It follows that if $S(n) \in O(f(n))$ w.h.p. then so is $T(n)$.

Proof:

$$\Pr(T(n) > (k+1)f(n)) \leq n \Pr(S(n) > (k+1)f(n)) \qquad \text{[Union Bound]}$$

Max Preserves with high probability

Max Preserves w.h.p.: Let $S(n)$ be a non-negative random variable, and let $T(n) = \max(S(n), \ldots, S(n))$, where there are n (not necessarily independent) copies of S(n) in the max. If $S(n) \leq f(n)$ w.h.p. then $T(n) \leq 2f(n)$ w.h.p. It follows that if $S(n) \in O(f(n))$ w.h.p. then so is $T(n)$.

Proof:

$$\Pr(T(n) > (k+1)f(n)) \leq n \Pr(S(n) > (k+1)f(n))$$ [Union Bound]

$\leq \frac{1}{n^{k+1}}$

But the RHS is at most $\frac{n}{n^{k+1}} = \frac{1}{n^k}$ by virtue of $S(n) \leq f(n)$ w.h.p.

$Pr(S(n) > k f(n)) < \frac{1}{n^k}$

Max Preserves with high probability

Max Preserves w.h.p.: Let $S(n)$ be a non-negative random variable, and let $T(n) = \max(S_1(n), \ldots, S_n(n))$, where there are n (not necessarily independent) copies of S(n) in the max. If $S(n) \leq f(n)$ w.h.p. then $T(n) \leq 2f(n)$ w.h.p. It follows that if $S(n) \in O(f(n))$ w.h.p. then so is $T(n)$.

Proof:

$$\Pr(T(n) > (k+1)f(n)) \leq n \Pr(S(n) > (k+1)f(n)) \quad \text{[Union Bound]}$$

But the RHS is at most $\frac{n}{n^{k+1}} = \frac{1}{n^k}$ by virtue of $S(n) \leq f(n)$ w.h.p.

Also note that $2k \geq k+1$, so we can write:

Let A be the event that $T(n) \geq (k+1)f(n)$
" E_i " " " " $S_i(n)$ " " "

$$\Pr(T(n) > k \cdot 2f(n)) \leq \Pr(T(n) > (k+1)f(n)) \leq 1/n^k$$

Then
$A = E_1 \cup E_2 \cup \cdots \cup E_n$

which is the definition of $T(n) \leq 2f(n)$ w.h.p. ■

Back to Quicksort

- With this theorem under our belt, we can now go back to proving the **Quicksort Depth Theorem**.
- We will also make use of the following theorem from the last lecture, the **QuickSelect Recursion Depth Theorem**.

Theorem (QuickSelect Recursion Depth): Ret $R(n)$ be a random variable that is the recursive depth (number of pivot selections) of `quickselect` on a sequence of length n, then $R(\mathrm{n}) \leq 5 \log_2 n$ w.h.p.

Back to the Quicksort Depth Theorem

Theorem (Quicksort Depth Theorem): Let $D(n)$ be a random variable that is the recursion depth of a run of Quicksort on a sequence of length n of unique elements. Then $D(n) \in O(\log n)$ with high probability. Or, more specifically, $D(n) \leq 10 \log_2 n$ with high probability.

Proof: The key idea of the proof is that there is a perfect correspondence between the Quicksort tree and running Quickselect on all elements.

Say we have a sequence of elements with (no duplicates).

Let $R(n, k)$ be the random variable which is the recursion depth Quickselect uses to find the element of the input sequence of rank k.

$R(n, k)$ is also equal to the depth of that element (counting the root as 1) in the quicksort tree.

Back to the Quicksort Depth Theorem

Theorem (Quicksort Depth Theorem): Let $D(n)$ be a random variable that is the recursion depth of a run of Quicksort on a sequence of length n of unique elements. Then $D(n) \in O(\log n)$ with high probability. Or, more specifically, $D(n) \leq 10 \log_2 n$ with high probability.

In proving the QuickSelect Recursion Depth Theorem we did not consider the role of k (the rank being searched) on the depth. We avoided it by making the pessimistic assumption that the recursion always chooses the larger half to recurse into.

Back to the Quicksort Depth Theorem

Theorem (Quicksort Depth Theorem): Let $D(n)$ be a random variable that is the recursion depth of a run of Quicksort on a sequence of length n of unique elements. Then $D(n) \in O(\log n)$ with high probability. Or, more specifically, $D(n) \leq 10 \log_2 n$ with high probability.

In proving the QuickSelect Recursion Depth Theorem we did not consider the role of k (the rank being searched) on the depth. We avoided it by making the pessimistic assumption that the recursion always chooses the larger half to recurse into.

Therefore the resulting random variable (call it $R(n)$) has its probability mass "pushed to the right" compared to $R(n, k)$.

Back to the Quicksort Depth Theorem

Theorem (Quicksort Depth Theorem): Let $D(n)$ be a random variable that is the recursion depth of a run of Quicksort on a sequence of length n of unique elements. Then $D(n) \in O(\log n)$ with high probability. Or, more specifically, $D(n) \leq 10 \log_2 n$ with high probability.

In proving the QuickSelect Recursion Depth Theorem we did not consider the role of k (the rank being searched) on the depth. We avoided it by making the pessimistic assumption that the recursion always chooses the larger half to recurse into.

Therefore the resulting random variable (call it $R(n)$) has its probability mass "pushed to the right" compared to $R(n, k)$.

There is compelling intuition for this statement. In our analysis of quickselect we let X_r denote the size of the array at depth r. Our modified algorithm operates in a way that makes X_r "bigger" (by choosing the larger half), and it therefore also increases the depth and thus makes $R(n)$ bigger. Rather than depending on this intuition we will make this more rigorous by using using stochastic dominance.

Stochastic Dominance

Definition (Stochastic Dominance): A random variable A is **stochastically dominant** over a random variable B if for all x we have:

$$\Pr(B \geq x) \leq \Pr(A \geq x).$$

And we denote this in symbols by $B \preccurlyeq A$

Stochastic Dominance

Definition (Stochastic Dominance): A random variable A is **stochastically dominant** over a random variable B if for all x we have:

$$\Pr(B \geq x) \leq \Pr(A \geq x).$$

And we denote this in symbols by $B \preccurlyeq A$

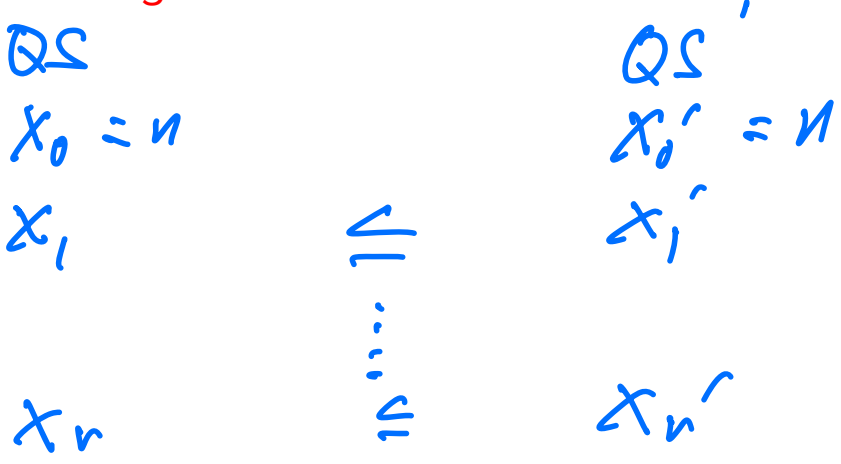
This definition has a couple of useful properties:

1. Any tail bound that we prove about A is also true of B, because any "tail of A" is an upper bound of the same tail of B.
2. Stochastic dominance is transitive. So, if $C \preccurlyeq B$ and $B \preccurlyeq A$ the $C \preccurlyeq A$.

Stochastic Dominance in Quickselect

Consider algorithms quickselect and quickselect'. The former is the original algorithm, the latter is the one which has been modified to always recurse into the larger part.

We're going to run these two algorithms in parallel starting from the same initial sequence. And using the same random bits.



Stochastic Dominance in Quickselect

Consider algorithms quickselect and quickselect'. The former is the original algorithm, the latter is the one which has been modified to always recurse into the larger part.

We're going to run these two algorithms in parallel starting from the same initial sequence. And using the same random bits.

We call X_0 the size of the starting sequence given to quickselect.

We call X_0' the size of the starting sequence given to quickselect'.

These are equal.

Denote these sizes at level r by X_r and X_r'. We will see that $X_r \leq X_r$'.

Stochastic Dominance in Quickselect

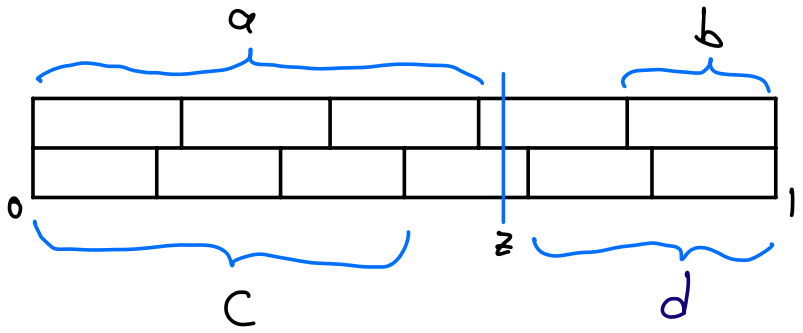
The algorithms will pick a pivot in the following fashion:

- The current size is X and the input sequence is S.
- A random real number $z \in [0,1]$ is selected.
- The pivot element is $S[\lfloor z \cdot X \rfloor]$ (equally likely to be $0,1 \ldots, X-1$.)

quickselect will apply this process to its size X_r and quickselect' will apply it to its size X_r', using the same random value z.

Next we show that $X_r \leq X_r$'.

Stochastic Dominance in Quickselect



$X_r = 5$
$X'_r = 6$

We have $c \geq a \not\geq d \geq b$

$\Rightarrow \max(c,d) \geq a$

$\geq b$

$X_{r+1} = a$ or b $\Longrightarrow X_{r+1} \leq X'_{r+1}$

$X'_{r+1} = \max(c,d)$

Scratch Work Space

$X_r : \Omega \to \mathbb{R}$

$X_r' : \Omega \to \mathbb{R}$

$X_r(q) \leq X_r$

$\forall q \in \Omega$

$\Downarrow$

$X_r \preccurlyeq X_r'$

Ω

1.1	6.7	1.3
3.4	8.8	1.2
2.2	7.7	2.2
4.5	2.8	2.4

Back to the Quicksort Depth Theorem

Theorem (Quicksort Depth Theorem): Let $D(n)$ be a random variable that is the recursion depth of a run of Quicksort on a sequence of length n of unique elements. Then $D(n) \in O(\log n)$ with high probability. Or, more specifically, $D(n) \leq 10 \log_2 n$ with high probability.

Claim: $R(n)$ (quickselect' depth) is stochastically dominant over $X(n, k)$.

This follows from the above proof that $X_r \leq X_r'$. Because it shows that quickselect will stop no later than quickselect' will stop.

Back to the Quicksort Depth Theorem

Theorem (Quicksort Depth Theorem): Let $D(n)$ be a random variable that is the recursion depth of a run of Quicksort on a sequence of length n of unique elements. Then $D(n) \in O(\log n)$ with high probability. Or, more specifically, $D(n) \leq 10 \log_2 n$ with high probability.

Claim: $R(n)$ (quickselect' depth) is stochastically dominant over $X(n, k)$.

This follows from the above proof that $X_r \leq X_r'$. Because it shows that quickselect will stop no later than quickselect' will stop.

We have shown that $R(n) \leq 5 \log_2 n$ w.h.p.

It follows immediately by stochastic dominance that $X(n, k) \leq 5 \log_2 n$ w.h.p.

Finishing the Quicksort Depth Theorem

Theorem (Quicksort Depth Theorem): Let $D(n)$ be a random variable that is the recursion depth of a run of Quicksort on a sequence of length n of unique elements. Then $D(n) \in O(\log n)$ with high probability. Or, more specifically, $D(n) <= 10 \log_2 n$ with high probability.

$$D(n) = \max\big(X(n,0), X(n,1), \ldots, X(n,n-1)\big)$$

We can now apply the **Max Preserves w.h.p.** Theorem.

Since each $X(n,k)$ is $5 \log_2 n$ w.h.p, it follows that $D(n)$ is $10 \log_2 n$ w.h.p.

Notice that the X(n,k) s are not independent. (If one is an ancestor of the other in the pivot tree, then they share random choices.) But the theorem does not require independence. ■

Proving the Expected Bounds of Quicksort

*not the same thing as with high probability

Expected Depth of Quicksort

Theorem (Expected Depth of Quicksort): The expected depth of quicksort is $O(\log n)$.

Proof: Again we let $D(n)$ be the random variable which is the depth of recursion of a run of quicksort.

Expected Depth of Quicksort

Theorem (Expected Depth of Quicksort): The expected depth of quicksort is $O(\log n)$.

Proof: Again we let $D(n)$ be the random variable which is the depth of recursion of a run of quicksort.

Next we make the observation that $D(n) \leq n$. i.e. quicksort always terminates by level n.

And we have also established the fact that $D(n) \leq 10 \log_2 n$ w.h.p.

Expected Depth of Quicksort

Theorem (Expected Depth of Quicksort): The expected depth of quicksort is $O(\log n)$.

If $n \leq 10 \log_2 n$ then

$$D(n) \leq 10 \log_2 n = O(\log n).$$

Expected Depth of Quicksort

Theorem (Expected Depth of Quicksort): The expected depth of quicksort is $O(\log n)$.

If $n \leq 10 \log_2 n$ then

$$D(n) \leq 10 \log_2 n = O(\log n).$$

If $n > 10 \log_2 n$, then in this case we have:

$$\Pr(D(n) \geq n) \leq \Pr(D(n) > 10 \log_2 n) < \frac{1}{n}.$$

Expected Depth of Quicksort

Theorem (Expected Depth of Quicksort): The expected depth of quicksort is $O(\log n)$.

If $n \leq 10 \log_2 n$ then

$$D(n) \leq 10 \log_2 n = O(\log n).$$

If $n > 10 \log_2 n$, then in this case we have:

$$\Pr(D(n) \geq n) \leq \Pr(D(n) > 10 \log_2 n) < \frac{1}{n}.$$

So the contribution to $E[D(n)]$ due to the range where

$D(n) \leq 10 \log_2 n$ is at most $10 \log_2 n \in O(\log n)$.

Expected Depth of Quicksort

Theorem (Expected Depth of Quicksort): The expected depth of quicksort is $O(\log n)$.

If $n \leq 10 \log_2 n$ then

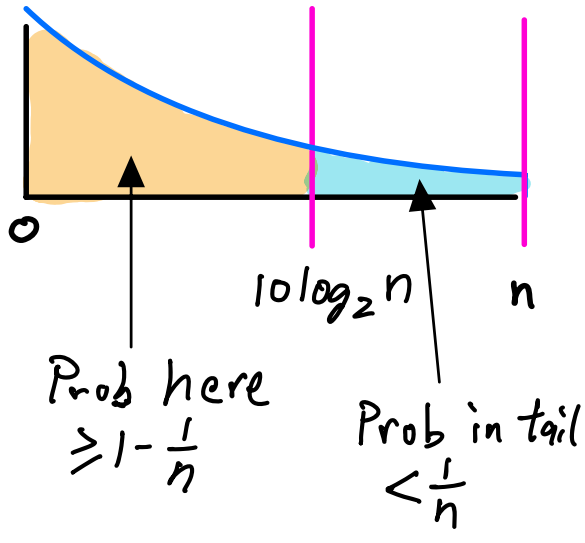
$$D(n) \leq 10 \log_2 n = O(\log n).$$

If $n > 10 \log_2 n$, then in this case we have:

$$\Pr(D(n) \geq n) \leq \Pr(D(n) > 10 \log_2 n) < \frac{1}{n}.$$

So the contribution to $E[D(n)]$ due to the range where $D(n) \leq 10 \log_2 n$ is at most $10 \log_2 n \in O(\log n)$

And the contribution to $E[D(n)]$ due to the range where $D(n) > 10 \log_2 n$ is at most $n\left(\frac{1}{n}\right) = 1$.



Expected Depth of Quicksort

Theorem (Expected Depth of Quicksort): The expected depth of quicksort is $O(\log n)$.

If $n \leq 10 \log_2 n$ then

$$D(n) \leq 10 \log_2 n = O(\log n).$$

If $n > 10 \log_2 n$, then in this case we have:

$$\Pr(D(n) \geq n) \leq \Pr(D(n) > 10 \log_2 n) < \frac{1}{n}.$$

So the contribution to $E[D(n)]$ due to the range where

$$D(n) \leq 10 \log_2 n \text{ is at most } 10 \log_2 n \in O(\log n)$$

And the contribution to $E[D(n)]$ due to the range where

$$D(n) > 10 \log_2 n \text{ is at most } n * \frac{1}{n} = 1.$$

We conclude that $E[D(n)] = O(\log n)$. ■

Quicksort Bounds

Theorem (Quicksort Bounds):
The span of Quicksort is $O(\log^2(n))$ w.h.p. and in expectation.
The work of Quicksort is $O(n \log n)$ w.h.p. and in expectation.

Proof:

These follow immediately from the fact filter has $O(\log n)$ span and $O(n)$ work, and that the depth of quicksort is $O(\log n)$ w.h.p. and in expectation. ■

Summary

- Quicksort depth is $O(\log n)$ with high probability and in expectation, shown by relating the quicksort recursion tree to Quickselect depth and using stochastic dominance plus max-preserves - w.h.p. arguments.
- Span and work follow from tree depth: span is $O(\log^2 n)$ w.h.p. ∈ expectation (tree depth × filter span), and work is $O(n \log n)$ w.h.p. ∈ expectation (tree depth × linear work per level).