From: **Daily Coding Problem** founders@dailycodingproblem.com
Subject: How To Solve a Hard Programming Interview Question
Date: February 10, 2019 at 5:22 PM
To: marco.olea@ciencias.unam.mx

DP

Hey there,

Hope your day is going well! I wrote a tutorial on how to solve hard interview questions and thought you might be interested.

So let's go over the thought process for solving tricky coding interview questions. I often find it's not enough to just be able to solve the problem; you really need to vocalize your thought process. This shows that you're a strong communicator and that you didn't just get lucky solving this one particular problem.

The question we'll work through is the following: **return a new sorted merged list from K sorted lists, each with size N.** Before we move on any further, you should take some time to think about the solution!

1. First, go through an example. This buys time, makes sure you understand the problem, and lets you gain some intuition for the problem. For example, if we had `[[10, 15, 30], [12, 15, 20], [17, 20, 32]]`, the result should be `[10, 12, 15, 15, 17, 20, 20, 30, 32]`.

2. Next, give any solution you can think of (even if it's brute force). It seems obvious that if we just flattened the lists and sorted it, we would get the answer we want. The time complexity for that would be `O(KN log KN)`, since we have `K * N` total elements.

3. The third step is to think of pseudocode—a high-level solution for the problem. This is where we explore different solutions. The things we are looking for are better space/time complexities but also the difficulty of the implementation. You should be able to finish the solution in 30 minutes. Here, we can see that we only need to look at K elements in each of the lists to find the smallest element initially. Heaps are great for finding the smallest element. Let's say the smallest element is E. Once we get E, we know we're interested in only the next element of the list that held E. Then we'd extract out the second smallest element and etc. The time complexity for this would be `O(KN log K)`, since we remove and append to the heap `K * N` times.

4. Initialize the heap. In Python this this is just a list. We need K tuples. One for the index for which list among the list of lists the element lives; one for the element index which is where the element lives; and the value of the element. Since we want the key of the heap to be based on the value of the element, we should put that first in the tuple.

5. While the heap is not empty we need to:
    - Extract the minimum element from the heap: (value, list index, element index)
    - If the element index is not at the last index, add the next tuple in the list index.

4. Write the actual code. Ideally, at this point, it should be clear how the code should look like. Here's one example:

```
def merge(lists):
    merged_list = []

    heap = [(lst[0], i, 0) for i, lst in enumerate(lists) if lst]
    heapq.heapify(heap)

    while heap:
        val, list_ind, element_ind = heapq.heappop(heap)

        merged_list.append(val)

        if element_ind + 1 < len(lists[list_ind]):
            next_tuple = (lists[list_ind][element_ind + 1],
                          list_ind,
                          element_ind + 1)
            heapq.heappush(heap, next_tuple)
    return merged_list
```

5. Think of test cases and run them through your interviewer. This shows that you're willing to test your code and ensure it's robust. I like to think of happy cases and edge cases. Our original example would be a happy case. Edge cases might be.
    - `lists` is `[]`.
    - `lists` only contains empty lists: `[[], [], []]`.
    - `lists` contains empty lists and non-empty lists: `[[], [1], [1,2]]`.
    - `lists` contains one list with one element: `[[1]]`.
    - `lists` contains lists of varying size: `[[1], [1, 3, 5], [1, 10, 20, 30, 40]]`.
6. Finally, the interviewer should ask some follow-up questions. One common question is: what other solutions are there? There's actually another relatively simple solution that would use a divide-and-conquer strategy. We could recursively merge each half of the lists and then combine the two lists. This would have the same asymptotic complexities but would require more "real" memory and time.

Doing all these steps will definitely help you crystallize your thought process, grasp the problem better, and show that you are a strong communicator and help you land that job offer!

Best of luck!

Marc

If you liked this guide, feel free to forward it along! As always, shoot us an email if there's anything we can help with!
