Chapter 8: Sorting in Linear Time

Lower Bounds for Comparison Sorting
Counting Sort
Radix Sort
Bucket Sort
Comparison Sorting Model

Sorting returns a permutation of its input. There are $n!$ permutations of $n$ elements.

$a_i < a_j$ determines if $a_i$ is before/after $a_j$.
$a_i < a_j$ chooses between two sets of permutations.

Map permutations to a binary decision tree.
Map each comparison to a node in a tree.
Map each subtree to a subset of permutations.
Map each leaf to a permutation.
A binary tree with height $h$ has $\leq 2^h$ leaves. A binary tree with $n!$ leaves has $h \geq \lg(n!)$.

$\lg(n!) \in \Theta(n \lg n)$, so $h \in \Omega(n \lg n)$.

Worst-case of comparison sorting is $\Omega(n \lg n)$.

Most leaves have $\Omega(n \lg n)$ depth. There are $\leq 2^d$ leaves at depth $\leq d$.

If $d < \lg(n!) - 1$, then $2^d < n!/2$.

This implies $\geq n!/2$ leaves have $d \geq \lg(n!) - 1$.

Average-case of comparison sorting is $\Omega(n \lg n)$. 
Counting Sort

Counting-Sort is $\Theta(n + k)$ and stable. It assumes that each $A[j] \in \{0, 1, \ldots, k\}$.

Counting-Sort($A$, $B$, $k$)

$C \leftarrow$ an array of $k$ zeros

for $j \leftarrow 1$ to length[$A$]

$C[A[j]] \leftarrow C[A[j]] + 1$

▷ $C[i]$ is the number of elements equal to $i$

for $i \leftarrow 2$ to $k$ do

$C[i] \leftarrow C[i] + C[i - 1]$

▷ $C[i]$ is the number of elements $\leq i$

for $j \leftarrow$ length[$A$] downto 1

$B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$
Radix Sort

Radix-Sort is $\Theta(d(n + k))$. It assumes that each value has $d$ digits. Each digit has one of $k$ values.

\[
\text{Radix-Sort}(A, d) \\
\quad \text{for } i \leftarrow 1 \text{ to } d \\
\quad \quad \text{use Counting-Sort to sort } A \text{ on digit } i
\]

Radix-Sort will outperform $\Theta(n \lg n)$ if $k$ is $O(n)$ and $d$ is $o(\lg n)$. 
Example Radix Sort

238   230   230   045
796   934   934   230
756   045   537   238
045  ⇒  796  ⇒  238  ⇒  537
537   756   045   756
230   537   756   796
934   238   796   934
↑      ↑      ↑      ↑
Bucket Sort

**Bucket-Sort** is $\Theta(n)$ on average if data is uniformly distributed over the interval $[0, 1)$.

**Bucket-Sort**($A$)

$\quad n \leftarrow A.length$

$\quad B \leftarrow$ an array of $n$ empty lists

$\quad$ for $i \leftarrow 1$ to $n$

$\quad \quad$ insert $A[i]$ into list $B[\lfloor n \cdot A[i] \rfloor]$

$\quad$ for $i \leftarrow 0$ to $n - 1$

$\quad \quad$ sort list $B[i]$ with **Insertion-Sort**

$\quad$ concatenate the lists $B[0]$ to $B[n - 1]$
Bucket Sort Analysis

Average Case:
There are $n$ elements and $n$ buckets. Let $n_i$ be the number of elements in bucket $i$. \textsc{Insertion-Sort} on $n_i$ elements is $O(n_i^2)$.
Need to bound $E[n_i^2]$ (expected value of $n_i^2$). $n_i$ is binomial with prob. $p = 1/n$ and $n$ trials.
$E[n_i] = np = 1$
$\text{Var}[n_i] = np(1 - p) = 1 - 1/n$
$\text{Var}[n_i] = E[n_i^2] - E[n_i]^2$ implies $E[n_i^2] < 2$
Expected time of second loop is $\Theta(n)$.

Alternative: Bound $\sum_{k=0}^{n} k^2 \binom{n}{k} \left(\frac{1}{n}\right)^k (1 - \frac{1}{n})^{n-k}$