**Count Sort**

An algorithm for sorting a collection of objects according to keys that are positive integers that is it is an integer sorting algorithm.

**Working of counts sort**

1. Find out the maximum elements let it be Max from the given array
2. Initialize N array of length Max plus one with all elements zero this array is used for storing the count of the elements in the array
3. Store the count of each element at their respective index in count array
4. Store accumulative relative sum of the elements of the count array. It helps in placing the elements into the correct index of the sorted array
5. Find the index of each element of the original array in the count array. This gives the accumulative count. Place the element at the index calculated.

**Time Complexity**: n+r **Space Complexity**: n+r

**Priority Queue**

Priority queue is a special type of queue, an abstract data type that is like a queue and every element has some priority value associated with it the priority of the elements in a priority queue determines the order in which elements are served. However, if elements with the same priority occur, they are served accordingly to their order in the queue.

The element with the highest value is considered the highest priority element however in other cases we can assume the element with the lowest value as the highest priority element

**Implementation**

Priority queue can be implemented using an array a link list or heap data structure or a binary search tree. Among these data structure hey heap data is structure provides an efficient implementation of priority queues.

|  |  |  |  |
| --- | --- | --- | --- |
| Operations | peek | insert | delete |
| Linked List | O(1) | O(n) | O(1) |
| Binary Heap | O(1) | O(log n) | O(log n) |
| Binary Search Tree | O(1) | O(log n) | O(log n) |

**Basic Operations**

Basic operation level priority QR inserting removing and peeking elements

**Applications**

Dijkstra algorithm, implementing stack etc