Sorting Algorithm Overview

Algorithm	**Description**	**Runtime**
Exhaustive-Search Sort	Enumerate every permutation of the list items, then for each, check whether each element is less than or equal to the next.	$O(n \cdot n!)$
Insertion Sort	Start with one element from the unsorted list, then consecutively, add an item to the list in its correct position.	$O(n^2)$
Merge Sort	Recursively call the algorithm to sort each half of the list, until the length of each sub-list is 1, then merge lists (paying attention to value order), to arrive at the correct sorting.	$O(n \cdot \log n)$
Counting Sort	Initialize an empty array of size U, where U is the size of the universe keys are drawn from, and each entry is the start of an empty linked list. Then, for each key-item pair (K, V), add V to the Kth index in the array, creating a linked list of values if multiple values are associated with the same key. Concatenate the array to create the sorted array.	$O(n + U)$
Radix Sort	Start by rewriting each key of the input as a list of the digits of the key in base b, in reverse order (each key represented by k digits). Starting with the least significant digit, call Counting Sort on the input with the kth significant digits as keys, iteratively repeating that for all digits. Finally, reconstruct the original keys from their list-in-base b versions, and return the sorted list.	$O((n+b) \cdot \lceil \log_b U \rceil)$