1. Analyzing an iterative algorithm (LOOPS)
   1.1 Problem instance and show how the algorithm works
   1.2 Correctness: Loop Invariant
   1.3 Time Analysis: number of comparisons
   Algorithms: MinSort, BubbleSort, InsertionSort, Linear Search, Binary Search, Summing Triple, Intersection

   about 30-35 P

2. Analyzing a recursive algorithm (NO LOOPS)
   2.1 Problem instance and show how the algorithm works
   2.2 Correctness: induction on n
   2.3 Time Analysis: unraveling the recurrence
   \( T(n) = T(n-1)+1 \) (#multiplications, Computing \( 2^n \))
   \( T(n)=T(n-1)+c \) (#comparisons, Recursive Merge)
   \( T(n)=2T(n-1)+1 \) (#moves, Towers of Hanoi)
   Algorithms: Exponentiation, Towers of Hanoi, Merging sorted arrays, Binary Strings, Ternary Strings

3. Divide-And-Conquer
   3.1 Problem instance and show how the algorithm works
   3.2 Correctness: strong induction on n
   3.3 Time Analysis: unraveling the recurrence
   \( T(n)=2T(n/2)+cn \) (#comparisons, MergeSort)
   \( T(n)=3T(n/2)+cn \) (#single-digit mult. Karatsuba)
   \( T(n)=4T(n/2)+cn \) (#single-digit mult. d&c)
   Algorithms: MergeSort, Karatsuba

   2. and 3. about 30-35 P

4. Big O
   4.1 Classifications of running times
   4.2 Dependencies between running time classes (e.g. \( O(n \log n) \) and \( O(n^2) \))

   about 10-15 P