**Implementation of Radix sort**

The effectiveness of radix sort heavily relies on the partitioning method for keys. The described radix sort partitions key pointers using efficient operations, initially determining partition sizes with a static table of 256 integer counters. This information constructs a linked list, aiding in partitioning key pointers in place. However, the algorithm's efficiency diminishes when sorting small numbers of keys, leading to the implementation of an insertion sort for sets of fewer than 16 keys. This decision was informed by theoretical and empirical studies, as well as observations regarding the operational complexities of insertion sort and Quicksort. Ultimately, sorting sets of less than 16 keys using insertion sort is favored due to its lower operational complexity compared to Quicksort.

*Reference:*  I.J.Davis(1992), A Fast Radix Sort, p.2, Retrieved from <https://academic.oup.com/comjnl/article-pdf/35/6/636/1009342/35-6-636.pdf>

**Advantage of Radix sort:**

Radix sort is a sorting algorithm that operates by sorting data elements based on their keys, typically represented as numbers or alphanumeric characters. It sorts each digit of the data elements, starting from the least significant digit and moving to the most significant digit. There are two classifications of radix sort: least significant digit (LSD) and most significant digit (MSD). The algorithm's efficiency is influenced by the number of digits in each array value, with a constant number of digits leading to better efficiency. However, if the array values have varying numbers of digits, the algorithm's efficiency decreases, comparable to other advanced sorting algorithms like quick sort and merge sort.

Advantages of radix sort include its efficiency regardless of the type and size of the data being sorted, as well as its ability to handle larger keys efficiently. However, it is less flexible and more complex to program compared to other algorithms, and it consumes more memory space. Despite these drawbacks, radix sort has the potential to sort N keys in O(N) operations, making it faster than many other sorting algorithms. Variants of radix sort, such as Fast Radix Sort and Forward Radix Sort, offer improved speed and performance.

*Reference:* Sai Krishna Kovi(2017), Comparative Analysis of Bucket and Radix Sorting With Their Applications and Advantages, p.IV(A), Retrieved from <https://www.researchgate.net/profile/Sai-Krishna-20/publication/317799415_Comparative_Analysis_of_Bucket_and_Radix_Sorting/links/594c2cbb458515e70348a9a9/Comparative-Analysis-of-Bucket-and-Radix-Sorting.pdf?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19>

**Founder and first development of Radix sort**

An American computer pioneer who founded The Tabulating Machine Company, later renamed IBM. Hollerith's inspiration stemmed from the challenge of processing the American census in the late 1880s using only pen and paper. In response, he invented the Hollerith machine, which utilized punch cards and tabulators to automate data recording and tallying, dramatically reducing the census processing time from ten years to three months.

Expanding on his invention, Hollerith developed a method to sort punch cards by columns, leading to the formulation of an algorithm that involved multiple passes, sorting by each digit or "radix." This iterative process of sorting by individual columns gave rise to the radix sort algorithm, marking the birth of the first sorting algorithm. Hollerith's innovative approach to sorting laid the foundation for the company that eventually evolved into IBM, known today as "Big Blue."

*Reference:* Nildesh D(2019), The first sorting algorithm and the founding of IBM..., p.2-4, Retrieved from <https://www.linkedin.com/pulse/first-sorting-algorithm-founding-ibm-nilesh-d/>

**Intro para:** It was the first sorting algorithm and was invented by Herman Hollerith, who later founded IBM. He created a machine called the Hollerith machine to speed up the American census. Radix sort organizes items by their digits or columns, sorting them one by one. This algorithm made sorting faster and more efficient. Herman Hollerith's innovation revolutionized computing history, yet sadly, many people have forgotten his contributions along with other pioneers like Tim Berners-Lee and Kenneth Rosen.

**Advantage and disadvantage of radix sort compared to quick sort**

Radix exchange cannot be bettered in respect to the amount of data looked at. Quicksort doesn't even come close. Quicksort on average performs O(log n) comparisons per string and inspects Ct(log n) bits per comparison. By this measure, the expected running time for quicksort is Θ(n log2⁡n), for radix exchange only O(n log n). However, the theoretical advantages of radix exchange are usually swamped by the cost of bit picking. This implies that while radix sort excels in terms of the amount of data inspected, its efficiency may be hindered by the overhead of bit manipulation.

*Reference:* Peter M. Mcllroy, Keith Bostic, M. Douglas MCllroy(1993), Engineering Radix Sort, p.1.2, Retrieved from [**https://www.usenix.org/legacy/publications/compsystems/1993/win\_mcilroy.pdf**](https://www.usenix.org/legacy/publications/compsystems/1993/win_mcilroy.pdf)