#include <iostream>

#include<bits/stdc++.h>

using namespace std;

// Constructs the Segment tree

void buildTree(int\* tree , int\* a ,int index , int s ,int e)

{

// Base case

if(s>e)

return ;

// Base Case of Leaf Node

if(s==e)

{

tree[index]=a[s];

return;

}

// Recurseive Case

int mid=(s+e)/2;

buildTree(tree,a,2\*index,s,mid);

buildTree(tree,a,2\*index+1,mid+1,e);

int left = tree[2\*index];

int right = tree[2\*index+1];

tree[index]=min(left,right);

}

// Returns min value in range qe ,qs;

int query(int\* tree, int index , int s , int e, int qs,int qe)

{

//No Overlapping Case

if(qs>e||s>qe)

return INT\_MAX;

// Complete Overlap case

if(s>=qs&&e<=qe)

return tree[index];

// Partial Overlap case

int mid=(s+e)/2;

int left = query(tree,2\*index ,s ,mid,qs,qe);

int right = query(tree,2\*index+1 ,mid+1,e,qs,qe);

return min(left,right);

}

// Updates a single Node

void updateNode(int\* tree, int index , int s , int e,int i,int value)

{

//No Overlapping Case

if(i>e||s>i)

return ;

// Complete Overlap case

if(s==e)

tree[index]=value;

return ;

// Partial Overlap case

int mid=(s+e)/2;

updateNode(tree,2\*index ,s ,mid,i,value);

updateNode(tree,2\*index+1 ,mid+1,e,i,value);

tree[index]=min(tree[2\*index],tree[2\*index+1]);

return;

}

// Updates a range from qs to qe

void updateNode(int\* tree, int index , int s , int e,int qs,int qe,int inc)

{

//No Overlapping Case

if(qs>e||s>qe)

return ;

// Complete Overlap case

if(s>=qs&&e<=qe)

tree[index] += inc;

// Partial Overlap case

int mid=(s+e)/2;

updateNode(tree,2\*index ,s ,mid,qs,qe,inc);

updateNode(tree,2\*index+1 ,mid+1,e,qe,qs,inc);

tree[index]=min(tree[2\*index],tree[2\*index+1]);

return;

}

// It's testing time guys !

int main() {

int n,q,i,s,e,l,r,index;

cin>>n>>q;

int a[100];

int\* tree = new int[4\*n+1];

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

cin>>a[i];

index=1;

s=0;

e = n-1;

buildTree(tree,a,index,s,e);

while(q--)

{

cin>>l>>r;

cout<<query(tree,index,s,e,l,r)<<endl;

}

return 0;

}