Heaps

Assignment Solutions

PW SKILLS

Q1. Given an integer array, find the kth largest element using priority queue.

Input 1: arr[] = {1,2,3,5,2,6,9} k = 3
Output 1: 5

Input 2: arr[] = {1,7,8,5,2,6,9} k = 6
Output 2: 2

Solution :

Code : ASS_Code1.java

Output :

```
k'th largest array element is 7
```

Approach :

- **Method 1 : Using maxheap,**
  - Insert all elements into a maxheap.
  - Remove first (k-1) elements.
  - The element at the top of the heap is the kth largest element.

Time complexity : O(nlogn) where n is the number of elements in the array.
Space complexity : O(n) because n elements are inserted in the heap.

- **Method 2 : Using minheap**
  - Insert first k elements into a minheap.
  - for remaining (n-k) elements if the ith element is greater than the top element of the queue, remove it from the top and insert this current element into the queue.
  - Since the element at the top will never be considered in k largest elements until another element greater than this peek element is outside the queue.
  - If the ith element is lesser then do nothing and move to the next iteration.
  - repeat this process until all the elements are traversed.
  - This way the kth largest element will be at the top of the queue in the end.

Time complexity : O(nlogk)
Space complexity : O(k) because in the worst case queue will be only holding k elements.

Code : ASS_Code2.java

Output :

```
k'th largest array element is 7
```

Q2. Given n ropes of different lengths, connect them into a single rope with minimum cost. Assume that the cost to connect two ropes is the same as the sum of their lengths.

For example,

Input: [5, 4, 2, 8]

Output: The minimum cost is 36

[5, 4, 2, 8] –> First, connect ropes of lengths 4 and 2 that will cost 6.
[5, 6, 8] –> Next, connect ropes of lengths 5 and 6 that will cost 11.
[11, 8] –> Finally, connect the remaining two ropes that will cost 19.

Therefore, the total cost for connecting all ropes is 6 + 11 + 19 = 36.

Solution :

Code : ASS_Code3.java

Output :

```
The minimum cost is 36
```

Approach :

- The idea is to connect the two lowest-cost ropes first. The resultant rope has a cost equal to the sum of the connected ropes.
- Repeat the process (with resultant rope included) until we are left with a single rope.
- At each iteration of the loop, we will be left with one less rope and the optimal cost is added to the total cost.
- The final cost for connecting n ropes will be minimal among all possible combinations.
- A priority queue implemented using min-heap is best suited for this problem.

Q3. Given an array of string 'words' and an integer k, return the k most frequent strings. Return the answer sorted by the frequency from highest to lowest. Sort the words with the same frequency by their lexicographical order.

Example 1:

Input: words = ["i","love","leetcode","i","love","coding"], k = 2
Output: ["i","love"]

Explanation: "i" and "love" are the two most frequent words.
Note that "i" comes before "love" due to a lower alphabetical order.
Example 2:

Input: words = ["the","day","is","sunny","the","the","the","sunny","is","is"], k = 4
Output: ["the","is","sunny","day"]

Explanation: "the", "is", "sunny" and "day" are the four most frequent words, with the number of occurrences being 4, 3, 2 and 1 respectively.

Solution :

Code : ASS_Code5.java

Output :

```
PW  PhysicsWallah  Code
```

Approach :

- The idea is to keep a count of each word in a HashMap and then insert in a Priority Queue.
- While inserting in pq, if the count of two words is the same then insert them based on string comparison of the keys.

Q4. You are given an array of integer stones where stones[i] is the weight of the ith stone. We are playing a game with the stones. On each turn, we choose the heaviest two stones and smash them together. Suppose the heaviest two stones have weights x and y with x <= y. The result of this smash is:
If x == y, both stones are destroyed, and
If x != y, the stone of weight x is destroyed, and the stone of weight y has new weight y - x.
At the end of the game, there is at most one stone left.

Return the weight of the last remaining stone. If there are no stones left, return 0.

Example 1:

Input: stones = [2,7,4,1,8,1]
Output: 1

Explanation:
We combine 7 and 8 to get 1 so the array converts to [2,4,1,1,1] then,
we combine 2 and 4 to get 2 so the array converts to [2,1,1,1] then,
we combine 2 and 1 to get 1 so the array converts to [1,1,1] then,
we combine 1 and 1 to get 0 so the array converts to [1] then that's the value of the last stone.
Example 2:

Input: stones = [1]
Output: 1

Solution :

Code : ASS_Code6.java

Output :

```
The desired output is : 1
```

Approach :

- Put all elements into a priority queue.
- Pop out the two biggest, push back the difference,until there are no more two elements left.