Codeforces Round #179

Problema A: *Greg and Array*

Prof. Edson Alves – UnB/FCTE

Problema

Greg has an array $a = a_1, a_2, \ldots, a_n$ and m operations. Each operation looks as: $l_i, r_i, d_i, (1 \leq l_i \leq r_i \leq n)$. To apply operation i to the array means to increase all array elements with numbers $l_i, l_i + 1, \ldots, r_i$ by value d_i.

Greg wrote down k queries on a piece of paper. Each query has the following form: $x_i, y_i, (1 \leq x_i \leq y_i \leq m)$. That means that one should apply operations with numbers $x_i, x_i + 1, \ldots, y_i$ to the array.

Now Greg is wondering, what the array a will be after all the queries are executed. Help Greg.

Entrada e saída

Input

The first line contains integers $n, m, k (1 \le n, m, k \le 10^5)$. The second line contains n integers: $a_1, a_2, \ldots, a_n (0 \le a_i \le 10^5)$ – the initial array.

Next m lines contain operations, the operation number i is written as three integers: $l_i, r_i, d_i, (1 \le l_i \le r_i \le n), (0 \le d_i \le 10^5)$.

Next k lines contain the queries, the query number i is written as two integers: $x_i, y_i, (1 \le x_i \le y_i \le m)$.

The numbers in the lines are separated by single spaces.

Output

On a single line print n integers $a_1, a_2, \ldots, a_n$ – the array after executing all the queries. Separate the printed numbers by spaces.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the cin, cout streams of the `%I64d` specifier.

Exemplo de entradas e saídas

Sample Input

```
3 3 3
1 2 3
1 2 1
1 3 2
2 3 4
1 2
1 3
2 3

1 1 1
1
1 1 1
1 1
```

Sample Output

```
9 18 17

2
```

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

- A solução tem três partes
- A primeira é acumular o número de vezes que cada operação deverá ser realizada
- Isto pode ser feito com uma árvore de Fenwick com suporte para *range update*
- Em seguida, deve-se acumular o impacto de cada operação no vetor original
- O número de vezes x que a operação i será aplicada pode ser feita com uma *point query* na árvore
- Novamente é necessária uma árvore de Fenwick com suporte para *range update*
- O intervalo $[L, R]$ deve ser atualizado com o valor dx
- Por fim, a cada posição do vetor i deve ser adicionado o valor y obtido pela *point query* do índice i da árvore

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

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

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

class BITree {
private:
    vector<ll> ts;
    size_t N;

public:
    BITree(size_t n) : ts(n + 1, 0), N(n) {}

    ll value_at(int i)
    {
        return RSQ(i);
    }
```

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

```
20      void range_add(size_t i, size_t j, ll x)
21      {
22          add(i, x);
23          add(j + 1, -x);
24      }
25
26  private:
27      int LSB(int n) { return n & (-n); }
28
29      ll RSQ(int i)
30      {
31          ll sum = 0;
32
33          while (i >= 1) {
34              sum += ts[i];
35              i -= LSB(i);
36          }
37
38          return sum;
39      }
```

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

```
41      void add(size_t i, ll x)
42      {
43          while (i <= N)
44          {
45              ts[i] += x;
46              i += LSB(i);
47          }
48      }
49  };
50
51  struct Op
52  {
53      int L, R;
54      ll d;
55  };
56
57  vector<ll> solve(int N, int M, const vector<int>& as,
58                   const vector<Op>& ops, const vector<ii>& qs)
59  {
60      BITree op_tree(M);
```

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

```
62      for (const auto& q : qs)
63          op_tree.range_add(q.first, q.second, 1);
64
65      BITree ft(N);
66
67      for (int i = 1; i <= M; ++i)
68      {
69          auto x = op_tree.value_at(i);
70          ft.range_add(ops[i].L, ops[i].R, x * ops[i].d);
71      }
72
73      vector<ll> ans(N + 1);
74
75      for (int i = 1; i <= N; ++i)
76          ans[i] = as[i] + ft.value_at(i);
77
78      return ans;
79  }
```

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

```
81 int main()
82 {
83     ios::sync_with_stdio(false);
84
85     int N, M, K;
86     cin >> N >> M >> K;
87
88     vector<int> as(N + 1);
89
90     for (int i = 1; i <= N; ++i)
91         cin >> as[i];
92
93     vector<Op> ops(M + 1);
94
95     for (int i = 1; i <= M; ++i) {
96         int L, R, d;
97         cin >> L >> R >> d;
98
99         ops[i] = Op { L, R, d };
100    }
```

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

```
102     vector<ii> qs(K);
103
104     for (int i = 0; i < K; ++i)
105         cin >> qs[i].first >> qs[i].second;
106
107     auto ans = solve(N, M, as, ops, qs);
108
109     for (size_t i = 1; i < ans.size(); ++i)
110         cout << ans[i] << (i + 1 == ans.size() ? '\n' : ' ');
111
112     return 0;
113 }
```