Problem Solving: Heaps

Relevel
by Unacademy

Generic Class for Heap

```
class Heap {
 constructor(comparator){
     this.data = [];
     this.comparator = comparator;
 }
  swap(index1, index2) {
     const temp = this.data[index1];
     this.data[index1] = this.data[index2];
     this.data[index2] = temp;
 }
  bottomUp(index){
     if(index === 0) return;
     const data = this.data;

     const parentIndex = Math.floor((index - 1) /
2);

     if(this.comparator(data[index],
data[parentIndex])){
         this.swap(parentIndex, index);
         this.bottomUp(parentIndex);
     }
 }
```

```
  topBottom(index){
     const n = this.data.length;
     const data = this.data;

     let nextIndexTochoose = index;
     const lefti = index * 2 + 1;
     const righti = lefti + 1;

     if(lefti < n && this.comparator(data[lefti],
data[nextIndexTochoose])){
         nextIndexTochoose = lefti;
     }

     if(righti < n && this.comparator(data[righti],
data[nextIndexTochoose])){
         nextIndexTochoose = righti;
     }

     if(nextIndexTochoose !== index){
         this.swap(index, nextIndexTochoose);
         this.topBottom(nextIndexTochoose);
     }
  }
```

```
   size() {
      return this.data.length;
  }
 peek(){
      if(this.size() <= 0) return null;
      return this.data[0];
  }
   add(val) {
      this.data.push(val);
      this.bottomUp(this.data.length - 1);
  }
  // delete top element
  poll(){
      if(this.size() <= 0) return;

      if(this.size() === 1){
          this.data.pop();
          return;
      }

      const data = this.data
      data[0] = data[data.length - 1];
      data.pop();
      this.topBottom(0);
  }
}
```

Find kth smallest element in an array

```
var findKthSmallest = function(nums, k) {
const comparator = (a,b) => {
   return a > b
 }
const heap = new Heap(comparator);
 for(let i=0; i<nums.length; i++){
     heap.add(nums[i]);
     if(heap.size() > k){
         heap.poll();
     }
 }
return heap.peek();
};

let arr = [2,3,45,6,79,1,15];
console.log(findKthSmallest(arr, 2));
```

Find kth smallest element in an array

```
var findKthSmallest = function(nums, k) {
const comparator = (a,b) => {
   return a > b
 }
const heap = new Heap(comparator);
 for(let i=0; i<nums.length; i++){
     heap.add(nums[i]);
     if(heap.size() > k){
         heap.poll();
     }
 }
return heap.peek();
};

let arr = [2,3,45,6,79,1,15];
console.log(findKthSmallest(arr, 2));
```

Sort array by frequency

```
var frequencySort = function(nums) {
 const frequency = {};
 const result = [];
  for(let i =0; i < nums.length; i++){
   if(nums[i] in frequency)
     frequency[nums[i]]++;
   else{
     frequency[nums[i]] = 1;
   }
 }
  const comparator = (a,b) => {
   return a[1] < b[1]
 }
 const heap = new Heap(comparator);
  for(let key in frequency){
   heap.add([key, frequency[key]]);
 }
  while(heap.size()>0){
   let val = heap.peek();
   console.log(val)
   heap.poll();
   for(let i = 0; i < val[1]; i++)
     result.push(val[0]);
 }
 return result;
};
let arr = [1,2,2,2,3,3,3,3,4];
console.log(frequencySort(arr));
```

Combine k sorted string

```
var mergeKSortedArrays = (arr,k) => {
 const comparator = (a,b) => {
   return a[0] < b[0]
 }
 const heap = new Heap(comparator);
  for(let i = 0; i < arr.length; i++)
 {
   heap.add([arr[i][0], i, 1]);
 }
 let result = [];
  for(let i = 0; i < arr.length*arr[0].length; i++){
   let root = heap.peek();
   heap.poll();
   result.push(root[0]);
   if(root[2] < arr[root[1]].length)
     heap.add([arr[root[1]][root[2]], root[1], root[2]+1]);
  }
  return result;
}
let arr = [[1, 3, 5, 7],
           [2, 4, 6, 8],
           [0, 9, 10, 11]
           ];
console.log(mergeKSortedArrays(arr));
```

Find median in real time data

```
class StreamMedian{
 constructor(){
   const minComparator = (a,b) => {
     return a > b
   }

   const maxComparator = (a,b) => {
     return a < b
   }
   this.minHeap = new Heap(minComparator);
   this.maxHeap = new Heap(minComparator);
 }
addNumber(num){
   this.maxHeap.add(num);
   this.minHeap.add(this.maxHeap.peek());
   this.maxHeap.poll();

if(this.maxHeap.size()!=this.minHeap.size())
   {
this.maxHeap.add(this.minHeap.peek());
       this.minHeap.poll();
   }
 }
findMedian(){

if(this.minHeap.size()==this.maxHeap.size())
   {
                return   (this.minHeap.peek()+
this.maxHeap.peek())/2;
   }
   else
   {
       return this.maxHeap.peek();
   }
 }
}
```

```
let streamMedian = new StreamMedian();
streamMedian.addNumber(1);
console.log(streamMedian.findMedian());
streamMedian.addNumber(2);
console.log(streamMedian.findMedian());
streamMedian.addNumber(5);
console.log(streamMedian.findMedian());
streamMedian.addNumber(3);
console.log(streamMedian.findMedian());
streamMedian.addNumber(7);
console.log(streamMedian.findMedian());
streamMedian.addNumber(4);
console.log(streamMedian.findMedian());
```

Job Schedular

```
let leastInterval= (employee, X) =>{
let employee_counter = {};
for(let i = 0;i< employee.length; i++) {
 if(employee[i] in employee_counter)
   employee_counter[employee[i]]++;
 else
   employee_counter[employee[i]] = 1;
}
let comparator = (a,b) => {return a[1] > b[1]}
let pq = new Heap(comparator);
for(let key in employee_counter){
 pq.add([key,employee_counter[key]])
}
let result = [];
while(pq.size() > 0) {
 let add_back = [];
 for(let i= 0; i <= X; i++) {
   if(pq.size() > 0) {
     let root = pq.peek();
     result.push(root[0]);
     pq.poll();
     root[1]--;
     if(root[1] > 0) {
       add_back.push(root);
     }
   }
   else
     result.push("idle")
   if(pq.size() == 0 && add_back.length == 0) {
     break;
   }
 }
 for(let i in add_back){
     pq.add(add_back[i]);
 }
}
return result;
}
const emp =
["Rohit","Rohit","Rohit","Rohit","Rohit","Rohit","Ankit","Astha","Vishw
a","Arnab","Sanket","Riya"]
console.log(leastInterval(emp,2));
```

Rearrange String

```
let rearrage= (str) =>{
 let char_counter = {};
 for(let i = 0;i< str.length; i++) {
   if(str[i] in char_counter)
     char_counter[str[i]]++;
   else
     char_counter[str[i]] = 1;
 }
 let comparator = (a,b) => {return a[1] > b[1]}
 let pq = new Heap(comparator);
 for(let key in char_counter){
   pq.add([key,char_counter[key]])
 }
 let result = "";
 let prev = [0,0];
 while(pq.size() > 0) {
   let p = pq.peek();
   result += p[0];
   pq.poll();
   p[1]--;
   if(prev[1] !=0) pq.add(prev);
   prev = p;
 }
 return result.length == str.length ? result : "";
}


console.log(rearrage("aba"));
```

Practice Problems

1) You have an array. And a frame of length x. You have to slide that frame on the array and find the median in each frame.
   I/P: nums = [1,3,-1,-3,5,3,6,7], x = 3
   O/P: [1.00000,-1.00000,-1.00000,3.00000,5.00000,6.00000]
   Explanation:

[1	3	-1]	-3	5	3	6	7	1
1	[3	-1	-3]	5	3	6	7	-1
1	3	[-1	-3	5]	3	6	7	-1
1	3	-1	[-3	5	3]	6	7	3
1	3	-1	-3	[5	3	6]	7	5
1	3	-1	-3	5	[3	6	7]	6

1) You have give a matrix of n*n and it is sorted row wise and column wise. You have to find the kth smallest element in the matrix.
   Example:
   I/P:[[1,5,9],[10,11,13],[12,13,15]], k = 8
   O/P: 13
   Explanation:[1,5,9,10,11,12,13,13,15], and the 8th smallest number is 13

Thank you