AlgoExpert Quad Layout JavaScript 12px Sublime Monok

Prompt Scratchpad Our Solution(s) Video Explanation Run Code

Solution 1 Solution 2

```javascript
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

// O(nlog(k) + k) time | O(n + k) space - where where n is the total
// number of array elements and k is the number of arrays
function mergeSortedArrays(arrays) {
  const sortedList = [];
  const smallestItems = [];
  for (let arrayIdx = 0; arrayIdx < arrays.length; arrayIdx++) {
    smallestItems.push({
      arrayIdx,
      elementIdx: 0,
      num: arrays[arrayIdx][0],
    });
  }
  const minHeap = new MinHeap(smallestItems);
  while (!minHeap.isEmpty()) {
    const smallestItem = minHeap.remove();
    const {arrayIdx, elementIdx, num} = smallestItem;
    sortedList.push(num);
    if (elementIdx === arrays[arrayIdx].length - 1) continue;
    minHeap.insert({
      arrayIdx,
      elementIdx: elementIdx + 1,
      num: arrays[arrayIdx][elementIdx + 1],
    });
  }
  return sortedList;
}

class MinHeap {
  constructor(array) {
    this.heap = this.buildHeap(array);
  }

  isEmpty() {
    return this.heap.length === 0;
  }

  buildHeap(array) {
    const firstParentIdx = Math.floor((array.length - 2) / 2);
    for (let currentIdx = firstParentIdx; currentIdx >= 0; currentIdx--) {
      this.siftDown(currentIdx, array.length - 1, array);
    }
    return array;
  }

  siftDown(currentIdx, endIdx, heap) {
    let childOneIdx = currentIdx * 2 + 1;
    while (childOneIdx <= endIdx) {
      const childTwoIdx = currentIdx * 2 + 2 <= endIdx ? currentIdx * 2 + 2 : -1;
      let idxToSwap;
      if (childTwoIdx !== -1 && heap[childTwoIdx].num < heap[childOneIdx].num) {
        idxToSwap = childTwoIdx;
      } else {
        idxToSwap = childOneIdx;
      }
      if (heap[idxToSwap].num < heap[currentIdx].num) {
        this.swap(currentIdx, idxToSwap, heap);
        currentIdx = idxToSwap;
        childOneIdx = currentIdx * 2 + 1;
      } else {
        return;
      }
    }
  }

  siftUp(currentIdx, heap) {
    let parentIdx = Math.floor((currentIdx - 1) / 2);
    while (currentIdx > 0 && heap[currentIdx].num < heap[parentIdx].num) {
      this.swap(currentIdx, parentIdx, heap);
      currentIdx = parentIdx;
      parentIdx = Math.floor((currentIdx - 1) / 2);
    }
  }

  remove() {
    this.swap(0, this.heap.length - 1, this.heap);
    const valueToRemove = this.heap.pop();
    this.siftDown(0, this.heap.length - 1, this.heap);
    return valueToRemove;
  }

  insert(value) {
    this.heap.push(value);
    this.siftUp(this.heap.length - 1, this.heap);
  }

  swap(i, j, heap) {
    const temp = heap[j];
    heap[j] = heap[i];
    heap[i] = temp;
  }
}

exports.mergeSortedArrays = mergeSortedArrays;

```