Radix Sort

Radix Sort

Non-comparative sorting algorithm

It orders keys by values, not by comparisons

H. Hollerith card sorting (1901)   

L. J. Comrie published LSD variant (1929) 

H. Seward improved storage (1954) 

Radix sort applies to data that can be sorted lexicographically (integers, words, cards etc.)

References/Notes/Image credits:
(Early history) https://www.ibm.com/ibm/history/ibm100/us/en/icons/tabulator/
D. E. Knuth, "The art of computer programming, vol. 3: sorting and searching", Chap 5 (1998)
(Hollerith image) https://en.wikipedia.org/wiki/Herman_Hollerith#/media/File:Hollerith.jpg
(Hollerith machine) https://americanhistory.si.edu/collections/search/object/nmah_694412
(CTR logo) https://en.wikipedia.org/wiki/Computing-Tabulating-Recording_Company#/media/File:CTR_Company_Logo.png

Radix sort runtime complexity

Two radix sort variants:

Least Significant Digit (LSD) | Most Significant Digit (MSD)

For n items with d-digit integer keys (in 0 to $k-1$):

LSD Radix Sort

$$$

Running time for any input: $\rightarrow \Theta(d(n+k))$ when using counting sort for inner loop

Storage: $\Theta(n+k)$ in addition to $\Theta(dn)$ for input

MSD Radix Sort *(worst case runtime matches LSD)*

Average runtime: $\Theta(n \log_k n)$ on random inputs

Storage: $\Theta(n+pk)$ in addition to $\Theta(dn)$ for input, where p is length of longest key prefix match

(IBM logo) https://en.wikipedia.org/wiki/File:IBM_logo.svg
L. J. Comrie, "The Hollerith and Powers Tabulating Machines", Transactions of the Office Machinery Users Assoc. (1929)
H. Seward, "Information sorting in the application of electronic digital computers to business operations", Dissertation MIT (1954)
T. Cormen et al., "Introduction to algorithms", Chap 8, MIT press (2022)
R. Sedgewick, "MSD Radix Sort", https://es.coursera.org/lecture/algorithms-part2/msd-radix-sort-gFxwG (2007)

Lexicographic Ordering And The Radix

Lexicographic ordering

Suppose each key is an integer pair (a, b)

Lexicographic ordering: $(a_1, b_1) < (a_2, b_2)$ if

- $a_1 < a_2$ or
- $a_1 = a_2$ and $b_1 < b_2$

This definition extends to tuples of > 2 elements

Radix sort operates on each key position in turn

Where does the name radix sort come from?

Radix is the latin for "root" (or base)

Radix of number system: num. of unique digits

E.g. 10 in decimal system: 0,1,...,9

Radix sort groups items by their radix

Reference:
M. T. Goodrich et al., "Algorithm design and applications", Chap. 9 (2015)

Least And Most Significant Digit Radix Sorts

Should we radix sort digits from right to left or from left to right?

Least Significant Digit - LSD (right to left)

Most Significant Digit - MSD (left to right)

LSD radix sort

Assumes all keys have same length

Approach used in card-sorting machines

Requires a stable sort in inner loop

Iteratively sorts d-digit data in d passes

MSD radix sort

Supports keys with different lengths

Sorts by recursion (exits early if < 2 keys share digit)

Runtime depends heavily on the input data distribution

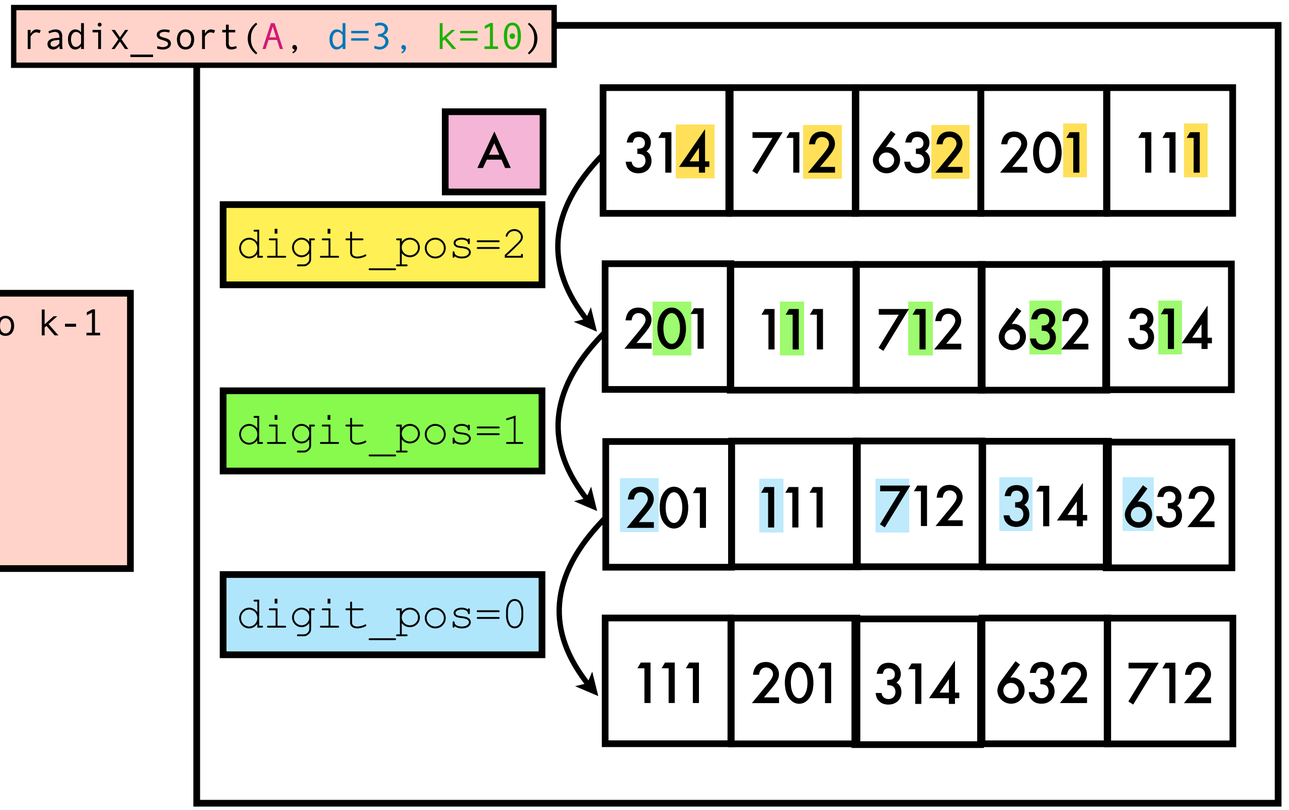
Can create lots of (small and inefficient) subarray sorts

This can be addressed with cutoff to insertion sort

Reference:
R. Sedgewick et al., "Radix Sorts", https://www.cs.princeton.edu/courses/archive/spr07/cos226/lectures/11RadixSort.pdf (2007)

LSD Radix Sort Implementation

```
def radix_sort(A, d, k): # keys w. d digits from 0 to k-1
    # loop from least to most significant digit
    for digit_pos in range(d-1,-1,-1):
        # typically use counting sort as stable sort
        stable_sort(A, k, digit_pos)
```



Reference:
T. Cormen et al., "Introduction to algorithms", Chap 8, MIT press (2022)

Digit Size Selection

The influence of digit size

For n numbers with b-bits and $r \leq b$ (digit size)

LSD radix sort runtime is $\Theta\left(\frac{b(n+2^r)}{r}\right)$

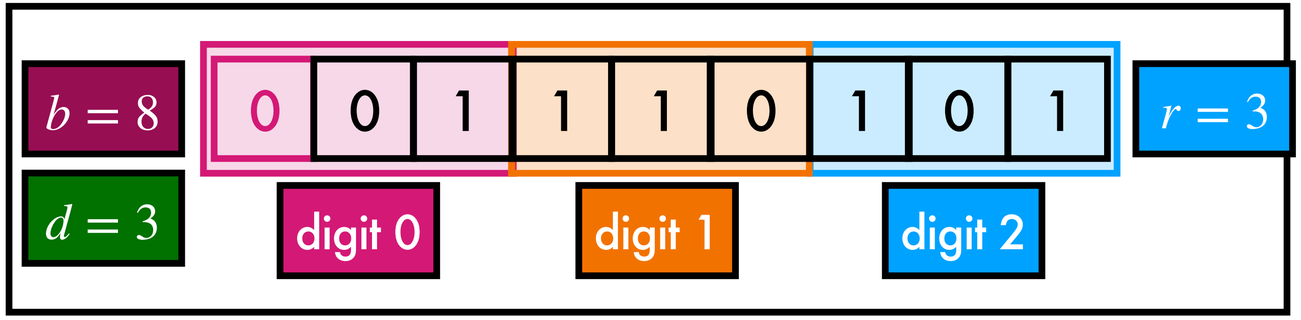
(assumes counting sort in inner loop)

Proof: interpret keys as $d = \lceil b/r \rceil$ digits

where each digit has r bits

Use counting sort with $k = 2^r$

Runtime is $\Theta(d(n+k)) = \Theta\left(\frac{b(n+2^r)}{r}\right)$



How should we choose digit size (r bits)?

Goal: minimise $\frac{b(n+2^r)}{r}$

If $b < \lfloor \log n \rfloor$: pick $r = b \implies \frac{b(n+2^b)}{b} = \Theta(n)$ (optimal)

If $b \geq \lfloor \log n \rfloor$: pick $r = \lfloor \log n \rfloor$

$\implies \frac{b(n+2^{\lfloor \log n \rfloor})}{\lfloor \log n \rfloor} = \Theta\left(\frac{bn}{\log n}\right)$ (best we can do)

References:
T. Cormen et al., "Introduction to algorithms", Chap 8, MIT press (2022)

Comparison To Quicksort

LSD radix sort vs quicksort

Quicksort runtime is typically $\Theta(n \log n)$

LSD radix sort $\Theta(dn)$ when $k = O(n)$

Quicksort - typically more cache-friendly & in-place

LSD radix sort - less cache-friendly & not in-place

Best choice of sorting algorithm will depend on

- data distribution
- storage requirements
- machine details

References:
R. Sedgewick et al., "Radix Sorts", https://www.cs.princeton.edu/courses/archive/spr07/cos226/lectures/11RadixSort.pdf (2007)
T. Cormen et al., "Introduction to algorithms", Chap 8, MIT press (2022)
D. E. Knuth, "The art of computer programming, vol. 3: sorting and searching", Chap 5 (1998)