3) Quicksort Recurrence Relations

- Recall that sequential Quicksort consists of
  - O(1) Picking a pivot
  - O(n) Partition data into
    - A: Less than pivot
    - B: Pivot
    - C: Greater than pivot
  - 2 T(n/2) – Recursively, sort each of the two halves, A and C.
- $T(n)=1+n+2T(n/2) = O(n \log n)$

To parallelize step 3 (recursion)

- Each partition can be done at the same, so 2T(n/2) becomes time 1 T(n/2)
- Whole relation becomes: T(n)=1+n+T(n/2)
- Ignoring the constant time pivot-picking:
- T(n) = n + T(n/2)

Solve recurrence relation

- $T(n) = n + T(n/2)$
- $T(n) = n + (n/2 + T(n/4))$
- $T(n) = n + (n/2 + (n/4 + T(n/8)))$
- $T(n) = n*(1+1/2+1/4+\ldots+1/2^{k-1})+T(n/2^k)$

Assume $T(1)=C$, that is, that to sort 1 element takes a constant C units of time.

Substitute in base case $T(1)=1$ and solve for k:
$n/2^k=1$
$k = \log n$

- $T(n) = n*(1+1/2+1/4+\ldots+1/2^{\log n-1})+C$
- Sum of geometric series (1+1/2+1/4+…) converges to 2
- $T(n) = 2n+C$ which is $O(n)$, linear

4) Parallelizing step 2, partition

- Do 2 filters, one to filter less-than-pivot partition, one to filter greater-than-pivot partition.
- Filter is work O(n), span O(log n)
- So total quicksort is now (partition+recursion):
- $T(n) = O(\log n) + T(n/2)$

Solve recurrence relation

- T(n) = log n + T(n/2) *expand out recurrence*
- T(n) = log n + (log(n/2) + T(n/4))
- T(n) = log n + log(n/2) + log(n/4) + T(n/8)
- T(n) = log n + log(n/2) + log(n/4) + log(n/8) + T(n/16)
- T(n) = log n +(log n – log 2) + (log n – log 4) + (log n – log 8) + T(n/16)
- T(n) = 4*log n – log 2 – log 4 – log 8 + T(n/16)
- T(n) = 4*log n – 1 – 2 – 3 + T(n/2^4) *because we're doing log base 2*
- T(n) = k*log n - (1+2+3+…+(k-1))+T(n/2^k)
- T(n) = k*log n – (k(k-1))/2 + T(n/2^k)
- As usual, assuming T(1)=C, set n/2^k=1, gives k=log n
- T(n) = (log n)*(log n) – ((log n-1)(log n))/2 + C
- T(n) = (log n)*(log n) – ((log n * log n)-log n)/2 + C
- Which is O(log n * log n)