JavaScript Merge k Sorted Lists

Challenge

You are given an array of `k` linked-lists `lists`, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

1st Example

```
Input: lists = [[1,4,5],[1,3,4],[2,6]]
Output: [1,1,2,3,4,4,5,6]
Explanation: The linked-lists are:
[
    1->4->5,
    1->3->4,
    2->6
]
merging them into one sorted list:
1->1->2->3->4->4->5->6
```

2nd Example

```
Input: lists = []
Output: []
```

3rd Example

```
Input: lists = [[]]
Output: []
```

Constraints

- `k == lists.length`
- `0 <= k <= 10^4`
- `0 <= lists[i].length <= 500`
- `-10^4 <= lists[i][j] <= 10^4`
- `lists[i]` is sorted in ascending order.
- The sum of `lists[i].length` will not exceed `10^4`.

Solution

```
const mergeKLists = (lists) => {
    const queue = new MinPriorityQueue(
                          {priority: x => x.val}
                      );

    for (const head of lists) {
        if (head) {
            queue.enqueue(head);
        }
    }
```

Solution continues on next page...

```
    let result = new ListNode();
    const head = result;

    while (!queue.isEmpty()) {
        const {val, next} = queue.dequeue().element;

        result.next = new ListNode(val);

        result = result.next;

        if (next) {
            queue.enqueue(next);
        }
    }

    return head.next;
};
```

Explanation

I've built a function called `mergeKLists` that takes an array of linked lists, `lists`, as input. The purpose of this function is to merge all the linked lists into one sorted linked list and return the head of the merged list.

Inside the function, a minimum priority queue called `queue` is created using the `MinPriorityQueue` class. The priority of each element in the queue is determined by its `val` property.

A `for...of` loop is used to iterate over each linked list in the `lists` array. Within the loop, the current linked list `head` is checked to see if it is not null or empty.

If the `head` is not null, it is enqueued into the `queue` using the `enqueue` method.

A new instance of the `ListNode` class called `result` is created. This will be used to store the merged list.

The value of `result` is assigned to a new variable called `head`. This variable represents the head of the merged list.

A `while` loop is entered that continues until the `queue` is empty.

Inside the loop, an element is dequeued from the `queue` using the `dequeue` method. The returned object is destructured to obtain the `val` and `next` properties of the dequeued element.

A new instance of the `ListNode` class is created with the value `val` and assigned to the `next` property of `result`.

The `result` variable is updated to point to the newly created node.

If the `next` property is not null, it means there are more elements in the linked list. In that case, the `next` element is enqueued into the `queue` using the `enqueue` method.

Once the `while` loop ends, the `next` property of the `head` variable, which represents the merged list, is returned as the output of the function.

In summary, this function merges multiple linked lists into one sorted linked list using a minimum priority queue. It iterates over each linked list, enqueues the non-null heads into the queue, and then dequeues the elements from the queue in sorted order to

create the merged list. The head of the merged list is returned as the output of the function.

Author: Trevor Morin

Copyright 2024 Superklok Labs