2 Chapter 9: Sorting in Linear Time

2.1 Counting Sort

(a) Rank each element
   i. Count how many times $i$ occurs
   ii. Prefix sum on counter array to find how many items $\leq i$

(b) Place at its location.
   Start from right, place item in the output array, decrease its corresponding count.
   $W(n) = O(n)$ if range is $O(n)$.

Stable sorts — counting sort, mergesort, Insertion sort, Bucket sort.

Nonstable sorts — Quicksort, heapsort.
2.2 Bucket Sort

$k$ Buckets.

Algorithm:

1. hash items among buckets
2. sort the buckets
3. Combine buckets

Let there be $k$ buckets, $n$ items

1. distribution $O(n)$
2. sort the buckets
   \[ w(n) = O(n \log n) \]
   \[ A(n) = O(k \frac{n}{k} \log \frac{n}{k}) = O(n \log(n/k)) \]
3. combine buckets $O(n)$.

Thus, bucket sort is

\[ w(n) = O(n \log n) \]
\[ A(n) = O(n \log \frac{n}{k}) \]

If \( k \) is constant,
\[ A(n) = O(n \log n - n \log k) = O(n \log n) \]
\[ A(n) = O(n) \text{ if } k = n/20, \ A(n) = O(n) \]

Good when item distribution is known so that bucket get equititable number of keys.

Space Usage

worst-case: each bucket should have space for \( n \) key (any allocation)
\[ \Rightarrow \text{total} = O(nk) \]
Thus, as \( k \) increases, average space increases but so does the space requirement.

If linked allocation is used
\[ \text{Space needed} = O(k) + O(n) = O(n + k) = O(n) \]

However sorting each bucket using quicksort, mergesort, and heapsort will be difficult which require array representation.
If insertion sort is used to sort linked list, (buckets),

\[ A(n) = O\left(\frac{n^2}{k^2}\right) \times k = O\left(\frac{n^2}{k}\right) = O(n) \text{ for } k = O(n). \]
2.3 Radix sort

for $i \leftarrow 1$ to $d$

stable sort $A$ on digit $i$.

(a) counting sort can be used
(b) bucket sort can also be used