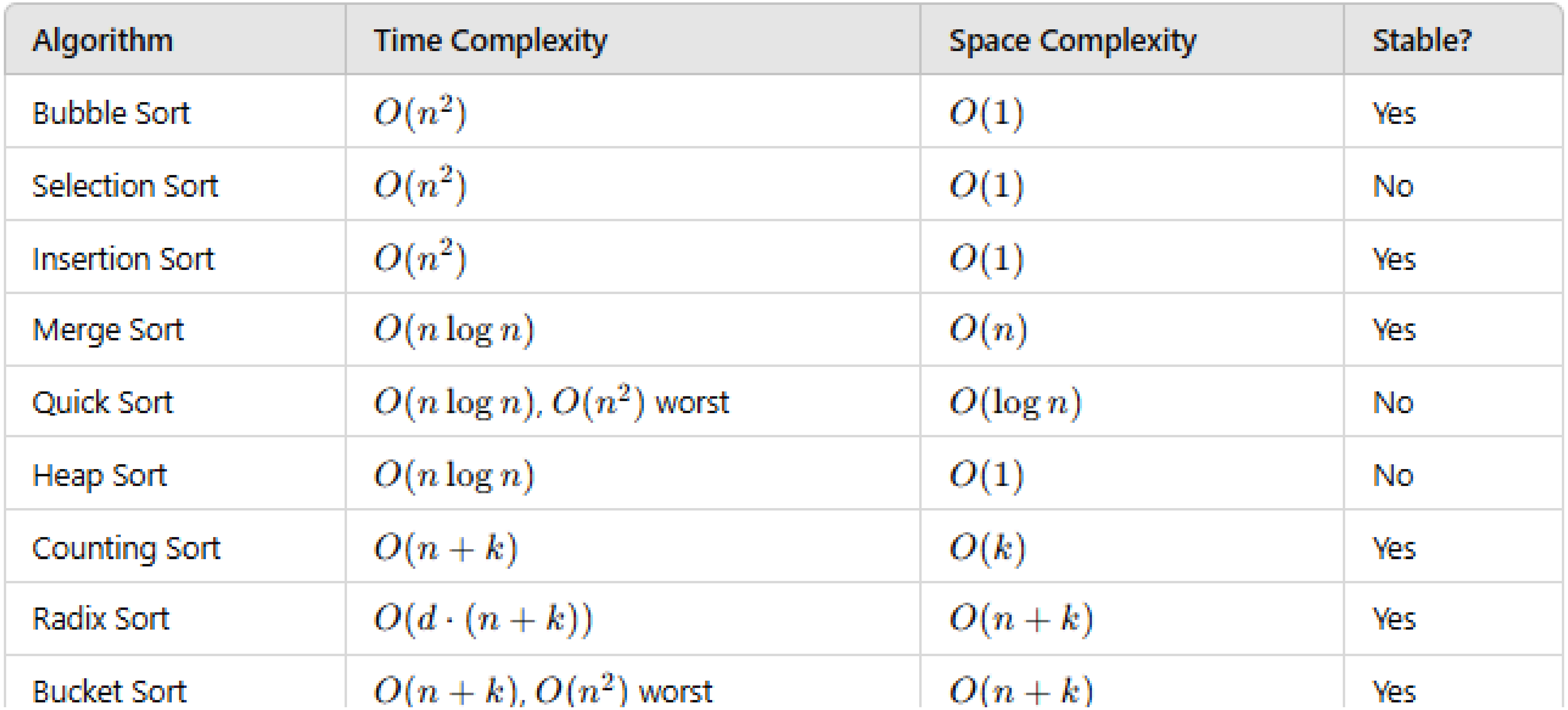
Foundations of Algorithm SCS1308

Dr. Dinuni Fernando PhD

Senior Lecturer

Sorting Algorithms



Algorithm	Time Complexity	Space Complexity	Stable?
Bubble Sort	$O(n^2)$	$O(1)$	Yes
Selection Sort	$O(n^2)$	$O(1)$	No
Insertion Sort	$O(n^2)$	$O(1)$	Yes
Merge Sort	$O(n \log n)$	$O(n)$	Yes
Quick Sort	$O(n \log n)$, $O(n^2)$ worst	$O(\log n)$	No
Heap Sort	$O(n \log n)$	$O(1)$	No
Counting Sort	$O(n + k)$	$O(k)$	Yes
Radix Sort	$O(d \cdot (n + k))$	$O(n + k)$	Yes
Bucket Sort	$O(n + k)$, $O(n^2)$ worst	$O(n + k)$	Yes

What is “stable” in a sort ?

- If it preserves the relative order of equal elements in the input list.
- When two elements have the same value, a stable sort ensures that their original order (as they appeared in the input) is maintained in the output.
- Input list = [(A, 2), (B, 1), (C, 2), (D, 1)]
- If sort by second number
- Stable sort output : [(B, 1), (D, 1), (A, 2), (C, 2)]
  - Relative order of (A,2)and (C,2)is preserved.
- Unstable Sort Output: [(D, 1), (B, 1), (C, 2), (A, 2)]
  - Relative order of (A,2and (C,2) is not preserved.

Table of Content

Why Sorting ?

- Fundamental techniques in computer science used to rearrange the elements of an array or list in a specific order,
- Ascending
- Descending.
- Sorting is essential in various applications, including data organization, searching, and optimization problems.

Shell Sort

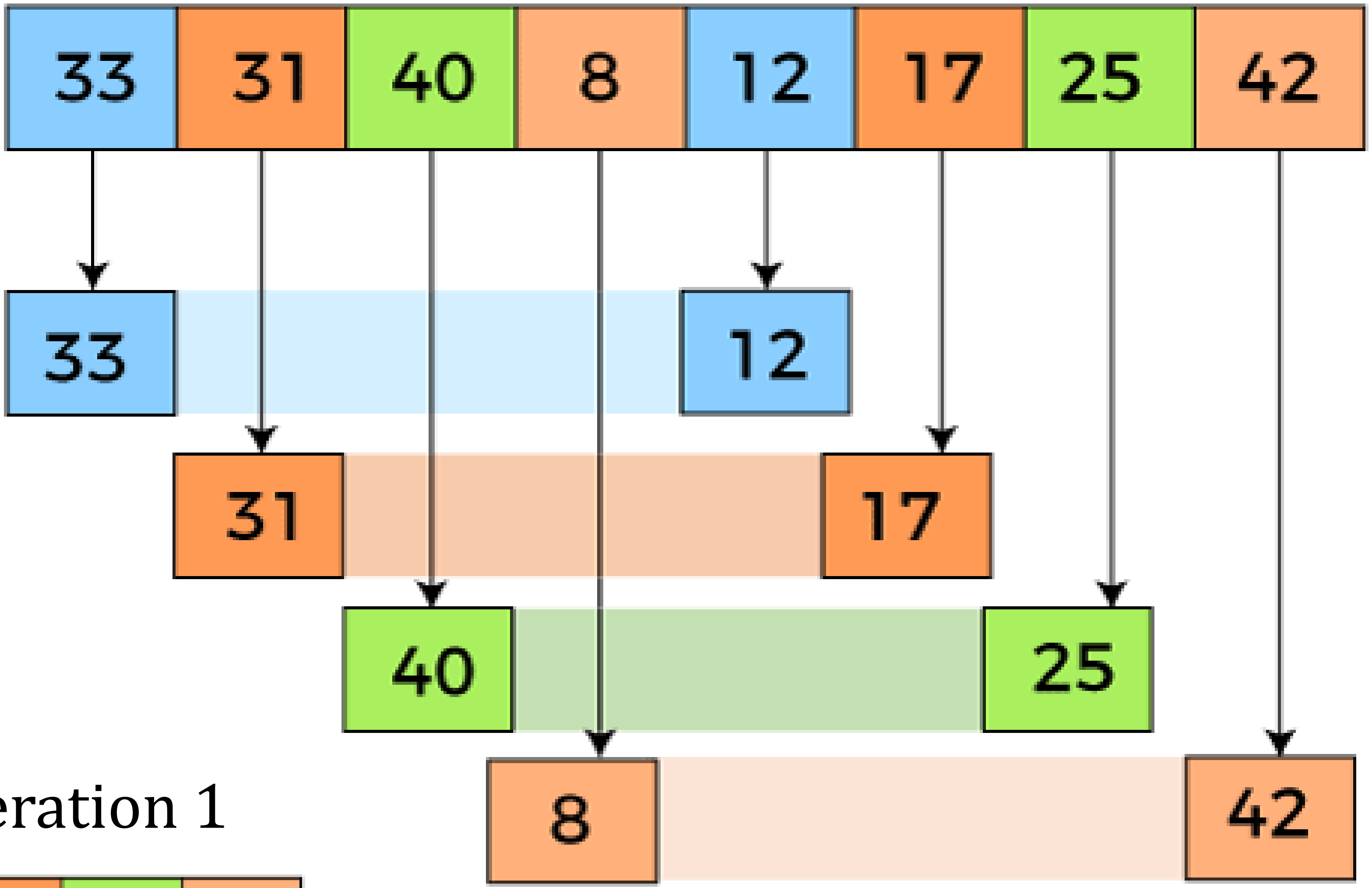
- Founded by Donald Shell in 1959.
- This is the first algorithm to break the quadratic time barrier.
- Is a highly efficient sorting algorithm and generalization of insertion sort.
- Also known as diminishing increment sort.
- Dividing interval varies from N/2, N/4, …..,1 intervals.

How does shell sort works ?

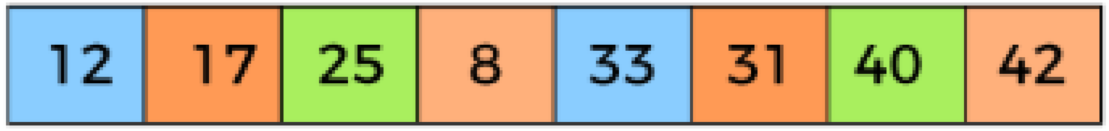
- Works by comparing distant elements rather than adjacent elements in an array or list where adjacent elements are compared.
- Uses increment sequence
- Makes multiple passes through a list.
- improves on the efficiency of insertion sort.
- decreases distance between comparisons.

Example – Shell Sort

- Interval/ Gap = 4



Finalized round after iteration 1

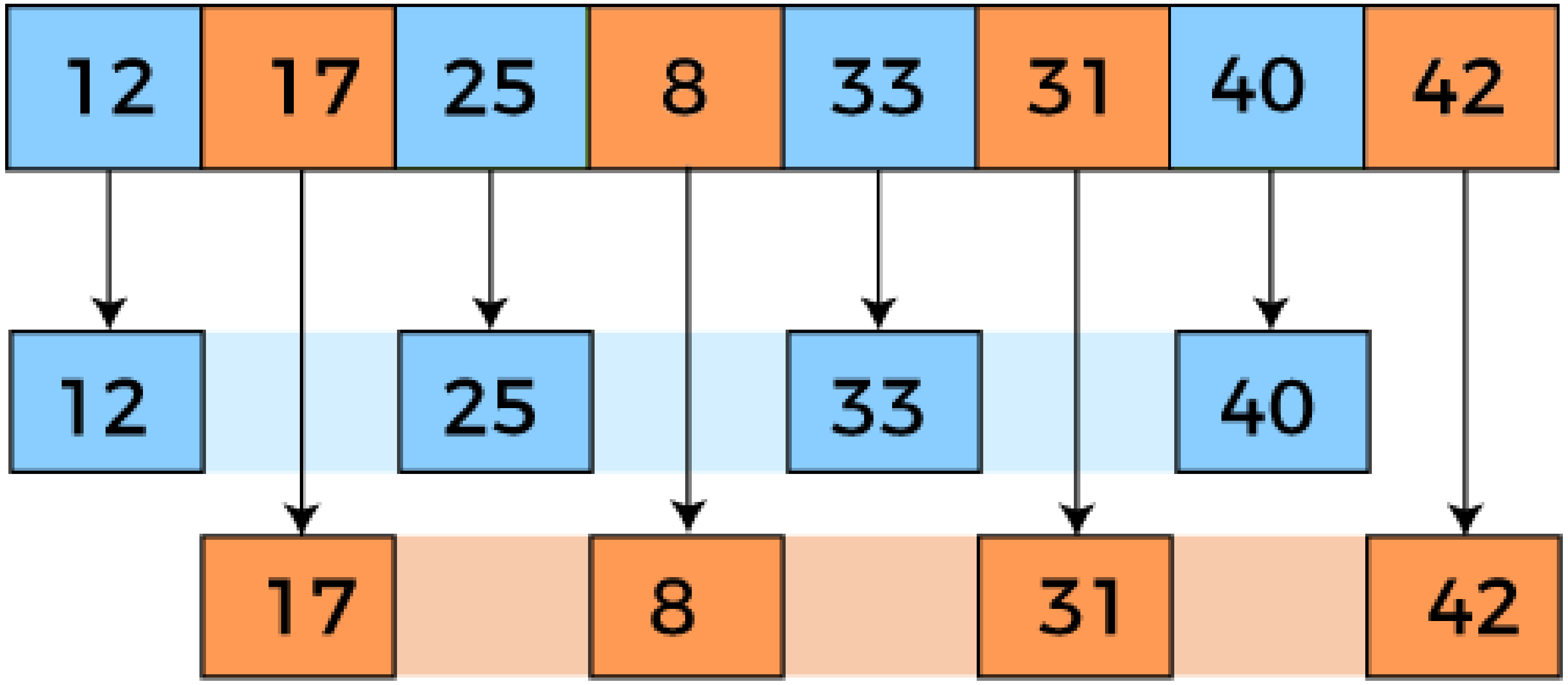


12	17	25	8	33	31	40	42

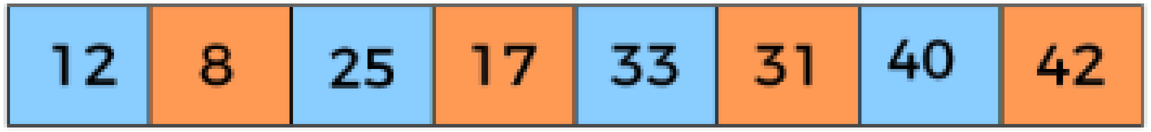
From Shell Sort - javatpoint

Example – Shell Sort

- Interval= 2



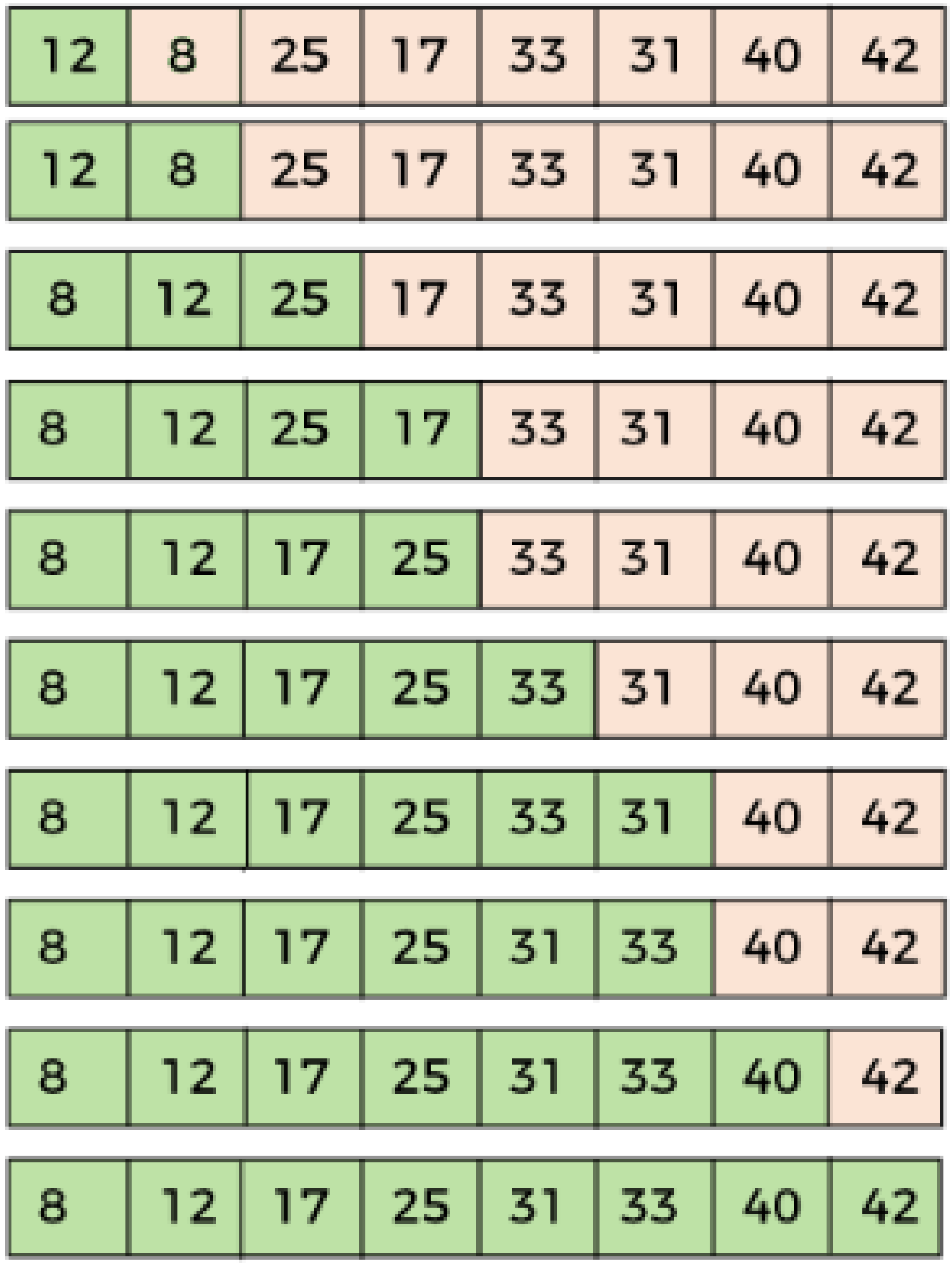
Finalized round after iteration 2



12	8	25	17	33	31	40	42

Example – Shell Sort

- Uses insertion sort



Shell Sort – Time complexity

Case	Time Complexity
Best case	O(nlogn)
Average case	$O(nlogn^2)$
Worst case	$O(n^2)$

Practical uses for Shell sort

- The dataset is relatively small.
- An in-place sorting algorithm is required to save memory.
    - In-place sorting: Shell Sort requires no additional memory for auxiliary arrays, unlike Merge Sort.
- Partial ordering of data can significantly reduce comparisons.
    - Efficiency for nearly sorted data: It can perform better than algorithms like Bubble Sort or Insertion Sort on partially sorted datasets.
- Sorting data in embedded systems or low-resource environments where simplicity and memory efficiency are critical.
    - Low-memory applications: Sorting in memory-constrained devices such as microcontrollers.

Practical limitations of shell sort

- Not suitable for large datasets: The time complexity ($O(n^3/2)$ to $O(n^2)$) makes it less efficient than Quick Sort or Merge Sort for large datasets.
- Dependence on gap sequence: The efficiency depends heavily on the choice of the gap sequence, which may require fine-tuning for specific use cases.

Radix Sort

- A non-comparative integer sorting algorithm that sorts data with integer keys by grouping keys by the individual digits which share the same significant position and value.
  - Works for integers or strings.
- Unlike other sorting methods, radix sort considers the structure of the keys.
  - Sorting is done by comparing bits in the same position.
- Time complexity O(d.(n+k))
  - d: Number of digits (or key length).
  - n: Number of elements.
  - k: Range of digits (typically 0–9 for decimal integers).

How does Radix sort works ?

- Sorts by grouping the numbers by their individual digits (or by their radix)
- It uses each radix digit as a key, and implements counting sort or bucket sort under the hood in order to do the work of sorting.
- Take the least significant digit of each key.
    - Sort numbers from least significant digit to most significant digit.
- Group the keys based on the digit, but otherwise keep the original order of keys.
- Repeat the grouping process with each more significant digit.

Example – Radix Sort

170,45,75,90,
802,24,2,66

#\position	100	10	1
	1	7	0
		4	5
		7	5
		9	0
	8	0	2
		2	4
			2
		6	6

Radix sort – sorting on 1^{s} position

#\position	100	10	1
	1	7	0
		9	0
	8	0	2
			2
		2	4
		4	5
		7	5
		6	6

Radix sort – sorting on 10s position

#\position	100	10	1
	8	0	2
			2
		2	4
		4	5
		6	6
	1	7	0
		7	5
		9	0

Radix sort – sorting on 100^s position

#\position	100	10	1
			2
		2	4
		4	5
		6	6
		7	5
		9	0
	1	7	0
	8	0	2

Practical uses of Radix sort

- The dataset consists of numbers or strings with a fixed range and length.
- Stability in sorting is important.
- Sorting is needed for massive datasets where comparisons are expensive.
- Practical applications
    - Sorting ZIP codes: ZIP codes are numeric and fixed in length, making them ideal for Radix Sort.
    - Telephone directories: Sorting phone numbers in ascending order.
    - Data indexing: Indexing records by numeric keys or hash codes in databases.
    - Genomic data: Sorting fixed-length DNA sequences.
    - Graphics applications: Sorting pixel intensities or RGB values for efficient image processing.

Limitations of Radix sort

- Memory usage: Requires extra space for auxiliary arrays (e.g., for counting sort).
- Limited flexibility: Not suitable for general-purpose sorting, as it works best for integers or fixed-length keys.
- Dependent on digit range: Sorting efficiency decreases if the range of digits (k) is very large.

Choosing Right Algorithm

- Small Data Sets: Bubble Sort, Insertion Sort.
- General Use Cases: Merge Sort, Quick Sort.
- Space Efficiency Required: Heap Sort.
- Special Use Cases:
  - Counting Sort: Limited range integers.
  - Radix Sort: Fixed-length keys like zip codes.
  - Bucket Sort: Uniformly distributed data.

Thank you!