Notebook um pouco dijkstrado

Notebook da equipe Dijkstrados

GRUPO SBC
SBC 1943 podcast
SBC
SBC
SBC SERVIÇO BRASILEIRO DE CERTIFICAÇÕES
SBC CREATIVE slogan here
SBC UTILITÁRIOS
SBC
sbc
SBC
MEUÔNIBUS SBC
SBC CLUBE
SBC LOREM IPSUM LOREM IPSUM
shutterstock.com · 1806517336
YHWH SBC MILITAR
SÃO BERNARDO DO CAMPO SBC CERVEJARIA
SBC AWARDS 2022
SBC Vans Auto Peças
Especializado em Vans e Pickups
sbclaw advogados
CENTRO DE FORMAÇÃO E INTEGRAÇÃO SOCIAL CAMP S.B.C.
JUDÔ CAMP SBC
SÃO BERNARDO DO CAMPO - SP

Referências

Pilhas e filas

```
from collections import deque
deque_vazio = deque()
deque_lista = deque([5,2,3,4])

#pegar da esquerda e da direita
deque_primeiro = fila.popleft()
deque_ultimo = fila.pop()

#adicionar à esquerda e a direita
deque_prioridade.appendleft(valor)
deque_fila.append(valor)
```

Conjuntos

```
set_vazio = set()
set_lista = set([5,2,3,2,2]) #resulta em [5,2,3]
conjunto.add(valor)
conjunto.remove(valor)
conjunto.discard(valor) # remove() sem dar erro se não tiver

s1,s2 = {1,2},{2,3}
união = s1|s2 #resulta em {1,2,3}
interseção = s1&s2 #resulta em {2}
diferença = s1-s2 #resulta em {1}
diferença = s1^s2 #resulta em {1,3}
```

Fila prioritária

```
from queue import PriorityQueue
fila = PriorityQueue()
fila.put(numero)
primeiro = fila.get()
```

Referências

Dicionários

```
mapa_vazio = []
dicionario[nova] = valor_novo
valor_adicionado = dicionario[mesma]
dicionario.pop(valor)

#formas de iterar
chaves = dict.keys()
fechaduras = dict.values()
for chave,fechadura in dict.items():
     continue
```

Entrada e Saída

```
#entradas
import sys
sys.setrecursionlimit(10**6)
a,b = map(int, input().split())
lista_entrada = list(map(int, input().split()))
#prints
print(a,b,c) == print(f”{a} {b}”)
print(*lista, sep = ‘ ‘) #printa todos os valores da lista
print(f”{valor_float:.3f}”)
print(f”{zeros_na_frente:0<2}”)
```

Hashing

```
#apagar os tabs antes de testar
#windows
> certUtil -hashfile questao.txt sha256
#linux
$ sha256 questao.txt
a b
c
```

sha256sum: 721a7916ccfa56849995f47004497d7d8cefa18b0033b017026036cbe016e171

Algoritmos

Soma Acumulada

```
sacumulada=[l[0]]
for i in range(1,len(l)):
    sacumulada.append(sacumulada[i-1]+l[i])
```

sha256sum: ee7fb5f6f7469c296082e49336fc5cf4f71d11843349aa790d468f3a4184aab1

Busca binária

```
def bsearch(i,l):
    p1=0
    p2=len(l)
    while p1!=p2:
        m=(p1+p2)//2
        if i>l[m]:p1=m+1
        else:p2=m
    return p1
```

sha256sum: 4f1bda4e7fe897662529c2b9a2826edb3bc83742f6d644cfd959e49f079d685b

Máximo Divisor Comum

```
def gcd(a,b):
    while a%b !=0:
    aux = b
    b = a%b
    a = aux
    return b
```

sha256sum: 88c1d7d2877ca1b31b7f18c26e75580c476e3c6648e412125f2977916efb5e9c

Mínimo Multiplicador Comum

```
def lcm(a,b):
    return (a/gcd(a,b))*b
```

sha256sum: 1fae73dcba5be425a044356dc3e5a5984570c8752f2359dd6de7042723a39997

Algoritmos

Divisores

```
import math
def divisores(n):
    divisores=[]
    i=1
    while i<=math.sqrt(n):
    if (n % i == 0):
        if(n/i==i):
            divisores.append(i)
        else:
            divisores.append(i)
            divisores.append(int(n/i))
    i=i+1
```

sha256sum: 2c6830bbcd7af64eec738fb86f55fb6e6ad3cda5414e5c0cc3c50bee3d0ed48a

Crivo De aristóteles

```
def crivo(n):
    prime = [True for i in range(n+1)]
    p = 2
    while (p**2<=n):
    if (prime[p] == True):
        for i in range(p**2, n+1, p):
            prime[i] = False
    p += 1
    return prime
```

sha256sum: 0de6f4ae8050662b07e5d8f5c74e16488da35ce9148b11dcccf1198542d8282f

Operações Modulares Simples

```
((a%m) + (b%m))%m = (a+b)%m
((a%m) - (b%m))%m = (a-b)%m
((a%m) * (b%m))%m = (a*b)%m
```

Algoritmos

Inverso multiplicativo

conceito: $b^{-1} = (b^{m-2})\%m$ #quando m é primo

$(a/b)\%m = (a * b^{-1})\%m$

```
#retorna o inverso multiplicativo
def pow_mod(x,n = m-2,m):
     y=1
     while n>0:
     if n%2:
          y=(y*x)%m
     n=n//2
     x=(x * x)%m
     return y
```

sha256sum: e16cb32d854cb62c9fcbcdb2652a254853f14e8971441594631607baf81fec7c

Exponenciação Binária

```
def power(x,y,p):
    res=1
    while (y>0):
        if ((y & 1)!=0):
            res=res*x
        y = y>>1
        x = x**2
    return res%p
```

sha256sum: 8918bdf57b5c79aa718ed6b36cb3c81225f67c89506eeb6ed21aea55961b5b7e

Algoritmos

Depth First Search

```
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited

graph = {'0': set(['1', '2']),
         '1': set(['0', '3', '4']),
         '2': set(['0']),
         '3': set(['1']),
         '4': set(['2', '3'])}
dfs(graph, '0')
```

sha256sum: 5c6cfdafdf4d6fc9cf3f8d26d5571765f3093bd0945df10478100d28edd2b960

Bredth First Search

```
import collections
def bfs(graph, root):
    visited, queue = set(), collections.deque([root])
    visited.add(root)
    while queue:
        vertex = queue.popleft()
        for neighbour in graph[vertex]:
            if neighbour not in visited:
                visited.add(neighbour)
                queue.append(neighbour)
if __name__ == '__main__':
    graph = {0: [1, 2], 1: [2], 2: [3], 3: [1, 2]}
    bfs(graph, 0)
```

sha256sum: 55811e845a83edb57d2ef93e7d272d57069e27e5b7142d7c02c90d4af835a86d

Algoritmos

dijkstra

```
from queue import PriorityQueue
class Graph:
    def __init__(self, num_of_vertices):
        self.v = num_of_vertices
        self.edges = [[-1 for i in range(num_of_vertices)]
for j in range(num_of_vertices)]
        self.visited = []
     def add_edge(self, u, v, weight):
        self.edges[u][v] = weight
        self.edges[v][u] = weight
def dijkstra(graph, start_vertex):
    D = {v:float('inf') for v in range(graph.v)}
    D[start_vertex] = 0
    pq = PriorityQueue()
    pq.put((0, start_vertex))
    while not pq.empty():
        (dist, current_vertex) = pq.get()
        graph.visited.append(current_vertex)
        for neighbor in range(graph.v):
            if graph.edges[current_vertex][neighbor] != -1:
                distance =
graph.edges[current_vertex][neighbor]
                if neighbor not in graph.visited:
                    old_cost = D[neighbor]
                    new_cost = D[current_vertex] + distance
                    if new_cost < old_cost:
                        pq.put((new_cost, neighbor))
                        D[neighbor] = new_cost
    return D
```

sha256sum: 5dd7d05e25c2883a64a9395d7f4f3a72962d706442497030faccd6f88164eb30

Algoritmos

dijkstra Teste

```
g = Graph(9)
g.add_edge(0, 1, 4)
g.add_edge(0, 6, 7)
g.add_edge(1, 6, 11)
g.add_edge(1, 7, 20)
g.add_edge(1, 2, 9)
g.add_edge(2, 3, 6)
g.add_edge(2, 4, 2)
g.add_edge(3, 4, 10)
g.add_edge(3, 5, 5)
g.add_edge(4, 5, 15)
g.add_edge(4, 7, 1)
g.add_edge(4, 8, 5)
g.add_edge(5, 8, 12)
g.add_edge(6, 7, 1)
g.add_edge(7, 8, 3)

D = dijkstra(g, 0)
print(D)
```

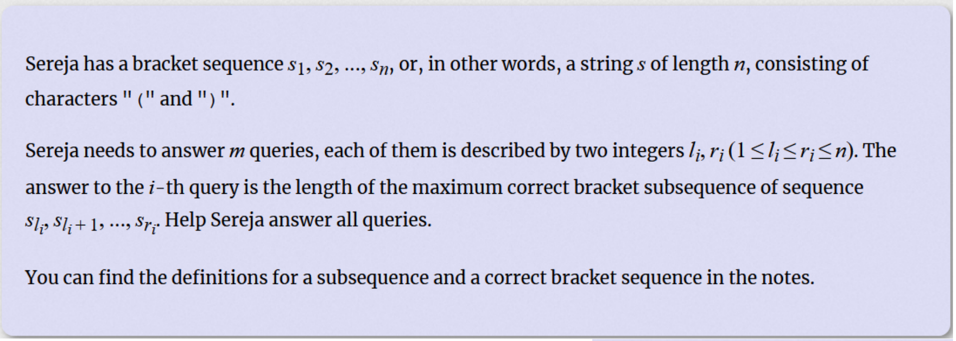
Questões

Dp

```
import sys
sys.setrecursionlimit(10**6)
n = int(input())
h = list(map(int, input().split()))

dp = [-1 for i in range(n)]
dp[0] = 0
dp[1] = abs(h[1]-h[0])
def solve(pos):
    if dp[pos] == -1:
    dp[pos] = min(
    solve(pos-1)+abs(h[pos]-h[pos-1]),
    solve(pos-2)+abs(h[pos]-h[pos-2])
    )
    return dp[pos]
print(solve(n-1))
```

Questões

Segtree - Sereja and Brackets

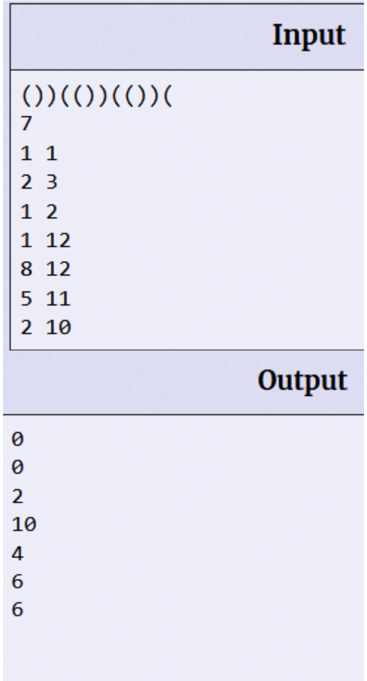


Sereja has a bracket sequence $s_1, s_2, ..., s_n$, or, in other words, a string s of length n, consisting of characters "(" and ")".

Sereja needs to answer m queries, each of them is described by two integers l_i, r_i $(1 \leq l_i \leq r_i \leq n)$. The answer to the i-th query is the length of the maximum correct bracket subsequence of sequence $s_{l_i}, s_{l_i+1}, ..., s_{r_i}$. Help Sereja answer all queries.

You can find the definitions for a subsequence and a correct bracket sequence in the notes.

```
#include <bits/stdc++.h>
using namespace std;
#define int long long

string entrada;
struct node {
        int abertos;
        int fechados;
        int loops;
};
node seg[4*1000000];

node merge(node a, node b){
        node c;
        int nloop = a.abertos-max((int) 0,a.abertos-b.fechados);
        c.abertos = a.abertos + b.abertos - nloop;
        c.fechados = a.fechados + b.fechados - nloop;
        c.loops = a.loops + b.loops + nloop;
        return c;
}

void build(int no, int l, int r){
        if (l == r) {
        if (entrada[l] == '(') seg[no] = {1,0,0};
        else if ( entrada[l] == ')') seg[no] = {0,1,0};
        else seg[no] = {0,0,0};
        }
        else{
        int mid = (l+r)/2;
        build(2*no,l,mid);
        build(2*no+1,mid+1,r);
        seg[no] = merge(seg[2*no],seg[2*no+1]);
        }
}

void update(int no, int l, int r, int i, int v){}

node query(int no, int l, int r, int lq, int rq){
        if (l > rq || r < lq) {return  {0,0,0};}
        if (l >= lq && r <= rq) {return seg[no];}
        int mid = (l+r)/2;
        node p1 = query(2*no,l,mid,lq,rq);
        node p2 = query(2*no+1,mid+1,r,lq,rq);
        return merge(p1,p2);
        }

signed main(){

        ios_base::sync_with_stdio(0);
        cin.tie(0);
        cout.tie(0);

        int n,lq,rq;
        cin >> entrada;
        build(1,0,entrada.size()-1);
        cin >> n;
        for (int i = 0; i < n; i++ ){
        cin >> lq >> rq;
        cout << 2*query(1,0,entrada.size()-1,lq-1,rq-1).loops << endl;
        }

        return 0;

}
```



Input

```
())(())(())(
7
1 1
2 3
1 2
1 12
8 12
5 11
2 10
```

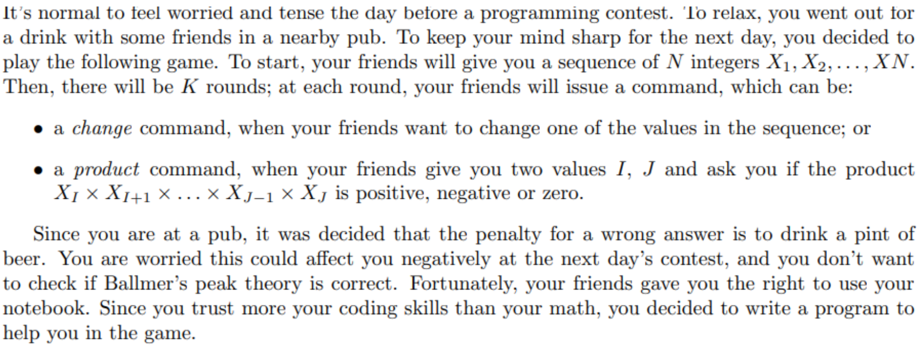
Output

```
0
0
2
10
4
6
6
```

Questões

segtree - Interval Product

```
#include <bits/stdc++.h>
using namespace std;
#define int long long

int entrada[10000000];
int tree[4 * 10000000];

void build(int no, int l, int r){
        if (l==r){tree[no] = entrada[l];}
        else {
        int mid = (l + r)/2;
        build(2*no,l,mid);
        build(2*no+1,mid+1,r);
        tree[no] = tree[2*no] * tree[2*no+1];
        }
}

void update(int no, int l, int r, int i, int val){
        if (l==r){tree[no] = val;}
        else {
        int mid = (l + r)/2;
        if (i <= mid) update(2*no,l,mid,i,val);
        else update(2*no+1,mid+1,r,i,val);
        tree[no] = tree[2*no] * tree[2*no+1];
        }
}

int query(int no, int l, int r, int i, int j){
        if (r < i || l > j) return 1;
        if (l >= i && r <= j) return tree[no];
        int mid = (l + r)/2;
        int p1 = query(2*no,l,mid,i,j);
        int p2 = query(2*no+1,mid+1,r,i,j);
        return p1 * p2;
}

signed main(){
        int n,m;
        while (cin >> n >> m){
        for (int i = 0; i < n;i++){
        cin >> entrada[i];
        if (entrada[i] > 0 || entrada[i] < 0)entrada[i] = entrada[i]/abs(entrada[i]);
        // cout << entrada[i];
        }
        build(1,0,n-1);
        for (int i = 0; i < m;i++){
        char q;
        cin >> q;
        int n1,n2;
        cin >> n1 >> n2;
        if (q == 'C') {
                if (n2 > 0)n2=1;
                if (n2 < 0)n2=-1;
                update(1,0,n-1,n1-1,n2);
        }
        if (q == 'P') {
                int out = query(1,0,n-1,n1-1,n2-1);
                if (out >= 1) cout << "+";
                else if (out <= -1) cout << "-";
                else cout << "0";
        }
        }
        cout << endl;
        }
        return 0;
}
```



It's normal to feel worried and tense the day before a programming contest. To relax, you went out for a drink with some friends in a nearby pub. To keep your mind sharp for the next day, you decided to play the following game. To start, your friends will give you a sequence of N integers $X_1, X_2, \ldots, XN$. Then, there will be K rounds; at each round, your friends will issue a command, which can be:

- a *change* command, when your friends want to change one of the values in the sequence; or
- a *product* command, when your friends give you two values I, J and ask you if the product $X_I \times X_{I+1} \times \ldots \times X_{J-1} \times X_J$ is positive, negative or zero.

Since you are at a pub, it was decided that the penalty for a wrong answer is to drink a pint of beer. You are worried this could affect you negatively at the next day's contest, and you don't want to check if Ballmer's peak theory is correct. Fortunately, your friends gave you the right to use your notebook. Since you trust more your coding skills than your math, you decided to write a program to help you in the game.

Sample Input

```
4 6
-2 6 0 -1
C 1 10
P 1 4
C 3 7
P 2 2
C 4 -5
P 1 4
5 9
1 5 -2 4 3
P 1 2
P 1 5
C 4 -5
P 1 5
P 4 5
C 3 0
P 1 5
C 4 -5
C 4 -5
```

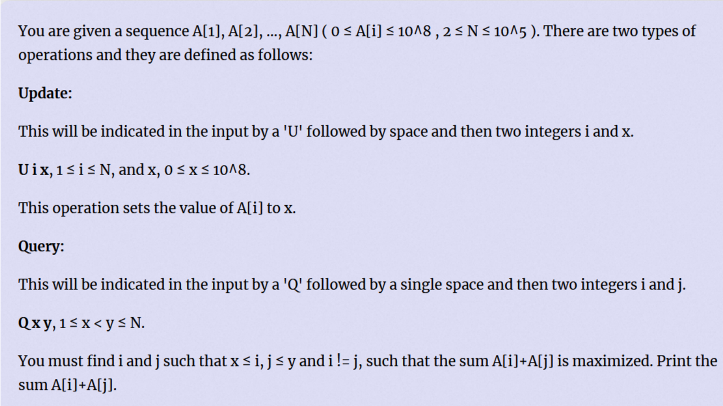
Sample Output

```
0+-
+-+-0
```

Questões

segtree - maximum sum

```
#include <bits/stdc++.h>
using namespace std;

int entrada[100000];
struct node { int big; int secbig; };
node seg[4 * 100000];

node merge(node a, node b){
        node c;
        c.big = max(a.big, b.big);
        c.secbig = min(max(a.big, b.secbig), max(a.secbig, b.big));
        return c;
}

void build(int no, int l, int r){
        if (l==r){seg[no] = node {entrada[l],0};}
        else {
        int mid = (l + r)/2;
        build(2*no,l,mid);
        build(2*no+1,mid+1,r);
        seg[no] = merge(seg[2*no], seg[2*no+1]);
        }
}

void update(int no, int l, int r, int i, int val){
        if (l==r){seg[no] = node {val,0};}
        else {
        int mid = (l + r)/2;
        if (i <= mid) update(2*no,l,mid,i,val);
        else update(2*no+1,mid+1,r,i,val);
        seg[no] = merge(seg[2*no], seg[2*no+1]);
        }
}

node query(int no, int l, int r, int i, int j){
        if (r < i || l > j) return node {0,0};
        if (l >= i && r <= j) return seg[no];
        int mid = (l + r)/2;
        node p1 = query(2*no,l,mid,i,j);
        node p2 = query(2*no+1,mid+1,r,i,j);
        return merge(p1,p2);

}

int main(){
        int n;cin >> n;
        for (int i = 0; i < n; i++) cin >> entrada[i];
        build(1,0,n-1);
        int m;cin >> m;
        for (int i = 0; i < m; i++){
        char opr;cin >> opr;
        if (opr == 'Q') {
        int l,r;
        cin >> l >> r;
        //for (int j = 0; j < 4*n; j++) cout << seg[j].big << " " << seg[j].secbig << endl;
        node ans = query(1,0,n-1,l-1,r-1);
        cout << ans.big + ans.secbig << endl;
        }
        else {
        int pos,val;
        cin >> pos >> val;
        update(1,0,n-1,pos-1,val);
        }
        }

        return 0;}
```



You are given a sequence A[1], A[2], ..., A[N] (0 ≤ A[i] ≤ 10^8 , 2 ≤ N ≤ 10^5). There are two types of operations and they are defined as follows:

Update:

This will be indicated in the input by a 'U' followed by space and then two integers i and x.

U i x, 1 ≤ i ≤ N, and x, 0 ≤ x ≤ 10^8.

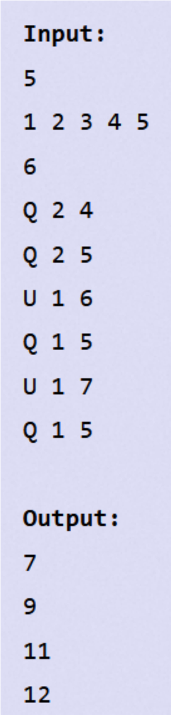
This operation sets the value of A[i] to x.

Query:

This will be indicated in the input by a 'Q' followed by a single space and then two integers i and j.

Q x y, 1 ≤ x < y ≤ N.

You must find i and j such that x ≤ i, j ≤ y and i != j, such that the sum A[i]+A[j] is maximized. Print the sum A[i]+A[j].

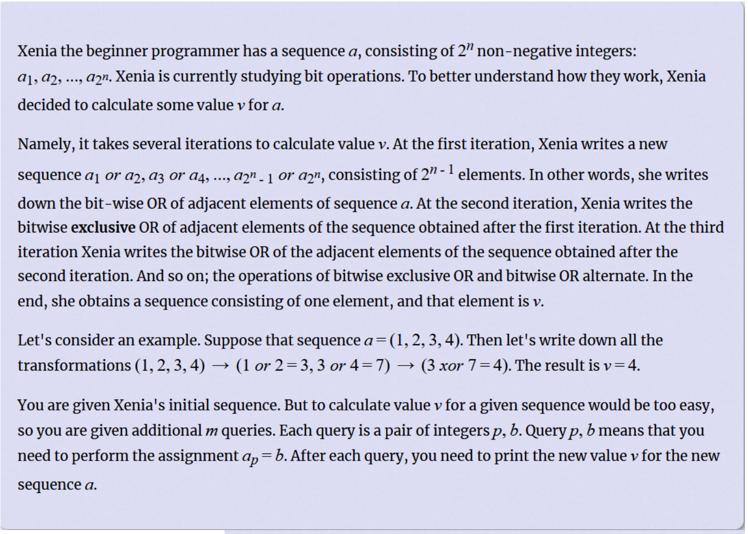


```
Input:
5
1 2 3 4 5
6
Q 2 4
Q 2 5
U 1 6
Q 1 5
U 1 7
Q 1 5

Output:
7
9
11
12
```

Questões

segtree - xenia and bit operations

```
#include <bits/stdc++.h>
using namespace std;

int entrada[1000000];
struct node{int valor;int camada;};
node tree[4 * 1000000];

node merge(node a, node b){
        node c;
        c.camada = a.camada + 1;
        if (a.camada%2 == 0){c.valor = a.valor|b.valor;}
        else{c.valor = a.valor^b.valor;}
        return c;
}

void build(int no, int l, int r){
        if (l==r){tree[no] = {entrada[l],0};}
        else {
        int mid = (l + r)/2;
        build(2*no,l,mid);
        build(2*no+1,mid+1,r);
        tree[no] = merge(tree[2*no],tree[2*no+1]);
        }
}

void update(int no, int l, int r, int i, int val){
        if (l==r){
        tree[no].valor = val;
        }
        else {
        int mid = (l + r)/2;
        if (i <= mid) update(2*no,l,mid,i,val);
        else update(2*no+1,mid+1,r,i,val);
        tree[no] = merge(tree[2*no],tree[2*no+1]);
        }
}

node query(int no, int l, int r, int i, int j){
        if (r < i || l > j) return {0,0};
        if (l >= i && r <= j) return tree[no];
        int mid = (l + r)/2;
        node p1 = query(2*no,l,mid,i,j);
        node p2 = query(2*no+1,mid+1,r,i,j);
        return merge(p1,p2);
}

signed main(){
        int n,m;
        cin >> n >> m;
        int ln = pow(2,n);
        for (int i = 0; i < ln;i++){cin >> entrada[i];}
        build(1,0,ln-1);
        for (int i = 0; i < m;i++){
        int p, b;
        cin >> p >> b;
        update(1,0,ln-1,p-1,b);
        // for (int j = 0; j < 4*ln; j++) cout << tree[j].valor << " " << tree[j].camada << endl;
        cout << query(1,0,ln-1,0,ln-1).valor << endl;

        }
        return 0;

}
```

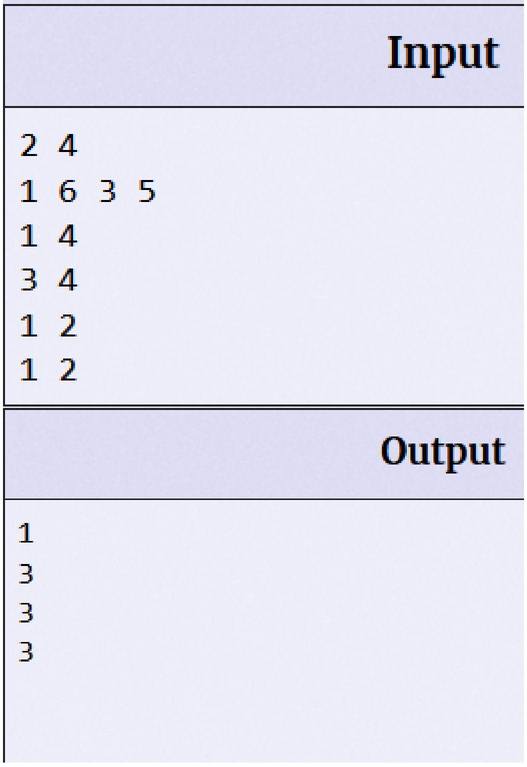


Xenia the beginner programmer has a sequence a, consisting of 2^n non-negative integers: $a_1, a_2, ..., a_{2^n}$. Xenia is currently studying bit operations. To better understand how they work, Xenia decided to calculate some value v for a.

Namely, it takes several iterations to calculate value v. At the first iteration, Xenia writes a new sequence a_1 *or* a_2, a_3 *or* $a_4, ..., a_{2^n - 1}$ *or* a_{2^n}, consisting of $2^{n - 1}$ elements. In other words, she writes down the bit-wise OR of adjacent elements of sequence a. At the second iteration, Xenia writes the bitwise **exclusive** OR of adjacent elements of the sequence obtained after the first iteration. At the third iteration Xenia writes the bitwise OR of the adjacent elements of the sequence obtained after the second iteration. And so on; the operations of bitwise exclusive OR and bitwise OR alternate. In the end, she obtains a sequence consisting of one element, and that element is v.

Let's consider an example. Suppose that sequence $a = (1, 2, 3, 4)$. Then let's write down all the transformations $(1, 2, 3, 4) \rightarrow (1 \text{ or } 2 = 3, 3 \text{ or } 4 = 7) \rightarrow (3 \text{ xor } 7 = 4)$. The result is $v = 4$.

You are given Xenia's initial sequence. But to calculate value v for a given sequence would be too easy, so you are given additional m queries. Each query is a pair of integers p, b. Query p, b means that you need to perform the assignment $a_p = b$. After each query, you need to print the new value v for the new sequence a.



Input

```
2 4
1 6 3 5
1 4
3 4
1 2
1 2
```

Output

```
1
3
3
3
```

Questões

Questões

Questões

Questões

Questões

Questões

Questões

Questões

Questões

Questões

Questões

Questões

Questões

Questões

Questões

Questões