Reading:

Lectures 16-17: 4.3; section at end of 4.4 on Merge Sort
Lectures 17-18: 5.1

Homework (due Monday, 10/13):

Odd-numbered self-checking exercises:

4.3: 1(d), 5(a,b,e), 13, 17, 39
4.4: 47
5.1: 15, 19, 29, 39

Problems to hand in:

4.3: 6(b), 12, 30, Ch. 4 Suppl. Ex. 18 [hint: determine \( f_n \pmod{3} \) for every \( n \)]
4.4: 48(a,b). For each problem, either find an algorithm that always uses fewer than the worst-case \( m + n - 1 \) steps required by 4.4, Algorithm 10, or show that no such algorithm is possible.
5.1: 16 [hint: subtract], 20(a-f), 28, 38, 42, 54, 58

(A) We define the height \( h(T) \) of a rooted tree \( T \) (4.3, Definition 4) analogously to how it is done for full binary trees in 4.3, Definition 7, by replacing the maximum over \( T_1, T_2 \) in the recursive step by the maximum over all the constituent trees \( T_1, \ldots, T_n \). We define the set of leaves recursively, analogously to the preamble to 4.3, Exercise 44, to be \( \{r\} \) in the case that \( T \) consists only of a root \( r \), and otherwise to be the union of the sets of leaves of the trees \( T_1, \ldots, T_n \). Let \( n(T) \) denotes the number of nodes in \( T \), and \( l(T) \) the number of leaves. Prove that \( n(T) \leq l(T)h(T) + 1 \) for every rooted tree \( T \).