

```

class Solution {
public:
    int shortestSubarray(vector<int>& nums, int k) {
        int n = nums.size();
        int shortestSubarrayLength = INT_MAX;
        long long cumulativeSum = 0;

        priority_queue<pair<long long, int>, vector<pair<long long, int>>,
                        greater<>>
            prefixSumHeap;

        for (int i = 0; i < n; i++) {

            cumulativeSum += nums[i];

            if (cumulativeSum >= k) {
                shortestSubarrayLength = min(shortestSubarrayLength, i + 1);
            }

            while (!prefixSumHeap.empty() &&
                cumulativeSum - prefixSumHeap.top().first >= k) {

                shortestSubarrayLength =
                    min(shortestSubarrayLength, i -
prefixSumHeap.top().second);
                prefixSumHeap.pop();
            }

            prefixSumHeap.emplace(cumulativeSum, i);
        }

        return shortestSubarrayLength == INT_MAX ? -1 :
shortestSubarrayLength;
    }
};

```