#include <bits/stdc++.h>

#include <climits>

using namespace std;

struct node {

int data;

node\* next;

node(int val) {

this->data = val;

this->next = nullptr;

}

};

struct nodeForDequeue {

int data;

node\* next;

node\* prev;

nodeForDequeue(int val) {

this->data = val;

this->next = nullptr;

this->prev = nullptr;

}

};

class Queue {

private:

node\* head, \*tail;

int capacity;

public:

Queue(int capacity);

void push(int val);

void pop();

int front();

bool isEmpty();

};

Queue::Queue(int capacity) {

this->capacity = capacity;

this->head = nullptr;

this->tail = nullptr;

}

void Queue::push(int val) {

if(this->head == nullptr) {

this->head = new node(val);

this->tail = this->head;

} else {

node\* temp = new node(val);

this->tail->next = temp;

this->tail = temp;

}

}

void Queue::pop() {

if(this->head == nullptr) {

return;

}

node\* newHead = this->head->next;

delete head;

this->head = newHead;

}

int Queue::front() {

if(this->head == nullptr) {

return -1;

}

return this->head->data;

}

bool Queue::isEmpty() {

return this->head == nullptr;

}

int main() {

Queue\* q = new Queue(10);

q->push(7);

q->push(10);

q->push(11);

while(!q->isEmpty()) {

cout<<q->front()<<endl;

q->pop();

}

delete q;

//

/\* queue<int>q1;

q1.push(10);

cout<<q1.size()<<endl;

cout<<q1.empty()<<endl;

q1.front();

q1.pop();

deque<int>dq;

dq.push\_back(10);

dq.push\_front(12);

cout<< dq.front() <<endl;\*/

int arr[] = {1, -7, 8, 10, 12, 16, -19, 10, -7};

int n = 9;

int pre[9];

pre[0] = arr[0];

for(int i = 1; i < 9; i++) {

pre[i] = arr[i] + pre[i-1];

}

int k = 40;

int ans = 1e9;

deque<int>dq1;

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

while(!dq1.empty() and pre[dq1.back()] >= pre[i]) {

dq1.pop\_back();

}

dq1.push\_back(i);

while(!dq1.empty() and k <= pre[dq1.back()] - pre[dq1.front()]) {

ans = min(ans, dq1.back() - dq1.front());

dq1.pop\_front();

}

}

cout<<ans<<endl;

return 0;

}