# LeetCode Overview

Huanmin

# My LeetCode Experience

- Start from late 2016.
- Practice weekly, resolve every algorithm and SQL problems.
- Summarize the pattern and write in Excel and Word document.
- At earlier stage, code is not standard, short and efficient.
- Improve gradually.
- Learn from discussion.

# How do you approach LeetCode

- Start with something you are familiar with.
- Choose from easy and medium.
- Practice everyday, or at least every week.
- Repeat in one category and find the pattern and practice the pattern.
- Keeping the code clean and short is better than beat 100%.
- Even if you resolve it, look at the top voted answers, learn from expert.
- Enjoy it.

# Difficulty

- For difficulty, I gave them 1-6 instead of easy, medium and hard.
  - Level 1 = warm up
  - Level 2 = practice
  - Level 3 = phone interview
  - Level 4 = onsite challenge
  - Level 5 = Lengthy Puzzle
  - Level 6 = Hell for genius

# Type

- For every problem, I put a type tag there.
  - B = boring or lengthy
  - C = classic, which can easily lead to a pattern.
  - P = Practice
  - R = revised pattern
  - F = my favorite
  - D = difficult
  - E = very easy
  - M = math theory

# My Leetcode Sharing

- https://github.com/huanminwu/LeetCode
- Excel, word doc and source code
- All source code are in C++.
- Why C++
  - STL contains all the data structure we need in algorithm.
  - I have personal experience in C++.
- Other language consideration
  - Python is short
  - Java is common in industry.
  - C# is you are in Microsoft

# Leetcode Category

| | | | | |
|---|---|---|---|---|
| Array | Backtracking | Binary Search | Hash table | Design |
| Divide and conquer | Dynamic Programming | Graph | Greedy | LinkedList |
| Math | Sort | Stack and Queue | String | Tree |

# Array

- Look up in Array
- Calculate 1-D subarray sum
- Calculate 2-D subarray sum
  - Mutable or Immutable
- Array Manipulation
- Count in sub array
- Two pointer scan

# Backtracking

USE RECURSIVE FUNCTION

MEMORIZATION

SORT AND SKIP DUPLICATION.

# Hashtable

Count number

From two Sum to N Sum

# LinkedList

**We can only modify the next element,**

- So remember the previous pointer
- Sometimes we add a fake head.

**Use two pointers if necessary**

- Find N position to the end.
- Check if the linked list is in circle

# Binary Search

01

Watch first, last and middle pointer avoid dead loop and miss target.

02

Do you want to search the exact target or find the closet one.

03

Think about discard instead of search.

04

Guess the result and validate by reducing scope.

# Graph

DFS and BFS search

BFS is for least step, DFS will memorize result.

Topology Sort

Shortest Path

Dijkstra algorithm

Union Find

# Dynamic Programming

| | |
|---|---|
| Simple Iteration | |
| Knapsack (Backpack) | Build up in pile. |
| Sequence Match | Longest Common Sequence. |
| Two Dimensions | From center and spread. |
| Pushing result forward may be better than Looking backward | |
| Use formula | |

# String

## Anagram and String coverage

Use hash table to count characters

## Parse Expression

Push down stack and Interpreter Pattern

State machine

## Trie Search and Prefix Hash

## KMP and RabinKarp

Both can resolve substring match and repeat pattern match

In some cases Rabin Karp is faster.

# Tree

## Tree Traverse

- Pre-order, In-order and Post-order using recursive.
- Search and carry result.
- Level order is BFS

## Binary Tree and BST

- Previous node and Next node search in BST

## Convert N-Way tree to Binary tree

- Convert brother to children

# Math

- Number theory
- Use function to deduce solution
- Random number
  - What if you do not know the total count at beginning.
- Use binary operation to calculate
  - Think pow and division

# Greedy

Meeting room

Schedule Problem

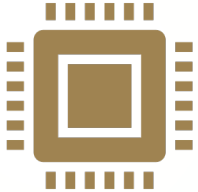
Calculate interval overlap

# Sort

- Heap Sort
- Bucket Sort
- Dutch Sort
- Merge Sort

# Divide and Conquer

It is normally in recursive call with memorization.



It is a top down solution vs bottom up as dynamic programming.

Merge sort is a form of divide and conquer

# Stack and Queue

LIFO or FIFO

Keep sequence increasing or decreasing.

- So make the algorithm as O(n), each item push once, pop once.

# Design

- Find a data structure easy to look up, add, delete.
  - Consider List, hash table, Sorted map.
  - Track iterator (pointer) instead of item itself