Árvore de segmentos

Definição e Implementação: problemas resolvidos

Prof. Edson Alves - UnB/FGA

2020

Sumário

Codeforces Round #197 (Div. 2) – Problem D: Xenia and Bit Operations

Problema

Xenia the beginner programmer has a sequence a, consisting of 2^n non-negative integers: $a_1, a_2, \ldots, a_{2^n}$. 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 $a_4, \ldots, a_{2^n-1}$ 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.

Problema

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

Entrada e saída

Input

The first line contains two integers n and m ($1 \leq n \leq 17, 1 \leq m \leq 10^5$). The next line contains 2^n integers $a_1, a_2, \ldots, a^{2n} (0 \leq a_i < 2^{30})$. Each of the next m lines contains queries. The i-th line contains integers $p_i, b_i (1 \leq p_i \leq 2^n, 0 \leq b_i < 2^{30})$ – the i-th query.

Output

Print m integers – the i-th integer denotes value v for sequence a after the i-th query.

Exemplo de entradas e saídas

Sample Input

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

Sample Output

```
1
3
3
3
```

Solução com complexidade $O(M \log N + N)$

- O problema consiste em um vetor cujo tamanho é uma potência de 2, cujas operações são a atualização de um elemento em particular ou uma consulta (*query*) em todo intervalo de índices $[1, 2^N]$
- Assim, ele pode ser solucionado de forma eficiente através do uso de uma árvore de segmentos com implementação *bottom-up*
- A alternância entre as duas operações (OR e XOR) pode ser feita por meio de ponteiros para funções e *swaps*
- Esta solução tem complexidade $O(M \log N + N)$

Solução com complexidade $O(M \log N + N)$

```
#include <bits/stdc++.h>

using namespace std;
using Op = int (*)(int, int);

class SegmentTree
{
    int N;
    std::vector<int> ns;

public:
    SegmentTree(const std::vector<int>& xs) : N(xs.size()), ns(2*N)
    {
        std::copy(xs.begin(), xs.end(), ns.begin() + N);

        int k = N;

        Op op = [](int x, int y) { return x | y; };
        Op next = [](int x, int y) { return x ^ y; };

```

Solução com complexidade $O(M \log N + N)$

```
21          while (k >>= 1)
22          {
23              for (int i = k; i < 2*k; ++i)
24                  ns[i] = op(ns[2*i], ns[2*i + 1]);
25
26              swap(op, next);
27          }
28      }
29
30      int update(int i, int value)
31      {
32          int a = i + N - 1;
33          ns[a] = value;
34
35          Op op = [](int x, int y) { return x | y; };
36          Op next = [](int x, int y) { return x ^ y; };
37
38          while (a >>= 1) {
39              ns[a] = op(ns[2*a], ns[2*a + 1]);
40              swap(op, next);
41          }
```

Solução com complexidade $O(M \log N + N)$

```
42
43          return ns[1];
44      }
45  };
46
47  int main()
48  {
49      ios::sync_with_stdio(false);
50
51      int n, m;
52      cin >> n >> m;
53
54      vector<int> xs(1 << n);
55
56      for (int i = 0; i < (1 << n); ++i)
57          cin >> xs[i];
58
59      SegmentTree tree(xs);
60
```

Solução com complexidade $O(M \log N + N)$

```
61      while (m--)
62      {
63          int p, b;
64          cin >> p >> b;
65
66          cout << tree.update(p, b) << '\n';
67      }
68
69      return 0;
70  }
```

SPOJ – Maximum Sum

Problema

You are given a sequence $A[1], A[2], \ldots, A[N]$ ($0 \leq A[i] \leq 10^8$, $2 \leq N \leq 10^5$). 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, $1 \leq i \leq N$, and x, $0 \leq x \leq 10^8$.

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, $1 \leq x < y \leq N$.

You must find i and j such that $x \leq i, j \leq y$ and $i \neq j$, such that the sum $A[i] + A[j]$ is maximized. Print the sum $A[i] + A[j]$.

Entrada e saída

Input

The first line of input consists of an integer N representing the length of the sequence. Next line consists of N space separated integers $A[i]$. Next line contains an integer $Q, Q \leq 10^5$, representing the number of operations. Next Q lines contain the operations.

Output

Output the maximum sum mentioned above, in a separate line, for each Query.

Exemplo de entradas e saídas

Sample 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
```

Sample Output

```
7
9
11
12
```

Solução

- Um algoritmo *naive*, que percorre o intervalo $[x, y]$ em busca destes valores tem complexidade $O(QN)$ no pior caso, o que leva ao TLE
- Para melhorar esta complexidade, observe primeiro que os índices i, j que maximizam a soma $A[i] + A[j]$ correspondem aos dois maiores elementos no intervalo $[x, y]$
- Assim, pode-se utilizar uma árvore de segmentos para manter, para cada intervalo, os valores de seus dois maiores elementos
- Nas folhas, devem ser armazenados os pares $(A[i], 0)$
- Em cada nó, é preciso avaliar os pares armazenados nos filhos à esquerda e à direita, e escolher dentre eles os dois maiores
- Esta solução terá complexidade $O(Q \log N)$ no pior caso, de modo que a solução será aceita

Solução AC com complexidade $O(Q \log N)$

```
#include <bits/stdc++.h>

using namespace std;
using ll = long long;
using ii = pair<int, int>;

class SegmentTree
{
public:

    SegmentTree(const std::vector<ii>& xs) : N(xs.size()), ns(4*N)
    {
        for (size_t i = 0; i < xs.size(); ++i)
            update(i, xs[i]);
    }

    void update(int i, const ii& value)
    {
        update(1, 0, N - 1, i, value);
    }

```

Solução AC com complexidade $O(Q \log N)$

```
22      ll query(int a, int b)
23      {
24          auto ans = RSQ(1, 0, N - 1, a, b);
25          return ans.first + ans.second;
26      }
27
28  private:
29
30      int N;
31      std::vector<ii> ns;
32
33      void update(int node, int L, int R, int i, const ii& value)
34      {
35          if (i > R or i < L)
36              return;
37
38          if (L == R)
39          {
40              ns[node] = value;
41              return;
42          }
```

Solução AC com complexidade $O(Q \log N)$

```
44          update(2*node, L, (L+R)/2, i, value);
45          update(2*node + 1, (L+R)/2 + 1, R, i, value);
46
47          vector<ll> ys { ns[2*node].first, ns[2*node + 1].first,
48              ns[2*node].second, ns[2*node + 1].second };
49
50          sort(ys.begin(), ys.end());
51
52          ns[node] = ii(ys[3], ys[2]);
53      }
54
55      ii RSQ(int node, int L, int R, int a, int b)
56      {
57          if (a > R or b < L)
58              return ii(0, 0);
59
60          if (a <= L and R <= b)
61              return ns[node];
62
63          auto x = RSQ(2*node, L, (L + R)/2, a, b);
64          auto y = RSQ(2*node + 1, (L + R)/2 + 1, R, a, b);
```

Solução AC com complexidade $O(Q \log N)$

```
66          vector<ll> ys { x.first, x.second, y.first, y.second };
67
68          sort(ys.begin(), ys.end());
69
70          return ii(ys[3], ys[2]);
71      }
72  };
73
74  int main()
75  {
76      ios::sync_with_stdio(false);
77
78      int N;
79      cin >> N;
80
81      vector<ii> xs(N, ii(0, 0));
82
83      for (int i = 0; i < N; ++i)
84          cin >> xs[i].first;
85
86      auto tree = SegmentTree(xs);
```

Solução AC com complexidade $O(Q \log N)$

```
87
88      int Q;
89      cin >> Q;
90
91      while (Q--) {
92          string cmd;
93          int x, y;
94
95          cin >> cmd >> x >> y;
96
97          switch (cmd.front()) {
98          case 'U':
99              tree.update(x - 1, ii(y, 0));
100             break;
101         default:
102             cout << tree.query(x - 1, y - 1) << '\n';
103         }
104     }
105
106     return 0;
107 }
```

Live Archive – Problem 6139: Interval Product

Problema

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 $X_1, X_2, \ldots, X_N$. 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.

Problema

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.

Entrada e saída

Input

Each test case is described using several lines. The first line contains two integers N and K, indicating respectively the number of elements in the sequence and the number of rounds of the game ($1 \leq N, K \leq 10^5$). The second line contains N integers X_i that represent the initial values of the sequence ($-100 \leq X_i \leq 100$ for $i = 1, 2, \ldots, N$). Each of the next K lines describes a command and starts with an uppercase letter that is either 'C' or 'P'. If the letter is 'C', the line describes a change command, and the letter is followed by two integers I and V indicating that X_I must receive the value $V (1 \leq I \leq N and -100 \leq V \leq 100)$. If the letter is 'P', the line describes a product command, and the letter is followed by two integers I and J indicating that the product from X_I to X_J, inclusive must be calculated ($1 \leq I \leq J \leq N$). Within each test case there is at least one product command.

Entrada e saída

Output

For each test case output a line with a string representing the result of all the product commands in the test case. The i-th character of the string represents the result of the i-th product command. If the result of the command is positive the character must be '+' (plus); if the result is negative the character must be '-' (minus); if the result is zero the character must be '`0`' (zero).

Exemplo de entradas e saídas

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
```

Solução com complexidade $O(K \log N + K)$

- Embora fique claro a necessidade do uso de uma árvore de segmentos (ou uma árvore de Fenwick), alguns cuidados são necessários
- Em primeiro lugar, se os números forem armazenados de acordo com os valores dados na entrada, ocorrerá o erro de *overflow*
- Além disso, o zero não é inversível na operação de multiplicação
- Assim, os zeros devem ser armazenados e tratados à parte
- Uma saída é manter duas árvores, que mantenham o registro dos números negativos e dos zeros
- Assim, em uma consulta, se o número de zeros no intervalo for maior que zero, a resposta será zero
- Caso contrário, o resultado depende da paridade do número de negativos no intervalo

Solução com complexidade $O(K \log N + K)$

```
#include <bits/stdc++.h>

using namespace std;

class SegmentTree
{
    int N;
    std::vector<int> ns;

public:
    SegmentTree(int n) : N(n), ns(4*N, 0) { }

    void update(int i, int value)
    {
        update(1, 0, N - 1, i, value);
    }

    int RSQ(int a, int b)
    {
        return RSQ(1, 0, N - 1, a, b);
    }
```

Solução com complexidade $O(K \log N + K)$

```
22
23 private:
24     void update(int node, int L, int R, int i, int value)
25     {
26         if (i > R or i < L)
27             return;
28
29         ns[node] += value;
30
31         if (L == R)
32             return;
33
34         update(2*node, L, (L+R)/2, i, value);
35         update(2*node + 1, (L+R)/2 + 1, R, i, value);
36     }
37
38     int RSQ(int node, int L, int R, int a, int b)
39     {
40         if (a > R or b < L)
41             return 0;
42
```

Solução com complexidade $O(K \log N + K)$

```
43          if (a <= L and R <= b)
44              return ns[node];
45
46          auto x = RSQ(2*node, L, (L + R)/2, a, b);
47          auto y = RSQ(2*node + 1, (L + R)/2 + 1, R, a, b);
48
49          return x + y;
50      }
51  };
52
53  int main()
54  {
55      int N, K;
56
57      while (cin >> N >> K) {
58          vector<int> xs(N);
59          SegmentTree zeroes(N), negatives(N);
60
61          for (int i = 0; i < N; ++i) {
62              int x;
63              cin >> x;
```

Solução com complexidade $O(K \log N + K)$

```
64
65              if (x == 0)
66                  zeroes.update(i, 1);
67              else if (x < 0)
68                  negatives.update(i, 1);
69          }
70
71          while (K--)
72          {
73              string cmd;
74              int x, y;
75
76              cin >> cmd >> x >> y;
77
78              switch (cmd.front()) {
79              case 'C':
80                  --x;
81
82                  zeroes.update(x, -zeroes.RSQ(x, x));
83                  negatives.update(x, -negatives.RSQ(x, x));
84
```

Solução com complexidade $O(K \log N + K)$

```
85                  if (y == 0)
86                      zeroes.update(x, 1);
87                  else if (y < 0)
88                      negatives.update(x, 1);
89
90                  break;
91              default:
92                  --x; --y;
93
94                  if (zeroes.RSQ(x, y))
95                      cout << 0;
96                  else
97                      cout << (negatives.RSQ(x, y) % 2 ? '-' : '+');
98              }
99          }
100
101         cout << '\n';
102     }
103
104     return 0;
105 }
```

Referências

1. Codeforces Round #197 (Div. 2) – Xenia and Bit Operations
2. SPOJ – Maximum Sum
3. ICPC Live Archive – Problem 6139: Interval Product