Fundamental Algorithms, Problem Set 2 Solutions

1. Illustrate the operation of PARTITION(A,1,12) on the array

$$A = (13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, 10)$$

(You may use either the text's program or the version given in class, but please specify which you are using.)
`Solution:` Using the class version we have $p = 1, q = 12$ and an auxiliary array B of length 12. We initialize $left = 1$, $right = 12$. We first set $x = 10$, the pivot element. Now for $j = 1$ to 11 we either put $A(j)$ as $B(left)$ and increment $left$ or as $B(right)$ and decrement $right$, depending on whether $A(j) \leq x$ or not. Here is what happens near the start:

j	newB	left	right
1	B[12]=13	1	11
2	B[11]=19	1	10
3	B[1]=9	2	10

etc., after $j = 11$ we get

$$B = (9, 5, 8, 7, 4, 2, 6, 8, 11, 12, 19, 13)$$

and now we have the left and right pointer with value 8 so we take our pivot value 10 (note that we saved it so it wouldn't be overwritten!) and make it $B[8]$, its correct position, giving

$$B = (9, 5, 8, 7, 4, 2, 6, 10, 11, 12, 19, 13)$$

We then reset the A vector to the B vector and we return the value 8. In `QUICKSORT[1,12]` we would now recursively `QUICKSORT` the first seven positions and the final four positions.

2. Let $L(n)$, ("L" for lucky) denote the number of comparisons that quicksort does if each time it is applied the pivot lies in the precise center of the array. For example, applying quicksort to an array of length 31, say $A(1) \cdots A(31)$ objects, there would be 30 comparisons (between $A(31)$ and all the other $A(j)$) and then $A(31)$ would end up in the 16^{th} place and there would be two recursive calls to quicksort on arrays each of size 15. Find the *precise* value of $L(1023)$. (Hint:

thats one less than 1024!)
`Solution:`Let $L(n) = p(n) + L(\ell) + L(r)$.
$L(1) = 0$.
If n is odd, $\ell = \frac{n-1}{2} = r$, thus

$$L(n) = p(n) + 2L(\frac{n-1}{2}).$$

W.l.o.g. If n is even, $\ell = \lceil \frac{n-1}{2} \rceil$ and $r = \lfloor \frac{n-1}{2} \rfloor$.

$$p(n) = n - 1$$

L(1023)	=	1022	+	2L(511)	=	1022	+	2*3586	=	8194
L(511)	=	510	+	2L(255)	=	510	+	2*1538	=	3586
L(255)	=	254	+	2L(127)	=	254	+	2*642	=	1538
L(127)	=	126	+	2L(63)	=	126	+	2*258	=	642
L(63)	=	62	+	2L(31)	=	62	+	2*98	=	258
L(31)	=	30	+	2L(15)	=	30	+	2*34	=	98
L(15)	=	14	+	2L(7)	=	14	+	2*10	=	34
L(7)	=	6	+	2L(3)	=	6	+	2*2	=	10
L(3)	=	2	+	2L(1)	=	2	+	2*0	=	0
L(1)	=	0								

`Note:` We have to work *backwards* to get $L(1023)$, doing $L(1), L(3), L(7)\cdots$ in that order.

3. You wish to sort five elements, denoted a, b, c, d, e. *Assume* that you already know that $a < b$, $c < d$ and $a < c$. Sort the elements with 4 further comparisons. (This actually gives a sorting of a, b, c, d, e under no assumptions with 7 comparisons. For if you begin by comparing a, b and then comparing c, d and then comparing the smaller of a, b to the smaller of c, d you will have something like $a < b$, $c < d$, $a < c$ except maybe with the letter interchanged. So the 4 more comparisons will suffice.
`Solution:`We have $a < c < d$. Ignoring b, we compare e with c and then with either a or d so that now $\{a, c, d, e\}$ are ranked, though not necessarily in that order. Now we insert b into the ranking of c, d, e, comparing it first to the middle and then to the top or bottom. Since we know $a < b$ this gives us the full ranking in 4 comparisons.

(Remark: This isn't as *ad hoc* as it may seem. After the given first three questions there are 15 possible permutations and 4 remaining questions. As $15 \leq 2^4$ this is not ruled out. But the next question must split the possibilities $8-7$. Suppose, for example, you make the next question "Is $b < c$?" On the Yes branch there are 5 possibilities ($abcd$ with e in any position) but on the No branch there are 10 possibilities (either $acbd$ or $acdb$ with e in any position) and there are only three questions remaining so only 8 branches so we would be dead. Note here that for a decision tree to work it has to work with all possible answers.)

4. Babu is trying to sort a, b, c, d, e with seven comparisons. First he asks "Is $a < b$" and the answer is yes. Now he asks "Is $a < c$?" Argue that (in worst-case) he will not succeed.
   `Solution:`Suppose (this being worst-case), he gets the answer Yes. At this stage of the original 120 permutations there are 40 left. (One way to see that is that a is the smallest of a, b, c and that happens precisely one-third of the time.) But $40 > 32 = 2^5$. From the Decision Tree Lower Bound he will need more than 5 further questions.

5. Illustrate the operation of `COUNTING-SORT` with $k = 6$ on the array $A = (6, 0, 2, 2, 0, 1, 3, 4, 6, 1, 3)$.
   `Solution:`We start with $C[0 \cdots 6]$ all zeroes. We go through A, incrementing $C[A[i]]$, at the end of which $C[j]$ gives the number of j's in A, so it is $2, 2, 2, 2, 1, 0, 2$. Then from $j = 1$ to 6 we set $C[j] \leftarrow C[j] + C[j-1]$ and now C has the cumulative sums $2, 4, 6, 8, 9, 9, 11$.
   Now we work our way down the array A:
   $A[11] = 2$ and $C[2] = 6$ so we set $B[6] = 2$ and reset $C[2] = 5$.
   $A[10] = 3$ and $C[3] = 8$ so we set $B[5] = 8$ and reset $C[3] = 7$.
   $A[9] = 1$ and $C[1] = 4$ so we set $B[4] = 1$ and reset $C[1] = 3$.
   $A[8] = 6$ and $C[6] = 11$ so we set $B[11] = 6$ and reset $C[6] = 10$.
   $A[7] = 4$ and $C[4] = 9$ so we set $B[9] = 4$ and reset $C[4] = 8$.
   $A[6] = 3$ and $C[3] = 7$ so we set $B[7] = 3$ and reset $C[3] = 2$.
   That last one was the clever one. Because $C[3]$ had earlier been decremented we are putting the second three into the appropriate empty space.
   $A[5] = 1$ and $C[1] = 3$ so we set $B[3] = 1$ and reset $C[1] = 2$.
   et cetera. At the end B is $0, 0, 1, 1, 2, 2, 3, 3, 4, 6$, the sorted output.

6. You are given a Max-Heap with n entries. Assume all entries are distinct. Your goal is to find the *third largest* entry. One way would

be to `EXTRACT-MAX` twice and then `MAXIMUM`. How long does this take? Find a better (by which we always mean faster for n large) way.
`Solution:` As `EXTRACT-MAX` takes $O(\lg n)$ and `MAXIMUM` takes $O(1)$ that method would take $2 \cdot O(\lg n) + O(1) = O(\lg n)$ steps. Better: The third largest is (previous problems!) one of $A[1] \cdots A[7]$. Sort those seven in $O(1)$ time and take the third.