The PQDictionary can be implemented in a way so that the minimum priority element at the front of the queue (reverse order of priority), and a balanced binary search tree which can link all the nodes of the priority queue to the binary search tree.

1. **Θ**(1) running time for size(), isEmpty(), min(): the front entry of the priority queue should be the one with the minimum key, isEmpty(), size() can use the bag operation in O(1) time.

2. **Θ**(log n) for insert(k, v): inserting in a priority queue is O(log n) and from the binary search tree it is also O(log n).

**Θ**(log n) for removeMin(): removing the first element with minimal key and from binary search tree, the running time is O(log n).

**Θ**(log n) for find(k) & remove(e): use a balanced binary search tree to achieve the O(log n) times and keep a mutual data structure to link from the queue to the binary search tree.

3. **Θ**(n) for findAll(k) & entries(): to iterate over the priority queue once in O(n) time to get all the values.