The Selection Problem is defined as follows: Given a sequence \( S = (a_0, a_1, \ldots, a_{n-1}) \) of \( n \) elements on which a linear ordering is defined, and an integer \( k, 1 \leq k \leq n \), find the \( k \)-th smallest element in the sequence. For instance, the minimum element in \( S \) would be the 1-smallest element, the median would be the \( \frac{n+1}{2} \)-smallest and the maximum would be the \( n \)-smallest element.

Write a method to generate a random list of elements of the type iNode, as described in class. Then implement 3 different algorithms to find the \( k \)-smallest elements in the list.

1. A method that sorts the list using selection sort and returns the \( k \)-th element in the sorted list. This should take \( O(n^2) \).
2. A method that sorts the list using quicksort and returns the \( k \)-th element in the sorted list. This should take \( O(n \log n) \), on average.
3. A modified version of quicksort that performs the partition just like regular quicksort and only searches for the \( k \)th smallest element in the appropriate sub-array. This should take \( O(n) \), on average.

For each method, experiment with various values of \( n \) and plot an estimate of the running times. For simplicity, use \( k = \frac{n+1}{2} \) for all experiments (that is, find the median). Since running times vary depending on circumstances that are independent of the method being evaluated (other applications running, garbage collection, O.S. operations, cache misses, etc.), we will use a global counter to keep track of the number of comparisons made by each algorithm and use that operation count as an estimate of the running time.

As usual, write a report describing your work.