Question 1 (Divide and Conquer Runtimes, 30 points)**.** *Consider the three following divide and conquer algorithms:*

- *Algorithm A when run on an input of size n solves three recursive subproblems of size $n/3$ in addition to $\Theta(n)$ overhead work.*
- *Algorithm B when run on an input of size n solves three recursive subproblems of size $n/2$ in addition to $\Theta(n^2)$ overhead work.*
- *Algorithm C when run on an input of size n solves three recursive subproblems of size $n/9$ in addition to $\Theta(1)$ overhead work.*

What are the asymptotic runtimes of these three algorithms?

For Algorithm A:
The runtime is given by the recurrence:

$$T(n) = 3T(n/3) + \Theta(n).$$

Applying the Master Theorem with $a = 3$, $b = 3$ and $d = 1$, we find that $a = b^d$, so the runtime is $\Theta(n^d \log(n)) = \Theta(n \log(n))$.

For Algorithm B:
The runtime is given by the recurrence:

$$T(n) = 3T(n/2) + \Theta(n^2).$$

Applying the Master Theorem with $a = 3$, $b = 2$ and $d = 2$, we find that $a < b^d$, so the runtime is $\Theta(n^d) = \Theta(n^2)$.

For Algorithm C:
The runtime is given by the recurrence:

$$T(n) = 3T(n/9) + \Theta(1).$$

Applying the Master Theorem with $a = 3$, $b = 9$ and $d = 0$, we find that $a > b^d$, so the runtime is $\Theta(n^{\log_b(a)}) = \Theta(n^{\log_9(3)}) = \Theta(\sqrt{n})$.

Question 2 (Crossing the Stream, 35 points)**.** *Roxy is a frog trying to make it across a stream by hopping between lilypads conveniently arranged along a line. From any lilypad, she can hop to any other within 5 feet and she is trying to make it from one bank of the stream to the other in as few hops as possible.*

Roxy thinks that she can minimize the number of hops by at each hop, leaping to the furthest lilypad she can reach. So for example, if there are lilypads at $1, 4, 6, 7, 10,$ *and* 13 *feet from the near bank and the far bank is at* 15 *feet, she should hop to the lilypads at* $4, 7,$ *and* 10 *feet before hopping to the far bank.*

Prove that this strategy always successfully reaches the far bank in the smallest possible number of hops.

Suppose that Roxy's strategy has her using lilypads located at $x_1, x_2, \ldots, x_m$ feet from the starting bank, and that another strategy uses lilypads at $y_1, y_2, \ldots, y_k$. We wish to show that $m \leq k$. We claim by induction on i that $x_i \geq y_i$. For $i = 1$, this is because Roxy's first hop is to the furthest reachable lilypad. Furthermore, assuming that $x_i \geq y_i$, we claim that $x_{i+1} \geq y_{i+1}$. This is because x_{i+1} is the furthest lilypad at distance at most $x_i + 5$. So this means that any other lilypad at location $z > x_{i+1}$ has $z > x_i + 5 \geq y_i + 5$ and thus is not reachable from the lilypad y_i. Since the far bank cannot be reached from x_{m-1}, the far bank can also not be reached from y_{m-1}, which implies that $k \geq m$.

Question 3 (Finding the Crossover Point, 35 points)**.** *Let A and B be two unordered lists of n distinct real numbers each so that the smallest element of A is less than the largest element of B. Your goal is to find the largest integer k so that the kth largest element of A is larger than the kth smallest element of B (you can return $k = 0$ if no element of A is bigger than any element of B).*

For example, if $A = (10, 3, 6, 8)$ and $B = (1, 9, 2, 7)$ the answer would be 2 since the 2nd largest element of A (i.e. 8) is larger than the 2nd smallest element of B (i.e. 2), but the 3rd largest element of A (i.e. 6) is not larger than the 3rd smallest element of B (i.e. 7).

Give an algorithm that given A and B finds this k. For full credit, your algorithm should run in expected $O(n)$ time or better.

The algorithm is as follows:

```
Crossing(A,B)
  If n = 1
    Return 0
  Let m = [n/2]
  x = mth largest of A
  y = mth smallest of B
  If x > y    \\ need k => m
    Let A' be the set of elements of A smaller than x
    Let B' be the set of elements of B bigger than y
    Return m+Crossing(A',B')
  If x <= y   \\ need k < m
    Let A' be the set of elements of A at least x
    Let B' be the set of elements of B at most y
    Return Crossing(A',B')
```

Note that using linear time order statistics, x and y can be computed in $O(n)$ time. A' and B' can be computed by doing a scan through each of A and B and keeping the appropriate elements. We then have to make a single recursive call on a list of roughly half the size. Thus, we have the runtime recurrence

$$T(n) = T(n/2) + O(n),$$

which by the Master Theorem leads us to a final runtime of $O(n)$.

To prove correctness, we use strong induction on n. If A and B have length 1, then, since we assumed that $\max(B) > \min(A)$, the largest k we can take is 0, so our answer is correct. Otherwise, if $x > y$, the m^{th} largest element of A is bigger than the m^{th} smallest element of B. For any $k > m$, we have that the k^{th} largest element of A is the $(k-m)^{th}$ largest element of A' and the k^{th} smallest element of B is the $(k-m)^{th}$ smallest element of B'. Thus, if k' is the largest number so that the $(k')^{th}$ largest element of A' is bigger than the $(k')^{th}$ smallest element of B' (note that this is correctly computed by `Crossing`(A', B') by induction), the answer is $m + k'$. Similarly if $x \leq y$, the optimal k is less than m. Since the k^{th} largest element of A is the k^{th} largest element of A' (and the same for B and smallest elements), `Crossing`(A, B) similarly returns the right answer. This completes our proof.

Alternative Solution: Let x be the n^{th} largest element of the concatenation of A and B. Then k is the number of elements of A that are at least x.

This can clearly be computed in $O(n)$ time using the linear time order statistics algorithm. To show correctness, note that the n largest elements of $A \cup B$ will consist of the k largest elements of A and the $(n-k)$ largest elements of B. This means that the k smallest elements of B will all be in the bottom half, meaning the k^{th} largest element of A is at least the k^{th} smallest element of B. However, the $(k+1)^{st}$ smallest element of B will be at least x while the $(k+1)^{st}$ largest element of A will be less than x.