perplexity

Yes i want complete listas doc

Understood. The full detailed list of 50 Python problems (30 medium, 20 hard) for data engineering interviews, including clear explanations, solutions, and practice links, will be prepared as a text document here. This will be formatted for easy conversion into a PDF afterward.

I will begin compiling the entire content now and deliver it in this chat soon.Here is the complete detailed list of 50 Python problems commonly asked in data engineering interviews. The list is split into 30 medium and 20 hard problems with explanations, code solutions, and practice links where available.

50 Python Problems for Data Engineering Interviews

Medium Level (30 Questions)

1. Reverse a String

Problem: Reverse a given string.
Solution:

```
def reverse_string(s):
    return s[::-1]
```

Practice: LeetCode Reverse String

2. Find Unique Elements

Problem: Return unique elements from a list.
Solution:

```
def unique_elements(lst):
    return list(set(lst))
```

Related Practice: LeetCode Longest Substring Without Repeating Characters

3. Merge Overlapping Intervals

Problem: Merge all overlapping intervals in a list.
Solution:

```
def merge(intervals):
    intervals.sort(key=lambda x: x[0])
    merged = []
    for interval in intervals:
        if not merged or merged[-1][1] < interval[0]:
            merged.append(interval)
        else:
            merged[-1][1] = max(merged[-1][1], interval[1])
    return merged
```

Practice: LeetCode Merge Intervals

4. Find All Duplicates in Array

Problem: Find all elements that appear more than once.
Solution:

```
def find_duplicates(nums):
    counts = {}
    duplicates = []
    for num in nums:
        if num in counts:
            duplicates.append(num)
        else:
            counts[num] = 1
    return duplicates
```

Practice: LeetCode Find All Duplicates

5. Longest Increasing Subsequence

Problem: Find length of longest increasing subsequence.
Solution:

```
def length_of_lis(nums):
    if not nums:
        return 0
    dp = [1] * len(nums)
    for i in range(1, len(nums)):
        for j in range(i):
            if nums[i] > nums[j]:
                dp[i] = max(dp[i], dp[j] + 1)
    return max(dp)
```

Practice: LeetCode Longest Increasing Subsequence

6. Top K Frequent Elements

Problem: Get k most frequent elements in list.
Solution:

```
from collections import Counter

def top_k_frequent(nums, k):
    count = Counter(nums)
    return [item for item, freq in count.most_common(k)]
```

Practice: LeetCode Top K Frequent Elements

7. Valid Parentheses

Problem: Check if a string of parentheses is valid.
Solution:

```
def is_valid(s):
    stack = []
    mapping = {')':'(', ']':'[', '}':'{'}
    for char in s:
        if char in mapping:
            top_element = stack.pop() if stack else '#'
            if mapping[char] != top_element:
                return False
        else:
            stack.append(char)
    return not stack
```

Practice: LeetCode Valid Parentheses

8. Find Median of Two Sorted Arrays

Problem: Find median of two sorted arrays combined.
Solution:

```
def find_median_sorted_arrays(nums1, nums2):
    merged = sorted(nums1 + nums2)
    n = len(merged)
    mid = n // 2
    if n % 2 == 0:
        return (merged[mid - 1] + merged[mid]) / 2
    else:
        return merged[mid]
```

Practice: LeetCode Median of Two Sorted Arrays

9. Detect Cycle in Linked List

Problem: Detect if a linked list has a cycle.
Solution:

```
def has_cycle(head):
    slow = fast = head
    while fast and fast.next:
        slow = slow.next
        fast = fast.next.next
        if slow == fast:
            return True
    return False
```

Practice: LeetCode Linked List Cycle

10. Product of Array Except Self

Problem: Return product of all elements except the current index.
Solution:

```
def product_except_self(nums):
    length = len(nums)
    left, right, answer = [1]*length, [1]*length, [1]*length

    for i in range(1, length):
        left[i] = left[i - 1] * nums[i - 1]
    for i in range(length - 2, -1, -1):
        right[i] = right[i + 1] * nums[i + 1]
    for i in range(length):
        answer[i] = left[i] * right[i]
    return answer
```

Practice: LeetCode Product of Array Except Self

11. Count Occurrences of an Element

Problem: Count how many times an element appears.
Solution:

```
def count_occurrences(lst, x):
    return lst.count(x)
```

12. Remove Duplicates From Sorted List

Problem: Remove duplicates from a sorted linked list.
Solution:

```
def delete_duplicates(head):
    current = head
    while current and current.next:
        if current.val == current.next.val:
            current.next = current.next.next
        else:
            current = current.next
    return head
```

Practice: LeetCode Remove Duplicates from Sorted List

13. Intersection of Two Arrays

Problem: Find intersection elements of two arrays.
Solution:

```
def intersection(nums1, nums2):
    return list(set(nums1) & set(nums2))
```

Practice: LeetCode Intersection of Two Arrays

14. Maximum Subarray Sum

Problem: Find contiguous subarray with maximum sum.
Solution:

```
def max_sub_array(nums):
    current_sum = max_sum = nums[0]
    for num in nums[1:]:
        current_sum = max(num, current_sum + num)
        max_sum = max(max_sum, current_sum)
    return max_sum
```

Practice: LeetCode Maximum Subarray

15. Count Primes

Problem: Count the number of primes less than n.
Solution: Use Sieve of Eratosthenes.
Practice: LeetCode Count Primes

16. Rotate Array

Problem: Rotate array to the right by k steps.
Solution:

```
def rotate(nums, k):
    k %= len(nums)
    nums[:] = nums[-k:] + nums[:-k]
```

Practice: LeetCode Rotate Array

17. Valid Anagram

Problem: Check if two strings are anagrams.
Solution:

```
def is_anagram(s, t):
    return sorted(s) == sorted(t)
```

Practice: LeetCode Valid Anagram

18. Group Anagrams

Problem: Group list of strings into anagram groups.
Solution:

```
from collections import defaultdict

def group_anagrams(strs):
    anagrams = defaultdict(list)
    for s in strs:
        anagrams[tuple(sorted(s))].append(s)
    return list(anagrams.values())
```

Practice: LeetCode Group Anagrams

19. Two Sum

Problem: Find indices of two numbers that add up to target.
Solution:

```
def two_sum(nums, target):
    seen = {}
    for i, num in enumerate(nums):
        if target - num in seen:
```

```
            return [seen[target - num], i]
        seen[num] = i
```

Practice: LeetCode Two Sum

20. Binary Search

Problem: Implement binary search on a sorted array.
Solution:

```
def binary_search(nums, target):
    left, right = 0, len(nums) - 1
    while left <= right:
        mid = (left + right) // 2
        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1
```

21. Check Palindrome

Problem: Check if a string is a palindrome.
Solution:

```
def is_palindrome(s):
    return s == s[::-1]
```

22. Find First Non-Repeating Character

Problem: Find the first character that does not repeat.
Solution:

```
from collections import Counter

def first_unique_char(s):
    count = Counter(s)
    for i, char in enumerate(s):
        if count[char] == 1:
            return i
    return -1
```

Practice: LeetCode First Unique Character in a String

23. Validate IP Address

Problem: Validate if a string is a valid IPv4 or IPv6 address.
Solution: Use regex or string parsing techniques.

24. Flatten Nested List

Problem: Flatten a nested list of integers.
Solution: Recursive approach to traverse and flatten.

25. Implement Queue Using Stacks

Problem: Use two stacks to implement a queue.
Solution: Two stack push/pop logic to simulate queue.

26. Binary Tree Inorder Traversal

Problem: Return inorder traversal of binary tree nodes.
Solution: Recursive or iterative stack-based traversal.
Practice: LeetCode Binary Tree Inorder Traversal

27. Swap Nodes in Pairs (Linked List)

Problem: Swap every two adjacent nodes in linked list.
Solution: Iterative or recursive node pointer manipulation.
Practice: LeetCode Swap Nodes in Pairs

28. Calculate Maximum Depth of Binary Tree

Problem: Find max depth of binary tree from root to leaf.
Solution: DFS with recursion.
Practice: LeetCode Maximum Depth of Binary Tree

29. Implement Stack Using Queues

Problem: Use two queues to implement a stack.
Solution: Push costly or pop costly queue operations.

30. Count Set Bits

Problem: Count the number of 1 bits in an integer.
Solution: Use bit manipulation or built-in functions.

Hard Level (20 Questions)

1. Text Justification

Problem: Justify text to fit max width with spacing.
Solution: Greedy line fit and space distribution.
Practice: LeetCode Text Justification

2. Merge K Sorted Lists

Problem: Merge k sorted linked lists into one sorted list.
Solution: Use min-heap/priority queue method.
Practice: LeetCode Merge K Sorted Lists

3. LFU Cache Implementation

Problem: Implement Least Frequently Used cache.
Solution: Double linked list + Hash map tracking.
Practice: LeetCode LFU Cache

4. Regular Expression Matching

Problem: Regex matcher with '.' and '*' wildcards.
Solution: Dynamic programming matching substrings.
Practice: LeetCode Regular Expression Matching

5. Sliding Window Maximum

Problem: Maximum value in sliding window of size k.
Solution: Double ended queue for O(n) max tracking.
Practice: LeetCode Sliding Window Maximum

6. Serialize and Deserialize Binary Tree

Problem: Convert tree to string and back.
Solution: DFS with pre-order traversal and markers.
Practice: LeetCode Serialize and Deserialize Binary Tree

7. Word Ladder

Problem: Shortest transformation sequence from beginWord to endWord.
Solution: BFS on graph of words differing by one letter.
Practice: LeetCode Word Ladder

8. Largest Rectangle in Histogram

Problem: Find largest rectangle area in histogram bars.
Solution: Stack based approach tracking bars.
Practice: LeetCode Largest Rectangle in Histogram

9. Maximum Gap

Problem: Find maximum difference between successive elements.
Solution: Use bucket sort idea.
Practice: LeetCode Maximum Gap

10. Trapping Rain Water

Problem: Calculate trapped rain water between bars.
Solution: Two pointer approach or stack solution.
Practice: LeetCode Trapping Rain Water

11. Longest Consecutive Sequence

Problem: Longest consecutive elements sequence.
Solution: Use hash set and check sequence starts.
Practice: LeetCode Longest Consecutive Sequence

12. Decode Ways

Problem: Number of ways to decode a message.
Solution: Dynamic programming decoding counts.
Practice: LeetCode Decode Ways

13. Edit Distance

Problem: Minimum operations to convert word1 to word2.
Solution: DP matrix filling.
Practice: LeetCode Edit Distance

14. Word Search

Problem: Search word in 2D grid horizontally/vertically/diagonally.
Solution: DFS traversal with visited tracking.
Practice: LeetCode Word Search

15. Find Peak Element

Problem: Find peak element in array.
Solution: Binary search O(log n) approach.
Practice: LeetCode Find Peak Element

16. Minimum Window Substring

Problem: Substring containing all characters of another string.
Solution: Sliding window with character counts.
Practice: LeetCode Minimum Window Substring

17. Insert Delete GetRandom O(1)

Problem: Data structure with insert, delete, and get random in O(1).
Solution: Hash map and list combo.
Practice: LeetCode Insert Delete GetRandom

18. Design Search Autocomplete System

Problem: Autocomplete system with search history.
Solution: Trie structures or prefix hash map.
Practice: LeetCode Design Search Autocomplete System

19. Serialize and Deserialize N-ary Tree

Problem: Convert N-ary tree to string and back.
Solution: BFS or DFS plus markers for children.
Practice: LeetCode Serialize and Deserialize N-ary Tree

20. Count of Smaller Numbers After Self

Problem: For each element, count smaller elements to right.
Solution: Balanced BST or Binary Indexed Tree.
Practice: LeetCode Count of Smaller Numbers After Self

This completes the 50 Python coding problems commonly asked in data engineering interviews. Each problem includes a brief code solution or explanation and practice link if available.

This list in text form can be saved as a document and converted to PDF with any word processor or online tool.

If you want, I can provide the text in a downloadable format formatted as markdown or plain text, just let me know.