P3

(a). Insertion sort will be faster.

Because all the elements in array are equal, we can consider it has been sorted.

So, time complexity will be $O(n)$.

(b). Heapsort is faster, and time complexity is $O(n \log n)$.

Because for quick, we always pick up the right-most element as our pivot, and Quicksort always put elements smaller than pivot in front of pivot, and put elements larger than pivot behind the pivot. and repeatedly execute smilar progress, so it's $O(n^2)$.

So, Heapsort is faster.

(c). Bucket sort is faster, time complexity $O(n+k)$.

Because the Bubble sort always compare adjacent elements, if the first is bigger, then swap, for this question, it will be $O(n^2)$, But Bucket sort will work best when the data are more or less uniformly distributed.

For the bucket sort, it depends on the time of sorting elements between buckets, and smaller the buckets are divided, the less elements there is between the buckets and the less time it takes to sort.

So, Bucket sort is $O(n+k)$.

(d). Counting Sort is faster.

Because the input elements are fixed and it's n integers between 0 and k, so the running time is $O(n+k)$.

So, time complexity is $O(n+k)$ for Counting Sort.