```
class Solution {
    public int shortestSubarray(int[] A, int K) {
        int N = A.length, res = N + 1;
        int[] B = new int[N + 1];
        for (int i = 0; i < N; i++) {
            B[i + 1] = B[i] + A[i];
        }
        Deque<Integer> d = new ArrayDeque<>();
        for (int i = 0; i < N + 1; i++) {
            while (d.size() > 0 && B[i] - B[d.getFirst()] >= K) {
                res = Math.min(res, i - d.pollFirst());
            }
            while (d.size() > 0 && B[i] <= B[d.getLast()]) {
                d.pollLast();
            }
            d.addLast(i);
        }
        return res <= N ? res : -1;
    }
}
```

Return the **length** of the shortest, non-empty, contiguous subarray of `A` with sum at least `K`.

If there is no non-empty subarray with sum at least `K`, return `-1`.

Example 1:

```
Input: A = [1], K = 1
Output: 1
```

**Example 2:**

```
Input: A = [1,2], K = 4
Output: -1
```

**Example 3:**

```
Input: A = [2,-1,2], K = 3
Output: 3
```