**Experiment 1.3**

**Student Name: Yash Gupta UID: 20BCS5009**

**Branch: BE-CSE Section/Group:20BCS\_DM-716 B**

**Semester: 6 Date of Performance: 27/02/23**

**Subject Name: CC LAB Subject Code: 20CSP\_351**

1. **Aim:**

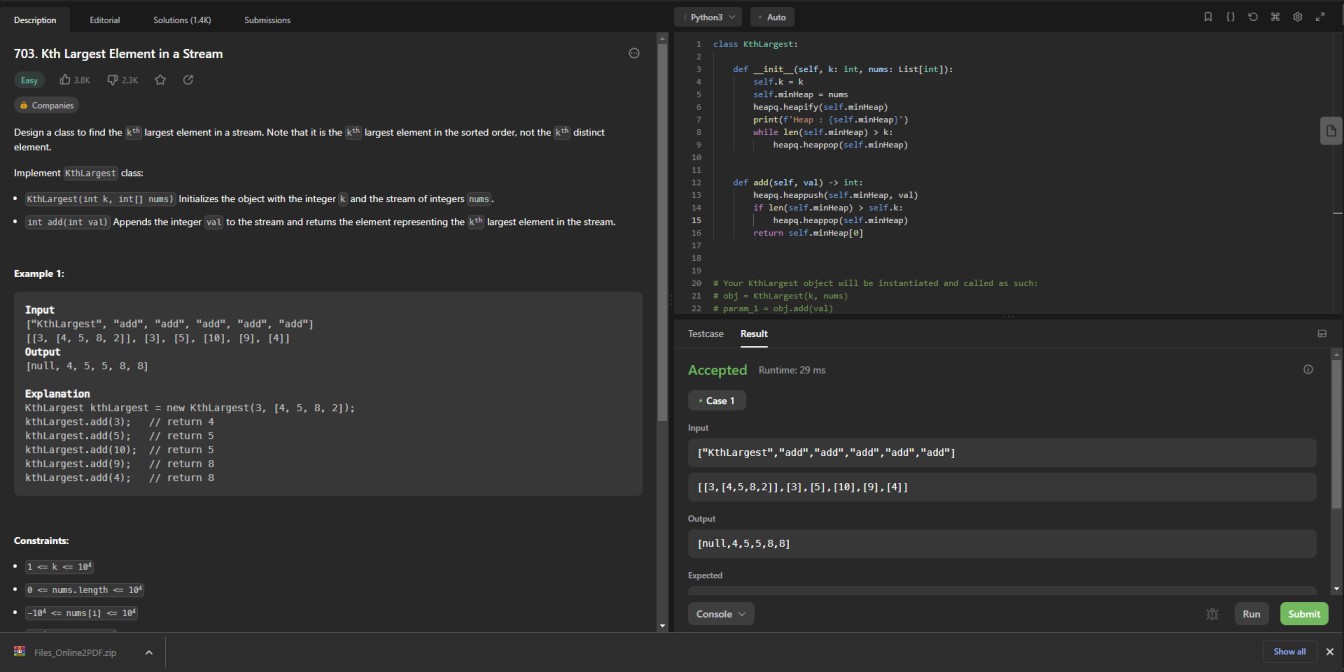
To implement the concept of heap modelling.

1. **Objective:** 
   * The objective is to build problem solving capability and to learn the basic concepts of data structures.
   * Understand the problem and find out better approach to solve particular problem

•

1. **LeetCode code and output:**

• **Kth largest element**



class KthLargest:

def \_\_init\_\_(self, k: int, nums: List[int]):

self.k = k

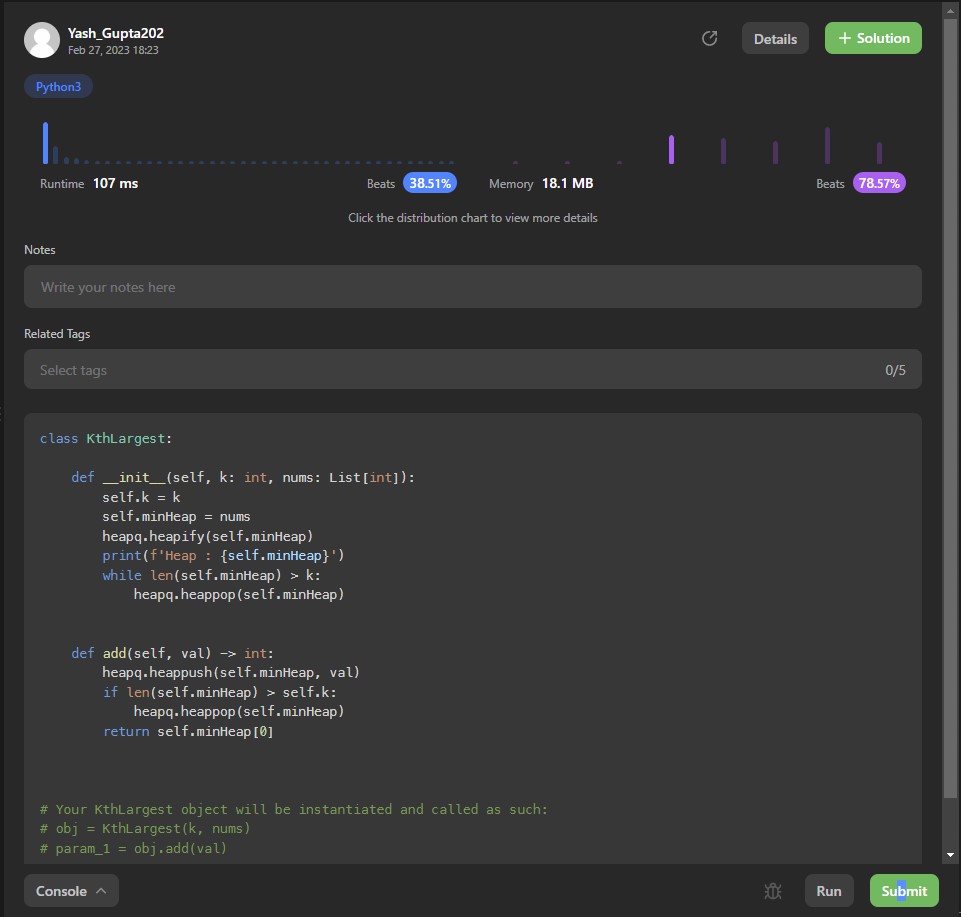
self.minHeap = nums heapq.heapify(self.minHeap) print(f'Heap : {self.minHeap}') while len(self.minHeap) > k:

heapq.heappop(self.minHeap)

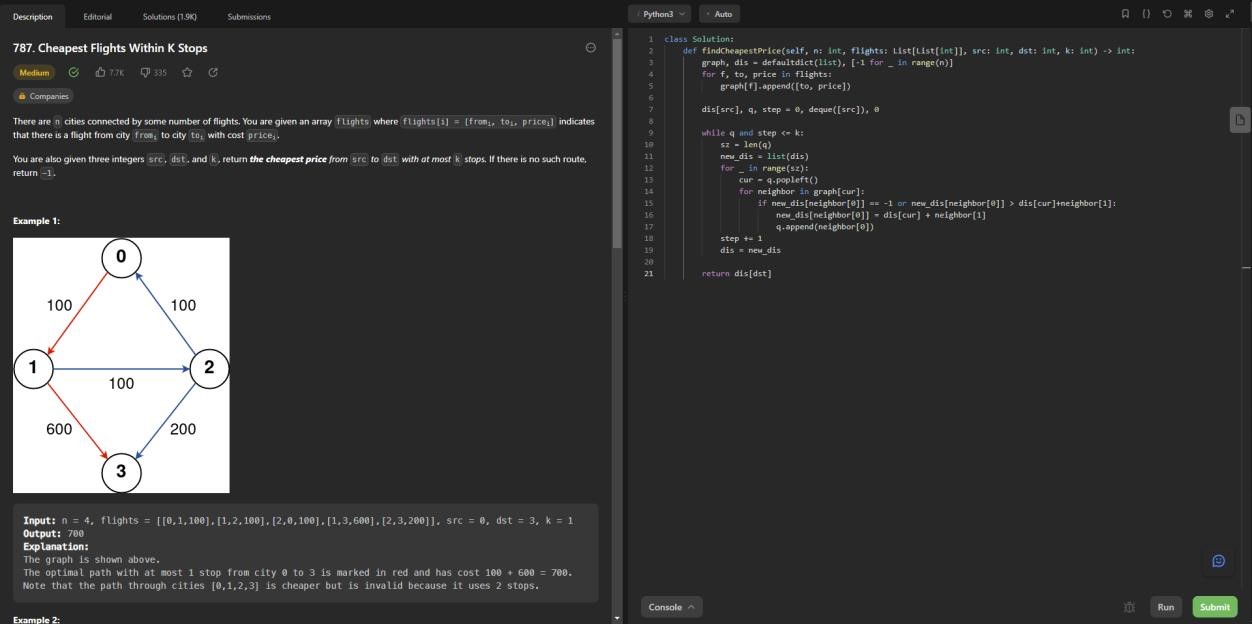
def add(self, val) -> int:

heapq.heappush(self.minHeap, val)

if len(self.minHeap) > self.k: heapq.heappop(self.minHeap) return self.minHeap[0]



• **Cheapest flight within k stops**



class Solution: def findCheapestPrice(self, n: int, flights: List[List[int]], src: int, dst: int, k: int) -> int:

graph, dis = defaultdict(list), [-1 for \_ in range(n)] for f, to, price in flights:

graph[f].append([to, price])

dis[src], q, step = 0, deque([src]), 0 while q and step <= k: sz = len(q) new\_dis = list(dis) for \_ in range(sz): cur = q.popleft() for neighbor in graph[cur]:

if new\_dis[neighbor[0]] == -1 or new\_dis[neighbor[0]] > dis[cur]+neighbor[1]:

new\_dis[neighbor[0]] = dis[cur] + neighbor[1]

q.append(neighbor[0]) step += 1 dis = new\_dis return dis[dst]

