

There are many ways to solve this problem. We present 2 solutions (but omit the correctness proofs). Suppose you have a routine $\text{Med}(A, B)$ for finding the median of $A++B$ (you probably did this in 15-451). It is actually easy to convert this into a parallel algorithm with work $O(\log n)$ and depth $O(\log n)$. Based on this routine, the following divide-and-conquer algorithm will give you the desired bounds:

```

Merge1(A, B)
1.  $m = \text{Med}(A, B)$ 
2.  $(A_L, A_R) = (\{x \in A : x \leq m\}, \{x \in A : x > m\})$ 
3.  $(B_L, B_R) = (\{x \in B : x \leq m\}, \{x \in B : x > m\})$ 
4. return Merge1( $A_L, B_L$ )++Merge1( $A_R, B_R$ )
    
```

Lines 2 and 3 can be accomplished in $O(\log n)$ work and depth by two binary searches, which locate where the split points should be in the arrays. Note that because m is the median of the merged array, we know that $|A_L| + |B_L| \leq (|A| + |B|)/2$ and $|A_R| + |B_R| \leq (|A| + |B|)/2$. Let $W(n)$ and $D(n)$ denote the work and depth (resp.) of running $\text{Merge1}(A, B)$, where $n = |A| + |B|$. We have the following recurrences:

$$W(n) \leq 2 \cdot W(n/2) + O(\log n)$$

$$D(n) \leq D(n/2) + O(\log n).$$

Therefore, we know that $W(n) = O(n)$ (by the Master formula) and $D(n) = O\left(\sum_{i=1}^{\log n} i\right) = O(\log^2 n)$.

Alternatively you can use a single binary search per recursive call. This will complicate the analysis—we had some (inconclusive) discussion about this in class.

```

Merge2(A, B) // Assume |A| ≥ |B|
0.  $\ell = \lfloor |B|/2 \rfloor$ 
1.  $m = B[\ell]$ 
2.  $(A_L, A_R) = (\{x \in A : x \leq m\}, \{x \in A : x > m\})$ 
3.  $(B_L, B_R) = (B[1..\ell], B[\ell + 1..|B|])$ 
4. return Merge2( $A_L, B_L$ )++Merge2( $A_R, B_R$ )
    
```

Line 3 is easy. Line 2 can be accomplished in $O(\log |A|)$ work and depth by a single binary search. In this case, it is no longer true that we will reduce the problem size by a half or any constant fraction of $|A| + |B|$. Howsoever, it can still be analyzed and surprisingly has depth $O(\log^2 n)$ and work $O(n)$. The analysis for depth is straightforward: each level has depth $O(\log n)$ and we have at most $O(\log n)$ levels. Let $W(a, b)$ be the work for $\text{Merge2}(A, B)$, where $a = |A|$ and $b = |B|$. It is easy to see that $W(a, b) = W(a', b/2) + W(a - a', b/2) + O(\log a)$.

If you draw the recursion tree, you will see that the i -th level (counting from $i = 0$ at the root) has 2^i nodes. Furthermore, in level i , if x_j is the size of A of the j -th node, then $\sum_j x_j = |A| = n$. By Jensen's inequality, we have $\sum_j \log x_j \leq \sum_j \log(|A|/2^i) = 2^i \cdot (\log n - i)$.

Therefore, the total work is

$$W(n, n) = \sum_{i=0}^{\log n} 2^i \cdot (\log n - i) \leq 2n \log n - \sum_{i=1}^{\log n} i \cdot 2^i = 2n \log n - (2 + 2n \log n - 2n) = O(n)$$