Sorting Multisets

CSCI2100 Tutorial 5
Problem: Sorting Multiset

- Problem Input:
  - An array containing $n$ integers from a total order

- Goal:
  - An array containing all the integers in **nondescending** order
Recall: Merge Sort

• Merge sort works as long as the objects are
  • distinct, and
  • can be compared.

• The algorithm does not care what the objects actually are (integers, documents, images, etc.).

• Same is true for any comparison-based algorithms, e.g., selection sort, distribution sort, quick sort, etc.
• Cannot use merge sort on $B$ as the objects are not distinct.
• **Idea:** convert them into distinct objects.

Assign IDs

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
</table>

$B'$

| (8,1) | (2,2) | (6,3) | (2,4) | (9,5) | (8,6) | (10,7) | (2,8) |

• Still need to define how to compare the new objects.
  • Want to make sure the sorter order on the new objects is a legal sorted order on the original objects.
Define a Comparator

• Given two objects \( e_1 = (v_1, id_1) \) and \( e_2 = (v_2, id_2) \):
  • If \( v_1 < v_2 \), then rule \( e_1 < e_2 \)
  • If \( v_1 > v_2 \), then rule \( e_1 > e_2 \)
  • If \( int_1 = int_2 \):
    • If \( id_1 < id_2 \), then rule \( e_1 < e_2 \)
    • If \( id_1 > id_2 \), then rule \( e_1 > e_2 \)

E.g.: \( (2,1) < (2,10) \) and \( (1,10) < (2,1) \)
Apply Merge Sort on B’

Sorted B’

(2,2)  (2,4)  (2,8)  (6,3)  (8,1)  (8,6)  (9,5)  (10,7)

Remove ID

Sorted B

2  2  2  6  8  8  9  10
Time Complexity

• Modifying the input array takes $O(n)$
• Merging sort takes $O(n \log n)$ time.
• Removing the IDs to produce the final output takes $O(n)$ time.

• Overall time complexity: $O(n \log n)$
Sorting a Multiset of integers in small range

• Problem Input:
  • An array containing a multiset of $n$ integers, each from the domain $[1, U]$.

• Goal:
  • An array containing all the integers in **nondescending** order.
Review: the Counting Sort in the Lecture

- Sort a set of integers in a domain [1, 9]

Initialize array $B$

<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>6</th>
<th>4</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<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>
</table>

Scan through $A$

<table>
<thead>
<tr>
<th>7</th>
<th>2</th>
<th>6</th>
<th>4</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<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>
</table>

Clear $A$ and scan through $B$

<table>
<thead>
<tr>
<th>2</th>
<th>4</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<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>
</table>
A Simple Extension

- Sort a **multiset** of integers in domain [1, 9]
- **B** stores **counters**.

### Initialize array B

<table>
<thead>
<tr>
<th></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>7</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

![Diagram of initializing array B]

### Scan through A

<table>
<thead>
<tr>
<th></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>7</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

![Diagram of scanning through A]

### Clear A and scan through B

<table>
<thead>
<tr>
<th></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>1</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

![Diagram of clearing A and scanning through B]
The “Real” Multi-set Sorting Problem

• Problem Input:
  • An array containing $n$ key-value pairs, where each key is an integer from [1, U].
    E.g.: (93, 1155123456)

• Goal:
  • An array storing all pairs in nondescending order of key.
Example

• Input:
\{(9, v_1), (7, v_2), \{2, v_3\}, \{6, v_4\}, \{2, v_5\}, \{7, v_6\}, \{1, v_7\}, \{2, v_8\}\}

• Initially we will have the following array

<table>
<thead>
<tr>
<th>Input Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>k_1   v_1</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>9</td>
</tr>
</tbody>
</table>

• Rearrange the elements so that their keys are sorted:

<table>
<thead>
<tr>
<th>Sorted Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>v_7</td>
</tr>
</tbody>
</table>
Example

Compute $B$

\[ 9 \ v_1 \ 7 \ v_2 \ 2 \ v_3 \ 6 \ v_4 \ 2 \ v_5 \ 7 \ v_6 \ 1 \ v_7 \ 2 \ v_8 \]

Compute the prefix sum of $B$

\[ 1 \ 2 \ 3 \ 4 \ 4 \ 5 \ 7 \ 7 \ 8 \]

$B'[i]$ indicates the last index of a particular key in the final sorted array.

The final sorted array

\[ 1 \ v_7 \ 2 \ v_8 \ 2 \ v_5 \ 2 \ v_3 \ 6 \ v_4 \ 7 \ v_6 \ 7 \ v_2 \ 9 \ v_1 \]
Example

Build array $A'$ by repeating: for a key-value pair $(k, v)$ in $A$, copy it to the $B'[k]$-th position in $A'$

Decrement $B'[k]$ such that it points the position for the next pair with key $k$. 
Example
Example
Example
Example
Example

<table>
<thead>
<tr>
<th>9</th>
<th>(v_1)</th>
<th>7</th>
<th>(v_2)</th>
<th>2</th>
<th>(v_3)</th>
<th>6</th>
<th>(v_4)</th>
<th>2</th>
<th>(v_5)</th>
<th>7</th>
<th>(v_6)</th>
<th>1</th>
<th>(v_7)</th>
<th>2</th>
<th>(v_8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[A\]

\[A'\]

\[B'\]
Example
Example

Overall time complexity: $O(n + U)$