Basic Algorithms, Problem Set 2
Solutions

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

$$A = (13, 18, 9, 5, 12, 8, 7, 4, 11, 2, 6, 10)$$

(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]=18	1	10
3	B[1]=9	2	10

etc., after $j = 11$ we get

$$B = (9, 5, 8, 7, 4, 2, 6, 8, 11, 12, 18, 13)$$

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

$$B = (9, 5, 8, 7, 4, 2, 6, 10, 11, 12, 18, 13)$$

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	=	2
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. 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.

4. 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:
$A[11] = 3$ 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.

5. 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.

6. Assume a program NEARMED[A,p,r] which returns an i such that $A[i]$ is uniform between the $0.49m$ and $0.51m$ positions ($m = p - r + 1$, the amount of data) when $A[p \cdots r]$ is sorted, and that NEARMED takes constant time K. (As a default assume NEARMED produces the median when given less than 100 values.) Create a variant $VQUICK[A, p, r]$ of QUICKSORT that calls on NEARMED to find pivot. Let $VT(n)$ be the expected time for VQUICK on an array of size n. Give (but do *not* attempt to solve!) the recursive equation for $VT(n)$. (Ignore initial conditions.)
`Solution:` VQUICK[A,p,r]
IF $p < r$ (*else done*)
$i \leftarrow NEARMED[A, p, r]$
IF $i \neq r$ THEN $A(i) \leftrightarrow A(r)$
$q \leftarrow PARTITION[A, p, r]$
$VQUICK[A, p, q-1]$
$VQUICK[A, q+1, r]$

Now the pivot i is averaged among the $.02n$ values so

$$T(n) = K + n - 1 + \frac{1}{.02n} \sum_{i=.049n}^{.051n} T(i) + T(n-i)$$

Comment: The extra time K for NEARMED is costly when n is small but is negligible when n is (very!) large. As the pivot is very close to the median the solution is quite close to that of "Lucky" $L(n)$ of problem 2 which is about 70 percent that of standard randomized QUICKSORT.