|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SORTING ALGORITHMS** | | | | | |
|  | **Counting** | **Bucket** | **Radix** | **Shell** | **Gnome** |
| **Who** | Harold Seward | ? | Herman Hollerith  Harold Seward | Donald Shell | Hamid Sarbazi-Azad |
| **When** | 1954 | 1955 / 1945 | 1887  1954 | 1959 | 2000 |
| **What** | sorts the elements of an array by counting the number of occurrences of each unique element in the array. The count is stored in an auxiliary array and the sorting is done by mapping the count as an index of the auxiliary array. | Sorting algorithm that divides the unsorted array elements into several groups called buckets. | an integer sorting algorithm that sorts data with integer keys by grouping the keys by individual digits that share the same significant position and value. | a generalized version of the insertion sort algorithm. It first sorts elements that are far apart from each other and successively reduces the interval between the elements to be sorted. | Gnome sort works by building a sorted list one element at a time, getting each item to the proper place in a series of swaps. |
| **Used** | Used as subroutine in radix sort. | Useful when input is uniformly distributed over a range | Used on data, such as words and integers. It is also used for stably sorting strings. | Used on medium sized lists of elements | Best used on data that is almost sorted / relatively close to their sorted position. |
| **Conditions** | * Stable algorithm * Out-of-place sorting algorithm * Non-comparison approach (distribution sort) * Non-negative Integer | * Stable algorithm * Out-of-place algorithm * Non-comparison approach | * Stable algorithm * Out-of-place sorting algorithm * Non-comparison approach | * Unstable algorithm * In-place sorting algorithm * Comparison approach | * Stable algorithm * In-place sorting algorithm * Comparison approach |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TIME COMPLEXITY** | | | | | | | | |
|  | **Counting** | **Bucket** | **Radix** | **Shell** | **Gnome** | **Bubble** | **Insertion** | **Selection** |
| **Best** | O (n) | O (n + k)  Uniformly distributed | O (nk) | O (n log n)  Already sorted | O (n) | **O (n)**  comparisons  **O (1)** swap | **O (n)**  comparisons  **O (1)** swap | **O (n2)**  comparisons  **O (1)** swap |
| **Average** | O (n + k) | O (n + k)  Randomly distributed | O (nk) | O (n log2 n)  Depends on gap sequence | O (n2) | **O (n2)**  comparisons  **O (n2)** swaps | **O (n2)**  comparisons  **O (n2)** swaps | **O (n2)**  comparisons  **O (n)** swaps |
| **Worst** | O (k) | O (n2)  Skewed distribution | O (nk) | O (n4/3)  O (n2) | O (n2) | **O (n2)**  comparisons  **O (n2)** swaps | **O (n2)**  comparisons  **O (n2)** swaps | **O (n2)**  comparisons  **O (n)** swaps |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SPACE COMPLEXITY** | | | | | | | | |
|  | **Counting** | **Bucket** | **Radix** | **Shell** | **Gnome** | **Bubble** | **Insertion** | **Selection** |
| **Worst** | O (k)  O (n + k) | O (n) | O (n + k) | O (1) | O (1) | O (1) | O (1) | O (1) |

|  |  |  |  |
| --- | --- | --- | --- |
| **SORTING ALGORITHMS** | | | |
|  | **Bubble** | **Insertion** | **Selection** |
| **Who** | Iverson – coined term | John Mauchly – mentioned | ? |
| **When** | 1962 | 1946 | ? |
| **What** | is a sorting algorithm that compares two adjacent elements and swaps them until they are in the intended order.  Just like the movement of air bubbles in the water that rise up to the surface, each element of the array moves to the end in each iteration. Therefore, it is called a bubble sort. | is a sorting algorithm that places an unsorted element at its suitable place in each iteration.  Insertion sort works similarly as we sort cards in our hand in a card game.  We assume that the first card is already sorted then, we select an unsorted card. If the unsorted card is greater than the card in hand, it is placed on the right otherwise, to the left. In the same way, other unsorted cards are taken and put in their right place.  A similar approach is used by insertion sort. | is a sorting algorithm that selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list.  This algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.  **The subarray which is already sorted. Remaining subarray which is unsorted.** |
| **Used** | Used on lists with small number of elements | * the array is having a small number of elements * there are only a few elements left to be sorted | * When the array is NOT partially sorted. * When we have memory usage constraints. * When a simple sorting implementation is desired. |
| **Conditions** | * Stable algorithm * In-place sorting * Comparison Approach * Linear complexity | * Stable algorithm * In-place sorting * Comparison approach | * Stable with O(n) space * unstable algorithm with O(1) * In-place sorting * Comparison approach |