30 Days Leetcode challenge - Day 18

public class leetday18

{

class Solution1 {

public int minDays(int[] arr, int m, int k)

{

int low = getLow(arr);

int high = getHigh(arr);

int ans = -1;

if(arr.length < m\*k)

{

return -1;

}

while(low<=high)

{

int mid = (low+high)/2;

if(possible(arr,mid,m,k)==true)

{

ans = mid;

high = mid -1;

}

else

{

low = mid +1;

}

}

return ans;

}

public static int getLow(int arr[])

{

int low = Integer.MAX\_VALUE;

for(int i=0;i<arr.length;i++)

{

if(arr[i]<low)

{

low = arr[i];

}

}

return low;

}

public static int getHigh(int arr[])

{

int high = Integer.MIN\_VALUE;

for(int i=0;i<arr.length;i++)

{

if(arr[i]>high)

{

high = arr[i];

}

}

return high;

}

public static boolean possible(int arr[], int day , int m , int k)

{

int count = 0;

int noOfBouquets = 0;

for(int i=0;i<arr.length;i++)

{

if(arr[i]<=day) // 7<=7

{

count++;

}

else

{

noOfBouquets = noOfBouquets + (count/k);

count = 0;

}

}

noOfBouquets = noOfBouquets + (count/k);

if(noOfBouquets >=m)

{

return true;

}

return false;

}

}

class Solution2 {

public int smallestDivisor(int[] arr, int threshold)

{

int low = 1;

int high = getHigh(arr);

int ans = 0;

while(low<=high)

{

int mid = (low+high)/2;

if(acceptable(arr,mid,threshold)==true)

{

ans = mid;

high = mid-1;

}

else{

low = mid +1;

}

}

return ans;

}

public static boolean acceptable(int arr[], int mid , int threshold)

{

int sum = 0;

for(int i=0;i<arr.length;i++)

{

sum = sum + (int) Math.ceil((double)arr[i]/mid);

}

if(sum>threshold)

{

return false;

}

return true;

}

public static int getHigh(int arr[])

{

int high = Integer.MIN\_VALUE;

for(int i=0;i<arr.length;i++)

{

if(arr[i]>high)

{

high = arr[i];

}

}

return high;

}

}

}