Recall that if two functions f and g eventually satisfy $f(n) \leq c \cdot g(n)$ for some constant c we say that $f(n) = O(g(n))$.

1. Given a sorted list of numbers the binary search algorithm finds the position of a specified number (if it is in the list). It works by selecting the midpoint of the given list, checking whether the element searched for is before or after this position and then recursing on the half of the list that the target must be in. Therefore the running time of this algorithm is given by
$$T(n) \leq \begin{cases} T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + c & \text{if } n > 1, \\ 1 & \text{if } n = 1, \end{cases}$$
for some constant c.

   (a) Find bounds for $T(3)$, $T(5)$, $T(10)$ and $T(15)$.

   (b) Solve the recurrence to find an explicit function f such that $T(n) = O(f(n))$.

2. The median of a set of n numbers is the element in the middle position if they placed in increasing order (assume n is odd if you like). In this question we will look at how quickly we can find the median of a set of numbers.

   (a) Give a $O(n \log n)$ algorithm to find the median of a set of numbers.

   (b) Give a $O(n)$ algorithm that will find the kth largest element of a set of numbers for k fixed.

   (c) It turns out it is possible to use a divide and conquer algorithm to find the median of a set of numbers quicker than the algorithm above[1]. This algorithm's runtime is given by a function $T(n)$ satisfying the following for some constant c:
$$T(n) \leq \begin{cases} T\left(\left\lfloor \frac{n}{5} \right\rfloor\right) + T\left(\left\lfloor \frac{3n}{4} \right\rfloor\right) + cn & \text{if } n > 5, \\ c & \text{if } n \leq 5. \end{cases}$$
   Find bounds for $T(8)$, $T(11)$ and $T(20)$.

   (d) Prove that $T(n) = O(n)$ and so the median can in fact be found in linear time!

   (e) Give an argument for why we cannot hope to beat this asymptotically.

[1] Google for "median of medians" or "linear time selection"