Sorting restricted ranges of numbers

• If the range is restricted, we can sort using more than comparisons and swaps.
• Assume each of the $n$ input elements is an integer in the range $1 \ldots k$. 
Sorting restricted ranges of numbers

- If the range is restricted, we can sort using more than comparisons and swaps.
- Assume each of the $n$ input elements is an integer in the range $1 \ldots k$.

**Idea** For each $A[i]$ compute the number of elements less than or equal to $A[i]$ use that to compute position.
Sorting restricted ranges of numbers

• If the range is restricted, we can sort using more than comparisons and swaps.
• Assume each of the \( n \) input elements is an integer in the range \( 1 \ldots k \).

Idea For each \( A[i] \) compute the number of elements less than or equal to \( A[i] \), and use that to compute position.

• Array \( A[1 \ldots n] \) – holds input
• Array \( C[1 \ldots k] \) – \( C[j] \) holds number of elements of \( A \) less than or equal to \( j \)

Example:

<table>
<thead>
<tr>
<th>index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sorting restricted ranges of numbers

• If the range is restricted, we can sort using more than comparisons and swaps.
• Assume each of the \( n \) input elements is an integer in the range \( 1 \ldots k \).

**Idea** For each \( A[i] \) compute the number of elements less than or equal to \( A[i] \) and use that to compute position.

• Array \( A[1 \ldots n] \) – holds input
• Array \( C[1 \ldots k] \) – \( C[j] \) holds number of elements of A less than or equal to \( j \)

**Example:**

<table>
<thead>
<tr>
<th>index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( C )</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

**Questions**

• How do we compute \( C \)?
Counting Sort

\[ Counting - Sort(A, B, k) \]

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

- Running Time \( O(n + k) \)
- No Comparisons
- Doesn’t work on all data
- Good when \( k \) is small
- When \( k = O(n) \) we have run-time \( O(n + k) = O(n) \)
- Examples?
Stable Sorting

- We want to sort $x_1, x_2, ..., x_n$
- If $x_i > x_j$ then put $x_i$ after $x_j$
Stable Sorting

• We want to sort \( x_1, x_2, \ldots, x_n \)

• If \( x_i > x_j \) then put \( x_i \) after \( x_j \)

• But what if \( x_i = x_j \)
Stable Sorting

- We want to sort $x_1, x_2, ..., x_n$
- If $x_i > x_j$ then put $x_i$ after $x_j$
- But what if $x_i = x_j$

**Stable Sorting:** if $i < j$ and $x_i = x_j$ then put $x_i$ before $x_j$
Stable Sorting

• We want to sort $x_1, x_2, ..., x_n$
• If $x_i > x_j$ then put $x_j$ after $x_i$
• But what if $x_i = x_j$

Stable Sorting: if $i < j$ and $x_i = x_j$ then put $x_i$ before $x_j$

Question: Is counting sort stable?
Improving Counting Sort

Question: Should we use counting sort to sort everyone in this class by initials?
Improving Counting Sort

Question: Should we use counting sort to sort everyone in this class by initials?

- \( n = 150 \)
- \( k = 27^2 > 700 \)
- Running time is \( 150 + 700 = 850 \)
Improving Counting Sort

**Question:** Should we use counting sort to sort everyone in this class by initials?

- $n = 150$
- $k = 27^2 > 700$
- Running time is $150 + 700 = 850$

**Improvement: Radix Sort**

- Sort second initial
- Then stable sort by first initial.
Improving Counting Sort

Question: Should we use counting sort to sort everyone in this class by initials?

- $n = 150$
- $k = 27^2 > 700$
- Running time is $150 + 700 = 850$

Improvement: Radix Sort

- Sort second initial
- Then stable sort by first initial.

Analysis

- Sorting a single letter: $150 + 27 < 200$
- Total running time: $2(150 + 27) < 400$
Radix Sort

Radix – Sort ($A, d$)
1 for $i \leftarrow 1$ to $d$
2 do use a stable sort to sort array $A$ on digit $i$

Example

<table>
<thead>
<tr>
<th>379</th>
<th>STABLE SORT</th>
<th>912</th>
<th>STABLE SORT</th>
<th>802</th>
<th>STABLE SORT</th>
<th>258</th>
</tr>
</thead>
<tbody>
<tr>
<td>912</td>
<td>$\Rightarrow$</td>
<td>802</td>
<td>$\Rightarrow$</td>
<td>803</td>
<td>$\Rightarrow$</td>
<td>259</td>
</tr>
<tr>
<td>258</td>
<td>823</td>
<td>804</td>
<td>269</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>269</td>
<td>803</td>
<td>912</td>
<td>279</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>823</td>
<td>804</td>
<td>823</td>
<td>379</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>259</td>
<td>258</td>
<td>258</td>
<td>802</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>803</td>
<td>379</td>
<td>259</td>
<td>803</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>279</td>
<td>269</td>
<td>269</td>
<td>804</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>804</td>
<td>359</td>
<td>379</td>
<td>823</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>802</td>
<td>279</td>
<td>279</td>
<td>912</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Radix Sort Correctness

*Radix – Sort*(A, d)

1. for *i* ← 1 to *d*
2. do use a stable sort to sort array *A* on digit *i*

**Loop Invariant:** After the *i*th iteration of the loop, the elements are sorted by their last *i* digits.
Radix Sort Correctness

Radix – Sort\( (A,d) \)

1. for \( i \leftarrow 1 \) to \( d \)
2. do use a stable sort to sort array \( A \) on digit \( i \)

Loop Invariant: After the \( i \)th iteration of the loop, the elements are sorted by their last \( i \) digits.

Inductive Step:

- Assume the invariant holds after \( i-1 \) iterations
- Need to prove that it holds after \( i \) iterations
Radix Sort Analysis

- $n$ elements
Radix Sort Analysis

• $n$ elements
• All elements have $d$ digits
  – Initials: $d = 2$
  – SSN: $d = 9$
  – Dictionary Words: $d = 30$
Radix Sort Analysis

• \( n \) elements
• All elements have \( d \) digits
  – Initials: \( d = 2 \)
  – SSN: \( d = 9 \)
  – Dictionary Words: \( d = 30 \)
• Digits are in base \( b \)
  – Numbers: \( b = 10 \)
  – Words: \( b = 27 \)
  – UNI (letter/number): \( b = 37 \)
Radix Sort Analysis

- $n$ elements
- All elements have $d$ digits
  - Initials: $d = 2$
  - SSN: $d = 9$
  - Dictionary Words: $d = 30$
- Digits are in base $b$
  - Numbers: $b = 10$
  - Words: $b = 27$
  - UNI (letter/number): $b = 37$

Radix Sort Running Time: $O(d(n + b))$
Radix Sort Analysis

- $n$ elements
- All elements have $d$ digits
  - Initials: $d = 2$
  - SSN: $d = 9$
  - Dictionary Words: $d = 30$
- Digits are in base $b$
  - Numbers: $b = 10$
  - Words: $b = 27$
  - UNI (letter/number): $b = 37$

Radix Sort Running Time: $O(d(n + b))$

Counting Sort Running Time: $O(n + k) = O(n + b^d)$
Example

Setup : Sort everyone in columbia by UNI. Say $n = 40,000$
Example

Setup: Sort everyone in columbia by UNI. Say \( n = 40,000 \)

Radix Sort:

- \( d = 7 \)
- \( b = 37 \)
- Running Time: \( d(n + b) = 7(40,000 + 37) \approx 280,000 \)
Example

Setup: Sort everyone in columbia by UNI. Say \( n = 40,000 \)

Radix Sort:

- \( d = 7 \)
- \( b = 37 \)
- Running Time: \( d(n + b) = 7(40,000 + 37) \sim 280,000 \)

Counting Sort:

- UNI = 7-digit number in base 37.
- \( k = b^d = 37^7 \sim 10^{11} \)
- Running Time: \( n + k = 40,000 + 37^7 \sim 10^{11} \)
Example

Setup: Sort everyone in columbia by UNI. Say $n = 40,000$

Radix Sort:
- $d = 7$
- $b = 37$
- Running Time: $d(n + b) = 7(40,000 + 37) \sim 280,000$

Counting Sort:
- UNI = 7-digit number in base 37.
- $k = b^d = 37^7 \sim 10^{11}$
- Running Time: $n + k = 40,000 + 37^7 \sim 10^{11}$

Merge Sort
- Running Time: $n\log(n) = 40,000 \cdot \log(40,000) \sim 600,000$