Exercise 1. Suppose you have the choice between three algorithms:

(a) Algorithm A solves your problem by dividing it into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

(b) Algorithm B solves problems of size $n$ by recursively solving two subproblems of size $n - 1$ and then combining the solutions in constant time.

(c) Algorithm C solves problems of size $n$ by dividing them into nine subproblems of size $\frac{n}{3}$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

Estimate the running times of each of these algorithms. Which one would you choose?

Exercise 2. Recall the recurrence for Mergesort: $T(1) = 0$; $T(n) = 2T(\frac{n}{2}) + (n - 1)$, for $n > 1$. Prove by induction that

$$T(n) = n \cdot (\log_2 n - 1) + 1 \quad \text{for} \quad n = 2^k, \quad k \geq 1$$

Exercise 3. Analyse the complexity of the following recursive algorithm to test whether a number $x$ occurs in an unordered list $L = [x_1, x_2, \ldots, x_n]$ of size $n$. Take the cost to be the number of list element comparison operations.

**Search**($x, L = [x_1, x_2, \ldots, x_n]$):

- if $x_1 = x$ then return yes
- else if $n > 1$ then return **Search**($x, [x_2, \ldots, x_n]$)
- else return no

*Exercise 4. Analyse the complexity of the following recursive algorithm to test whether a number $x$ occurs in an ordered list $L = [x_1, x_2, \ldots, x_n]$ of size $n$. Take the cost to be the number of list element comparison operations.

**BinarySearch**($x, L = [x_1, x_2, \ldots, x_n]$):

- if $n = 0$ then return no
- else
  - if $x[\lceil \frac{n}{2} \rceil] > x$ then return **BinarySearch**($x, [x_1, \ldots, x[\lceil \frac{n}{2} \rceil]-1]$)
  - else if $x[\lceil \frac{n}{2} \rceil] < x$ return **BinarySearch**($x, [x[\lceil \frac{n}{2} \rceil]+1, \ldots, x_n]$)
  - else return yes