Introduction to Priority Queues using Binary Heaps**

07 October 2024 21:41

```
class Solution {
  public int extractMax()
   {
      GFG obj=new GFG();
       int xx= obj.H[0];
       obj.H[0]=obj.H[obj.s];
       obj.s--;
       obj.shiftDown(0);
       return xx;
    }
};
```

Min Heap and Max Heap Implementation

07 October 2024 21:42

```
class MinHeap {
   int[] harr;
   int capacity;
   int heap_size;
   MinHeap(int cap) {
      heap_size = 0;
      capacity = cap;
      harr = new int[cap];
   }
   int parent(int i) { return (i - 1) / 2; }
   int left(int i) { return (2 * i + 1); }
   int right(int i) { return (2 * i + 2); }

   //Function to extract minimum value in heap and then to store
   //next minimum value at first index.
   int extractMin()
   {
      if(heap_size==0)
      {
         return -1;
      }
      if(heap_size==1)
      {
         return harr[--heap_size];

      }
      int ans=harr[0];
      harr[0]=harr[heap_size-1];
      heap_size--;
      MinHeapify(0);

      return ans;

      // Your code here.
   }

   //Function to insert a value in Heap.
   void insertKey(int k)
   {
      if (heap_size == capacity)
      return; // or throw an exception
      heap_size++;
      int i = heap_size - 1;
      harr[i] = k;
      while (i != 0 && harr[parent(i)] > harr[i])
      {
         int temp = harr[i];
         harr[i] = harr[parent(i)];
         harr[parent(i)] = temp;
         i = parent(i);
      }
```

```
    }

    //Function to delete a key at ith index.
    void deleteKey(int i)
    {
        if(i<0||i>=heap_size)
        {
            return;
        }
        decreaseKey(i,Integer.MIN_VALUE);
        extractMin();
    }

    //Function to change value at ith index and store that value at first index.
    void decreaseKey(int i, int new_val)
    {
        harr[i] = new_val;
        while (i != 0 && harr[parent(i)] > harr[i]) {
            int temp = harr[i];
            harr[i] = harr[parent(i)];
            harr[parent(i)] = temp;
            i = parent(i);
        }
    }

    /* You may call below MinHeapify function in
      above codes. Please do not delete this code
      if you are not writing your own MinHeapify */
    void MinHeapify(int i) {
        int l = left(i);
        int r = right(i);
        int smallest = i;
        if (l < heap_size && harr[l] < harr[i]) smallest = l;
        if (r < heap_size && harr[r] < harr[smallest]) smallest = r;
        if (smallest != i) {
            int temp = harr[i];
            harr[i] = harr[smallest];
            harr[smallest] = temp;
            MinHeapify(smallest);
        }
    }
}
```

Check if an array represents a min-heap or not

07 October 2024 21:42

```
public boolean countSub(long arr[], long n)
  {
    //int n=arr.length;
    for(int i=0;i<n;i++)
    {
      int l=2*i+1;
      int r=2*i+2;

      if(l<n&&arr[i]<arr[l])
      {
        return false;
      }
      if(r<n&&arr[i]<arr[r])
      {
        return false;
      }
    }
    return true;
  }
```

Convert min Heap to max Heap*****

07 October 2024 21:42

Kth largest element in an array [use priority queue]

07 October 2024 21:42

```
class Solution {
    public int findKthLargest(int[] nums, int k)
    {
        PriorityQueue<Integer> pq = new PriorityQueue<>();//MIN HEAP
        int n=nums.length;
        for(int i=0;i<n;i++)
        {
            pq.add(nums[i]);
            if(pq.size()>k)
            {
                pq.remove();
            }
        }
        return pq.peek();

    }
}
```

Kth smallest element in an array [use priority queue]

07 October 2024 21:42

```
class Solution {
    public static int kthSmallest(int[] arr, int k)
    {
        PriorityQueue<Integer> pq = new PriorityQueue<>((a, b) -> b - a); //MAX HEAP
        int n=arr.length;
        for(int i=0;i<n;i++)
        {
            pq.add(arr[i]);
            if(pq.size()>k)
            {
                pq.remove();
            }
        }
        return pq.peek();
    }
}
```

Sort K sorted array

07 October 2024 21:42

```
class Solution
{
   //Function to merge k sorted arrays.
   public static ArrayList<Integer> mergeKArrays(int[][] arr,int K)
   {
      ArrayList<Integer>ans=new ArrayList<>();
      PriorityQueue<Integer> pq = new PriorityQueue<>();//MIN HEAP

      int n=arr.length;
      int m=arr[0].length;
      for(int i=0;i<n;i++)
      {
         for(int j=0;j<m;j++)
         {
            pq.add(arr[i][j]);
            // if(pq.size()>=K)
            // {
            //    int a=pq.peek();
            //    ans.add(a);
            //    pq.remove();
            // }
         }
      }
      while(pq.size()>0)
      {
         ans.add(pq.peek());
         pq.remove();
      }

      return ans;
   }
}
```

Merge M sorted Lists****

07 October 2024 21:42

```
class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        PriorityQueue<ListNode> pq = new PriorityQueue<ListNode>(new Comparator<ListNode>(){
                            public int compare(ListNode head1,ListNode head2){
                                return head1.val-head2.val;
                            }
                        });
        for(ListNode list:lists){
            if(list!=null)
            pq.add(list);
        }
        ListNode result=new ListNode(-1);
        ListNode temp1=result;
        while(!pq.isEmpty()){
            ListNode temp=pq.poll();
            temp1.next=new ListNode(temp.val);
            temp1=temp1.next;
            if(temp.next!=null){
                pq.add(temp.next);
            }
        }
        return result.next;
    }

}
```

From <https://leetcode.com/problems/merge-k-sorted-lists/solutions/5629319/best-solution-in-java-and-beats-100-users-in-java/>

Replace each array element by its corresponding rank

07 October 2024 21:43

Task Scheduler

07 October 2024 21:43

Hands of Straights

07 October 2024 21:43

Design twitter

07 October 2024 21:43

Connect `n` ropes with minimal cost

07 October 2024 21:43

Kth largest element in a stream of running integers

07 October 2024 21:43

Maximum Sum Combination

07 October 2024 21:43

Find Median from Data Stream

07 October 2024 21:44

K most frequent elements

07 October 2024 21:44