

## In-class exercises: Quicksort

1. Below is the pseudocode for Quicksort that we talked about in class. As usual with recursive functions on arrays, we see the array indices  $p$  and  $r$  as arguments.  $\text{Quicksort}(a, p, r)$  sorts the part of the array between  $p$  and  $r$  inclusively. The initial call (that is, to sort the entire array) is  $\text{Quicksort}(A, 0, n - 1)$ .

```
QUICKSORT( $A, p, r$ )
IF  $p < r$  THEN
     $q = \text{PARTITION}(A, p, r)$ 
    QUICKSORT( $A, p, q - 1$ )
    QUICKSORT( $A, q + 1, r$ )
FI
```

```
PARTITION( $A, p, r$ )
 $x = A[r]$ 
 $i = p - 1$ 
FOR  $j = p$  TO  $r - 1$  DO
    IF  $A[j] \leq x$  THEN
         $i = i + 1$ 
        Exchange  $A[i]$  and  $A[j]$ 
    FI
OD
Exchange  $A[i + 1]$  and  $A[r]$ 
RETURN  $i + 1$ 
```

Let  $A = \{3, 6, 1, 5, 8, 2, 4, 1, 3\}$ , and assume we call  $\text{Quicksort}(A, 0, 8)$ . Show what happens during the first invocation of Partition. What is the value of  $q$  returned, and what are the two recursive calls made?

2. What is the running time of QUICKSORT when all elements of array  $A$  have the same value?
  3. Briefly sketch why the running time of QUICKSORT is  $\Theta(n^2)$  when the array  $A$  contains distinct elements and is sorted in decreasing order.
- C-2.31 Develop an algorithm that computes the  $k$ th smallest element in a set of  $n$  distinct integers in  $O(n + k \lg n)$  time.