When solving interval-related questions on LeetCode or any other platform, it's important to understand several key concepts and techniques. Here's a list of some essential ones:

**Interval Merging/Overlap Detection**

Often, you'll need to merge overlapping intervals or detect if two intervals overlap. This involves comparing start and end points of intervals.

**Sorting**

Many interval problems can be solved efficiently by sorting intervals based on their start or end points. This helps in various operations such as merging intervals or finding overlapping intervals.

**Greedy Algorithms**

Greedy algorithms are often useful for interval-related problems, especially when trying to maximize or minimize something within the given intervals.

**Stacks**

Stack data structure can be utilized for certain interval-related problems, especially when dealing with nested intervals or finding the maximum number of overlapping intervals.

**Prefix Sum**

In some scenarios, converting interval-related problems into prefix sum problems can lead to efficient solutions, especially when dealing with overlapping intervals.

**Binary Search**

Binary search can be employed in interval-related problems when searching for certain properties or values within intervals, especially when intervals are sorted.

**Dynamic Programming**

Dynamic programming techniques can be used in interval-related problems, particularly when you need to optimize some value over a range of intervals.

**Interval Tree**

For more advanced problems, implementing or utilizing an interval tree data structure can lead to efficient solutions, especially when dealing with a large number of intervals.

**Two Pointers**

Two pointers technique can be useful in some interval-related problems, especially when trying to find overlapping intervals or determining if intervals are disjoint.

**Graphs**

Certain interval problems can be modeled as graphs, and graph algorithms can be applied for efficient solutions, especially when dealing with interval scheduling or dependency problems.

**Segment Trees**

Segment trees can be used for efficient interval-related queries, such as finding minimum, maximum, or sum of values within a range of intervals.

**Sweep Line Algorithm**

This algorithm is particularly useful for solving geometric problems involving intervals by efficiently handling events along a line.

**Bit Manipulation**

Bit manipulation techniques can be applied in certain interval-related problems, especially when representing intervals or performing operations efficiently.

By mastering these concepts and techniques, you'll be better equipped to tackle interval-related questions on LeetCode or any other platform effectively. Practice and familiarity with different types of problems will also enhance your problem-solving skills in this domain.

Here's a list of widely used algorithms for solving interval-related problems on LeetCode:

**Merge Intervals**

Merge overlapping intervals into a set of non-overlapping intervals. This often involves sorting the intervals based on their start times and then merging overlapping intervals.

**Interval Intersection**

Given two lists of intervals, find the intersection of these intervals. This typically involves iterating over both lists and finding overlapping intervals.

**Interval Scheduling**

Given a set of intervals, maximize the number of non-overlapping intervals that can be selected. This problem can be solved using greedy algorithms by sorting the intervals based on their end times.

**Insert Interval**

Given a list of non-overlapping intervals sorted by their start time, insert a new interval into the list and merge if necessary.

**Interval Coverage**

Given a list of intervals and a target interval, determine if the target interval is covered by the given intervals. This can be solved by checking if there exists an interval that completely covers the target interval.

**Interval Partitioning**

Given a list of intervals, partition them into the minimum number of disjoint intervals. This problem can be solved using greedy algorithms or dynamic programming.

**Interval Sum**

Given a list of intervals with associated values, perform operations such as finding the sum, minimum, or maximum value over a given range of intervals. This can be solved using segment trees or prefix sum techniques.

**Interval Union**

Given a set of intervals, find the union of these intervals. This involves merging overlapping intervals and returning the resulting set of non-overlapping intervals.

**Interval Trees**

Implement interval trees data structure to efficiently perform operations such as insertion, deletion, and querying of intervals.

**Interval Overlaps Count**

Given a set of intervals, find the number of overlapping intervals for each interval. This can be solved using techniques such as sweep line algorithm or segment trees.

**Interval Overlap Count**

Given two lists of intervals, find the number of intervals in the first list that overlap with intervals in the second list.

**Interval Intersection Area**

Given two rectangles represented as intervals on the x-axis and y-axis, find the intersection area between them.

These algorithms cover a wide range of interval-related problems commonly found on LeetCode and other platforms. Mastering these algorithms will greatly enhance your ability to solve interval-related problems efficiently.