Suhail Basalama | HW3

6-1

- ***a.*** No, they don't. They produce different heaps. For example, given an array [1,4,7,8,9,6,3,2,5] building a heap using the two methods produces the following heaps:
  `1- BUILD-MAX-HEAP:  [9, 8, 7, 5, 4, 6, 3, 2, 1]`
  `2- BUILD-MAX-HEAP`: [9, 8, 6, 5, 7, 4, 3, 1, 2]`
- ***b.*** For upper bound:
  Each insertion takes at most $O(\lg(n))$.
  We have n insertions.
  Therefore, the runtime for `BUILD-MAX-HEAP`` is $O(\text{n}\lg(n))$
  For lower bound:
  If the array is sorted, each call to "`BUILD-MAX-HEAP``" makes "`HEAP-INCREASE-KEY`" to go up to the root. The length of the node i is $\lfloor lgi \rfloor$, and the time will be:

$$\sum_{i=1}^{n} \Theta(\lfloor \lg i \rfloor) \geq \sum_{i=\lceil n/2 \rceil}^{n} \Theta(\lfloor \lg \lceil n/2 \rceil \rfloor)$$
$$= \sum_{i=\lceil n/2 \rceil}^{n} \Theta(\lfloor \lg n - 1 \rfloor$$
$$\geq \frac{n}{2} \Theta(\lg n)$$
$$= \Omega(n \lg n)$$

Therefore, in the worst-case `BUILD-MAX-HEAP`` takes $\Theta(n \lg n)$.

7.2-2

If all elements have the same value the running time will be n^2 because the last element of the array will always be used for partitioning, which yields the worst-case running time of the QUICKSORT algorithm.

7.2-3

PARTITION's worst-case running time is when elements are in decreasing order and the pivot is the last and smallest element, where it needs to reduce the size of the subarray by 1 each step, which gives a running time of $\Theta(n)$. With this partitioning, the recurrence of the QUICKSORT becomes $T(n) = T(n-1) + \Theta(n)$. Solving the recurrence gives $T(n) = \Theta(n^2)$.