Amazon | OA 2020 | Disk Space Analysis

LEETCODE : <https://leetcode.com/discuss/interview-question/808348/>

Amazon is performing an analysis on the computers at one of its offices.

The computers are spaced along a single row. The analysis is performed in the following way:  
Choose a contiguous segment of a certain number of computers, starting from the beginning of the row.

Analyze the available hard disk space on each of the computers. Determine the minimum available disk space within this segment. After performing these steps for the first segment, it is then repeated for the next segment, continuing this procedure until the end of the row (i.e. if the segment size is 4, computers 1 to 4 would be analyzed, then 2 to 5, etc.)

***Given this analysis procedure, write an algorithm to find the maximum available disk space among all the minima that are found during the analysis.***

##### Input:

The input to the function/method consists of 3 arguments:  
***numComputer***, an integer representing the number of computers;  
***hardDiskSpace***, a list of integers representing the hard disk space of the computers;  
***segmentLength***, an integer representing the length of the contiguous segment of computers to  
be consider in each iterations.

##### Output:

Return an integer representing the maximum available disk space among all the minima that are found during the analysis.

##### Constraints:

1 ≤ numComputer ≤ 10^6  
1 ≤ segmentLength ≤ numComputer  
1 ≤ hardDiskSpace[i] ≤ 10^9  
0 ≤ i < numComputer

#### **Example:**

##### Input:

numComputer = 3  
hardDiskSpace = [8,2,4]  
segmentLength = 2

##### Output:

2

##### Explanation:

In this array of computers, the subarrays of size 2 are [8,2] and [2,4].  
Thus, the initial analysis returns 2 and 2 because those are the minima for the segmenets.  
Finally the maximum of these values is 2.  
Therefore, the answer is 2.

Ans :

Time Complexity : O(n) & space complexity : O(segmentLength) as we will be remove if its is greater than segment Length

<https://leetcode.com/playground/MShqbPZu>

public class Diskspace {  
public int analyse(int numComputer, int[] hardDiskSpace, int segmentLength) {  
Deque deq = new ArrayDeque<>();  
int maxMin = Integer.MIN\_VALUE;  
for(int i = 0; i < numComputer; i++) {  
while(!deq.isEmpty() && hardDiskSpace[deq.peekLast()] > hardDiskSpace[i]) {  
deq.pollLast();  
}  
if(!deq.isEmpty() && deq.peekFirst() <= (i - segmentLength)) {  
deq.pollFirst();  
}  
deq.offerLast(i);  
if(i >= (segmentLength - 1)) {  
maxMin = Math.max(maxMin, hardDiskSpace[deq.peekFirst()]);  
}  
}  
return maxMin;  
}  
public static void main(String[] args) {  
Diskspace space = new Diskspace();  
System.out.println(space.analyse(3, new int[] {8, 2, 4}, 2));  
System.out.println(space.analyse(6, new int[] {8, 2,4,3,7,6}, 2));//6  
System.out.println(space.analyse(6, new int[] {8, 2,4,3,7,6}, 3));//3  
System.out.println(space.analyse(7, new int[] {2,4,3,7,8,6,5}, 4));//5  
System.out.println(space.analyse(13, new int[] {2,4,3,7,8,6,5,16,19,33,32,34,35}, 5));  
}  
}

Alternate : import java.util.\*;

public class MaxInMinimal {

int get(int computers, List<Integer> hardDisks, int length) {

int ans = -1;

Deque<Integer> q = new ArrayDeque<>();

for (int right=0; right<hardDisks.size(); right++) {

int current = hardDisks.get(right);

while (q.size()>0 && current<q.getLast()) {

q.removeLast();

}

q.addLast(current);

if (right >= length-1) {

ans = Math.max(ans, q.getFirst());

}

if (q.size() >= length) {

q.removeFirst();

}

}

return ans;

}

public static void main(String[] args) {

MaxInMinimal case1 = new MaxInMinimal();

System.out.println(case1.get(3, Arrays.asList(new Integer[]{8,2,4}), 2));

}

}