Sorting in Linear Time

Textbook, Chapter 9, Sections 9.2 and 9.3

CSCE310: Data Structures and Algorithms
www.cse.unl.edu/~choueiry/S01-310/

Berthe Y. Choueiry (Shu-we-ri)
Ferguson Hall, Room 104
choueiry@cse.unl.edu, Tel: (402)472-5444
\[ O(n \lg n): \]
- Mergesort, heapsort: worst-case
- Quicksort: average-case

\[ \Omega(n \lg n): \]
- Mergesort, heapsort, quicksort

**Interesting common property:** sorted order is based *only* on comparisons between the input elements

\[ \rightarrow \text{Comparison sorts algorithms} \]

We can prove that: \[ \text{any comparison sort algorithm is in } \Omega(n \lg n) \] (Section 1.1)
Non-comparison sort algorithms

- Counting sort: assumes something about input
  \(O(n)\), stable
- Radix sort, \(\Theta(dn + kd)\)
  When \(d\) constant, \(k = O(n) \Rightarrow\) linear time
- Bucket sort: assumes something about input
  \(O(n)\)
Counting sort

Assumes that each of the $n$ input element is an integer in the range of 1 to $k$

When $k = O(n)$, counting sort is linear

Principle

- Determine for each input element $x$, the number of elements less than $x$
- Every element $x$ can be placed directly in its position in output array

Example

If $\exists 17$ elements less than $x$, $x$ must be in position 18

Slight modification for same value cases
Counting sort

Input: Array \( A[1 \ldots n], \text{length}[A] = n \)
Output: Array \( B[1 \ldots n] \)
Temporary working storage: Array \( C[1 \ldots 2] \)

\[
\text{COUNTING-SORT}(A, B, k)
\]

1. \( \text{for } i \leftarrow 1 \text{ to } k \)
2. \( \quad \text{do } C[i] \leftarrow 0 \)
3. \( \text{for } j \leftarrow 1 \text{ to } \text{length}[A] \)
4. \( \quad \text{do } C[A[j]] \leftarrow C[A[j]] + 1 \)
5. \( \triangleright C[i] \text{ now contains the number of elements equal to } i. \)
6. \( \text{for } i \leftarrow 2 \text{ to } k \)
7. \( \quad \text{do } C[i] \leftarrow C[i] + C[i - 1] \)
8. \( \triangleright C[i] \text{ now contains the number of elements less than or equal to } i. \)
9. \( \text{for } j \leftarrow \text{length}[A] \text{ downto } 1 \)
10. \( \quad \text{do } B[C[A[j]]] \leftarrow A[j] \)
11. \( \quad C[A[j]] \leftarrow C[A[j]] - 1 \)
Counting sort

\textbf{COUNTING-SORT}$(A, B, k)$

1 \hspace{1em} \textbf{for} $i \leftarrow 1$ \textbf{to} $k$
2 \hspace{1em} \textbf{do} \hspace{1em} $C[i] \leftarrow 0$
3 \hspace{1em} \textbf{for} $j \leftarrow 1$ \textbf{to} $\text{length}[A]$
4 \hspace{1em} \hspace{1em} \textbf{do} \hspace{1em} $C[A[j]] \leftarrow C[A[j]] + 1$
5 \hspace{1em} $\triangleright C[i]$ now contains the number of elements equal to $i$.
6 \hspace{1em} \textbf{for} $i \leftarrow 2$ \textbf{to} $k$
7 \hspace{1em} \hspace{1em} \textbf{do} \hspace{1em} $C[i] \leftarrow C[i] + C[i - 1]$
8 \hspace{1em} $\triangleright C[i]$ now contains the number of elements less than or equal to $i$.
9 \hspace{1em} \textbf{for} $j \leftarrow \text{length}[A]$ \textbf{downto} $1$
10 \hspace{1em} \hspace{1em} \textbf{do} \hspace{1em} $B[C[A[j]]] \leftarrow A[j]$
11 \hspace{1em} $C[A[j]] \leftarrow C[A[j]] - 1$

Lines 1–2: initialization

Lines 3–4: inspect each element, get values of $C[i]$

\hspace{2em} $C[i]$: number of elements equal to $i$

Lines 6–7: number of elements $\leq i$ (a running sum of $C$

Lines 9–11: $A[j]$ is placed in correct position in $B$
Counting sort

COUNTING-SORT($A, B, k$)
1 \hspace{1em} \textbf{for} $i \leftarrow 1$ \textbf{to} $k$
2 \hspace{1em} \textbf{do} $C[i] \leftarrow 0$
3 \hspace{1em} \textbf{for} $j \leftarrow 1$ \textbf{to} $length[A]$
4 \hspace{2em} \textbf{do} $C[A[j]] \leftarrow C[A[j]] + 1$
5 $\triangleright$ $C[i]$ now contains the number of elements equal to $i$.
6 \hspace{1em} \textbf{for} $i \leftarrow 2$ \textbf{to} $k$
7 \hspace{2em} \textbf{do} $C[i] \leftarrow C[i] + C[i - 1]$
8 $\triangleright$ $C[i]$ now contains the number of elements less than or equal to $i$.
9 \hspace{1em} \textbf{for} $j \leftarrow length[A]$ \textbf{downto} 1
10 \hspace{2em} \textbf{do} $B[C[A[j]]] \leftarrow A[j]$
11 \hspace{2em} $C[A[j]] \leftarrow C[A[j]] - 1$

Lines 9–11: $A[j]$ is placed in correct position in $B$
correct final position for $A[j]$ is $C[A[j]]$
Since some $x$ may not be different,
need to decrement $C[A[j]]$ when placing an $A[j]$ into $B$
Counting sort

\[
\text{COUNTING-SORT}(A, B, k)
\]

1. for \( i \leftarrow 1 \) to \( k \)
2. \( \text{do } C[i] \leftarrow 0 \)
3. for \( j \leftarrow 1 \) to \( \text{length}[A] \)
4. \( \text{do } C[A[j]] \leftarrow C[A[j]] + 1 \)
5. \( \triangleright C[i] \text{ now contains the number of elements equal to } i \).
6. for \( i \leftarrow 2 \) to \( k \)
7. \( \text{do } C[i] \leftarrow C[i] + C[i - 1] \)
8. \( \triangleright C[i] \text{ now contains the number of elements less than or equal to } i \).
9. for \( j \leftarrow \text{length}[A] \) downto 1
10. \( \text{do } B[C[A[j]]] \leftarrow A[j] \)
11. \( C[A[j]] \leftarrow C[A[j]] - 1 \)

Lines 1–2: \( O(k) \)
Lines 3–4: \( O(n) \)
Lines 6–7: \( O(k) \)
Lines 9–11: \( O(n) \)

Counting sort: \( O(k + n) \)

Usually, used with \( k = O(n) \), this in \( O(n) \)
**Counting sort:** stable

Numbers with the same value appear in $B$ in same order as in $A$

Important in presence of satellite data

Exercise: 9.2-1
Try again in 9.2-3
Radix sort

Given numbers of $d$-digit, Radix-sort:

1. Starts with the least significant digit first
2. Sorts the numbers according this digit using a stable sorting algorithm
3. Moves to the next least-significant digit
4. Repeats from 2, until last digit $d$
5. .. and the numbers are sorted!

Digit sorting must be stable
Radix sort is stable

Example: sort records by dates (years, months, and days)
General use: sort records keyed by multiple fields
Input: A, array of \( n \) elements, each of \( d \) digits: 1 lowest-order digit, \( d \) highest-order digit

For \( i \leftarrow 1 \) to \( d \)

\[ \begin{array}{cccc}
329 & 720 & 720 & 329 \\
457 & 355 & 329 & 355 \\
657 & 436 & 436 & 436 \\
839 & \Rightarrow & 457 & \Rightarrow & 839 & \Rightarrow & 457 \\
436 & 657 & 355 & 657 \\
720 & 329 & 457 & 720 \\
355 & 839 & 657 & 839 \\
\uparrow & \uparrow & \uparrow
\end{array} \]

Correctness: proof by induction on column being sorted

Running time: depends on intermediate sorting algorithm

Exercise: 9.3-1
Running time: of Radixsort

When each digit is in the range of 1 to $k$ ($k$ not too large)
  Use Counting sort
  Each pass over $n$ $d$-digit numbers is $\Theta(n + k)$
  $d$-passes: $\Theta(dn + kd)$
    When $d$ constant and $k = O(n)$, Radixsort is linear!!

Unlike Quicksort and Insertionsort,
Countingsort does not sort in place

When space is at stake, use Quicksort