Toggle()

Time Complexity O(n)
Toggle() switches the heapState between Min-Heap and Max-Heap. After toggling the state, the heap is rebuilt using the heapify method. Each downheap operation is log(n), but the combined cost of all downheap operations is O(n) because most nodes are closer to the leaves, and fewer downheap operations involve the deeper parts of the heap

Remove(e)

Time Complexity: O(n)
Remove() searches through the array, scanning through all elements which requires O(n). Once the entry is found, the entry is replaced with the last entry in the heap. The heap property is then restored using either upheap or downheap which take O(log n). The O(n) search dominates the overall time complexity.

ReplaceKey(e,k)

Time Complexity: O(n)
ReplaceKey() begins by scanning through each element to find the target element which takes O(n). Once found, the key is then swapped which takes O(1). The overall time complexity is dominated by O(n).

replaceValue(e,v)

Time Complexity: O(n)
ReplaceValue() begins by scanning through each element to find the target element which takes O(n). Once found, the value is then swapped which takes O(1). The overall time complexity is dominated by O(n). Once the swap is complete the heap property must be restored using the upheap or downheap method. Both operations take O(log n). The O(n) search dominates the overall time complexity