Nathaniel Daniel

Nicolescu

CS 477

3 October 2020

HW 5

1. The worst case performance is achieved when the highest element is chosen as the pivot at each partition.
2. Finding and deleting the smallest element in a max-heap is O(n). See problem2.cpp.
3. Finding an element and deleting it in a max-heap is O(n). See problem3.cpp.
4. The solution to this problem can use a modified counting sort. First create a count array of k elements. Preprocess the inputs by incrementing their values in the count array. Then modify the count array by making each slot's value the sum of itself and the previous slot. This finishes preprocessing. In order to answer queries about how many elements fall into the range [a...b] in O(n) time, simply find the values of indexes a and b in the counting array. Then, subtract the value of slot a from slot b. This results in the number of elements that fall in the range a..b, in O(1) time. The preprocessing is $\Theta(k + n)$ since one iteration of an array of length n and an iteration of another array of length k is needed, yielding a runtime of $\Theta(k + n)$.