# RADIX SORT UNRAVELED

Sabbir Islam [1] Anika Tabassum [2] Sadnan Yasar Tanvir[3] Samia Reza Maisha [4] Saiful Islam [5]

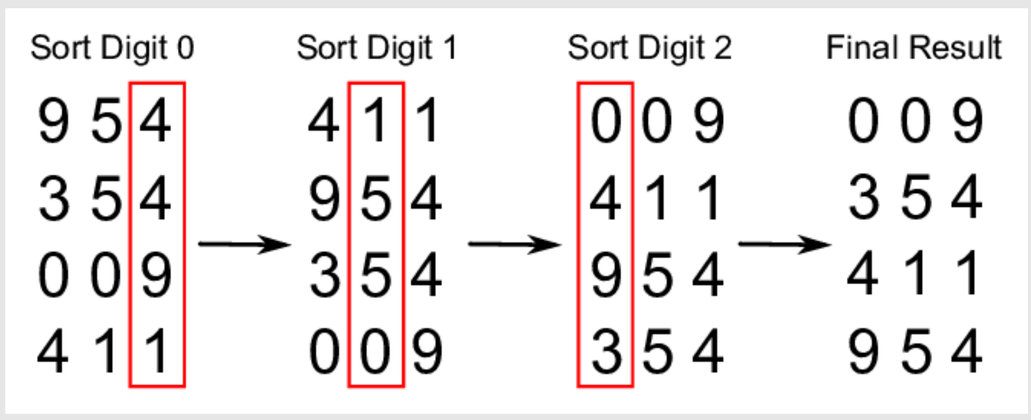
Department of Computer Science and Engineering
Independent University, Bangladesh
Dhaka, Bangladesh.
{[1]2211176,[2]2321188,[3]2321524,[4]2211510,[5]2321267}@iub.edu.bd

## Abstract

This research focuses on the usage of radix sort in daily applications. Radix sort is a fascinating algorithm with many intricacies to explore and implement. The goal of this research is to thoroughly investigate Radix Sort, clarifying its underlying ideas, practical applications, and performance attributes. Radix sort is appropriate for sorting because, under certain circumstances, it can handle a wide range of data formats and achieve linear time complexity by sorting items efficiently by processing individual digits or characters from the least significant to the most significant. A practical issue that can be resolved with this technique is explored and the algorithm paradigm is described. In conclusion, this article offers a thorough review of radix sort.

## Introduction

Radix sort, first conceptualized by **Herman Hollerith** in **1887**, became a practical sorting method for punched cards by 1923. It operates by processing individual digits or characters of numbers or strings from least to most significant, sorting elements based on their radix (the base of the numbering system used). Radix sort handles elements of varying lengths independently and can be more efficient than comparison-based algorithms in certain scenarios, despite its linear time complexity $O(nw)$ and space complexity $O(n+k)$. It can also maintain stability, preserving the relative order of elements with equal keys, which is important for specific applications.




## Rationale for the Algorithm's Selection

Radix sort is preferred for sorting hospital records due to its efficiency, stability, adaptability, and space complexity. It excels in handling **variable-length keys**, such as patient IDs and medical codes, by sorting them digit by digit or character by character. Radix sort requires less additional space compared to other sorting algorithms like counting sort and bucket sort, making it more memory efficient, especially in scenarios with large datasets. Its stable sorting nature preserves the **relative order of equal elements**, which is crucial for maintaining the integrity of medical records. Additionally, radix sort can be easily adapted to sort hospital records based on different criteria, such as patient name, age, admission date, or medical condition.

## Methodology

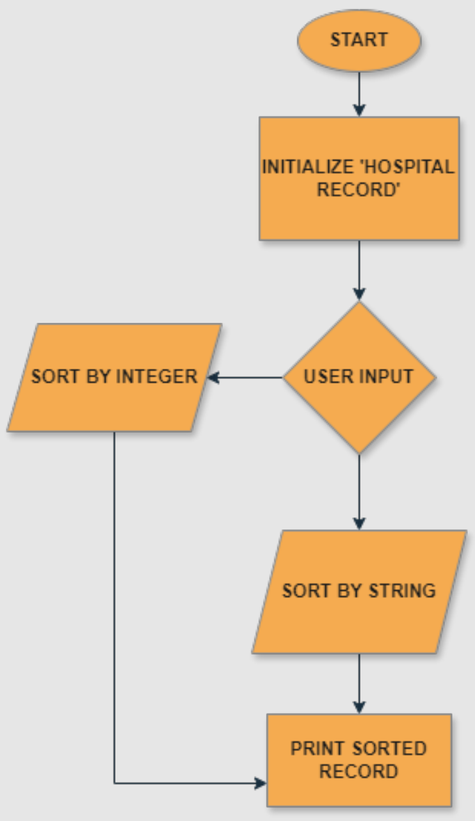
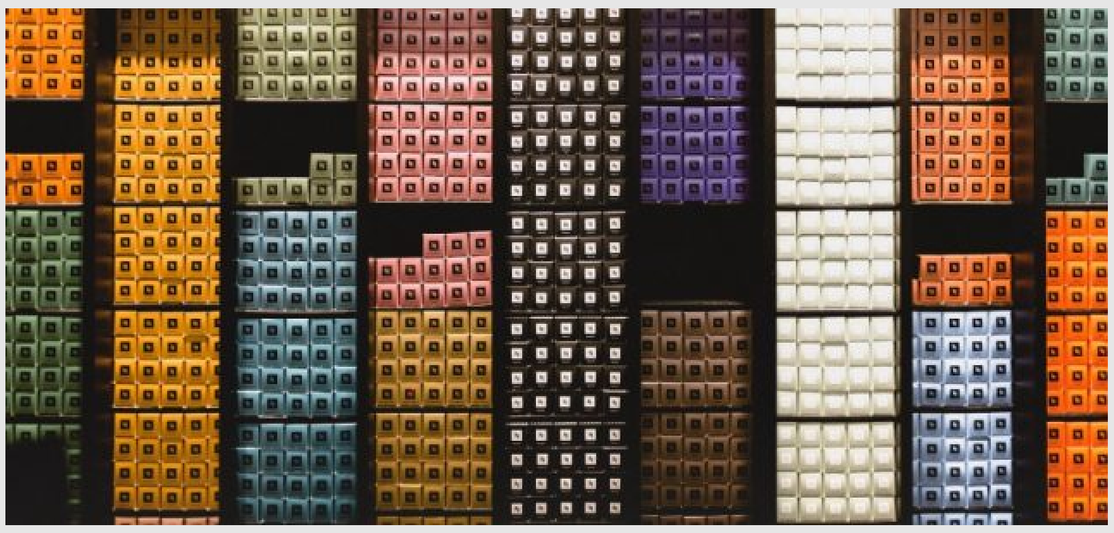



Fig: **Flowchart**

The radix sort methodology involves using counting sort to numerically sort patient records by IDs or ages and to lexicographically sort string attributes. For numerical sorting, the maximum value is calculated to determine sorting iterations per digit, organizing records based on digit values. String sorting involves standardizing string lengths and sorting characters to achieve lexicographical order. The counting sort method categorizes and tallies occurrences of specific digit values or characters, updating the original array with sorted records after processing all digits or characters.

## Substitute Use of the Algorithm



Radix sort finds applications beyond sorting hospital records and can be effectively used in various scenarios where keys are represented in a fixed-length format or can be processed digit by digit. One such application is in sorting **large volumes of numerical data**, particularly in **data analysis and scientific computing**.

In data analysis tasks, datasets often contain numerical values, such as sensor readings, financial transactions, or scientific measurements. Radix sort can efficiently sort these datasets based on numerical values, providing valuable insights and facilitating further analysis.

*Runtime Complexity:*

- Radix sort has a linear time complexity of $O(d\ (n+k))$, where $d$ is the maximum number of digits in the numerical values, $n$ is the number of elements to be sorted, and k is the range of the input.
- For numerical data with a fixed number of digits or a limited range, radix sort can achieve optimal performance, often outperforming comparison-based sorting algorithms like quick sort or merge sort.

*Space Complexity:*

- Radix sort typically has a space complexity of $O(n+k)$, where n is the number of elements to be sorted and $k$ is the range of the input.
- In the context of sorting numerical data, the space complexity remains efficient and scales well with the size of the dataset.

In conclusion, radix sort's efficiency and scalability make it a valuable algorithm for sorting numerical data in various applications, including data analysis, scientific computing, and financial analysis. Its linear runtime complexity and space efficiency make it well-suited for handling large volumes of numerical data effectively.

## Alternative Solution

Hospital records can be sorted using quick sort, an adaptable and effective sorting algorithm, based on a variety of criteria, including string and numeric fields. Quick sort can effectively handle a variety of data types and is not limited by specific properties of the data, such as fixed-length keys, as radix sort is. Quick sort is appropriate for situations where performance is crucial because, on average, it is faster than radix sort for larger datasets, with an $O(n\ log\ (n))$ time complexity. Furthermore, quick sort is an in-place sorting algorithm that uses less memory than radix sort, which might need extra storage in proportion to the range of key values.

## Conclusion

Radix sort's linear time complexity makes it faster than comparison-based algorithms like quick sort and merge sort for large datasets. It maintains relative order, making it valuable in real-world applications, and is efficient for sorting large numbers of integers or strings. However, it's not suitable for sorting floating-point numbers, requires significant memory, and isn't efficient for small datasets or those with few unique keys. Despite these drawbacks, radix sort is useful in digital forensics for sorting and analyzing file system metadata, aiding investigators in reconstructing timelines and identifying patterns.