Sorting in linear time

(CLRS 8.2, 8.3)

We know that it is not possible to sort n elements faster than $\Omega(n \lg n)$ in the worst case **when using only comparisons** (i.e. in the comparison-model). The question is: Are there other kinds of sorting? What else could we use besides comparisons?

Exercise: Give an $O(n)$ algorithm to sort n integers in the range 0..999.

As we can see from this simpleexample, faster sorting algorithm exist, but they require special assumptions about the input. The most common ones are bucket sort, counting sort and radix sort, which we discuss below.

Bucket sort

Input: integers in the range $\{0, .., N-1\}$, for some $N \geq 2$.

```
BUCKET-SORT(A)
1  Create an array B[0..N − 1]
2  For i = 0 to n − 1
3        insert A[i] into B[A[i]] (at end)
4        For i = 0 to N − 1
5              traverse B[i]
```

Analysis: $O(n + N)$ time and $O(N + n)$ extra space.
How does $\Theta(n + N)$ compare with $\Theta(n \lg n)$? Put differently, when is Bucket-sort efficient?

- When N is small. For e.g. if $N = O(n)$, then Bucket-sort runs in $O(n)$ time.

Counting sort

Input: integers in the range $\{0, .., N-1\}$, for some $N \geq 2$.

Counting-sort uses the same idea as bucket sort, except that, instead of creating buckets (with linked lists), it stores everything in an array. It's more elegant.

```
COUNTING-SORT(A)
 1  Create an auxiliary array C[0..N − 1]
 2  For i = 0 to N − 1
 3        C[i] = 0
 4  For i = 0 to n − 1
 5        C[A[i]] + +
 6  For i = 1 to N − 1
 7        C[i] = C[i] + C[i-1]
 8  For i = n − 1 down to 0
 9        B[C[A[i]]] = A[i]
10        C[A[i]]–
```

Analysis: $O(n + N)$ time and $O(N + n)$ extra space.
Counting-sort is stable.

A sorting algorithm is called *stable* if it leaves equal elements in the same order that it found them.

To understand why stability is important, assume you want to sort a bunch of names by Last name. Let's say they were previously sorted by first name. If we use a stable sorting algorithm to sort by Last name, it would keep all the people with same last name in the order they were in the input, which means that people with same last name would appear in the order of their first names.

Radix sort