1. Merge Two Sorted Lists

Python
```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def mergeTwoLists(list1: ListNode, list2: ListNode) -> ListNode:
    dummy = curr = ListNode(0)
    while list1 and list2:
        if list1.val < list2.val:
            curr.next = list1
            list1 = list1.next
        else:
            curr.next = list2
            list2 = list2.next
        curr = curr.next
    curr.next = list1 or list2
    return dummy.next
```

2. Merge k Sorted Lists

Python
```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

import heapq

def mergeKLists(lists: List[ListNode]) -> ListNode:
    dummy = curr = ListNode(0)
    minHeap = []
    for head in lists:
        if head:
            heapq.heappush(minHeap, (head.val, head))
    while minHeap:
        val, node = heapq.heappop(minHeap)
        curr.next = node
        curr = curr.next
        if node.next:
            heapq.heappush(minHeap, (node.next.val, node.next))
    return dummy.next
```

3. Remove Duplicates from Sorted Array

Python
```
def removeDuplicates(nums: List[int]) -> int:
    i = 0
    for j in range(1, len(nums)):
        if nums[i] != nums[j]:
            i += 1
            nums[i] = nums[j]
    return i + 1
```

4. Search in Rotated Sorted Array

Python
```
def search(nums: List[int], target: int) -> int:
    if not nums:
        return -1
    n = len(nums)
    pivot = 0
    for i in range(1, n):
        if nums[i] < nums[i - 1]:
            pivot = i
            break
    left, right = 0, n - 1
    while left <= right:
        mid = left + (right - left) // 2
        sorted_mid = (mid + pivot) % n
        if nums[sorted_mid] == target:
            return sorted_mid
        elif nums[sorted_mid] > target:
            right = mid - 1
        else:
            left = mid + 1
    return -1
```

5. Find First and Last Position of Element in Sorted Array

Python
```
def searchRange(nums: List[int], target: int) -> List[int]:
    left, right = 0, len(nums) - 1
    while left <= right:
        mid = left + (right - left) // 2
        if nums[mid] == target:
            start = end = mid
            while start >= 0 and nums[start] == target:
                start -= 1
            while end < len(nums) and nums[end] == target:
                end += 1
            return [start + 1, end - 1]
        elif nums[mid] > target:
            right = mid - 1
        else:
            left = mid + 1
    return [-1, -1]
```

6. Sort Colors

Python
```
def sortColors(nums: List[int]) -> None:
    """
    Do not return anything, modify nums in-place instead.
    """
    low = 0
    mid = 0
    high = len(nums) - 1
    while mid <= high:
        if nums[mid] == 0:
            nums[low], nums[mid] = nums[mid], nums[low]
            low += 1
            mid += 1
        elif nums[mid
```
content_copy

7. Remove Duplicates from Sorted List

Python
```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def deleteDuplicates(head: ListNode) -> ListNode:
    curr = head
    while curr and curr.next:
        if curr.val == curr.next.val:
            curr.next = curr.next.next
        else:
            curr = curr.next
    return head
```

8. Merge Sorted Array

Python
```
def merge(nums1: List[int], m: int, nums2: List[int], n: int) -> None:
    """
    Do not return anything, modify nums1 in-place instead.
    """
    i = m - 1
    j = n - 1
    k = m + n - 1
    while i >= 0 and j >= 0:
        if nums1[i] > nums2[j]:
            nums1[k] = nums1[i]
            i -= 1
        else:
            nums1[k] = nums2[j]
            j -= 1
        k -= 1
    while j >= 0:
        nums1[k] = nums2[j]
        j -= 1
        k -= 1
```

9. Convert Sorted Array to Binary Search Tree

Python
```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def sortedArrayToBST(nums: List[int]) -> TreeNode:
    def helper(left, right):
        if left > right:
            return None
        mid = (left + right) // 2
        root = TreeNode(nums[mid])
        root.left = helper(left, mid - 1)
        root.right = helper(mid + 1, right)
        return root
    return helper(0, len(nums) - 1)
```

10. Insertion Sort List

Python
```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def insertionSortList(head: ListNode) -> ListNode:
    dummy = curr = ListNode(float('-inf'))  # Use negative infinity for
easier handling
    while head:
        next_node = head.next
        prev = dummy
        while curr.next and curr.next.val < head.val:
            prev = curr
            curr = curr.next
        head.next = curr.next
        curr.next = head
        curr = prev
        head = next_node
    return dummy.next
```

11. Sort Characters By Frequency

Python
```
from collections import Counter

def frequencySort(s: str) -> str:
    char_counts = Counter(s)
    sorted_chars = sorted(char_counts.items(), key=lambda x: x[1],
reverse=True)
    return ''.join(c * count for c, count in sorted_chars)
```

12. Max Chunks To Make Sorted

Python
```
def maxChunksToMakeSorted(arr: List[int]) -> int:
    max_so_far = -float('inf')
    chunks = 0
    for num in arr:
        max_so_far = max(max_so_far, num)
        if num == max_so_far:
            chunks += 1
    return chunks
```

13. Intersection of Three Sorted Arrays

Python
```
def intersect(nums1: List[int], nums2: List[int], nums3: List[int]) ->
List[int]:
    i, j, k = 0, 0, 0
    result = []
    while i < len(nums1) and j < len(nums2) and k < len(nums3):
        if nums1[i] == nums2[j] == nums3[k]:
            result.append(nums1[i])
            i += 1
            j += 1
```

```
            k += 1
        elif nums1[i] <= nums2[j] and nums1[i] <= nums3[k]:
            i += 1
```

13. Intersection of Three Sorted Arrays (continued)

Python

```
        elif nums2[j] <= nums1[i] and nums2[j] <= nums3[k]:
            j += 1
        else:
            k += 1
    return result
```

14. Sort the Matrix Diagonally

Python

```
def diagonalSort(mat: List[List[int]]) -> List[List[int]]:
    m, n = len(mat), len(mat[0])
    diagonals = {}
    for i in range(m):
        for j in range(n):
            diag = i - j  # key for the diagonal dictionary
            diagonals.setdefault(diag, []).append(mat[i][j])
    for diag, nums in diagonals.items():
        nums.sort()
        i, j = 0, max(diag, 0)  # Handle negative diagonals
        while i < len(nums) and j < m:
            mat[i][j] = nums[i]
            i += 1
            j += 1
    return mat[i]
```