Radix Sort

Analysis

http://www.cs.unh.edu/~ruml/cs758

1 handout: slides
Radix Sort
Counting Sort

For $n$ numbers in the range 0 to $k$:

1. for $x$ from 0 to $k$ $\quad O(k)$
2. $\text{count}[x] \leftarrow 0$
3. for each input number $x$ $\quad O(n)$
4. increment $\text{count}[x]$
5. for $x$ from 0 to $k$ $\quad O(k)$ times around loop
6. do $\text{count}[x]$ times $\quad$ iterates $O(n)$ times total
7. emit $x$ $\quad O(1)$ each time

$O(k + n + k + n) = O(2n + 2k) = O(n + k) \neq O(n \lg n)$
\[
O(g(n)) = \{ f(n) : \text{there exist positive constants } c, n_0 \\
\text{such that } f(n) \leq cg(n) \text{ for all } n \geq n_0 \}
\]

ignore constant factors
ignore ‘start-up costs’
upper bound

We can upper-bound \( f \)
(except perhaps at start) by
scaling \( g \) by a constant.

eg, running time is
\[
10n \log \frac{n}{c} = O(n \log n)
\]
How to sort one million records?
Radix Sort

How to sort one million records?

How to sort one trillion 4-bit integers?
Radix Sort

How to sort one million records?

How to sort one trillion 4-bit integers?

How to sort one billion 16-bit integers?
Radix Sort

How to sort one million records?

How to sort one trillion 4-bit integers?

How to sort one billion 16-bit integers?

How to sort one billion 32-bit integers?
How to sort one million records?

How to sort one trillion 4-bit integers?

How to sort one billion 16-bit integers?

How to sort one billion 32-bit integers?

For \( n \) numbers with \( d \) digits (each digit has \( k \) values):
How to sort one million records?

How to sort one trillion \(4\)-bit integers?

How to sort one billion \(16\)-bit integers?

How to sort one billion \(32\)-bit integers?

For \(n\) numbers with \(d\) digits (each digit has \(k\) values):

1. for \(i\) from 0 to \(d\)
2. \textbf{stable} sort on digit in place \(i\) from right
Input array contains \( n \) numbers in the range 0 to \( k \)

1. set \( \text{count}[x] \) to number of items \( = x \)
2. set \( \text{pos}[x] \) to total number of items \( < x \)
3. for each input number \( x \) (in order)
4. write \( x \) in output array at position \( \text{pos}[x] \)
5. increment \( \text{pos}[x] \)

Complexity?
Invariants?
Analysis
Correctness

What’s the invariant in radix sort?
What’s the space complexity?
What’s the time complexity?
## Limitations

<table>
<thead>
<tr>
<th>Radix Sort</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correctness</td>
</tr>
<tr>
<td></td>
<td>Complexity</td>
</tr>
<tr>
<td><strong>Limitations</strong></td>
<td>Break</td>
</tr>
<tr>
<td></td>
<td>Lower Bounds</td>
</tr>
<tr>
<td></td>
<td>Quick Sort</td>
</tr>
<tr>
<td></td>
<td>Insertion Sort</td>
</tr>
<tr>
<td></td>
<td>EOLQs</td>
</tr>
</tbody>
</table>

Why not implemented more?
• room
• coming to class
• hw vs exams
• piazza.com
• textbook
• asst 1: agate, valgrind
What is the minimum that a sorting algorithm must do?
What is the minimum that a sorting algorithm must do? How many possible outputs are there for sorting $n$ items?
What is the minimum that a sorting algorithm must do?
How many possible outputs are there for sorting \( n \) items?

binary tree with \( n! \) leaves
What is the minimum that a sorting algorithm must do?

How many possible outputs are there for sorting $n$ items?

Binary tree with $n!$ leaves has height at least $\lg(n!)$
What is the minimum that a sorting algorithm must do? How many possible outputs are there for sorting $n$ items? A binary tree with $n!$ leaves has height at least $\lg(n!)$.

Stirling: $n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$
Lower Bounds

What is the minimum that a sorting algorithm must do?

How many possible outputs are there for sorting $n$ items?

A binary tree with $n!$ leaves has height at least $\log(n!)$.

**Stirling:**

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n (1 + \Theta\left(\frac{1}{n}\right))$$

so:

$$\log(n!) = \log\left(\sqrt{2\pi n} \left(\frac{n}{e}\right)^n (1 + \Theta\left(\frac{1}{n}\right))\right)$$
What is the minimum that a sorting algorithm must do? How many possible outputs are there for sorting $n$ items?

A binary tree with $n!$ leaves has height at least $\lg(n!)$.

Stirling: $n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n (1 + \Theta(\frac{1}{n}))$

So:

$$\lg(n!) = \lg(\sqrt{2\pi n} \left(\frac{n}{e}\right)^n (1 + \Theta(\frac{1}{n})))$$

$$= \lg \sqrt{2\pi} + \lg \sqrt{n} + \lg \left(\frac{n}{e}\right)^n + \lg(1 + \Theta(\frac{1}{n}))$$
What is the minimum that a sorting algorithm must do? How many possible outputs are there for sorting $n$ items?

A binary tree with $n!$ leaves has height at least $\lg(n!)$.

Stirling: $n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n (1 + \Theta\left(\frac{1}{n}\right))$

So:

$$\lg(n!) = \lg\left(\sqrt{2\pi} n \left(\frac{n}{e}\right)^n (1 + \Theta\left(\frac{1}{n}\right))\right)$$

$$= \lg \sqrt{2\pi} + \lg n + \lg\left(\frac{n}{e}\right)^n + \lg(1 + \Theta\left(\frac{1}{n}\right))$$

$$= \Theta(\lg \sqrt{n} + n \lg\left(\frac{n}{e}\right)) + \lg(1 + \Theta\left(\frac{1}{n}\right))$$
What is the minimum that a sorting algorithm must do?
How many possible outputs are there for sorting \( n \) items?

binary tree with \( n! \) leaves has height at least \( \lg(n!) \)

Stirling: \( n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right) \)

so:

\[
\begin{align*}
\lg(n!) &= \lg\left(\sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)\right) \\
     &= \lg \sqrt{2\pi} + \lg \sqrt{n} + \lg\left(\frac{n}{e}\right)^n + \lg\left(1 + \Theta\left(\frac{1}{n}\right)\right) \\
     &= \Theta(\lg \sqrt{n} + n \lg\left(\frac{n}{e}\right)) + \lg\left(1 + \Theta\left(\frac{1}{n}\right)\right) \\
     &= \Theta(n \lg n)
\end{align*}
\]
What is the minimum that a sorting algorithm must do?

How many possible outputs are there for sorting $n$ items?

A binary tree with $n!$ leaves has height at least $\lg(n!)$.

Stirling: $n! = \sqrt{2\pi n} \left( \frac{n}{e} \right)^n (1 + \Theta(\frac{1}{n}))$

So:

$$\lg(n!) = \lg\left( \sqrt{2\pi n} \left( \frac{n}{e} \right)^n (1 + \Theta(\frac{1}{n})) \right)$$

$$= \lg\sqrt{2\pi} + \lg\sqrt{n} + \lg\left( \frac{n}{e} \right)^n + \lg(1 + \Theta(\frac{1}{n}))$$

$$= \Theta(\lg\sqrt{n} + n \lg\left( \frac{n}{e} \right)) + \lg(1 + \Theta(\frac{1}{n}))$$

$$= \Theta(n \lg n)$$

So comparison-based sorting takes $\Omega(n \lg n)$ time.
Quick Sort

Radix Sort

Analysis
- Correctness
- Complexity
- Limitations
- Break
- Lower Bounds
- Quick Sort
- Insertion Sort
- EOLQs

in-place
potentially less copying than mergesort
*expected* $O(n \lg n)$
needs tricks to avoid worst case
how fast if almost sorted?
What’s still confusing?
What question didn’t you get to ask today?
What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

*Thanks!*