Counting Based Sorting
Beating the Worst Case

S.V.N. (vishy) Vishwanathan

University of California, Santa Cruz
vishy@ucsc.edu

February 17, 2016
Outline

1 Counting Sort

2 Radix Sort

3 Bucket Sort
Assumptions about the Input

1. Each of the $n$ input elements is an integer in the range 0 to $k$
2. Moreover, $k = O(n)$
Counting Sort

Partition

Crucial property: Stability (order preserving)
Pseudocode

COUNTING-SORT($A, B, k$)

1. let $C[0..k]$ be a new array
2. for $i = 0$ to $k$
   3. $C[i] = 0$
4. for $j = 1$ to $A.length$
   6. // $C[i]$ now contains the number of elements equal to $i$
7. for $i = 1$ to $k$
   8. $C[i] = C[i] + C[i - 1]$
   9. // $C[i]$ now contains the number of elements less than or equal to $i$
10. for $j = A.length$ downto 1
Time Complexity

- The `for` loop in lines 2–3 takes $\Theta(k)$ time
- The `for` loop in lines 4–5 takes $\Theta(n)$ time
- The `for` loop in lines 7–8 takes $\Theta(k)$ time
- The `for` loop in lines 10–12 takes $\Theta(n)$ time

Since $k = O(n)$, the overall time complexity is $\Theta(n)$
Why is Counting Sort beating the worst case complexity of comparison based sorting?
Outline

1. Counting Sort
2. Radix Sort
3. Bucket Sort
Assumptions about the Input

- Each of the $n$ input elements is an integer with at most $d$ digits
Basic Idea

329  720  720  329
457  355  329  355
657  436  436  436
839  457  839  457
436  657  355  657
720  329  457  720
355  839  657  839
Radix Sort

Pseudocode

RADIX-SORT(A, d)
1    for i = 1 to d
2        use a stable sort to sort array A on digit i
Time Complexity

- Given $n$ numbers with $d$ digits each
- Each digit takes up to $k$ possible values
- Stable sort for each digit takes $\Theta(n + k)$ time
- Then radix sort takes $\Theta(d(n + k))$ time

If $d$ is a constant, and $k = O(n)$, then radix sort takes $O(n)$ time
Outline

1. Counting Sort
2. Radix Sort
3. Bucket Sort
Assumptions about the Input

- Each of the $n$ input elements is drawn from a uniform distribution over the interval $[0, 1)$
Bucket Sort

Basic Idea

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.78</td>
</tr>
<tr>
<td>2</td>
<td>.17</td>
</tr>
<tr>
<td>3</td>
<td>.39</td>
</tr>
<tr>
<td>4</td>
<td>.26</td>
</tr>
<tr>
<td>5</td>
<td>.72</td>
</tr>
<tr>
<td>6</td>
<td>.94</td>
</tr>
<tr>
<td>7</td>
<td>.21</td>
</tr>
<tr>
<td>8</td>
<td>.12</td>
</tr>
<tr>
<td>9</td>
<td>.23</td>
</tr>
<tr>
<td>10</td>
<td>.68</td>
</tr>
</tbody>
</table>

| 0   | .12   |
| 1   | .21   |
| 2   | .39   |
| 3   | .68   |
| 4   | .72   |
| 5   | .94   |
| 6   | .78   |
| 7   | .17   |
| 8   | .26   |
| 9   | .45   |
Pseudocode

**BUCKET-SORT(A)**

1. let $B[0\ldots n-1]$ be a new array
2. $n = A.length$
3. for $i = 0$ to $n-1$
   4. make $B[i]$ an empty list
5. for $i = 1$ to $n$
   6. insert $A[i]$ into list $B[\lfloor nA[i]\rfloor]$
7. for $i = 0$ to $n-1$
8. sort list $B[i]$ with insertion sort
9. concatenate the lists $B[0], B[1], \ldots, B[n-1]$ together in order
Bucket Sort

Analysis

- Let $n_i$ be number of elements in bucket $B[i]$.
- Time complexity is

$$T(n) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

- Average case

$$\mathbb{E}[T(n)] = \mathbb{E}\left[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)\right]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(\mathbb{E}[n_i^2])$$

- Claim: $\mathbb{E}[n_i^2] = 2 - 1/n$ for each $i = 0, 1, \ldots, n - 1$
- Therefore $\mathbb{E}[T(n)] = \Theta(n) + \sum_{i=0}^{n-1} 2 - 1/n = \Theta(n)$
Proof of Claim - I

- Define

\[ X_{ij} = I \{ A[j] \text{ falls in bucket } i \} \]

- Therefore

\[ n_i = \sum_{j=1}^{n} X_{ij} \]

and

\[ n_i^2 = \left( \sum_{j=1}^{n} X_{ij} \right)^2 = \sum_{j=1}^{n} X_{ij}^2 + \sum_{j=1}^{n} \sum_{k=1, k \neq j}^{n} X_{ij} X_{ik} \]

- Taking expectations

\[ \mathbb{E} \left[ n_i^2 \right] = \sum_{j=1}^{n} \mathbb{E} \left[ X_{ij}^2 \right] + \sum_{j=1}^{n} \sum_{k=1, k \neq j}^{n} \mathbb{E} \left[ X_{ij} \right] \mathbb{E} \left[ X_{ik} \right] \]
Proof of Claim - II

\( X_{ij} = 1 \) with probability \( \frac{1}{n} \) and 0 with probability \( (1 - \frac{1}{n}) \)

\[
\mathbb{E}[X_{ij}] = 1 \cdot \frac{1}{n} + 0 \cdot \left(1 - \frac{1}{n}\right) = \frac{1}{n}
\]

and

\[
\mathbb{E}[X_{ij}^2] = 1^2 \cdot \frac{1}{n} + 0^2 \cdot \left(1 - \frac{1}{n}\right) = \frac{1}{n}
\]

Therefore

\[
\mathbb{E}\left[ n_i^2 \right] = \sum_{j=1}^{n} \mathbb{E}[X_{ij}^2] + \sum_{j=1}^{n} \sum_{k=1, k \neq j}^{n} \mathbb{E}[X_{ij}] \mathbb{E}[X_{ik}]
\]

\[
= \sum_{j=1}^{n} \frac{1}{n} + \sum_{j=1}^{n} \sum_{k=1, k \neq j}^{n} \frac{1}{n^2}
\]

\[
= 1 + n \cdot (n - 1) \cdot \frac{1}{n^2} = 2 - \frac{1}{n}.
\]
Why is Radix Sort beating the worst case complexity of comparison based sorting?
Questions?