Author: Wasiur Rahman

Interview Preparation

Version 1.0

Table of Contents

1 Yelp

1.1 Merge Intervals

Given a collection of intervals, merge all overlapping intervals. For example,

```
Given [1,3],[2,6],[8,10],[15,18],
return [1,6],[8,10],[15,18].
```

The idea is to sort them based on their starting times. Then we look at all the intervals in increasing starting order. For each one, if we see that their end time is greater than or equal to the starting time of the previous one, we simply merge it.

For instance, if we had [1,3], [2,6], we see that the starting time of the second interval is less than the end of the first interval. Thus we merge those intervals into [1, 6].

Another edge case is that the intervals can be [1,3], [3, 6]. ie the end and starting times are equal. In that case we still want to merge the two intervals. We will take care of that scenario in the code.

Finally, the benefit of the initial sort is that it guarantees that the starting of the first integer is always less than or equal to the second interval's starting. As a result, by merging the two intervals, we will never overwrite the starting of the initial interval.

```java
/**
 * Definition for an interval.
 * public class Interval {
 *     int start;
 *     int end;
 *     Interval() { start = 0; end = 0; }
 *     Interval(int s, int e) { start = s; end = e; }
 * }
 */
public class Solution {
    public List<Interval> merge(List<Interval> intervals) {
        Collections.sort(intervals, new Comparator<Interval>(){
            @Override
            public int compare(Interval obj0, Interval obj1) {
                return obj0.start - obj1.start;
            }
        });

        List<Interval> ret = new ArrayList<>();
        Interval prev = null;
        for (Interval inter : intervals) {
            if (  prev==null || inter.start>prev.end ) {
                ret.add(inter);
                prev = inter;
            } else if (inter.end>prev.end) {
                // Modify the element already in list
                prev.end = inter.end;
            }
        }
        return ret;
    }
}
```

2 Facebook

2.1 Move Zeroes

Given an array nums, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements.

For example, given nums = [0, 1, 0, 3, 12], after calling your function, nums should be [1, 3, 12, 0, 0].

Note: You must do this in-place without making a copy of the array. Minimize the total number of operations.

```
/**
 * Definition for an interval.
 * public class Interval {
 *     int start;
 *     int end;
 *     Interval() { start = 0; end = 0; }
 *     Interval(int s, int e) { start = s; end = e; }
 * }
 */
public class Solution {
    public List<Interval> merge(List<Interval> intervals) {
        if (nums == null || nums.length == 0) return;

        int insertPos = 0;
        for (int num: nums) {
            if (num != 0) nums[insertPos++] = num;
        }

        while (insertPos < nums.length) {
            nums[insertPos++] = 0;
        }
    }
}
```

2.2 Palindrome

Given a string s, determine is you can scramble its letters to make its a palindrome.

2.3 Search Binary Rotated

Given a rotated, sorted array, and an element to be searched, write an efficient search function.

2.4 Search a given element in a binary tree (not bst)

Like title says.

2.5 Find the shift location in a sorted array

Find the shift location in an array with a single shift shuffle.

2.6 Select Kth Element

Find Kth largest value in an unordered array. Bonus points for complexity better than O(nlog(n)).

2.7 Set Powerset

Write a function to generate a set's powerset.

2.8 XML Parser

Given an XML string, return the parsed tree structure.

2.9 Small Fixed Map Search

Search google...

2.10 Snakes on a Plane

Other machines on your wifi network are consuming all available bandwidth. Figure out a way to make them stop without having access to their machines.

2.11 Sort using a custom alphabet

Sort an input of characters into a specific order.

2.12 Sorted Iterator over K Sorted Lists

Given K sorted lists of up to N elements in each list, return a sorted iterator over all the items.

2.13 Spell Check with Wildcard

This is a medium to difficult string processing question. There are a variety of reasonable solutions that require some familiarity with data structures.

2.14 Status Search

This question is best for infrastructure candidates who should have a good understanding of distributed systems. You probably dont want use this question if they are well versed with working with search. They should know the various trade-offs. You can ask this question for both experienced and new grad candidates as long as you calibrate differently.

2.15 Stock profit

This is a low pass filter question I use to quickly eliminate candidates. Its on the easier with a single aha insight that can easily be discovered once they have the brute force solution.

2.16 Store millions of items efficiently in memory whose values can be 0,1,2

We want to optimize mem space as much as possible.

2.17 strace causes programs to fail

Do this after

2.18 String regexp matcher simple

Write a regular expression matching function.

2.19 String reverse words

Given a string, return a string with the orders of the words reversed.

2.20 String strstr

Implement strstr in C.

2.21 Super Enumerator

Do this later. iOS.

2.22 System

Ask candidate to take something that they have already done and ask how to scale it to Facebook level.

2.23 Tail

Write a simple version of tail that prints out the last 10 lines in a file.

2.24 Task dependency duration

Given a dependency graph of tasks with a set duration, calculate the amount of time it takes to finish a task and all of its dependencies when you can 1. work on one task at a time, 2. work on infinite, 3. work on k tasks at a time.

2.25 Task scheduler

Given a list of input tasks to run, and the cooldown interval, output the minimun number of time slots required to run them.

2.26 Telnet port 80

What is this question asking?

2.27 The number of subsets with min + max <= k

For a given vector of integers and the integer k, find the number of non-empty subsets S such that min(S) + max(S) <= k.

For example, for k=8, and vector [2,4,5,7], the solution is 5 and these are all the subsets that satisfy the requirement: [2], [4], [2,4], [2,4,5], [2,5]

2.28 Three non-overlapping intervals with maximal sum of elements

Given array of N positive integers...

2.29 Three non-overlapping intervals with maximal sum of elements

Given array of N positive integers 1. Find interval of size k with the largest sum. 2. Find 3 non-overlapping intervals, each of size k with the largest total sum.

2.30 Tic Tac Toe Winner

Implement the move method of tictactoe that returns whether the player's move had won. The candidate needs to define the state of the game and the move() method should take care of state manipulation and return whether the player made the move, has won the game.

2.31 Top 10 Listens

Imagine we want to build a product that shows a user's favorite songs recently. We've decided that we want to show the top 10 songs for a particular user over the past 7 days. Design the backend.

2.32 Trade ops case - apac licensed shipment

Wtf question

2.33 Tree Print all Paths

For a given binary tree, print paths from root to all leaf nodes, one path per line.

2.34 Tree Print by Levels

Print out a binary tree level by level on a separate line.

2.35 Tree to list

Given a binary tree made of these nodes, convert it, in-place (dont alloc new nodes) into a circular doubly linked list in the same order as an in-order traversal of the tree.

2.36 Trie String Match

Check if a word is valid given a trie.

2.37 Typeahead suggestions

Design a system to power typeahead suggestions for a site like google or youtube.

2.38 Type checking a program with arithmetic expressions, variables, and strings

wtf... do it later

2.39 UIE AdUnit

Use HTML/CSS to build a three col ad unit component

2.40 UIE Array Flatten

Builf a function that takes an array that may contain levels of nested arrays and returns a flattened version of the array.

2.41 UIE Pinterest

Build a url shortener service.

2.42 Unified address book

Design an application which manages contacts from multiple providers. For example, facebook, linkedin, gmail.

2.43 Uniformly pick random max integers in an array

Given an array of integers, return the index of one of the largest element, chosen uniformly at random.

2.44 Union of two lists of intervals on the real number line

Given two sorted lists of non-overlapping intervals on the real number line, return the union. It should also be sorted and contain exclusively non-overlapping intervals (as in a minimal number of intervals).

2.45 Unique Files

Warmup. Tests working with data structures as well as CLI args, opening files and managing exit codes.

2.46 Unknown server

The idea is you're given the root password to a system with no other knowledge of whats on the system. The idea is your friend gives you IP address and root password to their server and then runs away. You know nothing and need to discover as much as possible.

2.47 Unmount FS in use

Combined "restart webserver without impacting users" and "unmount FS in use". This is an experimental question being developed by ...

2.48 Validate BST

Given a binary tree with numeric values, determine if the tree is a valid Binary Search Tree.

2.49 Vmstat anomaly

A simple yet effective question. Paste some vmstat output into collabedit and ask them to tell you what they see. Another good warm up that is quick and can have reasonable signal.

2.50 Web browser no network

...