Radix Sort

Adapted from the CLRS book slides

RADIX SORT

How IBM made its money. IBM made punch card readers for census tabulation in early 1900's. Card sorters worked on one column at a time. It's the algorithm for using the machine that extends the technique to multi-column sorting. The human operator was part of the algorithm!

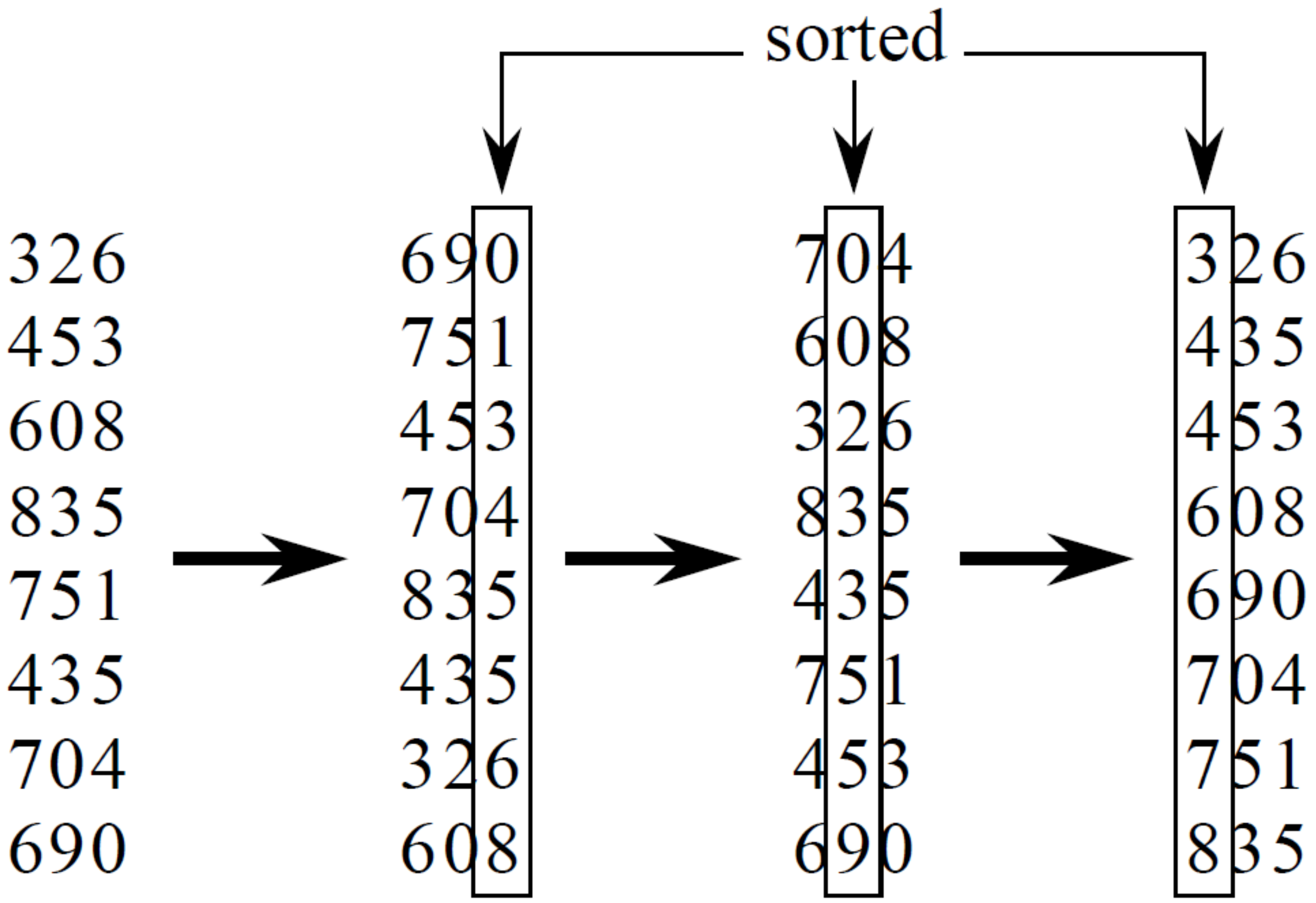
Key idea: Sort least significant digits first.

To sort d digits:

RADIX-SORT(A, n, d)

for $i = 1$ **to** d
    use a stable sort to sort array $A[1:n]$ on digit i

EXAMPLE


sorted
326
453
608
835
751
435
704
690
690
751
453
704
835
435
326
608
704
608
326
835
435
751
453
690
326
435
453
608
690
704
751
835

Correctness

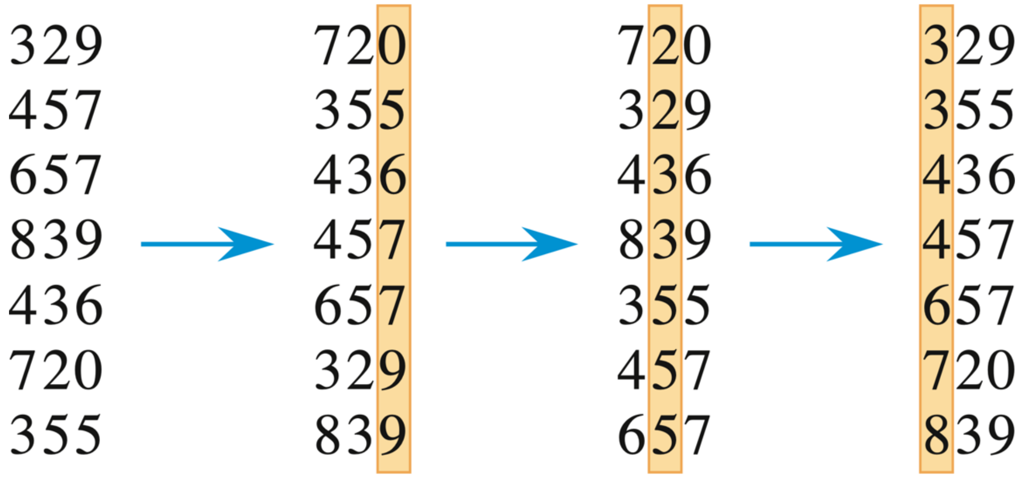
Induction on number of passes (i in pseudocode).

Assume digits $1, 2, \ldots, i - 1$ are sorted.

Show that a stable sort on digit i leaves digits $1, \ldots, i$ sorted:

- If two digits in position i are different, ordering by position i is correct, and positions $1, \ldots, i - 1$ are irrelevant.
- If two digits in position i are equal, the numbers are already in the right order (by inductive hypothesis). The stable sort on digit i leaves them in the right order.

This argument shows why it's so important to use a stable sort for intermediate sort.



ANALYSIS

Assume that we use counting sort as the intermediate sort.

- $\Theta(n + k)$ per pass (digits in range $0, \ldots, k$)
- d passes
- $\Theta(d(n + k))$ total
- If $k = O(n)$, time $= \Theta(dn)$.

How to break each key into digits?

- n words.
- b bits/word.
- Break into r-bit digits. Have $d = \lceil b/r \rceil$.
- Use counting sort, $k = 2^r - 1$.

  Example: 32-bit words, 8-bit digits. $b = 32$, $r = 8$, $d = \lceil 32/8 \rceil = 4$, $k = 2^8 - 1 = 255$.
- Time $= \Theta\left((b/r)(n + 2^r)\right)$.

ANALYSIS (continued)

How to choose r? Balance b/r and $n + 2^r$: decreasing r causes b/r to increase, but increasing r causes 2^r to increase.

If $b < \lfloor \lg n \rfloor$, then choose $r = b \Rightarrow (b/r)(n + 2^r) = \Theta(n)$, which is optimal.

If $b \geq \lfloor \lg n \rfloor$, then choosing $r \approx \lg n$ gives $\Theta\left((b/\lg n)(n + n)\right) = \Theta(bn/\lg n)$.

- Choosing $r < \lg n \Rightarrow b/r > b/\lg n$, and $n + 2^r$ term doesn't improve.
- Choosing $r > \lg n \Rightarrow n + 2^r$ term gets big. Example: $r = 2\lg n \Rightarrow 2^r = 2^{2\lg n} = (2^{\lg n})^2 = n^2$.

ANALYSIS (continued)

So, to sort 2^{16} 32-bit numbers, use $r = \lg 2^{16} = 16$ bits. $\lceil b/r \rceil = 2$ passes.

Compare radix sort to merge sort and quicksort:

- 1 million (2^{20}) 32-bit integers.
- Radix sort: $\lceil 32/20 \rceil = 2$ passes.
- Merge sort/quicksort: $\lg n = 20$ passes.
- Remember, though, that each radix sort "pass" is really 2 passes—one to take census, and one to move data.

ANALYSIS (continued)

- How does radix sort violate the ground rules for a comparison sort?
  - Using counting sort allows us to gain information about keys by means other than directly comparing two keys.