3.1. Give an example when quicksort works $\Omega(n^2)$ time, if the separator is: a) the leftmost element of the segment, b) the rightmost element of the segment, c) the central element of the segment (`a[(l + r) / 2]`).

3.2. Imagine that the attacker knows what algorithm is used to select the separator (for example, if he knows what kind of random number generator you have). How can he write a test that makes quicksort work $\Omega(n^2)$ time?

3.3. How much extra memory does quicksort use on average and in the worst case?

3.4. You have n bolts and n matching nuts, all bolts (and, accordingly, all nuts) have different diameters. Looking at two bolts (or two nuts), it is difficult to understand which is larger and which is smaller, so the only operation that you have is take some bolt and some nut, and compare their diameters. Find a matching nut for each bolt in $O(n \log n)$ operations.

3.5. There is an array. We need to get the first k elements of the array in sorted order. What is the minimum time it can take?

3.6. What happens if in the Blum–Floyd–Pratt–Rivest–Tarjan algorithm we replace the constant 5 with 3 or 7?

3.7. In a sorted array of size n, k elements were changed (it is not known which ones). Sort the resulting array in $O(n + k \log k)$ time.

3.8. There are n boxes in a row. You need to sort them by numbers. You have a crane that can do one command: `swap(i, j)`, which swaps i and j boxes. Build a work plan for a crane that will sort the boxes in the minimum number of swaps. Working time (of your program, not the crane) should be $O(n \log n)$.