1. For *n* items Insertion sort runs in \(8n^2\) steps and Merge Sort runs in \(64n \log n\) steps. For what value of *n* does Insertion Sort beat Merge Sort? \(8n^2 \leq 64n \log n\) hence *n* \(\leq 8 \log n\). This inequality holds for \(2 < n < 43\). The crossover value between *n* and \(8 \log n\) occurs between \(n = 32\) and \(n = 64\). Using calculator you can narrow it down to 43. To speed up Merge Sort for small data sets, modify the algorithm such that it only keeps on dividing if *n* \(> 43\) otherwise call Insertion Sort.

2. Selection Sort(A, n)
   for *i* = 1 to *n* - 1 do
     % find the smallest element in A[i] ... A[n]
     min = A[i]; loc = *i* ;
     for *j* = *i* + 1 to *n* do
       if A[*j*] < min then
         min = A[*j*]; loc = *j*;  
       endif
     end
     swap A[*i*] and A[loc]
   end
   We have two loops from 1...*n* - 1 and *i*...*n*:
   \[
   T(n) = \sum_{i=1}^{n-1}(c_1 + \sum_{j=i+1}^{n} c_2) = \sum_{i=1}^{n-1}(c_1 + c_2(n - i))
   \]
   \[
   = c_1(n - 1) + c_2 \sum_{i=1}^{n} i = c_1(n - 1) + c_2n(n - 1)/2 \Rightarrow T(n) = \Theta(n^2)
   \]

3. Binary search. Since the array is already sorted we can check the middle element, when the element which we are looking for is smaller then median we recursive call the binary search on the first half of the array, otherwise on the second half of the array. If the two elements are equal then finish. The recurrence \(T(n) = T(\frac{n}{2}) + \Theta(1)\). Solving the recurrence yields \(T(n) = \Theta(\log n)\).

4. Compare the functions \((\frac{3}{2})^n, n^3, \log^2 n, 2^x, 1\).
   \(1 = O(\log^2 n), \log^2 n = O(n^3), n^3 = O((\frac{3}{2})^n), \frac{3}{2})^n = O(2^n)\).

5. a. \(T(n) = 4T(n/2) + n^2\)
   Using Master’s theorem: \(a = 4, b = 2, f(n) = n^2, \log_2 4 = 2\)
   Case 2 applies: \(f(n) = \Theta(n^2) \Rightarrow T(n) = \Theta(n^2 \log n)\).
   b. \(T(n) = T(n - 1) + n \text{ for } n \geq 2 \text{ and } T(1) = 1\)
   \(T(n) = T(n - 2) + n - 1 + n = \ldots = 1 + 2 + 3 + \ldots + (n - 1) + n = n(n - 1)/2 \Rightarrow T(n) = \Theta(n^2)\)
   c. \(T(n) = 1 \text{ for } n = 1 \text{ and } T(n) = 3T(\frac{n}{3}) + n \log n \text{ for } n > 1\)
   Master’s theorem: \(a = 3, b = 2f(n) = n \log n, \log_3 3 = 1.585... \Rightarrow 1\). Then \(f(n) = O(n^{\log_3 3 - \epsilon})\) for some \(\epsilon > 0\) (e.g. using limits). Case 1 applies \(T(n) = \Theta(n^{\log_3 3}).\)
   d. \(T(n) = 6T(\frac{n}{2}) + n \log^6 n\)
   \(a = 6, b = 5; f(n) = n \log^6 n, \log_5 6 = 1.11...\)
   Case 1 applies: \(f(n) = O(n^{1.1 - \epsilon}) \Rightarrow T(n) = \Theta(n^{\log_5 6}).\)

6. \(T(n) = 2T(\frac{n}{3}) + 2\) and \(T(1) = 1\). Prove by induction that \(T(n) = 3n - 2\).
   Base case: \(T(1) = 3.2 - 2 = 1\). Induction hypothesis: \(T(k) = 2T(\frac{k}{3}) - 2\) for \(0 < k < n\). To prove: \(T(n) = 2T(\frac{n}{3}) + 2\) use induction hypothesis for \(T(\frac{n}{3})\). Then \(T(n) = 2.3(\frac{n}{3}) - 2 + 2 = 3n - 2\).

7. (10) Problem 4.2-2 (page 72 new book) (same number page 60 old book) \(T(n) = T(n/3) + T(2n/3) + cn\)
   Lower bound is provided by the length of the shortest path in the recurrence tree. In this case the shortest branch has length \(\log_3 n\) and since at each level the cost is \(cn\) then \(T(n) = \Omega(n \log_3 n) = \Omega(n \log n)\).

8. Quicksort. If the pivot is the largest element of the list we will get the worst possible partitioning, where one array will have \(n - 1\) elements. If the quicksort pivots each time with the median value, we get the best case partitioning and the total number of comparisons will be \(T(n) = \Theta(n \log n)\).