

← Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

13% completed

Search Course

We'll cover the following

Next Interval (hard)

Solution

Code

Time complexity

Space complexity

Solution Review: Problem Challenge 1

Next Interval (hard)#

Given an array of intervals, find the next interval of each interval. In a list of intervals, for an interval `i` its next interval `j` will have the smallest 'start' greater than or equal to the 'end' of `i`.

Write a function to return an array containing indices of the next interval of each input interval. If there is no next interval of a given interval, return -1. It is given that none of the intervals have the same start point.

Example 1:

Input: Intervals [[2,3], [3,4], [5,6]]

Output: [1, 2, -1]

Explanation: The next interval of [2,3] is [3,4] having index '1'. Similarly, the next interval of [3,4] is [5,6] having index '2'. There is no next interval for [5,6] hence we have '-1'.

Example 2:

Input: Intervals [[3,4], [1,5], [4,6]]

Output: [2, -1, -1]

Explanation: The next interval of [3,4] is [4,6] which has index '2'. There is no next interval for [1,5] and [4,6].

Solution#

A brute force solution could be to take one interval at a time and go through all the other intervals to find the next interval. This algorithm will take $O(N^2)$ where N is the total number of intervals. Can we do better than that?

We can utilize the **Two Heaps** approach. We can push all intervals into two heaps: one heap to sort the intervals on maximum start time (let's call it `maxStartHeap`) and the other on maximum end time (let's call it `maxEndHeap`). We can then iterate through all intervals of the `maxEndHeap` to find their next interval. Our algorithm will have the following steps:

- Take out the top (having highest end) interval from the `maxEndHeap` to find its next interval. Let's call this interval `topEnd`.
- Find an interval in the `maxStartHeap` with the closest start greater than or equal to the start of `topEnd`. Since `maxStartHeap` is sorted by 'start' of intervals, it is easy to find the interval with the highest 'start'. Let's call this interval `topStart`.
- Add the index of `topStart` in the result array as the next interval of `topEnd`. If we can't find the next interval, add '-1' in the result array.
- Put the `topStart` back in the `maxStartHeap`, as it could be the next interval of other intervals.
- Repeat steps 1-4 until we have no intervals left in `maxEndHeap`.

Code#

Here is what our algorithm will look like:

Java

Python3

C++

JS

```
1 from heapq import *
2
3
4 class Interval:
5     def __init__(self, start, end):
6         self.start = start
7         self.end = end
8
9
10 def find_next_interval(intervals):
11     n = len(intervals)
12
13     # heaps for finding the maximum start and end
14     maxStartHeap, maxEndHeap = [], []
15
16     result = [0 for x in range(n)]
17     for endIndex in range(n):
18         heappush(maxStartHeap, (-intervals[endIndex].start, endIndex))
19         heappush(maxEndHeap, (-intervals[endIndex].end, endIndex))
20
21     # go through all the intervals to find each interval's next interval
22     for _ in range(n):
23         # let's find the next interval of the interval which has the highest 'end'
24         topEnd, endIndex = heappop(maxEndHeap)
25         result[endIndex] = -1 # defaults to -1
26         if -maxStartHeap[0][0] >= -topEnd:
27             topStart, startIndex = heappop(maxStartHeap)
28             # find the the interval that has the closest 'start'
29             while maxStartHeap and -maxStartHeap[0][0] >= -topEnd:
30                 topStart, startIndex = heappop(maxStartHeap)
31             result[endIndex] = startIndex
```

Run

Save

Reset

Time complexity#

The time complexity of our algorithm will be $O(N\log N)$, where N is the total number of intervals.

Space complexity#

The space complexity will be $O(N)$ because we will be storing all the intervals in the heaps.

← Back

Next →

Problem Challenge 1

Introduction

Mark as Completed

Report an Issue

Ask a Question