Kyle Wolf

Problem Set #5 Part 1

a)

**public static int** findKthLargest(**int**[] s, **int** p, **int** q, **int** k, **int** size){  
 **int** pivotIndex = *partition*(s, p, q); *// get position of pivot* **if** (pivotIndex == size - k){ *// since pivot is always in right place, if pivotIndex and k are same, pivotIndex is kth* **return** s[pivotIndex]; *// using size - k instead because this is actually finding kth Smallest if just k. In array of size 10, 3rd smallest = 7th largest* }  
  
 **if** (pivotIndex > size - k){ *// k -1 for correct position normally but that's rolled into this* **return** *findKthLargest*(s, p, pivotIndex -1, k, size);  
 }  
  
 **return** (*findKthLargest*(s, pivotIndex + 1, q, k, size));  
}

b)

T(n) = T(3n/4) + n

c)

a = 1, b = 4/3, f(n) = n, h(n) = nlog4/31

Case 3

Regularity Condition:

af(n/b) = f(3n/4) = c(3n/4) = ¾ cn = ¾ f(n)

af(n/b) ≤ c’f(n) for c’ = ¾ which is between 0 and 1. Passed.

T(n) = Ө(n)

d)

Ө(n)