6.1

Give a brief argument that the running time of PARTITION on a subarray of size n is $\Theta(n)$.

Solution:

From 'PARTITION' procedure given in the example in textbook, we can obtain that for a subarray $A[p:r]$, (r - p) iterations are needed in for loop. In addition, the part outside the loop takes at most constant time, and since (r - p) is the size of subarray, the time that the procedure takes is proportional to the size of array, which is $\Theta(n)$.

6.2

Show that the running time of QUICKSORT is $\Theta(n^2)$ when the array A contains distinct elements and is sorted in decreasing order.

Proof:

Choose the last element in array A as reference, which is same as the example given in textbook. Regardless of the order of sorted array after the procedure is increasing or decreasing, the order of all elements before 'pivot' is monotonic. Thus, one of subarrays has (n - 1) elements, the other one is 0.

$$\Rightarrow T(n) = T(n-1) + \Theta(n) = \Theta(n^2)$$

6.3

Why do we analyze the expected running time of a randomized algorithm and not its worst-case running time.

Solution:

We analyze the expected running time of a randomized algorithm because it represents more typical time cost.

6.4

When RANDOMIZED-QUICKSORT runs, how many calls are made to the random number generator RANDOM in the worst case? How about in the best case? Give your answer in terms of Θ-notation.

Solution:

Since the random number generator RANDOM is called once the program calls 'RANDOMIZED-PARTITION'. Thus, we can obtain the result by computing the time cost of 'RANDOMIZED-PARTITION'.

Worst case: Elements in array are monotonic, thus in terms of 'partition',

$$T(n) = T(n-1) + \Theta(1)$$

Rewrite $T(n) = \begin{cases} c_0 & n = 1 \\ T(n-1) + c_1 & n > 1 \end{cases} \Rightarrow T(n) = \sum_{i=0}^{n-1} c_0 + c_1 = \Theta(n)$

Best case: The array is balanced.

$$T(n) \leqslant 2T\left(\frac{n}{2}\right) + \Theta(1)$$

Rewrite $T(n) \leqslant 2T\left(\frac{n}{2}\right) + c (c > 0)$

When $T(n) = 2T\left(\frac{n}{2}\right) + c$, from master theorem case 1,

we can obtain $T(n) = \Theta(n)$.

When $T(n) < 2T\left(\frac{n}{2}\right) + c$, use substitution method:

Guess: $T(n) \leqslant dn - k (d, k > 0)$

Inductive hypothesis: $T\left(\frac{n}{2}\right) \leqslant d \cdot \frac{n}{2} - k$

$$\begin{aligned} \Rightarrow T(n) &< 2T\left(\frac{n}{2}\right) + c \\ &\leqslant 2\left(d \cdot \frac{n}{2} - k\right) + c \\ &= dn + c - 2k \\ &\leqslant dn \left(\text{if } k \geqslant \frac{c}{2}\right) \end{aligned} \quad \Rightarrow T(n) = O(n)$$

Likewise, prove the lower bound:

Guess: $T(n) \geq dn \, (d > 0)$

Inducive hypothesis: $T\left(\frac{n}{2}\right) \geq d \cdot \frac{n}{2}$

$$\Rightarrow T(n) < 2T\left(\frac{n}{2}\right) + c$$

$$\geq 2d \cdot \frac{n}{2} + c \quad \Rightarrow T(n) = \Omega(n)$$

$$= dn + c \geq dn$$

Thus, $T(n) = \Theta(n)$