Alexander, Arash, Jason

# Counting Sort Handout

**What is Counting Sort?**

Assigns each element value an integer key. Each key stores information on what position each element should be inserted into the sorted array. The process is as follows:
1: Create an int array of keys (e.g. int[] keys), where the index represents the key. Create an empty array that will store the sorted version of the original.
2: Iterate through the original array. Store how many times each key's associated value appears in the array in the key array.
3: Shift the values in the keys array to the right by 1 index. The last element is discarded.
3: Do prefix sum on the key array. For each key k, keys[k] is the index in the original array at which its associated value should appear in the sorted array. (You're basically seeing how much space the previous elements take up in the sorted array.)
4. Iterate through the original array. For each element, for its associated key k. Insert that element into the sorted array at index keys[k].  Then, increment keys[k] by 1.

**Efficiency**

Time complexity is always (O)n+k, space complexity is always (O)n+k.
Counting sort can be used with values that can be given an integer representation (e.g. byte, int, long, char, Objects containing an integer field, String hashes).
Counting sort should not be used where the range of values is too large (e.g. array of size 10, with range from 0 to 100).
Counting sort is the most time-efficient method for sorting arrays where the range of values is smaller than the number of elements. It is also much more space-efficient than mergesort and is nearly as space-efficient as most other fast sorting algorithms.

**Quick Tips/Troubleshooting**

- The range of values in the array should be minimal for maximum efficiency (e.g. 2x the length of the array to sort). Best works when dealing with a large amount of small range keys.
- Counting sort is defined as a stable sorting algorithm, meaning that the order of a specific value in the original array will stay the same in the sorted array. While this isn't very useful when the values have no meaning such as strictly integers, a stable algorithm becomes very useful when sorting through values that have meaning behind the values(objects). For example, while sorting grades, a stable algorithm will ensure all students with one particular grade will be in the exact same order in the sorted array as the original one.
- Counting sort is an integer sorting algorithm rather than a comparison-based sort. So objects such as Strings cannot be sorted using Counting Sort.
- Can be used as a subroutine for radix sort in order to sort objects.

Alexander, Arash, Jason

**Conclusion**

Counting sort is the most time efficient algorithm for sorting arrays of integers (such as int, short, byte, char, etc.) of any size with a small range of values.
If objects in an array can be easily represented using an integer key, counting sort will also work. (e.g. string hashing.)
If the range of values is large (even with a small array), other methods should be used.

**Additional Resources**

https://brilliant.org/wiki/counting-sort/
http://www.java67.com/2017/06/counting-sort-in-java-example.html
https://www.geeksforgeeks.org/counting-sort/
https://www.interviewcake.com/concept/java/counting-sort
https://www.cs.usfca.edu/~galles/visualization/CountingSort.html (different implementation)
https://youtu.be/OKd534EWcdk