Week 7 Tutorial

CSCI 2100 Teaching Team, 2021 Fall
Outline

• Counting sort – a linked list version

• Counting inversions (Ex List 5 Problem 4)
Counting Sort on Key-Value Pairs

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

• Output:
  • An array storing all the pairs in nondescending order of key.
Counting Sort on Key-Value Pairs

• 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 have the following array

  Input Array

  \[
  \begin{array}{cccccccccccc}
  k_1 & v_1 & k_2 & v_2 & k_3 & v_3 & k_4 & v_4 & k_5 & v_5 & k_6 & v_6 & k_7 & v_7 & k_8 & v_8 \\
  9 & v_1 & 7 & v_2 & 2 & v_3 & 6 & v_4 & 2 & v_5 & 7 & v_6 & 1 & v_7 & 2 & v_8 \\
  \end{array}
  \]

• Rearrange the elements so that their keys are sorted:

  Sorted Array

  \[
  \begin{array}{cccccccccccc}
  1 & v_7 & 2 & v_3 & 2 & v_5 & 2 & v_8 & 6 & v_4 & 7 & v_2 & 7 & v_6 & 9 & v_1 \\
  \end{array}
  \]
Counting Sort (Linked List Ver.)

Compute $B$

$B = \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}$

$A = \begin{array}{cccccccc} 9 & v_1 & 7 & v_2 & 2 & v_3 & 6 & v_4 & 2 & v_5 & 7 & v_6 & 1 & v_7 & 2 & v_8 \end{array}$

⊥: A null pointer
Counting Sort (Linked List Ver.)

\[
\begin{array}{cccccccc}
9 & v_1 & 7 & v_2 & 2 & v_3 & 6 & v_4 & 2 & v_5 & 7 & v_6 & 1 & v_7 & 2 & v_8 \\
\end{array}
\]
Counting Sort (Linked List Ver.)

\[ \begin{align*}
9 & \quad \nu_1 & 7 & \nu_2 & 2 & \nu_3 & 6 & \nu_4 & 2 & \nu_5 & 7 & \nu_6 & 1 & \nu_7 & 2 & \nu_8 \\
\end{align*} \]
Counting Sort (Linked List Ver.)

<table>
<thead>
<tr>
<th></th>
<th>v₁</th>
<th>v₂</th>
<th>v₃</th>
<th>v₄</th>
<th>v₅</th>
<th>v₆</th>
<th>v₇</th>
<th>v₈</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>7</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Diagram:

```
  1  2  3  4  5  6  7  8  9
  ↓  ↓  ↓  ↓  ↓  ↓  ↓  ↓  ↓
 v₇  v₃  v₄  v₂  v₁
  ↓  ↓  ↓  ↓  ↓  ↓
 v₅  v₄  v₂  v₁
  ↓  ↓  ↓  ↓
 v₈  v₄  v₂  v₁
  ↓  ↓  ↓
 v₈  v₂  v₁
  ↓  ↓
 v₈  v₁
  ↓
 v₈
```
Counting Sort (Linked List Ver.)

How do we produce the sorted array $A'$?

Scan array $B$. For each cell referencing a non-empty linked list, enumerate all the pairs therein.

Overall time complexity: $O(n + U)$
Our next problem will demonstrate a somewhat unusual way to apply recursion. To solve the problem, we will need to manually add new output requirements.
Counting Inversions

- Input:
  - Let $A$ be an array of $n$ integers (not necessarily sorted). We call $(i, j)$ an inversion if $i < j$ but $A[i] > A[j]$.

- Goal: design an algorithm to count the number of inversions.

| 10 | 15 | 7 | 12 |

Inversions: (1, 3), (2, 3) and (2, 4)
Output: 3
Counting Inversions

• Input:
  • Let $A$ be an array of $n$ integers (not necessarily sorted).

• Goal: design an algorithm to
  • count the number of inversions, and
  • sort $A$ in ascending order.

Output: 3 inversions, and

$\begin{array}{cccc}
10 & 15 & 7 & 12 \\
\end{array}$

$\begin{array}{cccc}
7 & 10 & 12 & 15 \\
\end{array}$
Counting Inversions

Array $A$:

Subproblems:

Subproblem outputs:

It remains to count inversions $(i, j)$ such that $i$ comes from the first half and $j$ from the second.
Counting Inversions

Crossing inversion: an \((i, j)\) where \(i\) comes from the first half and \(j\) from the second.

\[
\begin{array}{ccc}
28 & 35 & 38 \\
9 & 31 & 40
\end{array}
\]

5 crossing inversions: (1,4), (2,4), (3,4), (2,5), (3,5)
Counting Inversions

We can count the crossing inversions as we merge the two halves into a sorted array.

# crossing Inversions: 0
Counting Inversions

# crossing Inversions: 3

Think: why
Counting Inversions

# crossing Inversions: 3

Think: why no change?
Inversions

In general, every time we move an element from the second half to the merged array, we increase the counter by the number of remaining elements in the first half.
Counting Inversions

# crossing Inversions: 5

no change if an element from the first half enters the merged array
Counting Inversions

- $T(1) = O(1)$
- $T(n) \leq 2T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + O(n)$
- Solving the recurrence gives $T(n) = O(n \log n)$